Construction d'un concept à l'interface entre informatique et algèbre élémentaire au collège : la variable dans un rôle de paramètre

Jean-Marc Legrand

COLLEGE DOCTORAL PAYS DE LA LOIRE
ED EDUCATION, COGNITION, LANGAGES, INTERACTIONS, SANTE

Nantes Université

THÈSE DE DOCTORAT

NANTES UNIVERSITE

ECOLE DOCTORALE N° 603
Education, Cognition, Langages, Interactions, Santé
Spécialité : « *Sciences de l'éducation* »

Par
Jean-Marc LEGRAND

Construction d'un concept à l'interface entre informatique et algèbre élémentaire au collège : la variable dans un rôle de paramètre

Enquête épistémologique et analyse de l'activité des élèves à l'aide d'une captation automatisée de leurs actions de programmation dans une situation de généralisation de motif informatisée

VOLUME 1

Thèse présentée et soutenue à Nantes, le 18/12/2023
Unité de recherche : CREN

Rapporteurs avant soutenance :

Ghislaine GUEUDET	Professeure, Université Paris-Saclay
Cécile OUVRIER-BUFFET	Professeure, UPEC

Composition du Jury :

Présidente :	Ghislaine GUEUDET	Professeure, Université Paris-Saclay
Examinateurs :	Cécile OUVRIER-BUFFET	Professeure, UPEC
	Lalina COULANGE	Professeure, Université de Bordeaux
	Simon MODESTE	Maître de Conférences, Université de Montpellier
Dir. de thèse :	Magali HERSANT	Professeure, Université de Nantes
Co-dir. de thèse :	Christophe DECLERCQ	Maître de Conférences, Université de La Réunion

Volume Un

Sommaire

Volume Un

Volume Deux : Annexes

Résumé

L'algorithmique et l'algèbre, et plus généralement l'informatique et l'algèbre, sont réputés être des cadres affichant une certaine proximité. Une étude épistémologique des concepts à l'interface entre ces cadres nous permet de souligner l'importance de la variable dans un rôle de paramètre, nécessaire pour la généralisation, mais aussi obstacle épistémologique potentiel. Dans ce rôle, la traditionnelle opposition entre variable informatique et variable mathématique est dépassée au profit d'une équivalence sémiotique. Afin d'identifier les conditions de la construction par les élèves de la nécessaire existence du paramètre et de sa représentation dans un certain registre sémiotique (algébrique ou informatique), nous étudions une situation de généralisation de motif informatisée, censée permettre la construction de ce concept. Pour analyser la dynamique de l'activité des élèves avec une granularité fine, nous avons conçu un dispositif de captation automatisée des actions de programmation des élèves dans l'environnement de programmation graphique par blocs mobilisé pour la situation étudiée (Snap !). L'analyse dans le Cadre de l'Apprentissage par Problématisation, qui s'appuie sur les traces de programmation de deux classes de 4ème, montre la nécessité de faire vivre aux élèves la tension fixe-variable du paramètre, entre l'unicité du symbole et la multiplicité de ses dénotés possibles. En outre, les faits construits par les élèves pour poser et construire le problème de la généralisation, sont éminemment liés aux sous-problèmes qu'ils se posent, et cela amène à identifier l'intérêt potentiel des caricatures dans les situations d'apprentissage de concepts informatiques.

Abstract

Algorithms and algebra, and more generally computer science and algebra, are reputed to be frameworks that display a certain proximity. An epistemological study of the concepts at the interface between these frameworks allows us to emphasize the importance of the variable in the role of parameter, necessary for generalization, but also a potential epistemological obstacle. In this role, the traditional opposition between computer variable and mathematical variable is overcome in favour of a semiotic equivalence. In order to identify the conditions under which students construct the necessary existence of the parameter and its representation in a certain semiotic register (algebraic or computerised), we study a situation involving the computerised generalisation of a pattern, which is supposed to enable the construction of this concept. In order to analyse the dynamics of the students' activity at a fine level of granularity, we designed a computerised device to capture the students' programming actions in the block-based graphical programming environment used for the situation under study (Snap!). Analysis within the Problem-Based Learning framework, based on the programming traces of two 4th year classes, shows the need for pupils to experience the fixed-variable tension of the parameter, between the uniqueness of the symbol and the multiplicity of its possible denotations. Furthermore, the facts constructed by the pupils to pose and construct the problem of generalisation are eminently linked to the sub-problems they pose themselves, and this leads us to identify the potential interest of caricatures in situations for learning computer science concepts.

Remerciements

Algorithme 0.0.0 : Remerciements

Entrées : $p \in R$, R étant l'ensemble des personnes qui, d'une manière ou d'une autre, m'ont permis d'aboutir à ce travail.
Sorties : Un message de remerciement personnalisé
initialisation ;
`dire` ('Merci à '+p)

Parmi les nombreux éléments de R, que je ne saurais tous citer, se trouvent notamment Magali Hersant et Christophe Declercq, pour m'avoir dirigé, suivi, conseillé et encouragé : Magali, qui par son exigence et sa capacité à localiser ce qui « gratte », m'a permis d'apprendre à (tenter de) toujours préciser et justifier mes pensées et mes propos, et Christophe, qui m'a ouvert les portes de la didactique de l'informatique, tout comme celles de sa maison. R contient aussi Lalina Coulange et Simon Modeste, pour m'avoir, eux aussi, conseillé et encouragé — nos échanges annuels me manquent déjà — et pour avoir accepté de surcroît de faire partie du jury. Je remercie aussi Ghislaine Gueudet et Cécile Ouvrier-Buffet pour avoir accepté d'être rapporteures.

Dans R se retrouvent aussi les nombreux et aimés collègues[1] du collège où j'ai passé de si bonnes années, sans oublier ni les élèves, ni notre principal P.R. qui m'a soutenu dans mon aventure vers l'Inspé. Parmi ces collègues, je ne peux pas ne pas nommer Aurélie et ses collègues de mathématiques, et de façon plus personnelle, mes amis de Segpa, Loïc et Isa, que j'ai toujours l'impression d'avoir abandonnés. R s'en aussi enrichi à mon arrivée à l'Inspé du site d'Angers, tout particulièrement avec mon moustachu préféré, pour qui on aurait dû empêcher la retraite avant au moins 80 ans !

Enfin, R ne serait rien sans mon père, d'une part, et surtout sans les femmes de ma vie, Nadia, ma seule et unique danseuse, et mes filles adorées Ambre et Agate, qui m'ont soutenu et supporté — dans les deux sens du terme — tout au long de ces six longues années.

R est bien pratique : même si je ne vous ai pas cité, vous savez que vous en êtes un élément, donc je ne vous ai pas oublié !

1. Sauf deux, rappelez-vous !

à mon père, qui m'a offert le goût du questionnement (et un VIC-20), et qui aurait apprécié, sans forcément le dire, de discuter de cette recherche.

Introduction

En tant qu'enseignant spécialisé exerçant en SEGPA [2], nous avons été confronté à la difficulté de faire entrer les élèves dans l'algèbre élémentaire au cours du cycle 4. Cette difficulté n'est pas spécifique à ces élèves, les collègues enseignantes en mathématiques dans le même collège l'observaient aussi. Squalli et Bronner (2017, p. 1) soulignent d'ailleurs que « l'enseignement et l'apprentissage de l'algèbre ont toujours posé et posent encore problème », quinze ans après que le rapport Kahane sur le calcul (Ministère de l'éducation nationale et Commission de réflexion sur l'enseignement des mathématiques, 2002, p. 4) évoquait « les difficultés résistantes [...] pour beaucoup d'élèves » qu'engendrait le passage de l'arithmétique à l'algèbre. Cette difficulté n'est pas spécifique non plus à la France :

> The algebra research carried out during the latter decades of the 20th century with 12- to 15-year-olds pointed to some of the shortcomings of an arithmetic way of thinking when students first experience algebra in high school. The countless research studies that had disclosed the difficulties involved in moving from an arithmetic to an algebraic form of reasoning (e.g., Kieran 1992 ; Linchevski 1995 ; Rojano and Sutherland 2001 ; Wagner and Kieran 1989) (Kieran et al., 2016, p. 3).

Ce passage délicat de l'arithmétique à l'algèbre est illustré par diverses difficultés recensées par (Squalli et Bronner, 2017, p. 5-6), notamment :

- le signe d'égalité est vu comme l'annonce d'un résultat,
- les élèves cherchent à obtenir une valeur numérique et n'arrivent pas à laisser des opérations en suspens,
- les élèves n'utilisent pas des symboles mathématiques pour représenter des relations entre des quantités,
- ils ne savent pas non plus utiliser les lettres comme nombres généralisés ou comme variables,
- et ils ont des difficultés à opérer sur les inconnues ou à transformer des expressions.

Kieran (2004, p. 140-141) relève des difficultés similaires pour les élèves afin de changer de mode de pensée. En effet, pour l'algèbre, il s'agit notamment de passer à :

1. A focus on relations and not merely on the calculation of a numerical answer ;
2. A focus on operations as well as their inverses, and on the related idea of doing / undoing ;
3. A focus on both representing and solving a problem rather than on merely solving it ;
4. A focus on both numbers and letters, rather than on numbers alone. This includes :
   (i) working with letters that may at times be unknowns, variables, or parameters ;
   (ii) accepting unclosed literal expressions as responses ;
   (iii) comparing expressions for equivalence based on properties rather than on numerical evaluation ;
5. A refocusing of the meaning of the equal sign.

En outre, le calcul algébrique ne se limite pas à la résolution d'équations, or « la fonction du calcul algébrique comme outil de généralisation et de preuve est mal assumée (Ministère de l'éducation nationale et Commission de réflexion sur l'enseignement des mathématiques, 2002, p. 30) ». De plus :

2. Section d'Enseignement Général et Professionnel Adapté

> Faire comprendre la puissance mathématique que donne le calcul algébrique, faire comprendre que, tout en ayant des modes de contrôle différents du calcul arithmétique, le calcul algébrique n'est pas un calcul aveugle, constituent des enjeux fondamentaux pour l'enseignement du calcul dans la scolarité obligatoire. Les diverses évaluations à grande échelle existantes comme les travaux de recherche montrent cependant que, pour beaucoup d'élèves, ces prises de conscience, sans doute insuffisamment préparées, ne se font pas. Le calcul algébrique est affaire de contrat didactique et les dérapages formels y abondent. (ibid., p. 29)

Ainsi, l'introduction du symbolisme formel engendre des difficultés tant pour les élèves que pour les enseignants (Bronner et Squalli, 2021, p. 4).

En France, une piste évoquée afin de remédier à ces difficultés est « le travail de production de formules, associé par exemple à des situations de dénombrement » (Ministère de l'éducation nationale et Commission de réflexion sur l'enseignement des mathématiques, 2002, p. 31) afin de saisir ce qu'est une démarche de généralisation, et ce travail gagnerait à être soutenu « par des visualisations, des traductions dans un autre cadre » (ibid., p. 29), cadre étant ici entendu au sens de Douady (1984).

À ce premier questionnement sur les difficultés d'entrée dans l'algèbre, s'est conjugué un intérêt doublé d'une opportunité : l'entrée de l'algorithmique et de la programmation dans les programmes du primaire et du secondaire. Ayant une formation informatique, l'introduction de l'enseignement de l'informatique dans les programmes de 2016, suite à un mouvement international d'ampleur à la fin des années 2000 afin de valoriser cet enseignement (Chiprianov, Coulange et Train, 2018, p. 20), a suscité notre intérêt. Intérêt d'un point de vue scientifique pour la discipline, mais aussi d'un point de vue didactique, avec les nouvelles questions et opportunités engendrées par ce nouvel enseignement. :

> la recherche de raisons d'être des savoirs informatiques constitue un problème premier de transposition didactique posé à la profession : comment développer un enseignement scolaire de l'informatique (ou de domaines particuliers de l'informatique) qui apparaisse à la fois fidèle à l'informatique et pertinent pour la formation scolaire des élèves (en tant que futurs citoyens) (ibid., p. 26)

Cet enseignement de la « Science Informatique » n'est pas disciplinaire : l'algorithmique et la programmation, au collège, sont intégrées dans les apprentissages en mathématiques et en technologie. Si la non-reconnaissance de l'informatique en tant que discipline scientifique à part entière est source de discussion, ce lien avec les autres disciplines est caractéristique de l'informatique, et « les activités de programmation ont été vues par beaucoup d'innovateurs comme pouvant contribuer aux apprentissages dans les domaines scientifiques existants, les mathématiques en premier lieu » (Lagrange et Rogalski, 2017, p. 4). Ainsi, la proximité qui est souvent établie entre algorithmique et algèbre pourrait servir de point d'appui pour travailler les difficultés identifiées d'entrée dans l'algèbre élémentaire. Lagrange et Rogalski (ibid., p. 14) soulignent d'ailleurs qu'« il est possible de faire le lien d'une part avec l'algèbre » :

> Concevoir les variables informatiques comme représentant des objets familiers mais avec un fonctionnement différent, présente en effet une certaine similarité avec la compréhension du symbolisme algébrique comme outil de modélisation du réel, une dimension importante de l'enseignement de l'algèbre soulignée par exemple par Chevallard (1989). (ibid., p. 14)

D'autre part, certains aspects de l'informatique sont purement algébriques (Guttag, Horowitz et Musser, 1978). Cependant, « les apprentissages algébriques restent peu assurés au moment où commencent les apprentissages en algorithmique et programmation » (Lagrange et Rogalski, 2017, p. 14). Vient alors une première question : *serait-il possible d'utiliser les apprentissages en algorithmique et programmation — ou plus généralement en informatique — afin de construire ou consolider certains concepts algébriques ?*

Cette question implique de dépasser la simple croyance commune d'une proximité algèbre-algorithmique (ou plus généralement algèbre-informatique), il faudra donc établir *quels sont les concepts à l'interface entre algèbre et informatique* (chapitre 1, p. 17).

Une fois cette étude épistémologique effectuée, nous en extrairons un concept qui apparait comme fondamental tant pour l'algèbre que pour l'informatique, *le paramètre* ou ce qui nous décrirons comme *la variable dans un rôle de paramètre* (1.9, p. 105). Dans le chapitre 2 (p. 127), nous chercherons ainsi *une situation informatique susceptible de permettre la construction de ce concept de variable dans un rôle de paramètre* (2.1, p. 131). Nous nous intéressons ici à la construction d'un concept comme une connaissance apodictique, et non seulement assertorique, l'analyse de la mise-en-œuvre de la situation dans une classe ordinaire sera ainsi faite dans le *Cadre de l'Apprentissage par Problématisation* (CAP) .

Nous intéressant à un processus dynamique dans un contexte de modification de programme, il nous faudra disposer des traces de programmation des élèves rendant compte de leur activité. Nous préciserons ainsi *le dispositif de captation automatisée des actions de programmation* que nous avons conçu (2.2, p. 155).

La description et l'analyse de l'activité des élèves dans une situation mobilisant des concepts algébriques, algorithmiques ou informatiques, nécessitera de convoquer des éléments issus d'autres cadres théoriques (2.4, p. 171). Ces différents éléments d'explicitation théorique nous permettront de tenter d'établir une méthodologie cohérente et objective de reconstitution et représentation des données (2.5, p. 196).

L'analyse de la situation expérimentée sera axée sur *les conditions de construction ou non de la nécessité de la variable dans un rôle de paramètre*, avec une granularité fine sur certains groupes, complétée par des éléments issus des autres groupes observés (chapitre 3, p. 217).

Nous conclurons enfin sur les conditions de possibilité de construction du concept, ainsi que sur la situation expérimentée, le dispositif de captation et la méthodologie le mobilisant.

Chapitre 1

Enquête épistémologique

Sommaire du chapitre

Liste des figures

Liste des tables

Les mots « algorithme » et « algèbre » ont une origine historique commune, en référence au nom latinisé du mathématicien Perse al-Khwārizmī pour le premier, et au terme « al-jabr » utilisé par celui-ci pour décrire une « opération qui consiste à faire disparaître un terme négatif qui figure dans l'un des membres d'une équation, en ajoutant la valeur absolue de ce terme aux deux membres » (Farès, 2015).

Chabert (2010, p. 5) rappelle par ailleurs que « dans l'Encyclopédie, d'Alembert décrit ainsi le mot algorithme : "Terme Arabe, employé par quelques auteurs, et singulièrement par les Espagnols pour signifier la pratique de l'algèbre [...] Le même mot se prend, en général, pour désigner la méthode et la notation de toute espèce de calcul." »

Mais cette origine commune n'est pas suffisante. Si nous voulons étudier la possibilité d'utiliser l'enseignement de l'algorithmique ou de l'informatique pour aider les élèves à entrer dans l'algèbre élémentaire, il nous faut établir les concepts à l'interface entre algèbre et algorithme, algorithmique, voire informatique. Les conditions épistémologiques de leur émergence seront aussi nécessaires, puisque le chercheur en didactique « est ainsi amené à chercher dans les études épistémologiques des appuis pour comprendre les processus d'enseignement et d'apprentissage des concepts scientifiques et pour nourrir des élaborations didactiques permettant d'agir sur ces processus » (Bächtold, Durand-Guerrier et Munier, 2017, p. 9).

Nous allons donc nous appuyer sur une enquête épistémologique croisée de l'algorithmique, de l'informatique, et de l'algèbre, afin d'en dégager les concepts à l'interface, et d'en extraire un concept qui apparait fondamental : le paramètre, ou ce qui nous définirons comme *la variable dans un rôle de paramètre*.

1.1 Algèbre et algèbre élémentaire

Il ne semble pas y avoir de définition « universelle » de l'algèbre élémentaire. Nous allons néanmoins tenter d'extraire une approche de définition utilisable dans ce travail à partir de différents points de vue concernant l'algèbre et son enseignement à l'école élémentaire ou au collège.

1.1.1 La résolution et l'étude d'équations

Le Larousse encyclopédique [1] nous indique que l'algèbre est la « branche des mathématiques qui, dans sa partie classique, se consacre à la résolution par des formules explicites des équations algébriques et, dans sa partie moderne, étudie des structures (groupes, anneaux, corps, idéaux) et se prolonge par les algèbres linéaire et multilinéaire et par l'algèbre topologique ». Cette description de la partie « classique » de l'algèbre se trouve chez Jean-Luc Verley qui précise dans l'Encyclopédia Universalis [2] que si « l'algèbre au sens moderne, à savoir l'étude des structures algébriques indépendamment de leurs réalisations concrètes, ne s'est dégagée que très progressivement au cours du XIX^{e} siècle, [. . .] jusqu'alors, le propos essentiel de l'algèbre avait été la résolution, par des formules explicites, des équations algébriques ». Ce point de vue est proche de celui de Jean Itard dans le même ouvrage, selon lequel « jusqu'au début du XIX^{e} siècle, l'étude des équations constitue l'unique préoccupation des algébristes. » Ce qui rejoint l'opinion de Farès (2017b, p. 1) qui considère que « [d]epuis le début de sa fondation dans le livre d'al-Khwārizmī, l'algèbre demeure la discipline qui s'occupe de la résolution des équations (ou

1. É. Larousse (14 août 2020b). *Encyclopédie Larousse en ligne - algèbre latin médiéval algebra de l'arabe al-djabr réduction*. URL : https://www.larousse.fr/encyclopedie/divers/alg%C3%A8bre/19868 (visité le 14/08/2020).

2. J.-L. Verley (15 août 2020). *Algèbre*. In : Encyclopædia Universalis. URL : http://www.universalis-edu.com.budistant.univ-nantes.fr/encyclopedie/algebre/ (visité le 15/08/2020).

des systèmes d'équations) algébriques, et du calcul polynomial ». Le terme « polynomial » étant par ailleurs repris dans la définition de l'algèbre élémentaire (ou algèbre classique) donnée dans Wikipédia [3] : « branche des mathématiques dont l'objet est l'étude des opérations algébriques (addition, multiplication, soustraction, division et extraction de racine) sur les nombres réels ou complexes, et dont l'objectif principal est la résolution d'équations polynomiales. » Vitrac (2005), qui considère que « la manière d'envisager l'algèbre comme spécialité mathématique est [. . .] quelque chose d'historiquement variable » distingue [4] trois moments spécifiques : le « moment arabe » centré sur les expressions de type polynomial, un moment lié à la résolution algébrique de problèmes géométriques (XVI^{e}-XVII^{e}), et « plus récemment encore, l'algèbre a été caractérisée comme l'étude de structures ».

Toutes ces conceptions de l'algèbre « classique » ou « élémentaire », nous amènent définir le sens que nous donnons dans ce travail à l'expression « algèbre élémentaire » :
Définition 1 (Algèbre élémentaire) *L'algèbre élémentaire vise à la résolution et l'étude d'équations polynomiales.*

1.1.2 al-Khwārizmī, aux origines de l'algèbre et/ou de l'algorithmique ?

On s'accorde généralement à situer l'origine de cette conception de l'algèbre dans l'ouvrage *Kitā al Jabr wa-al-muqābala (Livre de l'algèbre et d'al-muqābala)* d'al-Khwārizmī, mathématicien perse de Bagdad, « né au cours des dernières décennies du VIII^{e} siècle [. . .][et qui] était encore en vie en 847 »(Rashed, 2007, p. 11). Selon Rashed (ibid., p. 13), parlant de cet ouvrage :

> N'est-il pas vrai que l'on y rencontre pour la première fois le projet d'une discipline mathématique différente de la géométrie et de l'arithmétique ? et que c'est seulement à partir de ce livre, et jamais avant, que se sont formées et développées les traditions de la recherche en algèbre ? N'est-il pas vrai que c'est dans ce livre que la discipline a trouvé son nom ?

Vitrac (2005, p. 37) a le même avis :

> Finalement, autant que nous puissions le savoir, le pas décisif — la constitution de l'algèbre comme discipline en tant que telle, autour de ce noyau qu'est la classification des équations — a été franchi dans la première moitié du IXe siècle, en Pays d'Islam, par al-Khwārizmī et ses pairs.

Pourtant, les sources de l'algèbre pourraient « dans un certain sens, remonter aux mathématiques égyptienne ou babylonienne, du fait que ces deux mathématiques comprenaient des pratiques "algébriques" (manipulations d'inconnues et d'équations) » (Farès, 2017b). Mais al-Khwārizmī n'attribue pas le même sens au concept d'équation : il commence par classer les équations en « six types canoniques d'équations du premier et du second degré » [5] (Rashed, 2007, p. 9). « Ce n'est pas lors de la résolution des problèmes qu'al-Khwārizmī trouve ces équations : la classification précède en effet tout problème » (ibid., p. 10). En fait, « Le concept d'équation apparaît dans le livre d'al-Khwārizmī pour désigner *une classe infinie de problèmes*, et non pas, comme chez les Babyloniens, au cours de la solution de l'un ou l'autre problème. Par ailleurs, les équations ne sont pas présentées au cours de la solution des problèmes à résoudre, comme chez les Babyloniens ou chez Diophante, mais dès le départ, à partir des termes primitifs dont les combinaisons doivent donner toutes les formes possibles » (Anawati et Rashed, 2020).

3. *Algèbre classique* (17 oct. 2019). In : *Wikipédia*. URL : https://fr.wikipedia.org/w/index.php?title=Alg%C3%A8bre_classique&oldid=163621400 (visité le 17/08/2020).
4. un peu arbitrairement, de son propre aveu.
5. Nous y reviendrons.

Ainsi la formulation de la méthode de résolution est en amont du problème : il s'agit d'établir *une procédure de résolution d'une classe infinie de problèmes*. Ceci renvoie à la définition d'un algorithme proposée par Modeste (2012, p. 25), définition que nous adopterons :

Définition 2 (Algorithme) *Un algorithme est une procédure de résolution de problème, s'appliquant à une famille d'instances du problème et produisant, en un nombre fini d'étapes constructives, effectives, non-ambigües et organisées, la réponse au problème pour toute instance de cette famille.*

Ainsi, si le mot *algorithme* vient de la latinisation du nom d'al-Khwārizmī dans les traductions latines de *al-jabr wa-al-muqābala*, son sens est lié à la démarche algébrique qui est, dans le sens premier de « algèbre élémentaire », la recherche d'une procédure de résolution d'une famille d'instance de problèmes formulables [6] par des expressions algébriques, soit des équations polynomiales — même si le domaine d'action des algorithmes ne saurait se limiter aux objets de l'algèbre. Nous pourrions modifier la Définition 1 ainsi :

Définition 3 (Algèbre élémentaire) *L'algèbre élémentaire vise à la recherche et l'étude d'algorithmes permettant la résolution d'une famille d'instances de problèmes, modélisables par des équations polynomiales.*

Cependant, si al-Khwārizmī fait un pas vers la généricité des expressions algébriques, on peut faire deux remarques :

1. D'une part, concernant notamment la modélisation, il manque « ce qui fait la force de l'algèbre » selon Chevallard (1989, p. 65) : les **paramètres**.
2. D'autre part, al-Khwārizmī n'était pas le premier à chercher à construire des procédures les plus génériques possibles : cet aspect était présent dès les premières traces écrites de problèmes algébriques.

Dans la section suivante, nous allons commencer par une incursion plus lointaine dans l'histoire [7] de l'algèbre afin notamment de justifier la deuxième remarque. En parcourant ensuite le temps vers notre époque, nous illustrerons peu à peu la première remarque, ainsi que d'autres concepts qui seront à la croisée de l'informatique et de l'algèbre. Nous établirons en particulier que la variable (informatique ou algébrique) en tant que *paramètre* est non seulement fondamentale pour l'algèbre, mais aussi fondatrice pour l'informatique. Nous verrons aussi que l'apparition de ce concept constitue une rupture épistémologique majeure, qui laisse augurer la possibilité d'un obstacle épistémologique chez des élèves de cycle 4 [8].

1.2 Premiers problèmes chez les Sumériens

Dans les premières civilisations de l'écrit, on retrouve des traces de résolution de problèmes modélisables sous une forme algébrique. Ainsi, dans plusieurs tablettes sumériennes de Shuruppak (2500 avant J.-C.), on retrouve la formulation et le résultat d'un même problème : le partage d'un grenier à grain entre un nombre inconnu d'hommes, de sorte que chacun reçoive la même quantité de grains, 7 « silà » (Chabert, 2010, p. 14). La description de l'algorithme (c'est-à-dire ici la procédure permettant de faire une division) n'est pas donnée, mais on retrouve dans deux tablettes au moins deux résolutions d'un même problème, formulable ainsi : $7x = 1152000$ [9]. S'agit-il ici de la présentation de différents algorithmes de résolution d'un même problème ? Sans

6. ou modélisables, nous y reviendrons.
7. On pourrait dire « la préhistoire ».
8. « Le créateur de la notion d'obstacle épistémologique, Gaston Bachelard, ne croyait pas qu'il puisse y avoir d'obstacles de ce genre en mathématiques. La théorie des situations didactiques a permis de prévoir des obstacles didactiques en partant des obstacles épistémologiques historiques et de les expliquer » (Brousseau, 2011, p. 3).
9. Un grenier à grain contient 1152000 « silà ».

doute pas, puisque seuls les résultats sont inscrits, l'algorithme utilisé dans chaque cas étant implicite [10]. S'agit-il d'un problème « type », utilisé à des fins d'enseignement, comme ce sera le cas pour la plupart des écrits liés à des pratiques algébriques ? Nous ne nous prononcerons pas sur ce cas, cela dépendant si la quantité de 7 « silà » de grains par personne a, ou n'a pas, un sens particulier pour cette civilisation : cette quantité était-elle considérée comme un exemple à caractère générique, — qui aurait tout aussi bien pu être 9 ou 5 —, ou bien correspondait-elle à, par exemple, une quantité nécessaire et minimale. Quoiqu'il en soit, le problème et l'exposé de sa solution ne semblent pas avoir de visée générique.

1.2.1 La « bulle-enveloppe » : une première variable informatique ?

Cette civilisation sumérienne nous offre aussi ce que l'on peut considérer comme une première construction matérielle d'une mémoire, d'un stockage de valeur, représentant « un certain nombre » : les « bulles-enveloppes »(figure 1.1, p. 24). Il s'agissait selon Laguës, Beaudouin et Chapouthier (2017) de « véritables contrats commerciaux » :

> Constitués de boules creuse d'argile, ceux-ci contenaient des calculi, petits objets de terre cuite représentant trois types de nombres : des sphères, des cônes, des cylindres, accompagnés de figurines illustrant les objets du contrat (brebis, récoltes...). La boule de glaise que le marchand marquait de son sceau était brisée en cas de contestation pour permettre la comparaison du nombre de calculi et de la marchandise livrée. (ibid., p. 48).

(a) Bulle-enveloppe et ses jetons de comptabilité. Terre cuite, période d'Uruk. Provenance : Tell de l'Acropole à Suse(Le Louvre)

(b) Calculi et tablettes d'argile - Oriental Institute Museum, University of Chicago

Figure 1.1 – Premières mémoires concrètes

10. $x = \frac{1152000}{7}$ pour la tablette 50 et $x = \frac{2400}{7}$ pour la tablette 671, 2400 « gur » étant équivalents à 1152000 « silà ».

Ainsi, une bulle-enveloppe est un objet concret, matériel, permettant de stocker la *représentation* (par les *calculi*) d'une certaine valeur (ici numérique), valeur qui est la mesure d'une propriété d'un autre objet défini (la quantité de sacs d'une cargaison, l'aire d'un champs...). Cette valeur n'est pas forcément connue *a priori* par un individu désirant opérer sur la propriété en question (comparer, tester l'égalité, partager...). Pour opérer, on peut faire référence à une certaine bulle-enveloppe qui dénote elle-même une certaine propriété.

Cette bulle-enveloppe peut être *initialisée* : lors de sa création, la représentation d'une valeur par des *calculi* lui est associée en la remplissant par ces *calculi*. Elle peut aussi être *consultée* : en cassant la bulle-enveloppe, on a accès à la représentation de la valeur dénotée [11]. Dans certains cas, la valeur dénotée par le contenu de la boule peut être inscrite (avec une représentation écrite des nombres) sur la bulle, rendant la consultation non destructive [12]. En revanche, elle ne peut être *modifiée* : toute modification effective détruit la mémoire, et elle entraîne nécessairement la création d'une nouvelle mémoire si on désire conserver la modification.

Il n'y a certes pas ici — à notre connaissance — de nom, ou de signe permettant de représenter cette « mémoire », mais nous sommes proches de la définition d'une *variable informatique*.

Variable informatique

En effet, selon Modeste (2012), une variable informatique est « une variable qui joue le rôle d'un emplacement en mémoire pouvant changer de valeur au cours du temps. Par une opération d'affectation, on peut associer une nouvelle valeur à la variable, l'ancienne valeur est alors perdue. » Ceci rejoint la définition donnée dans Wikipédia [13] : « En informatique, une variable est un symbole (habituellement un nom) qui renvoie à une position de mémoire dont le contenu peut prendre successivement différentes valeurs pendant l'exécution d'un programme. » Plus précisément, « Dans les langages de programmation impératifs, ce que les informaticiens appellent des variables sont des repères de valeurs qui évoluent au cours du temps, on parle aussi de références. Il s'agit donc plutôt de l'identification d'emplacements en mémoire. Si une variable informatique n'est pas initialisée, sa valeur est non définie. » [14]. Par ailleurs, Rutishauser (1951) différencie clairement, dans sa proposition de formalisation d'un langage informatique :

— `(n)` : le nombre contenu dans la cellule mémoire d'adresse (ou de numéro) n,
— `<x>` : l'adresse (ou le numéro) de la cellule mémoire dans laquelle le nombre x est stocké,
— `x` $\rightarrow$ `n` : l'action de stocker le nombre x dans la cellule mémoire n.

Ainsi, nous pourrons considérer qu'une définition simple pourrait être celle utilisée dans (Calmet, Hirtzig et Wilgenbus, 2016) :

Définition 4 (Variable informatique) *Une variable informatique est un nom que l'on donne à une zone de mémoire. Elle permet de stocker une valeur et de la réutiliser plus tard, ou de la modifier.*

Une définition plus complète serait :

Définition 5 (Variable informatique) *Une variable informatique est un nom, ou un* identificateur*, que l'on donne à une zone de mémoire. Elle permet de stocker* la représentation d'*une valeur et de la réutiliser plus tard, ou de la modifier.*

11. La bulle-enveloppe a aussi propriété de permettre la validation, au sens didactique du terme.
12. L'objectif d'une inscription de la valeur contenue était de permettre une vérification. Plus tard, l'inscription suffira, supprimant la nécessité de la bulle et amenant les premières représentations écrites du nombre.
13. *Variable* (30 août 2019). In : *Wikipédia*. URL : https://fr.wikipedia.org/w/index.php?title=Variable&oldid=162222557 (visité le 15/04/2020).
14. *Variable (informatique)* (27 mars 2020). In : *Wikipédia*. URL : https://fr.wikipedia.org/w/index.php?title=Variable_(informatique)&oldid=168856866 (visité le 15/04/2020).

En effet, pour toute variable informatique, la valeur est nécessairement *représentée*, c'est-à-dire *codée* en utilisant un système de codage dépendant de la machine. Modeste (2012) précise ainsi que « [l]a variable informatique est un modèle de la mémoire d'une machine. » Ainsi, dans le cas des bulles-enveloppes, si l'on s'intéresse à l'implémentation de la variable du point de vue de la mémoire, la valeur est représentée, codée, par les *calculi* (figure 1.2, p. 26). En outre, les bulles-enveloppes contenaient aussi des « figurines illustrant les objets du contrat » (Laguës, Beaudouin et Chapouthier, 2017, p. 48). On a ainsi non seulement la représentation d'une valeur, mais aussi le « type » de valeur représentée (ici, le « nombre de » sacs, unités de récolte etc ...). De son côté, dans son *Rechenplanfortigung*, Rutishauser (1951) représente une valeur de type « nombre décimal » sous la forme $x = a \times 10^b$ en utilisant un codage sur 48 bits (un pour le signe, 40 pour la valeur a, 6 pour l'exposant b).

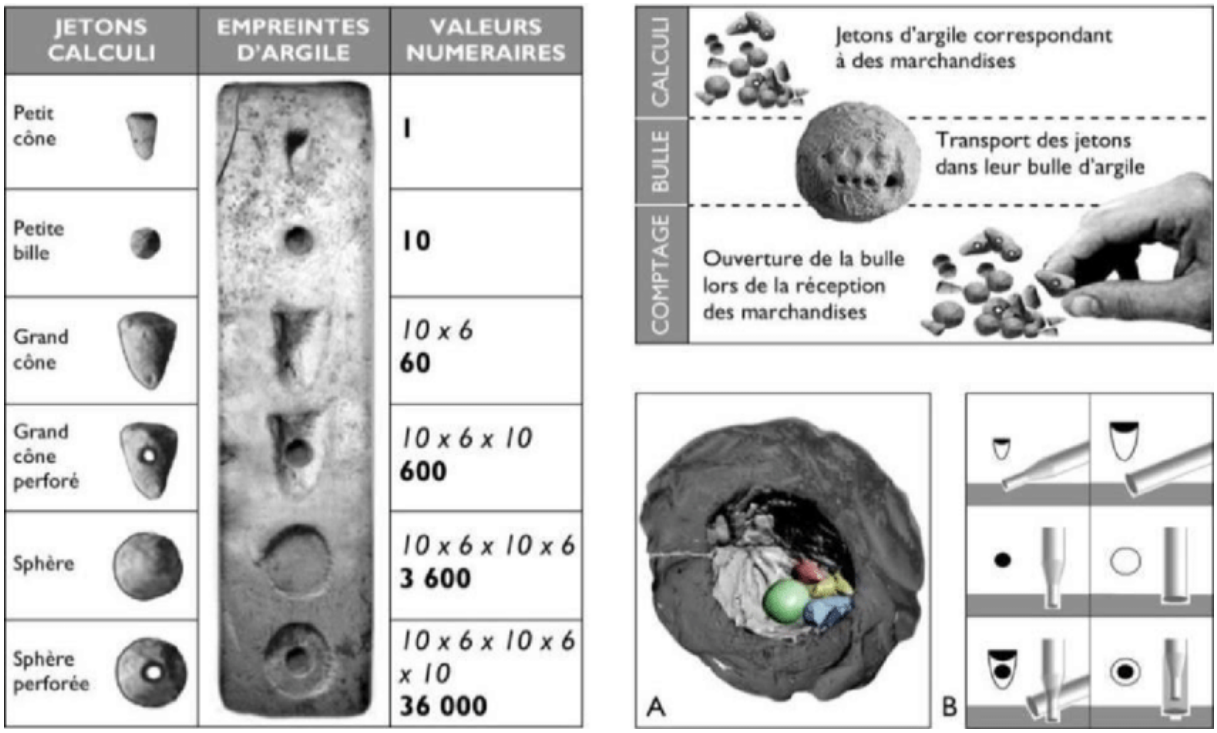
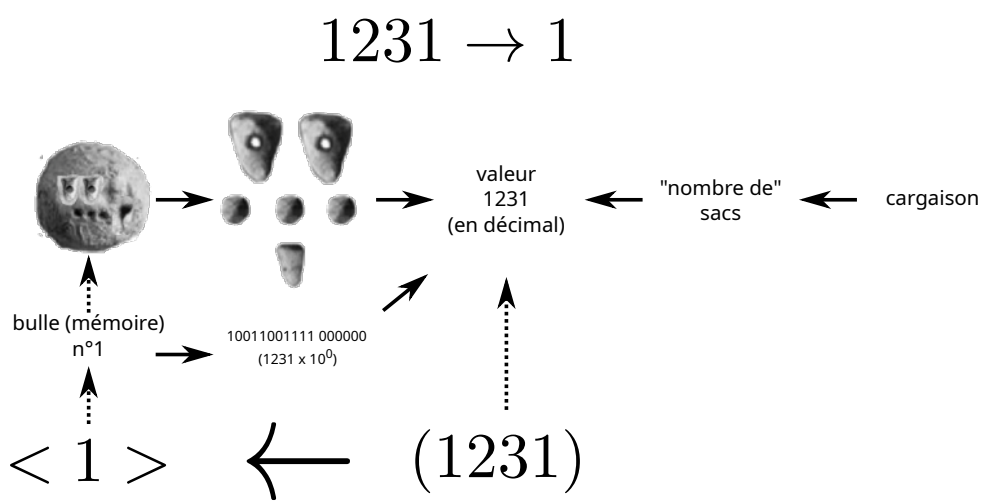


Figure 1.2 – « (à gauche) Liste des calculi, des marques numéraires et des valeurs numéraires semblant leur correspondre. (en haut à droite) La découverte de bulles d'argile contenant des jetons de comptabilité a conduit au concept selon lequel les bulles garantissaient lors du transport la non-dépréciation de la qualité et de la quantité des marchandises. (en bas à droite) A) Image 3D par tomographie numérique d'une bulle d'argile de Choga Mish contenant des calculi de différentes valeurs ; B) les encoches géométriques, réalisées avec deux calames sur la surface de la bulle d'argile, correspondent aux jetons qu'elle contient. »(Philippe Roi, 2013)

On a ainsi une chaîne de références que l'on pourrait représenter, en indiquant le codage sumérien et celui de Rutishauser par :



Ainsi, d'un *objet* « cargaison » est extrait une *propriété* « nombre de sacs ». Cette propriété a une certaine *valeur* « 1231 » qui est elle-même représentée (ou codée) : dans notre exemple, en suivant le codage numérique sumérien, ou le codage binaire de Rutishauser. Cette *représentation* de la valeur est stockée, mémorisée — et cette mémorisation est une des différences entre variable informatique et variable mathématique. Dans le cas de Rutishauser, `<1>` dénote cette mémoire physique, stockant la représentation d'une valeur. Par rapport à notre définition 5 (p. 25), il manque cependant un élément essentiel : le *nom*, ou l'identificateur, un élément du langage dans lequel s'exprime l'algorithme, qui à la fois fait référence à la mémoire physique, mais aussi, dans un algorithme, dénotera la valeur de la propriété de l'objet.

Sens et dénotation On voit ici que la bulle-enveloppe *dénote* le nombre de sacs de la cargaison : elle fait référence à une « portion de réalité ». `<1>` est un objet linguistique différent, ayant la même dénotation. Nous utilisons ici « dénotation » dans la définition qui en est donnée par Frege (1994) (« Bedeutung »), et mobilisée par Drouhard (1992) : « La dénotation d'un nom propre est l'objet même que nous désignons par ce nom » (Frege, 1994, p. 106), ou, plus précisément, la dénotation est la « portion de réalité » qu'une expression désigne. Un même objet peut ainsi avoir diverses expressions le dénotant. L'exemple donné par Frege est celui de Vénus : les expressions « l'étoile du soir » et « l'étoile du matin » sont différentes [15] mais elles désignent le même objet, la planète que nous désignons par ailleurs avec le nom « Vénus ». Drouhard (1992, p. 268) précise que la dénotation (d'une expression algébrique en l'occurrence, mais on peut l'étendre à toute expression) n'est pas une valeur, mais une fonction, notée δ. Nous conserverons cette notation dans la suite du texte : ainsi pour une certaine expression E, $\delta(E)$ désignera la dénotation de l'expression E. Notons que Drouhard précise davantage cette fonction en précisant qu'il s'agit d'« une fonction des assignations vers les valeurs », qu'il formalise ainsi :

$$u \overset{\delta}{\mapsto} val_u(E)$$

Où u est une assignation, une application de l'ensemble des variables dans $\mathbb{R}$, et $val_u(E)$ la valeur de l'expression E pour une assignation u. Ainsi, pour une expression $E = 2a + b$ et une assignation u_0 telle que $u_0(a) = 2$ et $u_0(b) = 3$, la valeur de E pour cette assignation u_0 est $val_{u_0}(E) = 7$ (ibid., p. 268). Pour une procédure manipulant une expression E, nous parlerons d'« instanciation », pour évoquer l'assignation au moment de l'exécution de cette procédure.

15. Drouhard dirait qu'elles n'ont pas la même connotation.

Rôles d'une variable Précisons ici que les bulles-enveloppes ont un rôle de *constante*, ce que Sajaniemi (2005, p. 7) nomme *fixed-value* : « A variable initialized without any calculation and not changed thereafter » : on verra que variable informatique et variable mathématique ont différents rôles ou différents statuts (voir 1.9.4, p. 117). Dans les sections suivantes, d'autres exemples nous permettront d'illustrer ce concept de variable informatique, dépendant de celui de machine, ainsi que leur rôle. Nous rencontrerons aussi les premières occurrences de variables mathématiques, liées aux premiers algorithmes. Le concept plus général de « variable » sera repris et étudié plus spécifiquement dans la section 1.9 (p. 105).

1.3 Les Babyloniens et la recherche de procédures génériques

Si les sumériens nous ont permis d'illustrer un aspect de l'informatique, la variable, nous ne nous avancerons pas à considérer qu'ils développaient ou utilisaient des procédés algorithmiques au sens de la Définition 2 (p. 23). Ces procédés semblent en revanche visibles chez les babyloniens (environ 2000 avant J.-C).

1.3.1 Des algorithmes instanciés

Parmi les nombreuses tablettes babyloniennes trouvées (-2000 à -1600 avant J.-C) (Høyrup, 2020), la traduction de la tablette VAT 6505 [16] donnée par Chabert (2010, p. 18) illustre tout à la fois la présence précoce d'algorithmes mathématiques, leur recherche de généricité, mais aussi la difficulté de représentation de ces algorithmes, faute d'outil de pensée suffisant.

Ainsi, l'extrait traduit présenté dans la figure 1.3 (p. 29) nous montre un exemple de résolution de problème algébrique, que l'on pourrait actuellement formuler ainsi : connaissant un nombre x, trouver son inverse u [17].

Au premier abord, cette tablette ne montre que des résolutions d'instances du problème, avec une résolution particulière spécifique à l'instance. Par exemple, l'énoncé 2 décrit la suite des opérations à effectuer afin de trouver l'inverse du nombre *4 ;10* [18], alors que l'énoncé 3 décrit la procédure permettant de trouver l'inverse de *8 ;20*. Cependant, en comparant ces différentes procédures, on constate qu'elles ne diffèrent que par les nombres utilisés (ce qui est mis en valeur dans la figure 1.3 (p. 29) par des couleurs identiques) [19]. Ainsi, cette tablette montre un même algorithme, pour plusieurs instances du problème. Tout se passe comme si le scribe souhaitait donner à voir l'algorithme [20] utilisé en donnant des exemples, ces instances prenant un caractère générique : à charge pour le lecteur, s'il veut résoudre une nouvelle instance du

16. Les tablettes babyloniennes sont référencées suivant leur collection et/ou le musée auquel elles sont rattachées. Par exemple, « YBC » pour « Yale Babylonian Collection », « BM » pour « British Museum ». « VAT » réfère à « Vorderasiatische Abteilung. Tontafeln », soit la collection des tablettes d'argiles du Vorderasiatisches Museum à Berlin. La Cuneiform Digital Dibrary Initiative (CDLI, https://cdli.mpiwg-berlin.mpg.de/) recense ces artefacts avec une numérotation particulière : la tablette VAT 6505 est référencée sous le numéro P254921.

17. C'est-à-dire le nombre u tel que $x \times u = 1$.

18. Chabert (2010, p. 16) précise cette écriture en chiffres décimaux de la base sexagésimale babylonienne. *2 ;13 ;20* désigne ainsi, normalement, le nombre $2 \times 60^2 + 13 \times 60 + 20$. Plus précisément, les babyloniens de cette époque ne représentant pas les absences de certaines puissances de 60, *2 ;13 ;20* désigne *un* des nombres $2 \times 60^m + 13 \times 60^n + 20 \times 60^p$, avec m, n, p entiers relatifs tels que $m > n > p$.

19. Notons qu'il en est de même pour la deuxième partie de la tablette, non montrée ici.

20. Précisons bien que cette vision en terme d'algorithme est actuelle, il ne semble pas que le concept d'algorithme soit connu à cette époque.

problème, de reconstituer l'algorithme et de procéder par substitution des nombres à caractère générique donnés dans cette tablette par les nombres caractérisant l'instance du problème en cours. L'algorithme pourrait être écrit ainsi, en écrivant x comme somme d'un nombre régulier [21]

Première partie de la tablette (début détruit) :

2 [Le nombre est **4;10**. Quel est son inverse ?
Procède comme suit.
Forme l'inverse de **10**, tu trouveras ***6***.
Multiplie ***6*** par **4**, tu trouveras ***24***.
Ajoute **1**, tu trouveras ***25***.
Forme l'inverse de ***25***] tu trouveras ***2;24***.
[Multiplie ***2;24*** par ***6***], tu trouveras ***14;24***.
[L'inverse est ***14;24***]. Telle est la façon de procéder.

3 [Le nombre est **8;20**]. Quel est son inverse ?
[Pro]cède comme suit.
[Forme l'inverse de **20**], tu trouveras ***3***.
Mul[tiplie ***3*** par **8**], tu trouveras ***24***.
[Ajoute **1**], tu trouveras ***25***.
[Forme l'inverse de ***25***] tu trouveras ***2;24***.
[Multiplie ***2;24*** par ***3***], tu trouveras [***7;12***.
L'inverse est ***7;12***. Telle est la façon de procéder].

4 [Le nombre est ***16;40***. Quel est son inverse ?
[Pro]cède comme suit.
Forme [l'inverse de ***6*** ;**40**], tu trouveras ***9***.
Multiplie [***9***]par **10**, tu trouveras ***1;30***.
[Aj]oute [**1**], tu trouveras ***2;30***.
Forme [l'inverse de ***2***]***;30***] tu trouveras ***24***.
[Multiplie ***24*** par ***9***, tu trouveras ***3;36***.
[L'inverse est ***3;36***]. Telle est la façon de procéder].

Figure 1.3 – Procédures de calcul des inverses - VAT6505 (entre 2000 et 1650 avant J.-C), (Chabert, 2010, p. 18), codage couleur ajouté par nous

y dont on connaît l'inverse et d'un autre nombre z (ibid., p. 19) :

Le nombre est $x = y + z$. Quel est son inverse ?
Procède comme suit.
Forme l'inverse de y, tu trouveras $\bar{y}$.
Multiplie $\bar{y}$ par z. Tu trouveras t.
Ajoute 1. Tu trouveras u.
Forme l'inverse de u. Tu trouveras $\bar{u}$.
Multiplie $\bar{u}$ par $\bar{y}$, tu trouveras v.
L'inverse est v. Telle est la façon de procéder.

Ce qui, en pseudo-langage algorithmique, donnerait l'algorithme 1.3.1.

21. Les nombres réguliers, dans les textes babyloniens, sont « les nombres dont l'inverse possède une écriture sexagésimale finie[...]. Ces nombres sont de la forme $2^p 3^q 5^r$, où p, q et r sont des entiers relatifs » (Chabert, 2010, p. 16).

Algorithme 1.3.1 : VAT6505 inverse d'un nombre régulier

Données : x, y, z avec $x = y + z$, et y un nombre régulier dont on connaît $\bar{y}$
Résultat : $\bar{x}$ (tel que $x \times \bar{x} = 60^n$)

début
  $\bar{y} \leftarrow$ `inverse(`y`)`;
  `t` $\leftarrow \bar{y} \times z$;
  `u` $\leftarrow$ `t` $+ 1$;
  $\bar{\texttt{u}} \leftarrow$ `inverse(u)`;
  `v` $\leftarrow \bar{\texttt{u}} \times \bar{y}$;
fin
retourner `v`

1.3.2 Procédures et programmes

L'expression « telle est la façon de procéder » en fin de procédure, ou « Procède comme suit » (« You, by your proceeding ») ou encore « Thus the procédure », juste après l'exposition du problème et des données de l'instance en cours de résolution, se retrouve dans diverses tablettes [22]. Selon Chaigneau (2019, p. 119), citant C. Proust, la tablette VAT 6505 fait partie des « textes à procédures » (*kibsu*), « se terminant par une notice (un colophon) qui précise le nombre de "kibsu", c'est-à-dire de "procédures" contenues dans la tablette ». La traduction utilisée par Chabert (2010) parlait de « 12 exemples », il semblerait qu'il faille plutôt traduire par « 12 procédures ». Cela renforce le caractère algorithmique de ces méthodes : il ne s'agit pas de « faire », mais plutôt de « faire faire »(Samurçay et Rouchier, 1985). Il s'agit non pas d'exécuter une procédure et de donner un résultat, mais de permettre à un « dispositif d'exécution » d'appliquer cette procédure pour une instance particulière du problème.

Procédé ou procédure ?

Il ne faut pas selon nous entendre ici le terme « procedure » dans le sens informatique du terme (méthode générique, ou de validité plus générale que celle de l'instance donnée, que l'on peut réutiliser dans un autre algorithme [23]). D'après Høyrup (2010, p. 37) :

> The terms "proceeding" (epēšum) and "procedure" (nēpešum) appear not to be used about any procedure of more general validity than the single paradigmatic example [...] Hence none of them corresponds to a notion of an algorithm or rule of general validity (as, say, the "rule of three").

Ainsi, il est peu probable que les babyloniens utilisaient un algorithme comme procédure intégrée dans un autre algorithme. En gardant l'exemple des inverses, la formulation « algorithmique » nous amène à utiliser une fonction (c'est-à-dire une procédure renvoyant un résultat) `inverse()`. Mais il s'agit ici d'utiliser une table répertoriant les inverses connus, et non de ré-appliquer l'algorithme de calcul de l'inverse, ni d'appliquer un autre algorithme de calcul de l'inverse [24]. Autrement dit, d'un point de vue informatique, nous pourrions réécrire l'algorithme correspondant à la tablette VAT 6505 comme la fonction récursive 1.3.2 (p. 31), où `choisirYZ(`x`)`

22. Par exemple, dans la tablette TMS IX(Høyrup, 2010, p. 16) ou AO 8862(ibid., p. 22).
23. Ce qui est aussi considéré comme un « sous-programme ».
24. Même si « chercher une information dans une table de données » est un problème algorithmique classique, dans le cas babylonien, il n'y a pas la formulation d'un algorithme de recherche, seulement une action de recherche (et le résultat de cette action). On pourrait considérer qu'il y a une abstraction de l'algorithme de recherche.

est une fonction décomposant x en $y + z$, y étant un nombre régulier dont l'inverse est connu (et donc renvoyé par `inverse(`y`)`). Chabert (2010, p. 20) fait l'hypothèse que cette définition récursive a été utilisée dans la tablette AO 6456 afin de dresser « une longue liste de nombres inverses qui lui permet d'obtenir l'inverse de $x = 1; 5; 32; 9; 36$ ». En effet, cette « tablette d'époque séleucide, provenant d'Uruk et exposée au Musée du Louvre (AO 6456), contient presque tous les nombres inversibles en base 60 commençant par 1 ou 2 et comportant jusqu'à 6 positions sexagésimales » (Proust, 2006). Il est cependant probable que ces tables soient construites non pas de façon récursive, mais séquentielle : d'abord $\overline{16}$, puis $\overline{4; 16}$ etc. Cazalas (1932) critique quant à lui l'hypothèse d'utilisation d'un procédé *méthode-crible*, soulignant que :

> On est en droit de se demander si les Sumériens étaient bien en état de comprendre cette simplification indéniable, mais qui nécessite une évolution mathématique assez développée, et, en définitive, il paraît bien difficile de préciser sans hésitation le mode de construction de leurs tables, qui est peut-être le fruit de pénibles tâtonnements. (ibid., p. 188)

Une illustration de cette conception non-procédurale du procédé est en outre donnée par Høyrup (2010, p. 25), qui relève que dans la tablette AO 8862, on se retrouve dans une situation identique à celle de la tablette TMS IX (problème 1), mais cette procédure est ré-expliquée, elle n'est pas référencée : il n'y a pas d'abstraction de la procédure de TMS IX.

Fonction 1.3.2 : inverseReq(x) Algorithme VAT 6505 en version récursive

Données : x
Résultat : $\bar{x}$ (ie $x \times \bar{x} = 1$)

début
  si *l'inverse de x est connu* **alors**
    v = `inverse(`x`)`
  sinon
    (y, z) ← `choisirYZ` (x);
    $\bar{y}$ ← `inverse(y)`;
    t ← $\bar{y}$ ×z ;
    u ← t + 1;
    ū ← `inverseReq(u)`;
    v ← ū × $\bar{y}$;
  fin
  retourner v
fin

Situation de programmation et dispositif d'exécution

On retrouve dans ces tablettes la manifestation de ce que Samurçay et Rouchier (1985, p. 242) définissent comme « une situation de programmation » :

> À la différence d'une situation habituelle de résolution de problème, la situation de programmation fait intervenir un élément essentiel qui est le dispositif d'exécution, défini par un ensemble d'opérations permises et les conditions de validité qui leur sont associées. La solution d'un problème doit alors être analysée à un double niveau :
>
> — les résultats qu'on veut obtenir par l'application du programme à des données ;

— le programme qui est l'expression dans un langage de programmation d'une procédure définie par référence au dispositif d'exécution.

Dans le cas des tablettes babyloniennes, il est visible que nous ne sommes pas dans une situation de « production de résultat »(Hoc, 1981, p. 264) pour laquelle « l'objectif d'élaboration de procédure s'identifie de fait à l'exécution »(Samurçay et Rouchier, 1985, p. 242). Ici, le « dispositif d'exécution » envisagé est l'étudiant scribe « calculateur » qui sera censé appliquer la procédure à de nouvelles instances de problèmes. Høyrup (2010) note que les problèmes traités dans certaines tablettes sont « artificiels » et que « Pas un seul d'entre eux ne correspond à une tâche qu'un scribe aurait pu rencontrer dans sa pratique professionnelle - à moins que son travail n'ait consisté à enseigner les mathématiques ! » [25].

L'explicitation de ces procédures est exprimée dans un *langage* spécifique, interprétable par le « dispositif d'exécution » : on retrouve ici la définition d'un « programme » selon Samurçay et Rouchier (1985, p. 242) : « Le *programme* [...] est l'expression dans un langage de programmation d'une procédure définie par référence au dispositif d'exécution ». Le « langage de programmation » étant quant à lui constitué d'un vocabulaire et d'une grammaire spécifique permettant de définir « l'ensemble d'opérations permises [par le dispositif d'exécution] et les conditions de validité qui lui sont associées » (ibid., p. 242). Il faut cependant noter que ce langage n'est pas dénué d'ambiguïté : « It was formulated in words, in a very standardized but not always unambiguous language » (Høyrup, 2010, p. 34). Høyrup (note 39 ibid., p. 34) souligne par ailleurs que l'ambiguïté n'est pas toujours levée par le contexte, mais plutôt par la séquence de problèmes proposée. Il signale en outre dans cette même note que l'usage d'abréviations logographiques n'implique pas que les babyloniens s'approchent du symbolisme, c'est un langage qu'il qualifie de « sténographique ».

On peut donc considérer, que les créateurs des procédures contenues dans ces tablettes étaient en « situation de programmation » : la situation ne se limite pas à la production de résultats, les procédures sont explicitées afin d'être exécutées de façon différée, et cette explicitation nécessite « le passage d'une planification des actions exécutables par le sujet lui-même à celle d'un plan d'actions dont l'exécution n'est pas contrôlée par [le sujet ... qui] n'aura pas l'occasion d'intervenir en cours d'exécution pour modifier le déroulement de la procédure. », en mobilisant de surcroit un langage spécifique, adapté au dispositif qui exécutera les procédures (Samurçay et Rouchier, 1985, p. 242).

1.3.3 Concepts informatiques présents

Un « programme » est une rencontre des concepts d'« algorithme » et de « langage ». Selon Dowek (2011, p. 21), « quatre concepts – algorithme, machine, langage et information – semblent suffisants pour couvrir l'ensemble de ce que nous appelons "informatique" ».

Le concept d'algorithme a déjà été présenté succinctement (voir définition 2, p. 23). Dowek (ibid.) précise la différence entre algorithme et programme : l'algorithme est l'idée du programme, sa structure logique mais non verbalisée. Il donne ainsi l'exemple de la recette de cuisine qui peut se transmettre sans être verbalisée (c'est un algorithme, une suite d'actions effectives et non ambigües), et qui devient un programme dès lors qu'elle est verbalisée.

Concernant le concept de « machine », Dowek (ibid., p. 23) précise :

25. It is noteworthy, but says more about the kind of real-world problems encountered by Babylonian calculators than about their mathematical technique that all problems of the second or higher degrees which we find in the texts are artificial. Not a single one of them corresponds to a task a scribe might encounter in his working practice – unless his work was to teach mathematics ! (note 38 Høyrup, 2010, p. 34)

> Pendant 4500 ans, nous avons conçu des algorithmes, pour les exécuter « à la main », mais depuis quelques décennies, nous utilisons pour cela des outils. Une telle utilisation d'outils semble naturelle, puisqu'un algorithme est fait pour être exécuté « sans réfléchir » et c'est ainsi que, depuis l'Antiquité, nous avons utilisé des abaques – baguettes à calculer, bouliers, échiquiers, ... – et des machines mécaniques, avant que nous ayons les connaissances techniques nécessaires pour construire les outils les plus courants aujourd'hui : les ordinateurs.

Nous pouvons étendre cette vision de la machine vue comme un artefact à l'ensemble des « dispositifs d'exécution » de Samurçay et Rouchier (1985). Ainsi, une machine, en informatique, n'est pas nécessairement un *ordinateur*. Rappelons cette phrase attribuée, semble-t-il à tort, à Dijkstra : « L'informatique n'est pas plus la science des ordinateurs que l'astronomie n'est celle des télescopes. »

Un « langage » permet, en informatique, de mettre en mot un algorithme, afin qu'il soit interprété et exécuté par une machine [26]. Il existe de nombreux langages informatiques, qui ont généralement en commun de chercher à être non ambigus [27]. Le langage dépend du dispositif d'exécution, et parfois le langage impose une mise-en-œuvre particulière de l'algorithme. Dowek (2011, p. 26) fait néanmoins remarquer :

> [...] les premiers langages de programmation universels – le lambda-calcul, le langage des machines de Turing sont antérieurs de quelques années à la construction des premières machines universelles, les premiers ordinateurs. En effet, écrire les algorithmes dans un langage de programmation est aussi un moyen de l'exprimer clairement et de le communiquer à d'autres. C'est aussi un moyen de lui donner une forme symbolique et de permettre à d'autres algorithmes, qui opèrent sur des données symboliques, d'opérer sur lui.

Le langage permet de représenter un algorithme, de le communiquer, de le faire exécuter par un dispositif d'exécution (une machine) et d'opérer sur cet algorithme.

Enfin, une des caractéristiques de l'informatique est de manipuler des informations, ou plus précisément des données représentant des informations, non forcément numériques.

Dans la suite de la succincte approche historique que nous présentons, nous préciserons lesquels de ces quatre concepts (algorithme, machine, langage, information) semblent présents.

Programme et langage

Comme nous l'avons déjà évoqué, on retrouve dans les tablettes babyloniennes un langage « standardisé » selon Høyrup (2010). Ce dernier a précisé que ses traductions suivaient le principe de « traduction conforme » [28], « afin de restituer autant que possible les structures conceptuelles, les distinctions et les connotations d'origine, et de clarifier ce qui est dit dans les textes et ce qui ne l'est pas » [29] (Høyrup, 2001, p. 9), ce qui est essentiel si l'on veut travailler cette question du langage utilisé par les babyloniens pour décrire les procédures. Høyrup (2019, p. 58) précise ainsi que, concernant les mots utilisés (qui ne sont pas les seuls vecteurs de transmission du savoir), les babyloniens utilisent : :

— des *noms pour les outils* : la main, les tables par exemple ;

26. Nous ne rentrerons pas ici dans les différences entre langage interprété et compilé ; ce qui nous intéresse étant le rapport qu'entretient le programmeur (l'élève) avec le langage, et nous considérons aussi les machines qui ne sont pas des ordinateurs.

27. Y compris lorsqu'ils cherchent à se rapprocher du langage naturel, comme le langage Cobol.

28. « Conformal translation ».

29. In order to render as much as possible of the original conceptual structures, distinctions and connotations, and to make clear what is told in the texts and what not[...]

— des *noms pour les méthodes* ;
— des *phrases et termes utilisés pour structurer* le texte mathématique, pour poser le problème, pour préciser les étapes successives ou pour présenter la solution ;
— des *noms pour les objets mathématiques* ;
— et des *termes pour les opérations mathématiques.*

Parmi les termes permettant de structurer le texte, on retrouve par exemple *.bi en.nam*, qui marque la question (« its... what ? », « quel est son inverse ? » dans la traduction de VAT 6505 dans (Chabert, 2010, p. 18)), et le suffixe possessif *.bi* qui marque la réponse — « its... », « L'inverse est... » dans VAT 6505 — (Høyrup, 2019, p. 63). D'autres termes permettent de la même façon de marquer le début d'une procédure et sa terminaison. Du côté des opérations, des termes différents sont utilisés pour désigner des additions différentes. Ainsi, *wasābum* « consists in joining one magnitude d to another one A. In this process, A conserves its identity but increases in magnitude ; the sum thus has no name of its own. » (ibid., p. 77) : si l'on utilisait des symboles actuels, cela équivaudrait à $A \leftarrow A + d$, opération que l'on qualifierait davantage d'informatique que de mathématique. Mais pour les babyloniens, A n'est pas une variable, et cette addition est une opération concrète, nécessitant que A et d soient du même type (par exemple une longueur).

Même si les babyloniens laissent parfois des actions implicites (par exemple pour trouver un inverse connu), ils mobilisent un langage de description de la procédure, de l'algorithme, avec des balises permettant de structurer le texte : on peut ici parler de programme. L'implicite, comme `inverse` ou `choisirYZ`, est constitué de procédures supposées connues du scribe (dispositif d'exécution) : on pourrait considérer que « forme l'inverse de y, tu trouveras $\bar{y}$ » (c'est-à-dire $\bar{y}$`=inverse(`y`)`) est une instruction de haut niveau, déjà implémentée dans la « machine », c'est un changement de niveau d'abstraction.

1.3.4 Concepts algorithmiques présents [30]

Pour concevoir tout algorithme, seuls quatre concepts sont nécessaires, quatre ingrédients (Dowek et al., 2010) :

— l'instruction (la séquence),
— l'itération (la boucle),
— la condition, l'instruction conditionnelle,
— la variable.

La séquence Les procédés, décrits par les babyloniens dans les tablettes mathématiques, destinés aux étudiants scribes sont des séquences d'instructions précises et (presque) non ambigües.

La variable On ne peut pas parler de variable informatique chez les babyloniens. Cependant, certains éléments s'en rapprochent. Ainsi, dans la VAT 8390 (ligne 9 et 10 ou 12 et 13, figure 1.4, p. 35), le terme traduit par Høyrup en « posit » est utilisé : « the given numbers are "posited", that is, taken note of materially (if only memorized, they would have to be "held in your head") »(Høyrup, 2010, p. 11). Il s'agit de mémoriser une valeur afin de la réutiliser ultérieurement [31], ce qui, d'un point de vue algorithmique, se rapproche d'une *variable temporaire*

30. Par « présents », nous entendons « identifiables *a posteriori* dans ces textes mathématiques ».
31. En général en faisant référence à la nature de la valeur mémorisée : par exemple dans la tablette VAT 8390 (Høyrup, 2017, p. 121) la ligne 13 indique « 3 to the width posit », et cette valeur est utilisée ligne 16 « 1 from 3 which to the width you have posited ».

(« Temporary » pour Sajaniemi (2005, p. 8)). Nous avons déjà signalé qu'une variable informatique avait pour particularité d'être une mémoire concrète : chez les babyloniens cette mémorisation temporaire pouvait se faire en utilisant une « table à poussière », sur laquelle on pouvait noter et effacer des valeurs. L'utilisation de la valeur mémorisée se fait en faisant référence à ce qui a été mémorisé, — à ce que représente cette valeur — et non à un lieu ou espace de stockage. De plus, il n'y a pas de représentation symbolique de cet objet mémorisé.

La boucle Dans la tablette VAT 8390 (figure 1.4, p. 35), on remarque (ligne 5) l'instruction « I have made hold, to 9 I have repeated » : il s'agit ici d'une version spécifique d'une forme de boucle : « *to repeat* or *repeat until n* (*esēpum*/TAB), a concrete doubling (e.g., of a right triangle into a rectangle) or n-doubling (*n* being sufficiently small to be intuitively graspable, $2 < n \leq 9$) »(Høyrup, 2010, p. 7). Nous n'avons pas ici à faire avec une version générique de la boucle, on peut y voir davantage une sorte d'instruction réitérée. Notons qu'il y a ici une part d'implicite que la plupart des langages de programmation demandent de lever. Ainsi, si les Environnements de Programmation Graphique par Blocs (EPGB) tels que Scratch ou Snap autorisent des formes comme 

, on devrait écrire `for i in range(9)` en Python ou `for (let i = 0; i < 9; i++)` en Javascript. Généralement, la variable d'itération (ici i) est visible, mais ce n'est pas le cas dans les EPGB [32], ni dans ce texte babylonien.

VAT 8390 #1[13]

Obv. I

1. [Length and width] I have made hold: 10` the surface.
2. [The length t]o itself I have made hold:
3. [a surface] I have built.
4. [So] much as the length over the width went beyond
5. I have made hold, to 9 I have repeated:
6. as much as that surface which the length by itself
7. was [ma]de hold.
8. The length and the width what?
9. 10` the surface posit,
10. and 9 (to) which he[14] has repeated posit:
11. The equalside of 9 (to) which he has repeated what? 3.
12. 3 to the length posit
13. 3 t[o the w]idth posit.
14. Since "so [much as the length] over the width went beyond
15. I have made hold", he has said
16. 1 from ⌊3 which t]o the width you have posited
17. tea[r out:] 2 you leave.
18. 2 which yo[u have l]eft to the width posit.
19. 3 which to the length you have posited
20. to 2 which ⟨to⟩ the width you have posited raise, 6.
21. Igi 6 detach: 10´.
22. 10´ to 10` the surface raise, 1`40.

Figure 1.4 – Tablette VAT 8390 : mise en équation, boucle et mémorisation (Høyrup, 2010)

32. Dans Snap !, il existe cependant aussi une instruction 

1.3.5 Concepts algébriques ou pseudo-algébriques présents

L'équation On ne trouve bien entendu pas d'équation avec des symboles dans ces tablettes babyloniennes. En revanche, il y a une mise en équation implicite dans la description du problème, qui précède la procédure de résolution. Par exemple, dans la tablette BM 13901 #1 (Høyrup, 2010, p. 12), dont l'énoncé et une représentation du problème se trouvent figure 1.5a (p. 36) : la ligne 1 explique que la somme d'une surface carrée et d'un côté du carré est 45' ($= \frac{3}{4}$). Cela correspondrait, avec une écriture actuelle, à poser l'équation $c^2 + c = \frac{3}{4}$. Pour donner un sens à cette somme (d'une longueur et d'une surface), il est expliqué qu'une « projection » du côté est faite de côté 1 : le côté (c sur le schéma) est projeté sur une longueur de 1, ce qui donne un rectangle de surface $1 \times c$. On voit ici la nécessité pour les babyloniens — et cela restera vrai jusqu'à Descartes — de respecter une certaine homogénéité : il n'y a pas de sens pour eux à ajouter une surface et une longueur, mais ajouter une surface c^2 et une surface $1 \times c$ en a. Ainsi, la représentation du problème est celle de l'équation que nous écririons $c^2 + 1 \cdot c = \frac{3}{4}$. Cette projection qui permet de contourner le problème de l'homogénéité est un procédé proche de celui qu'utilisera Descartes, avec le même but mais rendu explicite et générique (voir 1.6.3, p. 63). Høyrup (ibid., p. 11) définit ainsi ce qu'est une « équation » babylonienne :

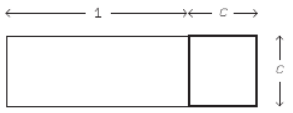
> [...]a statement that (the measure) of a (mostly composite) entity is so and so much, or that (the measure of) one entity is « as much as » (the measure of) another entity. This is no different from the equations of any applied algebra ; the difference, as we shall see, is that the Babylonians did not operate on their equations.

Cette « équation » pose le problème, mais ne sert pas de base sémiotique à la résolution du problème par une transformation et une réécriture des termes de l'équation.

Obv. I

1. The surfa[ce] and my confrontation I have accu[mulated]: 45′ is it. 1, the projection,
2. you posit. The moiety of 1 you break, [3]0′ and 30′ you make hold.
3. 15′ to 45′ you append: ⌜by⌝ 1, 1 is equal. 30′ which you have made hold
4. in the inside of 1 you tear out: 30′ the confrontation.

(a) Formalisation du problème



(b) Représentation du problème

Figure 1.5 – Tablette BM 13901, problème 1 (sur 24) (Høyrup, 2010)

L'inconnue et sa dénomination Ces équations ne mobilisent pas de représentation symbolique de l'inconnue. Cependant, il y a une dénomination de la nature de l'objet recherché. Dans la tablette YBC 6967 (ibid., p. 20), deux valeurs inconnues sont cherchées et dénommées *igûm* et *ibigûm* qui sont un nombre et son inverse (« Akkadianized versions of IGI and IGI.BI, "the IGI" and "its IGI". ») : « *igûm* and *ibigûm* what ? ». Ces valeurs ne pas géométriques, mais leur produit est représenté par une surface. Dans un problème impliquant des valeurs géométriques, les quantités inconnues seront dénommées par exemple *longueur* et *largeur* (« the length and the width what ? »). Les valeurs recherchées, propriétés d'objets différents, sont nommées, et

nommées différemment selon leur sens (dans l'acceptation de Frege). On retrouve un mécanisme sémiotique similaire dans le « Hau-Calcul » égyptiens : la dénomination de l'inconnue était ce que l'on cherchait (un tas, « Hau »). Cependant, pour le Hau-Calcul, il existait aussi un symbole pour désigner le tas (Serfati, 1997, p. 39).

Ces dénominations babyloniennes des inconnues ne sont pas symboliques, et elles ne sont pas manipulées comme si leur valeur était connue.

La question de la preuve Dans ces textes babyloniens, il n'y a pas de preuve, ni du procédé, ni du raisonnement. Il y a seulement une vérification des résultats obtenus : c'est une preuve sur une instance. Il peut cependant y avoir des traces des interprétations géométriques des équations (Høyrup, 2010, p. 18). On trouve aussi des explications didactiques :

> In line 8 we observe once more a quotation from the statement used as argument for at particular step. This is certainly no deductive proof of anything. It is a nice didactical explanations of concepts relevant to the understanding of the equations and of what goes on in their transformation. Already Neugebauer argued that many Babylonian problems were so complex that solutions could never have been found without good understanding, and supposed that such explanations were imparted orally. The present text is from Susa, a peripheral area (in a valley inthe Zagros), which may explain that an oral tradition of this kind had to be put into writing. However, once we recognize the style, we can find traces elsewhere in the corpus of similar though more rudimentary expositions, combined with problem solutions. (ibid., p. 16)

Cela semble confirmer que les babyloniens passait bien du « faire » au « faire-faire », mais sans doute avec un support d'explicitation oral : dans ce cas, il n'a pas de nécessité de représentation du paramètre.

1.3.6 Bilan

Une citation de Radford (1991) va nous permettre de résumer cette première recherche d'algèbre, d'informatique et d'algorithmique et des liens que ces disciplines entretiennent. Radford évoque le problème suivant :

« *Trouver les dimensions d'un rectangle d'aire 96 et dont la somme de la base et de la hauteur est 20* »

Ce qui s'écrirait de nos jours :

$$\begin{cases} x + y &= 20 \\ xy &= 96 \end{cases}$$

La résolution rapportée par Radford est celle-ci :

> La résolution babylonienne consiste à opérer sur les chiffres donnés, d'après une suite d'instructions données par le scribe :
>
> 1. Diviser par 2 la somme des nombres : $20 \div 2 = 10$.
> 2. Élever au carré : $10^2 = 100$.
> 3. Retrancher l'aire donnée, 96, à 100 : $100 - 96 = 4$.
> 4. Extraire la racine carrée : 2.
> 5. La base est $10 + 2 = 12$.
>   La hauteur est $10 - 2 = 8$.

> Cet exemple nous permet de voir que la résolution babylonienne, telle qu'elle nous est parvenue, repose sur un enchainement d'opérations numériques. Il n'y a pas une inconnue mêlée aux calculs. (Radford, 1991, p. 2)

Ainsi, on voit que :

— le problème est un problème que l'on pourrait généraliser ($x + y = a \wedge xy = b$), mais qui est formulé comme une instance. Les nombres (20 ; 96) sont les éléments potentiellement variables du problème. Ce sont des données déterminées, des paramètres instanciés.
— la résolution montre une séquence organisée d'instructions effectives et non ambigües, qui résout une instance du problème : c'est un algorithme.
— cet algorithme est mis en mot, avec un langage spécifique : c'est un programme.
— il est difficile de suivre ce que l'on fait des inconnues : à chaque instruction on perd de l'information. Il n'y a ni inconnue, ni paramètres.
— il n'y a pas de symbolisation.
— la valeur inconnue est montrée et dénommée par sa nature :
  — il y a une marque dans le texte indiquant ce que l'on cherche, et son résultat (« its ... what ? » et « its » collée à la réponse).
  — il y a des expressions différentes pour des objets différents : ainsi les longueurs et largeurs d'un rectangle support de la procédure de résolution portent différents noms suivant qu'elles soient des mesures réelles d'un objet géométrique (longueur d'un mur, côté d'un champ...) ou la représentation abstraite d'une valeur (Høyrup, 2019, p. 14).

1.4 Diophante et les premières formalisations

Il n'y a pas trace de représentation symbolique chez les Babyloniens, mais la dénomination existait déjà, notamment liée à la culture et la nature de ce qui était recherché :

> D'un autre côté, ce que l'on recherchait fut, bien souvent aussi, ce que l'on désirait. Et donc, en des temps naïfs, on le désigna comme le bien, l'héritage, la possession (les Arabes), la couleur (les Hindous), dénominations qui, au bout d'un temps, parurent trop spécifiques, au regard de la profonde analogie entre les procédés de recherche. En Europe, au Moyen-Age et à la Renaissance, se substitua ce mot-valise, porteur d'images ambiguës et multiples, qui rapidement trouva sa place en latin, puis dans les langues communes :*res*, la Chose, *coss* en allemand, *cosa* en italien, puis, à la suite de Rudolff, l'écriture *cossique*, qu'utilisa entre autres Clavius. Depuis Descartes, on dit l'inconnue. (Serfati, 1997, p. 42)

Il y a donc, à l'époque des babyloniens, dénomination mais pas représentation de l'inconnue, ce qui pour Serfati est nécessaire afin de pouvoir être manipulée pour résoudre (« nécessité de représentation symbolique ») :

> nous avons brièvement conclu que, dans tout problème contenant une inconnue, il était indispensable de représenter celle-ci symboliquement, afin de la manipuler pour résoudre, et qu'il n'aurait certes pas suffi de la dénommer. Une conclusion qui nous parait aujourd'hui s'imposer avec tant de force qu'on serait tente de parler de nécessité de la représentation symbolique. Ce serait oublier que semblable nécessité ne fut pas historiquement perçue avant Diophante. (ibid., p. 42)

Cela changera avec Diophante, qui dans les *Arithmétiques*, introduira notamment en représentation symbolique et manipulable de l'inconnue, qu'il nomme *arithme*. Un exemple de problème traité par Diophante est donné figure 1.6 (p. 39), il nous servira d'illustration au fil de cette partie.

Proposition I.1 : « Partager un nombre proposé en deux nombres dont la différence est donnée ».
écriture actuelle : trouver deux nombres entiers x et y tels que $x + y = a$ et $x - y = b$.
Texte de la solution :

> Que le nombre donné soit 100 et que la différence soit 40 unités ; trouver les nombres.
> Posons que le plus petit nombre est un arithme ; donc, le plus grand nombre est 1 arithme plus 40 unités. En conséquence, la somme des deux nombres devient deux arithmes plus 40 unités. Or, les 100 unités données sont cette somme ; donc 100 unités sont égales à deux arithmes plus 40 unités. Retranchons les semblables des semblables, c'est-à-dire 40 unités de 100 et, de même, 40 unités de 2 arithmes plus 40 unités. Les deux arithmes restants valent 60 unités, et chaque arithme devient 30 unités. Revenons à ce que nous avons proposé : le plus petit nombre sera 30 unités ; tandis que le plus grand sera 70 unités, et la preuve est évidente.

Avec une écriture actuelle (ς désigne l'arithme)

$$x + \varsigma = 100 \wedge x - \varsigma = 40 \tag{1.1}$$
$$x = \varsigma + 40 \tag{1.2}$$
$$x + \varsigma = 2\varsigma + 40 \tag{1.3}$$
$$2\varsigma + 40 = 100 \tag{1.4}$$
$$2\varsigma = 100 - 40 \tag{1.5}$$
$$2\varsigma = 60 \tag{1.6}$$
$$\varsigma = 30 \tag{1.7}$$
$$x = 70 \wedge y = 30 \tag{1.8}$$

Figure 1.6 – Diophante, Proposition I.1, d'après (Farès, 2017b, p. 17)

1.4.1 Brève présentation des *Arithmétiques*

Selon Rashed et Houzel (2013, p. 3), « de la vie de Diophante, nous ignorons presque tout. » Il aurait vécu entre le II^{e} avant notre ère et le IV^{e} siècle de notre ère, mais Farès (2017b, p. 9) situe l'écriture des *Arithmétiques* entre le III^{e} et le IV^{e} de notre ère. Dès son premier livre, Diophante annonce que l'ouvrage devait comprendre treize livres. En cumulant les versions arabes et grecques, dix livres étaient disponibles en 2017 (ibid., p. 10), représentant 290 problèmes. Certains de ces problèmes avaient déjà été traités par les babyloniens (Rashed et Houzel, 2013, p. 10), avec une différence notable : les problèmes sont formulés de façon générique (figure 1.6, p. 39). Les problèmes traités sont traduisibles en « équations algébriques de degré inférieur à 6 (ou à 9 si l'on tient compte des livres arabes) » (ibid., p. 35), mais selon Farès (2017b, p. 14), « le sujet des Arithmétiques n'est pas la résolution d'équations polynomiales », mais plutôt le « nombre ». Même si certains désaccords subsistent pour dire que les *Arithmétiques* est un livre d'algèbre ou pas, il est indéniable que les problèmes sont des problèmes que l'on considère actuellement comme algébriques, et Diophante manipule des concepts et des outils qui sont utilisés dans l'algèbre (substitution,

élimination). Il ne semble pas y avoir de règles fixes pour établir des solutions (contrairement à ce que nous verrons chez al-Khwārizmī), mais plutôt des « méthodes ingénieuses qui utilisent des moyens appelés de nos jours -après l'introduction de l'algèbre au IX^{e} siècle- algébriques » (Farès, 2017b, p. 14). Christianidis (2018, p. 5) parle de son côté d'« algèbre pré-moderne ».

Il semble établi que le livre de Diophante a une « intention didactique » (Rashed et Houzel, 2013, p. 22) [33], comme l'avaient sans doute aussi les tablettes mathématiques babyloniennes évoquées précédemment. « Les mathématiques de type algorithmique semblent quasi universelles, du moins si l'on se limite aux cultures qui ont développé des mathématiques. On les rencontre en Égypte et au Proche-Orient ancien, en Inde, en Chine et aussi en Grèce ancienne. Dans les premières civilisations nommées, nous les connaissons essentiellement grâce à des documents d'origine scolaire. » (Vitrac, 2005, p. 7)

Diophante s'adresse à un personnage fictif nommé Dionysius, à qui il va « exposer la nature et la puissance des nombres » (Farès, 2017b, p. 10). Il définit ainsi le nombre comme étant formé d'une certaine quantité d'unités, et précise six types de nombres (« espèces ») : outre le nombre, les carrés, cubes, bicarrés, carré-cubes, et cubo-cubes. Mais l'objectif de cet ouvrage ne parait pas être d'établir des procédures de résolution ou des solutions, mais selon Christianidis (2007, p. 298), d'enseigner une méthode générale pour résoudre des problèmes arithmétiques, la « voie » (ὁδός).

1.4.2 Le langage de Diophante

Une fois avoir présenté les nombres et les espèces, Diophante va introduire leur « désignation abrégée » (figure 1.7a, p. 41). Ces symboles sont des abréviations, et non des écritures symboliques algébriques [34]. Ces abréviations seront composées avec les écritures de nombres (accumulation par défaut, signe ⍋ pour les éléments ôtés), complétés par le symbole de l'unité « $\overset{o}{\mathrm{M}}$ ». Il va ensuite présenter un mot désignant l'inconnue « arithme », « qui possède en soi une quantité indéterminée d'unités », désignée par la lettre ς (figure 1.7b, p. 41).

1.4.3 Le langage et la voie

Selon Christianidis (ibid., p. 298-299), la « voie » de Diophante est constituée des étapes suivantes :

1. « invention » : les nombres recherchés du problème doivent être traduits, formulés, par leur abréviation dans le langage de la « théorie arithmétique ». On passe ainsi du *langage du problème* au *langage de la théorie*. La formation de l'équation dans le langage de la théorie clôt cette première étape.
2. « disposition » : transformation de l'équation pour aboutir à la forme finale, en appliquant des méthodes de substitution ou d'élimination (que al-Khwārizmī nommera *al-jabr* — ce qui donnera le mot « algèbre » — et *al-muqābala*). La transformation vise à aboutir à une forme permettant de facilement déterminer le nombre inconnu.
3. Calculer les valeurs des nombres cherchés à partir de l'arithme (établissement de la solution).
4. Vérification de la validité des solutions.

33. Voir aussi (Christianidis, 2018) pour une présentation plus développée de cet aspect.

34. « Les historiens des sciences s'accordent sur le fait qu'il ne s'agit pas là de symbolisation algébrique ni même d'un début d'une telle symbolisation qui, elle, a commencé au XVIe siècle » (Farès, 2017b, p. 15), mais « The majority of historians, however, agree with the view that the work of Diophantus is a work with algebraic characteristics, these characteristics being usually found in its use of symbolism, its operating with εἶδη (“species”), and its algorithmic features » (Christianidis, 2007, p. 302).

Diophante introduit des *symboles* pour désigner les nombres précédents : ce sont des "désignations abrégées", dit-il.

carré : Δ^Y
cube : K^Y
carré-carré (ou bicarré) : $\Delta^Y\Delta$
carré-cube : ΔK^Y
cubo-cube : K^YK.

(a) Les puissances de l'inconnue

A l'aide de ces symboles il va construire un langage (avec une syntaxe bien définie) qui rend possible de représenter et les espèces et les expressions (ces dernières étant – rappelons-le – d'espèces (i. e. des monômes) opérées entre elles).

Les nombres invariants déterminés sont représentés en utilisant la notation alphabétique grecque [9] ; par exemple :

α désigne **1**, β désigne **2**, γ désigne **3**, ..., θ désigne **9**,
ι désigne **10**, κ désigne **20**, λ désigne **30**, ..., q désigne **90**.

Le nombre **12** est désigné par $\iota\beta$, le nombre **23** est désigné par $\kappa\gamma$, et ainsi de suite. Le tableau suivant résume la correspondance pour les premiers entiers :

1 - 9 : $\alpha\ \beta\ \gamma\ \delta\ \varepsilon\ \varsigma\ \zeta\ \eta\ \theta$
10 - 90 : $\iota\ \ \kappa\ \lambda\ \mu\ \nu\ \xi\ o\ \pi\ q$.

Dans le langage algébrique de Diophante, notre monôme $3x^2$ correspond à l'espèce $\Delta^\gamma\gamma$. De même, $4x^5$ correspond à $\Delta K^\gamma\delta$, alors que $33x$ correspond à $\varsigma\lambda\gamma$.

Pour écrire $33x+8$, par exemple, Diophante place à droite le 8, qui est un nombre invariant déterminé (alors que 33 n'en est pas un : il indique qu'on a 33 arithmes), séparés par la lettre M du reste de l'écriture : $\varsigma\lambda\gamma M\eta$. De même, $3x^2+12$ s'écrit

$$\Delta^\gamma\gamma M\iota\beta.$$

Les *expressions* se forment par juxtaposition. Ainsi, l'expression $\Delta K^\gamma\delta K^\gamma\varepsilon\Delta^\gamma\kappa\varsigma\lambda\gamma M\eta$ correspond à $4x^5+5x^3+20x^2+33x+8$.

La soustraction est dénotée par $\dot\Lambda$. Les termes négatifs dans une expression sont placés à la fin. Ainsi, $2x^3-x^2-2x+8$ s'écrit :

$$K^\gamma\beta M\eta\dot\Lambda\Delta^\gamma\alpha\varsigma\beta.$$

(b) Écriture symbolique

Figure 1.7 – Langage chez Diophante, d'après (Radford, 1991)

La plupart du temps, Diophante ne traite que très brièvement ces deux derniers aspects.

Dans le problème rapporté figure 1.6 (p. 39), l'invention, après avoir défini l'arithme en fonction des nombres cherchés, se termine sur « la somme des deux nombres devient deux arithmes plus 40 unités » (1.3). « Les deux arithmes valent 60 unités » est l'aboutissement de la disposition (1.6). Le calcul des nombres cherchés est présenté comme une évidence, de même que la preuve.

Christianidis (2018, p. 8) considère que la « voie » est similaire à la démarche utilisée pour résoudre un problème par l'algèbre pré-moderne (donc jusqu'à Viète et Descartes) :

1. Nommer les inconnues du problème.
2. Exécuter les opérations décrites dans l'énoncé, en utilisant les termes nommés.
3. Établir une équation, formulée dans le langage des noms, comme résultat des deux procédures précédentes.

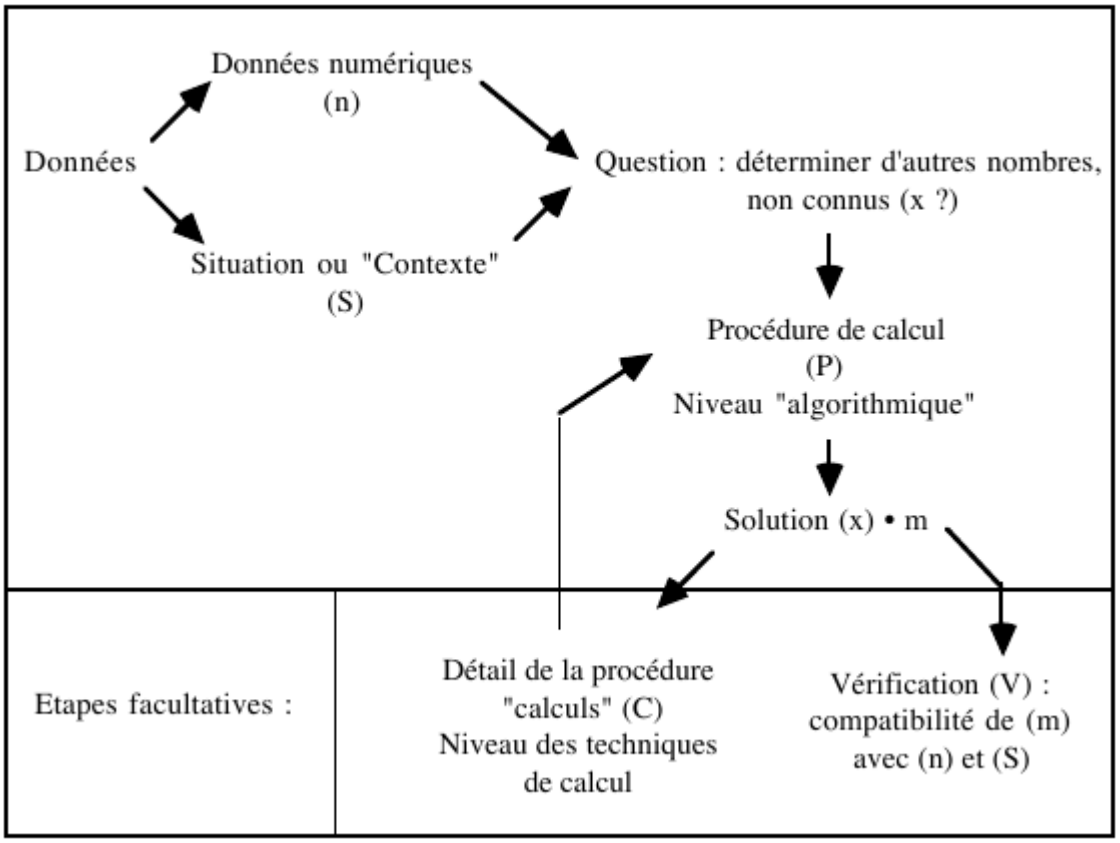
4. Simplifier et résoudre l'équation en appliquant chaque fois l'algorithme approprié.
5. Répondre au problème, en se servant de la solution de l'équation.

On peut aussi rapprocher cette organisation, ce « meta-algorithme » en quelque sorte, du programme de Viète (Christianidis, 2007, p. 303) :

A Dissocier l'algèbre des nombres ; la rendre générale.
B Élaborer un canon pour la résolution de problèmes :

B1 Traduire le problème dans le langage algébrique.
B2 Utiliser l'algèbre.
B3 Traduire le résultat algébrique (équation, solution) et donner la solution.
B4 Prouver que la solution est exacte.

Vitrac (2005, p. 7) quant à lui, affirme que « les mathématiques de type algorithmique semblent *quasi* universelles » pour les civilisations anciennes qui ont développé des mathématiques, les différenciant des mathématiques démonstratives. Il précise que pour ces cultures, la résolution du problème calculatoire suit une même organisation, reproduite figure 1.8 (p. 43).



(**i**) Des données numériques connues (**n**) sont insérées dans
(**ii**) Une situation (**S**) plus ou moins concrète, plus ou moins artificielle, voire abstraite.
(**iii**) Une question, explicite ou non, propose de déterminer une ou plusieurs autres quantités présentées comme non connues.
(**iv**) La procédure de calcul (**P**) à suivre pour la (les) déterminer est exposée sous forme d'étapes.
(**v**) La ou les réponses sont données.
(**vi**) Dans certains cas, on trouve, en outre, le détail des calculs (**C**) réalisés au cours de la mise en œuvre de (**P**) : mise en évidence d'opérations fondamentales (duplication, dimidiation; extraction de racines carrées; inversion d'un nombre) … Cette analyse peut conduire à la constitution de tables numériques lesquelles, à terme, dispenseront de l'exposé du détail des calculs ou à la mise au point d'instrument type "abaque" qui a le même genre d'effet. Elle met aussi en évidence des problèmes-types que tout calculateur doit d'abord maîtriser. C'est un lieu d'exercice privilégié pour l'artificialité pédagogique.
(**vii**) Parfois une justification de ce que les résultats sont exacts, en fait la vérification que les quantités déterminées dans (**v**) satisfont les conditions imposées dans (**i-ii**). C'est notamment le cas dans les papyri mathématiques égyptiens.

Figure 1.8 – Forme des problèmes, Vitrac (2005, p. 8)

1.4.4 L'inconnue

L'introduction par Diophante d'une représentation de l'inconnue, l'arithme, « modifia considérablement les choses » (Serfati, 1997, p. 43). Pour Radford (1991, p. 3), cette représentation d'« une quantité indéterminée d'unités » fait que Diophante considère « deux classes d'objets : les *nombres*, en tant que nombres invariants déterminés [...] et l'*arithme* ». L'arithme est un nombre, mais c'est un nombre différent, puisque indéterminé. Serfati (1997) considère d'ailleurs que le symbole choisi [35] montre que ce symbole lui-même est à part, loin des symboles des autres nombres (eux aussi représenté par des lettres ou des combinaisons de lettres). Il souligne aussi que ce côté *différent* sera aussi rendu visible plus tard par les cossistes, avec un symbole très spécifique, mais aussi par Viète et Descartes qui opposeront respectivement les voyelles (pour les inconnues) et les consonnes, ou les lettres finales (x, y, z) et les lettres du début de l'alphabet. En fait, ce symbole répond à une obligation : « refuser toute confusion avec la représentation du donné, afin de spécifier l'inconnu, une catégorie dont on ne connaissait en vérité que l'absence de détermination. » (ibid., p. 43). Cette différenciation est aussi la marque d'un problème, d'une question, ce que les babyloniens précisaient avec les formules « its ... what ? ».

La particularité de l'arithme, contrairement aux dénominations des quantités recherchées par les babyloniens, est qu'il est manipulable comme un nombre, comme s'il était connu : « on va opérer avec elle » (Radford, 1991, p. 5).

Précisons enfin que chez Diophante, les nombres recherchés ne pouvaient être que des nombres rationnels positifs.

1.4.5 Concepts algorithmiques présents

On le voit avec Diophante comme on l'a vu avec les babyloniens : la démarche de résolution de problèmes que nous qualifions de nos jours d'algébrique, est une démarche que nous qualifions de nos jours d'algorithmique.

Algorithmes et dispositif d'exécution Dans le cas de Diophante comme des babyloniens, le dispositif d'exécution est un étudiant, ici représenté par le personnage de Dionysius. L'aspect algorithmique est néanmoins un peu plus générique que chez les babyloniens : les énoncés sont cette fois génériques, mais ils restent traités sur des instanciations. Cet algorithme commence à être « un mode de présentation codifiée pour un certain nombre de procédures de calculs » (Vitrac, 2005, p. 7) : des éléments de langage sont créés pour l'occasion et clairement définis : « l'introduction de la syntaxe permet de créer un langage avec lequel on peut représenter (d'une façon fort efficace d'ailleurs) les opérations entre l'inconnue, les puissances de celle-ci et les nombres invariants déterminés. » (Radford, 1991, p. 6)

Un début de généralisation malgré tout ? L'absence de généricité des solutions n'implique pas l'absence de volonté de généricité : « Diophante ne se pose pas le problème d'expliciter toutes les solutions : il cherche à montrer comment sa méthode [...] fonctionne et peut produire autant de solutions qu'on voudra : elle apparait ainsi comme un programme. » (ibid., p. 10). Cette volonté est aussi visible dans quelques problèmes qui sont laissés « dans l'indétermination », notamment les problèmes GIV 19 et 20 (Vitrac, 2005, p. 17-23).

Le problème GIV 19 est celui-ci :

35. Serfati conserve la notation qu'il affirme être le plus fréquente, ζ, mais le signe ς est lui aussi particulier, puisque représentant un σ final. Radford (1991, p. 4) souligne aussi que « pour désigner les nombres invariants déterminés, Diophante utilise le symbole $\overset{o}{\text{M}}$, alors que pour l'arithme il utilise la lettre grecque ς ».

« *Trouver trois nombres dans l'indétermination de telle manière que le produit de deux quelconques d'entre eux, accrue d'une unité, forme un carré* »

Quant au problème GIV 20 il est traduit ainsi :

« *Trouver quatre nombres de telle manière que le produit de deux quelconques d'entre eux, augmenté d'une unité, forme un carré* »

Les solutions proposées pour le GIV 19 sont exprimées « dans l'indétermination », c'est-à-dire sans donner lieu à un calcul final effectif, à une instanciation : Diophante établit que les solutions seront les nombres $\varsigma+2$, ς et $4\varsigma+4$. Pour résoudre le problème GIV 20 on peut alors s'appuyer sur ces résultats pour établir les solutions. L'algorithme traitant GIV 19 serait un sous-programme de celui traitant GIV 20. Cependant, ce n'est pas tout à fait le cas : d'une part, comme nous l'avions vu chez les babyloniens, il n'est pas *fait appel* à la solution générique produite par le traitement du problème GIV 19, mais la méthode permettant de résoudre GIV 19 est *reprise*, et avec des variantes, dans GIV 20. D'ailleurs, dans GIV 20, Diophante choisit comme triplet de nombres solutions de GIV 19 $\{\varsigma, \varsigma+2, 4\varsigma+4\}$ et non $\{\varsigma+2, \varsigma, 4\varsigma+4\}$: ce changement d'ordre étant dû à une « tentative manquée » de GIV 19. Ainsi, GIV 19 n'est pas une procédure ou un sous-programme appelé pour résoudre GIV 20, mais l'algorithme résolvant GIV 19 est réécrit comme une partie du programme résolvant GIV 20. GIV 19 n'est pas un algorithme générique non instancié, une fonction à un paramètre, `résoudreGIV19(`p`)` ; c'est un problème non encore résolu, qui le sera lorsqu'on rajoutera des conditions (GIV20 reprend l'algorithme, pas l'expression de la solution).

Il s'agit néanmoins d'un pas important fait dans le changement de niveau d'abstraction : « lorsqu'il y a N conditions à satisfaire, on résout le problème de manière indéterminée en en satisfaisant N - 1, puis on détermine l'arithme grâce à la dernière condition » (ibid., p. 19).

Pour aboutir à un niveau générique, les algorithmes devraient manipuler des paramètres. Radford (1991, p. 10) relève :

> Il est clair que Diophante distingue parfaitement le statut des paramètres et celui des inconnues. Mais il ne symbolise point les paramètres ceux-ci restent au niveau numérique. En fait, la symbolisation des paramètres est inconcevable chez Diophante. Cette symbolisation est beaucoup plus tardive dans l'histoire de l'algèbre : elle se trouve dans l'œuvre de Viète, et permet d'écrire la solution d'un problème en toute généralité.

Le niveau de généralité augmente, mais il manque encore les outils de pensée pour pouvoir les exprimer.

Langage Comme on l'a vu, Diophante définit un langage spécifique pour représenter et résoudre le problème. Il s'agit d'une part, grâce aux abréviations, d'alléger la description du problème, mais aussi, d'autre part, de représenter les relations entre l'arithme et les nombres déterminés. L'écriture n'est pas totalement symbolique, elle est syncopée. Radford (ibid., p. 6-7) précise que « le langage mis en œuvre par Diophante aide la pensée à se dégager du contexte numérique dans lequel le problème est posé et de se concentrer sur un *calcul formel*, c'est-à-dire un calcul qui tient compte seulement de la forme ($\epsilon\iota\delta o\varsigma$) des expressions ». En passant du langage du problème au langage de la théorie arithmétique, on change de registre de représentation sémiotique (Duval, 2002).

Ainsi par exemple le nombre carré décrit dans le problème « tetragônos » devient « dynamis » dans le langage de la théorie, ou encore, un « tetragônos multiplié par lui-même » dans le langage du problème deviendra un « dynamodynamis » dans le langage de la théorie : on utilise des signes différents pour désigner un même objet, on a donc deux systèmes sémiotiques différents. Duval (ibid., p. 81) précise :

> Un système sémiotique est constitué : a) De règles organisatrices pour combiner ou regrouper des éléments en unités significatives, c'est-à-dire des expressions ou des unités figurales élémentaires ; b) D'éléments prenant une valeur de sens qu'en opposition de choix par rapport à d'autres éléments (par exemple, les chiffres d'une base d'un système de numération), leur utilisation selon les règles organisatrices permettant de désigner des objets.

Dans le langage de la théorie arithmétique de Diophante, un signe *nombre* suivi d'un signe *nombre* correspond à l'ajout des deux quantités représentées : c'est une règle organisatrice. Le système de signes mis en place rend le traitement algorithmisable (Duval, 2002, p. 81), tout au moins potentiellement : le langage reste syncopé, les signes sont des abréviations.

Finalement, que l'on se situe dans le registre de l'algèbre, dans le registre d'un langage informatique (permettant de mettre en mots des algorithmes) L_1 ou dans le registre d'un autre langage informatique L_2, un traitement algébrique du problème nécessite la transformation de son expression en langage naturel en une expression d'un des langages cibles, dans lequel s'opèrera la solution. En référence au « langage de la théorie arithmétique » de Diophante, ces langages cibles, ces « langages de la théorie » par opposition au langage du problème, seront désignés génériquement par la lettre θ. Ainsi, si le langage cible est le langage de programmation de Snap ! [36] et qu'il n'y a pas d'ambiguïté possible, nous parlerons d'une « expression de θ » pour signifier « une expression du langage Snap ! » (et qui serait donc une expression différente avec un autre langage).

Variable Comme on l'a vu, le symbole de l'arithme permet de dépasser la simple marque de l'inconnu (du fait que ce soit dans le calcul toujours une quantité non connue, donc qu'on a affaire à un problème) à la représentation de la valeur d'une grandeur permanente dans le texte mais non encore connue. Cela semble correspondre à l'idée que l'on pourrait se faire d'une variable. Il pourrait être tentant de comparer cette représentation à la « variable de sortie » d'un programme, ce que Sajaniemi (2005, p. 8) désignerait comme ayant un rôle de « Most-wanted holder ». La différence est cependant notable : en informatique, on ne peut opérer sur une variable non encore connue, c'est-à-dire non encore initialisée [37]. Tant que, à l'exécution, on n'a pas établi la valeur de l'inconnue, on ne peut la manipuler. En outre, en tant que *Most-recent holder*, une valeur a déjà pu être assignée, sans que cette valeur soit la solution. Par exemple, considérons le simple programme :

$x \leftarrow 0$;
tant que $\neg(3 \times x + 9 > 20)$ **faire**
| $x \leftarrow x + 1$;
fin
Afficher(x) ;

36. https://snap.berkeley.edu/
37. En fait, c'est possible dans un certain nombre de langages, mais le résultat est — généralement —imprévisible.

La variable solution x affichée peut difficilement être considérée comme l'équivalent informatique ou algorithmique de l'inconnue, puisqu'elle ne dénote pas que la réponse (4) au problème. Dans la version suivante, on pourrait éventuellement considérer x comme une inconnue, mais uniquement pour cet algorithme, et uniquement parce qu'on n'opère pas avec la représentation de l'inconnue avant qu'elle ne soit déterminée.

$n \leftarrow 0$;
tant que $\neg(3 \times n + 9 > 20)$ **faire**
| $n \leftarrow n + 1$;
fin
$x \leftarrow n$;
Afficher(x) ;

1.4.6 Concepts algébriques ou pseudo-algébriques présents

L'inconnue L'arithme de Diophante est difficilement identifiable à une variable informatique, sauf à n'opérer dessus que lorsqu'elle a été déterminée — qu'elle n'est donc plus une inconnue. Mais l'arithme est-il la représentation de l'inconnue au sens algébrique du terme ? Comme on l'a dit, c'est une représentation, dans θ, dénotant la valeur cherchée. Cette représentation est manipulable dans une expression de θ comme si elle était connue. Cependant, l'arithme n'est pas totalement univoque : certes dans un même problème p_1, l'arithme ς dénote toujours la même valeur, mais s'il y a un sous-problème p_2 nécessitant une deuxième inconnue, Diophante garde la représentation ς. Comme p_2 fait partie de p_1, pour lever toute ambiguïté et rendre le traitement algorithmisable (ce qui est le propre du langage algébrique moderne), donc exécutable par un dispositif automatisé, il faudrait avoir $\delta(p_2) = \delta(p_1)$, ce qui n'est pas le cas. Or, selon Duval (2006b, p. 53), « [l']opération de désignation, qui crée la référence à un objet, est soumise à une condition d'unicité pour qu'il n'y ait pas d'ambiguïté dans la communication sur l'objet qui est ainsi désigné ». ς dans ce cas est toujours une marque de l'inconnue, mais ce n'est pas « l'inconnue du problème ». On peut noter aussi que les abréviations des puissances de l'arithme limitent les expressions possibles : Diophante ne peut écrire, avec son langage, une expression équivalente à $(x^2 + x + 7)^2$. Notons qu'il en sera de même avec les cossistes.

Procédés et voie Diophante, pour résoudre ces problèmes, met en œuvre des procédés algébriques : substitution, élimination. Ce qui fait dire à Radford (1991, p. 1) que « L'algèbre est donc, au 9ème siècle, l'art de réduire et résoudre des équations ». La méthode décrivant la solution manipule une représentation de l'inconnue comme si elle était un nombre connu. De plus, la méthode générale d'approche de la résolution d'un problème, la « voie », correspond à une démarche algébrique. Le langage de Diophante n'est pas algébrique, mais ses solutions sont très proches d'être des solutions algébriques.

Pensée algébrique En fait, si on ne peut pas encore parler d'algèbre, on peut sans doute parler de pensée algébrique. Radford (2014, p. 260) suggère trois conditions qui caractériseraient la pensée algébrique[38] :

38. « there are three conditions, I would like to suggest, that characterise algebraic thinking : (1) indeterminacy : the problem involves not-known numbers (unknowns, variables, parameters, etc.) ; (2) denotation : the indeterminate numbers involved in the problem have to be named or symbolised. Now this symbolisation may be accomplished in various ways. One can use alphanumeric signs—but not necessarily. The denotation of indeterminate quantities

1. « indeterminacy » (indétermination) : le problème concerne des nombres non connus (inconnues, variables, paramètres etc.) ;
2. « denotation » (dénotation) : les nombres indéterminés concernés par le problème sont dénommés ou symbolisés (pas forcément grâce à des signes alpha-numériques, mais aussi le langage naturel, les gestes, des signes non conventionnels...) ;
3. « analycity » (analycité) : les quantités indéterminées sont manipulées comme si elles étaient connues.

Si l'on peut constater que, chez les babyloniens, indétermination et dénotation sont bien présents, l'analycité ne l'est pas. En revanche, chez Diophante, cette analycité est effective. Diophante met en pratique ce qui relève de la pensée algébrique.

1.4.7 Bilan

Diophante marque une évolution quant au caractère algébrique, non de ses problèmes, mais de ses méthodes :

— Les problèmes sont présentés de façon générique (mais la solution est instanciée).
— Une forme de pensée algorithmique, un « meta-algorithme » est un des objectifs d'apprentissage des *Arithmétiques*.
— Un changement de registre de représentation sémiotique du langage du problème à un langage de la théorie θ est impliqué par cette méthode générale de traitement des problèmes.
— Cela implique notamment un langage plus spécifique et explicité, mais ce langage n'est pas un langage formel : c'est un langage syncopé, mêlant langue naturelle, abréviations, et symboles.
— Enfin, l'inconnue est représentée par un symbole, une expression de θ, et ce symbole est manipulé dans des expressions de θ comme si sa valeur était connue. Ce symbole n'est cependant pas toujours univoque.

1.5 al-Khwārizmī et les débuts de l'algèbre

Si Diophante utilise une représentation de l'inconnue, et la manipule, les solutions sont spécifiques aux problèmes proposés. al-Khwārizmī, dans son ouvrage *Kitāb al-Jabr wa-al-muqābala* proposera des solutions de cas génériques avant de poser les problèmes. Son ouvrage est considéré comme marquant les débuts de l'algèbre, ou « fondateur de l'algèbre » selon Rashed (2007, p. vii), notamment parce qu'il a permis de poser les bases d'une « discipline mathématique indépendante de la géométrie et de l'arithmétique », et que cela a ouvert de « nouvelles possibilités inhérentes à l'algèbre ».

1.5.1 al-Khwārizmī et al-jabr

Muhammad ibn Mūsā al-Khwārizmī est un mathématicien (mais aussi astronome, géomètre...) de l'école de la Sagesse, à Bagdad, et a écrit son ouvrage sur *al-jabr et al-muqābala* dans les années 820 de notre ère [39]. Ce livre est une commande du calife al-Ma'mūn, destinée à permettre de régler les problèmes parfois très complexes d'héritage. Ainsi, parlant de son livre, al-Khwārizmī écrit :

can also be symbolised through natural language, gestures, unconventional signs, or even a mixture of these ; (3) analyticity : the indeterminate quantities are treated as if they were known numbers. That is, although the are not known, one starts from the indeterminate quantities and operates on them (i.e., adds, subtracts, multiplies, divides them) as if they were known : This is what analycity means ».

39. Pour une liste de ses ouvrages attestés, voir (Farès, 2017a, p. 4-7).

> j'ai voulu qu'il enferme ce qui est subtil dans le calcul et ce qui en lui est le plus noble, ce dont les gens ont nécessairement besoin dans leurs héritages, leurs legs, leurs partages, leurs arbitrages, leurs commerces, et dans tout ce qu'ils traitent les uns avec les autres lorsqu'il s'agit de l'arpentage des terres, de la percée des canaux, de la mensuration, et d'autre choses relevant du calcul et de ses sortes[...] (ibid., p. 94)

Il y a une fois de plus une intention didactique dans cet ouvrage, dont on voit qu'il s'appuie sur des problèmes concrets, mais qu'il réussira à abstraire.

Son ouvrage est constitué de deux parties équilibrées : la première posera les bases de la théorie qu'il mobilisera dans la deuxième partie. Cette deuxième partie est-elle même constituée de problèmes issus de trois domaines d'application différents : les nombres, la géométrie, et les problèmes d'héritages et de legs (Farès, 2017a, p. 9-10). Comme les *Arithmétiques* de Diophante, la partie théorique débute par la définition des objets manipulés, de leur dénomination et de leurs opérations. Il distingue ainsi la *racine* (*al-jadhr*) ou la *chose* (*shay*), qui représente l'inconnue — notre x —, du *māl*, le produit de la racine par elle-même — notre x^2— et le *nombre*, qui n'est ni *māl* ni *chose*. Ensuite, alors que Diophante montrait la « voie » par des exemples résolus, al-Khwārizmī établit les six formes canoniques d'une équation du second degré :

$$\begin{array}{lll} 1) ax^2 = bx & 3) bx = c & 5) ax^2 + x = bx \\ 2) ax^2 = c & 4) ax^2 + bx = c & 6) ax^2 = bx + c \end{array}$$

Ces six types ne sont pas formulés, comme nous venons de le faire, par des symboles, mais en mobilisant le vocabulaire qu'il a défini plus tôt.

Il propose alors pour chacun de ces types un algorithme de sa solution (le *procédé*), à partir d'un exemple, accompagné de la justification géométrique de ces algorithmes (ibid., p. 11). La partie suivante présente les opérations sur les expressions binomiales (de la forme $(ax \pm b) \cdot (dx \pm c)$) et trinomiales ($(ax^2 \pm bx \pm c) \pm (a'x^2 \pm b'x \pm c')$). Enfin les opérations algébriques *al-jabr* et *al-muqābala* sont précisées au travers de six problèmes, qui amènent à une des six formes canoniques. Suivront ensuite divers problèmes, toujours présentés de façon générique et résolus sur une instance, très souvent en laissant la dernière étape (la résolution effective d'une des formes canoniques) à la charge du lecteur.

Rashed (2007, p. 49) précise que cet ouvrage se fonde finalement sur trois idées essentielles :

— Les équations et leurs solutions sont données *avant* les problèmes : « [l'algorithme]prend le pas sur la solution ». Il ne s'agit cependant pas d'inverser simplement la démarche de Diophante, ou d'établir les équations et leurs solutions comme des abstractions d'un ensemble de problèmes. Ces équations ont été établies par un raisonnement sur les objets algébriques manipulés.

— L'algèbre est donc une discipline algorithmique, et « si l'on veut que cette discipline soit mathématique, il faut que les algorithmes prouvent leur apodicticité ». Il ne s'agit pas d'accepter une solution spécifique et vérifiable sur un cas, mais d'accepter une méthode générale. On doit donc prouver la validité de la méthode, du procédé, de l'algorithme.

— Pour prouver la validité de l'algorithme, al-Khwārizmī procédera de deux façons, d'une part en démontrant « par la cause », c'est-à-dire en s'appuyant sur la géométrie, et d'autre part « par l'expression », c'est-à-dire en s'appuyant sur des règles de transformations d'expressions : on voit là une caractéristique de l'algèbre, mais aussi de l'algorithmique.

1.5.2 Le langage de la théorie de al-Khwārizmī

al-Khwārizmī définit donc un langage spécifique pour sa théorie, ce langage cible que nous désignons de façon générique par θ, et dans lequel il faut transformer la formulation du problème puis le résoudre.

Inconnue et indéterminée Comme on l'a dit, les éléments de base de ce langage sont la *racine* (*jidhr*) ou la *chose* (*shay'*), le *māl* ou le carré de la *chose*, et les nombres, qui ne sont ni des *choses* ni des *māl*. al-Khwārizmī n'utilise pas de symboles, il utilise des mots déjà existants, et leur prête un sens spécifique dans le registre sémiotique de sa théorie. Ainsi, le *mot*, par exemple « shay' » prend un statut de *nom propre*, désignant un objet mathématique (Duval, 2006b, p. 52). Rashed (2007, p. 15) souligne que le mot *shay'*, la *chose*, n'est pas choisi de façon anodine : les grammairiens de l'époque le considéraient comme « le plus indéfini des indéfinis », et en théologie, « une existence certaine, mais dont la connaissance que nous en avons est encore indéterminée ». Dans le langage d'al-Khwārizmī, la *chose* est à la fois le signe de l'inconnu, mais aussi, selon Farès (2017a) ou Houzel (2015, p. 3), de l'indéterminée : en effet, la solution est souvent irrationnelle, au contraire des solutions envisagées par Diophante, et l'irrationnel est non seulement traité comme un nombre dans les calculs, mais aussi il exprime l'indéterminée. al-Khwārizmī précise ainsi que :

> Sache que si tu veux doubler la racine de tout carré (*māl*), connue ou irrationnelle [...] alors il faut que tu multiplies deux par deux et ensuite le carré et la racine produite sera égale à deux fois la racine de ce carré [...] (Rashed, 2007, p. 130)

Formulé en langage algébrique moderne, on a le raisonnement $2\sqrt{a} = \sqrt{2 \times 2 \times a} = \sqrt{2^2 \times a}$. al-Khwārizmī donne un autre exemple pour une multiplication par trois, et indique qu'il faut faire de même « pour les multiples plus grands ou plus petits », soit $n\sqrt{a} = \sqrt{n^2 a}$. Dans ce même chapitre, il évoque, avec son langage, l'égalité $(\sqrt{200} - 10) + (20 - \sqrt{200}) = 10$: l'irrationnel est ici le signe que ce nombre peut être remplacé par n'importe quel nombre. Il prend le rôle d'indéterminée : l'égalité n'est pas valable *que* pour ce nombre, mais pour *un ensemble* de nombres [40]. La *chose* ou l'irrationnel sont manipulés dans les expressions de θ, comme s'ils étaient connus.

Opérations sur les éléments du langage Concernant les opérations sur ces objets primitifs que constituent ces différents types de nombres, on peut remarquer qu'elles sont présentées d'abord de façon abstraite, avant d'être illustrées par un exemple. Ainsi, le « chapitre sur la multiplication » (ibid., p. 122), concernant les expressions binomiales (de la forme $(ax \pm b) \cdot (cx \pm d)$), commence par ces mots :

> Je t'enseigne comment multiplier les unes par les autres les choses qui sont les racines, si elles sont seules, si elles sont avec un nombre, si elles sont diminuées d'un nombre ou si elles sont retranchées d'un nombre ; et comment les additionner les unes aux autres et comment les retrancher les unes des autres.

La multiplication d'un nombre par un nombre est présentée comme une addition réitérée : « il est nécessaire d'additionner l'un des nombres autant de fois que l'autre contient d'unités ». On reste ici sur une formulation générique. En revanche, lorsqu'il s'agit de préciser comment faire les calculs $(a + b) \cdot (c + d)$, al-Khwārizmī va passer par une représentation générique intermédiaire, avec des dizaines et des unités. Il résout ainsi « des dizaines auxquelles on a ajouté des unités, ou dont on a retranché des unités » que l'on multiplie :

40. Voir par exemple, sur le sujet du statut des lettres dans l'algèbre élémentaire, (Briant, 2013, p. 33).

> il est nécessaire de les multiplier quatre fois : les dizaines par les dizaines, les dizaines par les unités, les unités par les dizaines et les unités par les unités [...].

Il exprime ici l'opération de distributivité [41] $(10a+b)\cdot(10c+d) = (10a\times 10a)+(10a\times d)+(b\times 10c)+(b+c)$ (et ses variantes en cas de soustraction). Cette solution est ensuite de nouveau exprimée sur une instance : $(10+2)\cdot(10-1)$, puis sur une instance impliquant des *choses*, $(10-x)\cdot(10-x)$. On voit bien ici toute la difficulté de formuler des solutions génériques sans disposer de paramètres. Les nombres deviennent des paramètres généraux [42].

al-jabr* et *al-muqābala Les deux opérations algébriques de la théorie sont elles-aussi définies par des exemples prenant un caractère générique, dans le « chapitre des six problèmes » (ibid., p. 144). Chaque problème illustre la transformation d'une équation en une des formes canoniques établies. Le premier problème, en algèbre actuelle, s'écrit $x^2 = 4x(10-x)$. Les opérations vues précédemment permettent d'établir ensuite que $4x(10-x) = 40x - 4x^2$, donc qu'on doit avoir $x^2 = 40x - 4x^2$. al-Khwārizmī indique qu'il faut alors *restaurer* les quatre carrés et les ajouter au carré : il s'agit de faire disparaitre un terme négatif en ajoutant sa valeur absolue aux membres de l'équation. « *al-muqābala* consiste à éliminer les termes communs aux deux membres » (Farès, 2015, p. 6). Il ne s'agit donc pas seulement de manipuler des nombres, mais de combiner des termes, sans se préoccuper de ce que représentent les valeurs en jeu (nombre, longueurs, valeur monétaire...) : al-Khwārizmī rend les objets manipulés « ontologiquement neutres ».

Combinatoire et ontologie Pour démontrer la validité de propositions telles que $(\sqrt{200}-10)+(20-\sqrt{200}=20-10)$, ou de façon plus générique $(\sqrt{x}-a)+(\sqrt{(}x)+b) = b-a$, al-Khwārizmī s'appuie sur la géométrie, en représentant les différents nombres en jeu par des segments (Rashed, 2007, p. 46-47). Pour d'autres égalités impliquant non les racines, mais les carrés, il raisonnera, comme les babyloniens, sur les surfaces. Cependant, selon lui, ces démonstrations « par la cause » ne sont possibles que si l'expression est composée de deux espèces seulement (ibid., p. 48, 54). Ainsi, pour $(100+x^2-20x)+(50+10x-x^2)$, al-Khwārizmī dit « nous pouvons en avoir une figure, mais non sensible » : « il ne lui convient pas de figure car cela est composé de trois genres différents, des māls, des racines et du nombre, et il n'y a pas avec eux ce qui leur est égal pour que cela puisse être représenté par une figure. » (Farès, 2015, p. 10). al-Khwārizmī est ici confronté à un problème d'homogénéité, l'ontologie des nombres, leur part de réalité, ne semble pas compatible avec une représentation figurée. Pourtant, les babyloniens, ou Descartes plus tard et de façon systématique, ont contourné ce problème d'homogénéité : si ax^2 est une surface, bx est aussi une surface, celle d'un rectangle de côtés de longueur b et x — ce que conçoit aussi al-Khwārizmī—, et un nombre c est la surface d'un rectangle de côtés c et 1 (« projection »).

Cela interroge : comment al-Khwārizmī, a-t-il pu établir avec certitude les six formes canoniques des polynômes de degré 1 et 2, sans disposer de paramètres et en étant limité par des considérations d'homogénéité dues à l'ontologie des valeurs en jeu ? Comment accéder à la généralité ? Rashed (2007, p. 22) souligne qu'à cette époque « on assiste à la construction de tout un groupe de disciplines — lexicographie, morphologie, prosodie, cryptographie, crypto-analyse, etc. — qui appliquent une nouvelle méthode ». Cette méthode consiste à « déterminer un ensemble

41. Il n'y a pas de justification de cette transformation, et « [al-Khwārizmī] admet la commutativité et l'associativité de l'addition et de la multiplication, la distributivité de la seconde par rapport à la première et l'associativité mixte dans l'algèbre des polynômes à coefficients rationnels. Les successeurs d'al-Khwārizmī, n'ont pas tardé à sentir le besoin d'exprimer explicitement ces propriétés » (Farès, 2015, p. 9).

42. Cette utilisation de nombres particuliers, différents des autres ou porteur d'une forme de généralité, que nous nommeront NPG pour Nombre Potentiellement Générique, sera rencontré lors de l'expérimentation dans de nombreux groupes (45f, 46a, 46b, 46c, 46e...)

d'éléments discrets et finis », d'en déterminer toutes les combinaisons possible, puis d'isoler parmi ces combinaisons celles qui sont effectivement possibles dans la discipline en question. Il donne l'exemple d'al-Khalīl (718-786), un linguiste qui a voulu constituer un dictionnaire exhaustif de l'arable en cherchant toutes les combinaisons possibles des racines des mots arabes (de bilitères à quinquilitères), puis en les combinaisons compatibles avec la phonologie (Rashed, 2007, p. 18-21). Il n'est plus question de *sens*, il est question de combinatoire. Il s'agit de manipuler des éléments linguistiques comme des éléments formels, indépendamment de leur ontologie, soit de les rendre « ontologiquement neutre » (ibid., p. 22). Les trois termes primitifs de sa théorie sont le nombre, la chose, le carré, et en les combinant deux à deux puis trois à trois, puis en éliminant les doublons tels que $ax^2 = n$ et $n = ax^2$, on obtient les six formes canoniques, que al-Khwārizmī énonce ainsi :

— « les carrés égaux à des racines » : $ax^2 = bx$ (ibid., p. 96)
— « les carrés égaux à un nombre » : $ax^2 = c$ (ibid., p. 98)
— « les racines égales à un nombre » : $bx = c$ (ibid., p. 98)
— « les carrés plus les racines égaux à un nombre » : $ax^2 + bx = c$ (ibid., p. 100)
— « les carrés et le nombre égaux à des racines » : $ax^2 + c = bx$ (ibid., p. 104)
— « les racines plus le nombre égaux aux carrés » : $bx + c = ax^2$ (ibid., p. 106)

On peut constater que cette écriture montre deux rôles non symétriques dans l'équation [43].

Ainsi, une fois les formes canoniques établies en ayant supprimé l'ontologie des termes manipulés, lorsque ces équations sont utilisées dans un problème, l'ontologie des termes est de nouveau considérée. al-Khwārizmī parait proche d'une conception formelle de l'algèbre, mais elle n'est pas totalement établie.

1.5.3 Concepts algorithmiques présents

Comme chez Diophante ou les babyloniens, les procédés, les algorithmes de résolutions proposés par al-Khwārizmī, sont destinés à être exécutés par autrui, il s'agit toujours de faire-faire une certaine chose à un certain dispositif d'exécution.

Le langage, comme on l'a vu, est spécifique et clairement défini. Des règles de manipulation de ce langage sont établies. S'il est rhétorique et ne contient pas de symbole, ce langage permet d'expliciter les actions successives sans ambiguïté. En outre, une fois la validité des règles de transformation établies (parfois de façon implicite), al-Khwārizmī raisonne sur le langage : il utilise le langage et ses transformations comme moyen de prouver la validité des algorithmes (preuve « par l'expression »). Modeste (2012, p. 40) considère la preuve comme étant consubstantielle de l'algorithme :

> Tout d'abord, pour présenter un algorithme et affirmer qu'il résout un problème donné, il convient d'en donner une démonstration. Il faut être certain qu'il aboutit au bon résultat quelle que soit l'instance, c'est ce que l'on appelle la *correction*, et pouvoir garantir que ce résultat sera atteint en un nombre fini d'étapes, c'est la *terminaison*.

Contrairement à Diophante, qui vérifie sur une instance, ou Euclide qui démontre la vérité des propositions, al-Khwārizmī démontre la validité du procédé. La question de la terminaison n'est pas abordée, elle est implicitement admise, mais la question de la correction est essentielle : al-Khwārizmī vise à construire des solutions apodictiques, et non assertoriques (comme peuvent l'être les solutions de Diophante).

43. Rashed (2007, p. 77) compare les expressions de l'équation $x^2 + 10x = 39$: al-Khwārizmī l'exprime « un carré plus dix de ses racines sont égaux à trente-neuf », tandis que Brahmagupta aurait écrit :

ya v 1 *ya* 10 *ru* 0
ya v 0 *ya 0 ru 39*

Concernant les autres concepts algorithmiques, on voit évidemment que al-Khwārizmī manipule la séquence. Des éléments concernant les structures de contrôles et les questions sont aussi visibles. Ainsi, une fois avoir établi et justifié le procédé permettant de résoudre un des cas canonique, cet algorithme n'est plus utilisé explicitement dans les problèmes : il laisse ce travail à la charge du lecteur, convoquant, à la manière d'une procédure, l'algorithme précédemment défini.

> [al-Khwārizmī] donne chaque cas idéal, chaque type, et formule l'algorithme qui lui correspond. Cet algorithme, il le désigne du vocable *bāb*, la porte, [...] la « voie poursuivie pour parvenir au but cherché ». Il s'agit donc bien de ce qu'en français on nomme "procédé", en anglais "procedure"[...] (Rashed, 2007, p. 51)

Une fois le procédé, le *bāb*, établi et validé, al-Khwārizmī y renverra le lecteur, en introduisant le terme *qiyās* que Rashed (ibid., p. 51) décrit ainsi :

> Le *qiyās* se présente donc comme une opération en deux temps :
> 1) mettre le problème particulier proposé en conformité avec un modèle général (un des six cas idéaux)[...] à l'aide des opérations algébriques.
> 2) résoudre, ou renvoyer à la solution déjà établie, de l'équation.

On trouve ainsi, une fois la transformation de l'équation faite et aboutissant à un des six cas, des formulations telles que « Réduis par cela comme nous l'avons expliqué au début du livre » (problème <10>) ou encore « Restaure par cela de la manière que je t'ai décrite dans la partition des racines, si Dieu le veut » (problème <21>). De même, dans la deuxième partie du livre, al-Khwārizmī ne détaille plus les étapes permettant d'établir les solutions à partir d'une forme canonique. Le lecteur doit exécuter la procédure, le *bāb* : al-Khwārizmī change ainsi de niveau d'abstraction (Abiteboul et Dowek, 2017) [44]. On retrouve ici une des caractéristiques importantes de la « pensée informatique » :

> Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively.(Wing, 2006) [45]

D'autre part, notamment lors de l'étude des problèmes d'héritage, al-Khwārizmī opère parfois à une disjonction des cas suivant une certaine condition. Par exemple dans le « chapitre sur l'avoir et la dette » (Rashed, 2007, p. 232-234), un même problème est décomposé en trois cas, dont les deux derniers commencent par « s'il [un père décédé] laisse deux fils et dix dirhams[...] on l'infère ainsi [...] » et « s'il laisse trois fils et a légué un cinquième [...] on l'infère ainsi [...] ». D'autres usages des conditions sont présents : « Si tu veux rendre entières les parts selon le droit [...] elles seront [...] » (ibid., p. 244). La locution « on l'infère ainsi » est en outre présente dans tous les problèmes : c'est la marque qui sépare l'énoncé du problème et de ses conditions, de sa résolution.

44. On peut d'ailleurs noter, toujours selon Rashed (ibid., p. 51), que le *qiyās* « désigne l'analogie entre des cas particuliers et un modèle général, ou l'analogie entre un nouveau cas et un autre qui, pour des raisons historiques, joue le rôle de modèle ».

45. « Penser informatiquement, c'est utiliser l'abstraction et la décomposition quand il s'agit d'affronter une tâche d'une grande complexité ou de concevoir un système d'une grande complexité. C'est être capable d'opérer une séparation judicieuse entre ce qui est plus ou moins difficile à résoudre, de choisir une représentation appropriée pour un problème donné ou de modéliser les aspects pertinents d'un problème pour le rendre abordable. C'est mettre en avant des propriétés stables pour décrire de manière synthétique le comportement d'un système » (Lescanne, 2009).

1.5.4 Concepts algébriques présents

Les algorithmes présentés sollicitent une variable (la *chose*), mais aussi ce qui se rapproche du concept de paramètre.

L'inconnue et les équations L'inconnue (la *chose*), comme on l'a vu, est présente en tant qu'objet disposant d'une représentation dans θ, dénotant de façon univoque une certaine valeur. La mise en équation est rendue explicite, certes de façon rhétorique, mais non ambigüe : par exemple le problème <20> (Rashed, 2007, p. 180) est énoncé « Si on dit : tu ajoutes vingt dirhams à un carré, on a douze fois la racine du carré », ce qui s'écrirait $x^2 + 20 = 12x$. Parfois, l'équation n'est pas formulée explicitement dans θ, lorsqu'il n'y a pas d'ambiguïté. Ainsi le problème <22> (ibid., p. 182) : « Si on dit : tu multiplies le tiers d'un bien par son quart ; on retrouve le bien ». Le *bien* est implicitement traduit en *chose*. L'équation correspondante serait $\frac{1}{3}x \times \frac{1}{4}x = x$, que l'on retrouve dans la formulation de la solution : « On l'infère ainsi : tu multiplies le tiers d'une chose par le quart d'une chose ; on a la moitié d'un sixième de carré égale à une chose [...] », soit $\frac{1}{3}x \times \frac{1}{4}x = x \equiv \frac{1}{2}\frac{1}{6}x^2 = x$. Les problèmes sont instanciés, avec des nombres qui prennent ainsi un caractère générique.

Des paramètres ? Cependant, d'autres algorithmes semblent être traités de façon générique. Ainsi, la présentation du « Chapitre sur les transactions » (ibid., p. 198) est :

> Sache que toutes les transactions [...] ont lieu selon deux modes et d'après quatre nombres prononcés par le demandeur, qui sont : quantité d'évaluation, taux, prix, quantité évaluée.

S'ensuit une explication rhétorique des relations entre ces nombres, par exemple « le nombre qui est la quantité d'évaluation n'est pas proportionnel à celui qui est le prix », et le problème est posé de trouver l'un des nombres connaissant les trois autres. Il s'agit ici de traiter des problèmes où l'inconnue est liée à d'autres valeurs, où l'on considère :

$$\frac{\text{quantité d'évaluation}}{\text{taux}} = \frac{\text{quantité évaluée}}{\text{prix}} \text{ soit : } \frac{a}{b} = \frac{c}{d}$$

La solution générale est rédigée ainsi, nous illustrons avec un exemple à droite :

On l'infère ainsi :	
tu examines les trois nombres évidents ;	par exemple, a, b, c, on cherche d
il est nécessaire que, parmi eux, il y en ait deux, dont chacun n'est pas proportionnel à son associé.	b et c
Tu multiplies les deux nombres évidents non proportionnels l'un par l'autre ;	$b \times c$
tu divises le produit par l'autre nombre évident, dont <l'associé> non proportionnel est inconnu ;	$\frac{b \times c}{a}$
ce que tu obtiens est le nombre inconnu cherché par le demandeur, et qui n'est pas proportionnel au nombre par lequel tu as divisé.	$\frac{a}{b} = \frac{c}{d} \implies d = \frac{bc}{a}$

Le texte de la solution est rhétorique, mais il est générique : peut-on considérer que al-Khwārizmī utilise ici des paramètres, ou bien des indéterminées ? Nous traiterons le sujet dans la partie suivante, consacrée à Viète et plus particulièrement Descartes, qui ont tous deux représenté les paramètres par des lettres.

La preuve Un autre aspect remarquable de l'ouvrage d'al-Khwārizmī est la question de la preuve, déjà évoquée plus haut pour le cadre algorithmique. Comme on l'a vu, al-Khwārizmī veut prouver que son procédé, son algorithme (*bāb*) est valide, et il procède pour cela de deux manières :

— D'une part en démontrant « par la cause », en se référant à la géométrie et au raisonnement sur ses objets : « la "cause" de l'algorithme, c'est-à-dire ce qui le rend nécessaire et garantit son universalité, réside précisément dans la nécessité et l'universalité des *objets* et des *relations* géométriques [...] », affirme Rashed (ibid., p. 50) qui conclut quelques phrases plus loin : « La notion de démonstration par la cause a donc deux dimensions : logique et ontologique ». Cette dimension ontologique étant parfois problématique (dans le cas des trinômes), un autre système de preuve doit être mobilisé.

— D'autre part en démontrant « par l'expression », indépendamment de toute « figure sensible ». Il s'agit d'une démonstration non pas extrinsèque, faisant appel à un autre registre ou un autre cadre (la géométrie), mais intrinsèque, en restant dans le registre du langage algébrique défini, les transformations dont la nécessité a été établie suffisant à justifier le raisonnement.

Une fois encore, al-Khwārizmī reconnait implicitement l'équivalence entre ces deux modalités de production de preuve. Ses successeurs se chargeront de les unifier, par exemple avec Thaābit ibn Qurra (ibid., p. 33-34)

1.5.5 Bilan

Le livre d'*al-jabr* et *al-muqbala* de al-Khwārizmī est généralement considéré comme marque le « commencement de l'algèbre ». Dans son ouvrage éponyme, Rashed (ibid., p. 12) le décrit ainsi :

> N'est-il pas vrai que l'on y rencontre pour la première fois le projet d'une discipline mathématique différente de la géométrie et de l'arithmétique ? et que c'est seulement à partir de ce livre, et jamais avant, que se sont formées et développées les traditions de la recherche en algèbre ? N'est-il pas vrai que c'est dans ce livre que la discipline a trouvé son nom ?

Il n'est bien entendu pas question de dire qu'il n'y avait pas d'algèbre avant al-Khwārizmī : on a vu dans les parties précédentes que des démarches, techniques, outils et éléments de langage relevant de l'algèbre ont existé avant al-Khwārizmī. Farès (2015, p. 6) précise ainsi :

> On comprend donc bien que, par l'expression "commencement de l'algèbre", on veut en fait dire que les inconnues, les équations et les polynômes n'ont jamais été traités avant ce livre du 9e siècle, comme des objets mathématiques ; on se contentait plutôt de les manipuler d'une façon aléatoire, au cours de la résolution de tel ou tel problème géométrique, arithmétique ou autre. Leur naissance comme des êtres mathématiques à part entière, accompagnée de celle des lois qui régissent leur traitement, est l'acte qualitatif nouveau qui a déterminé la naissance de l'algèbre.

Ainsi on a pu relever que :

— Les équations et leur méthode de résolution précèdent tout problème (changement de niveau d'abstraction).

— La preuve concerne le procédé, et non le résultat.

— Le langage est rhétorique, mais les termes et expressions de ce langage suivent des règles précises permettant de manipuler et représenter les objets sur lesquels porte ce langage. Sont ainsi rendues possibles les trois activités que Duval (1993, p. 41) estime être fondamentales pour qu'un système sémiotique soit un registre de représentation sémiotique : la formation d'une représentation identifiable, le traitement et la transformation de cette représentation dans le registre auquel elle appartient, et les conversions de représentation vers d'autres registres (géométrique, mais aussi marchand, lié aux héritages...)
— Les problèmes, comme avec Diophante, sont posés sous une forme générique, puis résolus par des exemples instanciés.

Cependant, concernant le dernier point, nous avons évoqué des procédures plus génériques. Si dans la plupart des algorithmes de résolution d'al-Khwārizmī on ne constate pas d'utilisation de variables dans un rôle de paramètre, la question se pose sur certains. Dans la partie suivante, nous préciserons ce concept de paramètre, dont une représentation par des lettres sera donnée par Viète et Descartes.

1.6 Viète et Descartes, le "donné" symbolisé

Dans les chapitres précédents, nous avons pu constater les similarités et évolutions entre différents textes mathématiques. Nous en faisons un rapide rappel ici, complété par quelques remarques, avant de préciser en quoi les propositions de Viète et Descartes furent déterminantes. Cela nous amènera à traiter de questions de vocabulaire, concernant notamment la variable et le paramètre.

1.6.1 Avant Viète et Descartes

Représentations de l'inconnue Nous avons pu voir, pour Diophante comme pour al-Khwārizmī, que l'inconnue était vue comme un terme du langage qui dénote une certaine valeur, et sur laquelle on peut opérer : ainsi, « $\delta\varsigma$ », « trois *choses* » et « $3x$ », dans un même problème, dénoteraient le même objet mathématique, mais dans un langage différent (le « langage de la théorie », θ), impliquant des registres de représentation sémiotique différents.

Par la suite, d'autres dénominations, représentations ou abréviations de l'inconnue vont émerger, que ce soit en Chine, en Inde ou en Europe. Voir par exemple le tableau 1.1 (p. 56) pour quelques dénominations. L'algèbre restera centrée sur l'inconnue et ses puissances. Ainsi,

Algèbre prémoderne				Algèbre moderne
Diophante	**Arabe**	**Latine**	**Italienne**	**Symbole moderne**
monas	*dirham / 'adad*	*dragma /numerous*	*dramma / numero*	1
arithmos	*jidhr / shay'*	*radix / res*	*radice / cosa*	x
dynamis	*māl*	*census*	*censo*	x^2
kybos	*ka'b*	*cubus*	*cubo*	x^3

Tableau 1.1 – Vocabulaire des algébristes prémodernes (Christianidis, 2018, p. 8)

on utilisera des représentations symboliques spécifiques pour désigner les « espèces » de Diophante, voir par exemple les symboles cossiques de Stifel (1486-1567) ou Rudolff, figure 1.9 (voir aussi (Heeffer et Van Dyck, 2010)).

Serfati (1998, p. 241) constate ainsi que :

Figure 1.9 – Symboles cossistes, Rudolff (Serfati, 1998, p. 249)

> A la fin du XV^{e} siècle, avec l'introduction de l'imprimé, les représentations de l'inconnue, de son carré, de l'addition, etc., dans le droit-fil des techniques diophantiennes, devinrent usuelles chez les calculateurs, tout en demeurant le plus souvent singulières, valides dans l'œuvre d'un seul auteur. D'où, à cette époque, une incroyable floraison de représentations insolites et de tentatives avortées, minutieusement inventoriées par Cajori.

Cependant, Oaks (2018, p. 245-246) souligne que cette diversité ne signifie pas incompatibilité, et que, avant 1590, on peut passer d'un auteur à l'autre sans encombre, du moment que les éléments du langage et les opérations sont décrites. Ainsi, il affirme que « The wide variety of notations and even terminology in these books mask the fact that the algebras they present are all conceptually and structurally compatible ».

Objectif didactique On a vu aussi que depuis les babyloniens, l'objectif de ces textes mathématiques était didactique : il ne s'agissait pas de produire des solutions, mais de transmettre et amener à comprendre les procédés, les algorithmes, permettant de produire des solutions. Cela situe ces textes comme mettant en place des « situations de programmation » (Samurçay et Rouchier, 1985, p. 242) :

> A la différence d'une situation habituelle de résolution de problème, la situation de programmation fait intervenir un élément essentiel qui est le dispositif d'exécution, définie par un ensemble d'opérations permises et les conditions de validité qui leur sont associées. La solution d'un problème doit alors être analysée à un double niveau : — les résultats qu'on veut obtenir par l'application du programme à des données ; — le programme qui est l'expression dans un langage de programmation d'une procédure définie par référence au dispositif d'exécution. Un premier aspect permet d'opposer comme le fait Hoc (1982), les situations de programmation aux situations de « production de résultat » dans lesquelles l'objectif d'élaboration de procédure s'identifie de fait à l'exécution. Un deuxième aspect concerne l'explicitation par le sujet des procédures qu'il utilise pour résoudre un problème. Cette explicitation est rendue nécessaire par le fait que l'exécution de la procédure se fait et doit se faire dans un temps distinct de celui de la construction du programme : l'exécution est différée. Le troisième

> aspect concerne le passage d'une planification des actions exécutables par le sujet lui-même à celle d'un plan d'actions dont l'exécution n'est pas contrôlée par l'opérateur humain. Le sujet n'aura pas l'occasion d'intervenir en cours d'exécution pour modifier le déroulement de la procédure.

L'objectif n'est pas non plus de transmettre les procédures de résolution, mais de permettre aux étudiants de les comprendre afin de pouvoir traiter de nouveaux problèmes : c'est la « voie » de Diophante.

Généralisation En outre, tous ces textes ont aussi en commun la recherche de généralité : il ne s'agit pas de résoudre *un* problème, mais *une famille d'instances* du problème. Le problème, depuis Diophante (mais aussi chez Euclide par exemple) est présenté de façon générique. Cette généralisation est au cœur de l'algèbre élémentaire. Ainsi, Bronner et Squalli (2021, p. 6), en parlant de la généralisation et de l'analycité évoqués par Radford (voir 1.4.6, p. 47) affirment :

> La généralisation et l'analycité sont deux caractéristiques essentielles de la pensée algébrique qui sont soulignées par la majorité des chercheurs [...]. La généralisation est un processus essentiel dans l'activité mathématique et tout particulièrement en algèbre. Elle est reconnue par plusieurs auteurs comme étant une composante très importante de la pensée algébrique (Lee, 1996 ; Kaput, 2008 ; Carraher et al., 2008 ; Radford, 2010, 2014 ; Squalli, 2000 ; Grugeon, 1995). Lee va jusqu'à penser qu'en algèbre, les activités de généralisation sont les plus importantes et constituent les seuls moyens d'initier les élèves à la culture algébrique. Elle ajoute : « Ce n'est pas non plus un défi de démontrer que les fonctions, la modélisation et la résolution de problèmes sont tous des types d'activités de généralisation, que l'algèbre et en fait toutes les mathématiques consistent à généraliser des modèles » (Lee, 1996, p. 102-103, traduction libre).

Cependant, si ces textes visent la généralisation, et si l'analycité commence à être décelée chez Diophante et al-Khwārizmī, la méthode de résolution est présentée sur une instance, à charge pour le lecteur d'effectuer lui-même la généralisation, en identifiant ce qui change et ce qui ne change pas, le pareil et le différent, si l'on prenait deux instances de ce problème. Le lecteur doit généraliser à partir d'exemples, et donc identifier des schémas valables sur toutes les instances possibles de ces exemples. Par exemple, en prenant un cas simple, avec notre écriture actuelle :

$$\begin{cases} 3x - 2 = 0 & \Longrightarrow \; x = \frac{2}{3} \\ 4x - 5 = 0 & \Longrightarrow \; x = \frac{5}{4} \end{cases}$$

Ces deux exemples (éventuellement complétés par d'autres) devraient permettre d'établir la formulation générique $ax - b = 0 \implies x = \frac{b}{a}$ [46]. Les nombres « 3 » et « 2 » d'une part, et « 4 » et « 5 » d'autre part, sont les *paramètres* des équations, ce qui les différencie dans une famille d'instances d'équations de même forme. Ni Diophante, ni al-Khwārizmī ne disposaient de moyen sémiotique permettant de dénommer et désigner les paramètres d'une façon générique. Chevallard (1989, p. 65) souligne ainsi :

> Mais la clé du succès [de l'algèbre] ne tient pas seulement dans ce petit x qui figure l'inconnue du problème. D'emblée la puissance algébrique est mise en relation avec le fait de désigner par des lettres, à côté des quantités inconnues, que l'on recherche, les quantités connues elles-mêmes - les données.

46. Si $a \neq 0$. Notons que al-Khwārizmī aurait écrit $3x = 2$, aucun des deux membres de l'équation ne pouvant être nul. D'autre part, Diophante, comme al-Khwārizmī ne considèrent comme nombre que les nombres positifs.

Cette désignation par des lettres opposant valeur connues et valeur inconnues sera proposée par Viète, puis Descartes.

1.6.2 Viète, une théorie formelle ou un calcul non interprété

François Viète (1540-1603) n'était pas mathématicien, mais juriste à la court d'Henri IV (Oaks, 2018, p. 246) et son ouvrage d'algèbre, *In artem Analycitem Isagoge* se démarque des autres ouvrages de l'époque. Comme ses prédecesseurs, son objectif était didactique :

> The aim of Viète's Analytic Art, following Ramus, to teach this method in a pedagogically effective fashion, that is, in a way that will enable students systematically to solve mathematical problems. (Kleiner, 2007, p. 90)

Cependant, il « introduisit un nouveau système de signes, uniquement constitué de lettres, et dont la fonction véritable consista, en dernière analyse, à faire prendre en charge par l'écriture symbolique deux concepts jusqu'alors considérés comme opposés : l'arbitraire et le fixé ou, plus significativement, le quelconque et le singulier » (Serfati, 1997, p. 129).

L'inconnue et ses puissances Comme Diophante ou les cossistes, dans son système de signes, Viète désigne l'inconnue et ses puissances. En revanche, il ne définit pas un terme pour l'inconnue, et un autre pour chacune de ses puissances : il utilise une lettre, une voyelle majuscule, par exemple A et la fait suivre par un mot indiquant sa puissance : A *cubum* signifiant ainsi le cube de l'inconnue. Ainsi, si dans les systèmes de signes précédents, la référence à l'inconnue n'était pas visible (*māl*, en tant qu'expression de θ, ne contient aucune référence à *'shay* ; ou encore Δ^y ne contient aucune référence au signe ς), on garde la trace de la présence de l'inconnue. D'autres avant Viète avaient utilisé des moyens similaires : par exemple Bombelli utilisait les symboles $\llcorner 1 \lrcorner$ pour l'inconnue, $\llcorner 2 \lrcorner$ pour son carré, $\llcorner 3 \lrcorner$ pour son cube etc. Comme le remarque Serfati (1998, p. 261), Viète, avec son suffixe marquant l'espèce, n'avait pas reconnu la relation entre les espèces et l'inconnue comme un nombre entier [47].

Le Requis et le Donné Pour Serfati (1997, p. 129), « Depuis l'antiquité, les deux catégories épistémologiques du Donné et du Requis avaient accompagné toute question revêtant la forme d'un problème ». Le Requis, le cherché, la *chose*, l'inconnue, a été depuis Diophante représenté par des symboles ou abréviations, et le Donné [48], le connu, est représenté par des chiffres, ce qui, comme on l'a signalé plus haut, obère toute généralisation formelle.

Dans l'*Isagoge* (cité par Serfati (ibid., p. 139)), Viète supprimera cette dichotomie nombre/signe :

> Or afin que ceci soit aidé par l'art, les grandeurs données seront distinguées des requises par un symbole constant perpétuel et apparent, en signifiant les grandeurs requises par l'élément Alphabétique A, ou quelques autres voyelles, E, I, O, U, Y, et les données par les éléments B, C, D, ou quelque autre des consonnes.

Comme le dit Kleiner, cette idée de représenter le donné, les paramètres, par des lettres (les consonnes) qui nous parait si naturelle, a permis un changement fondamental dans l'algèbre : la possibilité de généraliser.

47. D'une façon proche, Stevin écrivait par exemple, l'expression « 2③+ 8②-24①-96 », qui s'écrirait de nos jours $2x^3 + 8x^2 - 24x - 96$ (Serfati, 1997, p. 197)

48. Serfati met des majuscules pour distinguer la catégorie de la « réalisation effective » : requis et donné sont spécifiques à un problème

> Viète's basic idea was to introduce arbitrary parameters into an equation and to distinguish these from the equation's variables. He used consonants (B,C,D,...) to denote parameters and vowels (A,E,I,...) to denote variables. [...]. To us this appears to be a simple and natural idea, but it was a fundamental departure in algebra : for the first time in over three millennia one could speak of a general quadratic equation, that is, an equation with (arbitrary) literal coefficients rather than one with (specific) numerical coefficients. (Kleiner, 2007, p. 20)

Kleiner (ibid., p. 21) poursuit :

> This was a seminal contribution, for it transformed algebra from a study of the specific to the general, of equations with numerical coefficients to general equations. Viète himself embarked on a systematic investigation of polynomial equations with literal coefficient.

Serfati (1997, p. 138) souligne enfin que cette nouvelle écriture permettra de « continuer d'user des considérables avantages des preuves "en nombres" (mécanismes et automaticité du calcul, écriture de séquences complexes d'opérations), tout en conservant le caractère universel des énoncés et des solutions ». Ainsi Chevallard (1989, p. 65) rapporte que :

> [...]dans une note à sa traduction (1630) de La nouvelle algèbre de M. Viète, Vauzélard écrit : « L'utilité que l'on tire de cete nouvelle Algebre, est admirable, au respect de la confusion, de laquelle sont farsies les Algebres des anciens (...), à cause qu'ils exerçoient et faisoient les operations de leurs Algebres par les nombres ; c'est pourquoy de ces Algebres ne peut estre nul Theoreme ny solution generale pour toute proposition semblable à celle dont elle doit estre tirée, comme il se fait en celle-cy nouvellement instituée, de laquelle les ratiocinations et operations se font soubs les especes ».

Cette généralité et le caractère formel de la notation de Viète permettent de voir l'équation en fonction des connus, et donc de pouvoir facilement substituer ces représentations de connus par leur valeur, pour résoudre une instance du problème :

> With this new foundation came an entirely new notation. Viète's letters stand not only for unknown magnitudes, but for known ones as well. For the first time the structure of solutions can be seen in a simplified equation through the operational relations of the known magnitudes. To paraphrase Ritter, this final equation is a formula, and if one wants to solve the same problem with different knowns (as in trigonometry), it suffices to substitute them into this formula to obtain the answer immediately. (Oaks, 2018, p. 297)

L'homogénéité Cependant, malgré le symbolisme de son langage, Viète ne s'est pas débarrassé des conditions d'homogénéité dans ces expressions, expressions elles-mêmes syncopées, puisque mêlant symbole et mots (Kleiner, 2007, p. 21). En effet, comme ses prédecesseurs, « Viète required "homogeneity" in algebraic expressions : all terms had to be of the same degree. ».

Cette nécessité d'homogénéité amène un fonctionnement différent pour les inconnues et les données. Ainsi, comme le signale Macbeth (2004, p. 90), dans l'expression « *A cubum* », qui est comparable à notre x^3, le « *A* » ne désigne pas la valeur, mais la racine : multiplier « *A quadratum* » par « *A* » donne « *A cubum* ». En revanche, pour les connus, l'écriture, si elle est similaire, n'a pas le même sens : si *A quadratum* dénote la valeur de *A* multiplié par *A*, *B plano* dénote *B* et *précise* qu'il s'agit d'un carré, pour des raisons d'homogénéité : *B plano*, ce n'est pas $B \times B$. *Plano* ou *solido* pour les connus ne servent que d'annotation, de remarque : « it serves to remind the analyst that if, at the last stage in solving a problem, he turns geometer (rather than arithmetician), he must put for 'B' something of the appropriate "scale." ». Pour Viète, « *A quadratum + B plano* » est valide, alors que « *A cubum + B plano* » ou « *A quadratum + B solido* » ne le sont pas — alors que la première et la dernière de ces expressions s'écrirait de nos jours $x^2 + b$.

Ainsi, Viète symbolise inconnues et connues, comme un langage formel, mais pour Macbeth (ibid., p. 92-93) ce n'est pas vraiment un langage, mais plutôt « un calcul non interprété » : le problème est représenté de façon formelle, et ontologiquement neutre (les signes ne sont pas interprétés), mais lorsqu'on effectue le calcul, l'ontologie doit être compatible avec l'interprétation :

> The only plausible reading of Viète's logistice speciosa is a reading of it as a formal theory or uninterpreted calculus. The first stage of the Analytic Art, zetetics, takes one out of a particular domain of inquiry, either arithmetical or geometrical, into a purely formal system of uninterpreted signs that are to be manipulated according to rules laid out in advance, and only at the last stage, exegetics, are the signs again provided an interpretation, either arithmetical or geometrical.

Descartes, entre autres choses, montrera comment dépasser cette contrainte d'homogénéité.

1.6.3 Descartes ou le dépouillement ontologique

Descartes, lorsqu'il étudiait au collège de Jésuites à La Flèche (1607-1615) (Jullien, 1996, p. 5), maitrisait le système cossique :

> La pratique cartésienne était ici certes dérivée de l'*Algebra* de Clavius, un ouvrage de référence majeur à La Flèche. Descartes utilisa cependant un système cossique archaïque, souvent ambigu, inférieur à celui de Clavius. (Serfati, 1998, p. 255)

Descartes s'éloignera des cossistes à qui selon Jullien (1996, p. 14) il reprochait « une imagination reproductrice trop réaliste », à quoi il opposera « une faculté de former des images qui, certes, expriment les objets qui sont l'occasion de ces images, mais peuvent bien ne pas leur ressembler ». Il citera ainsi Descartes dans *La Dioptrique* :

> Il faut au moins que nous remarquions qu'il n'y a aucunes images qui doivent en tout ressembler aux objets qu'elles représentent [...] mais qu'il suffit qu'elles leur ressemblent en peu de choses, et souvent même, que leur perfection dépend de ce qu'elles ne leur ressemblent pas tant qu'elles pourraient faire.[...]

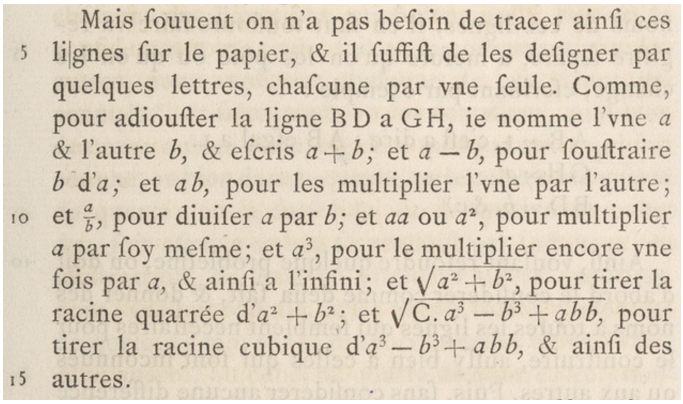
Selon Serfati (1998, p. 285), « Descartes cependant, qui n'était pas un formaliste, ne se préoccupait guère de questions de symbolique. ». Quoi qu'il en soit :

> The concern of mathematics, according to Descartes, « is with questions of order and measure and it is irrelevant whether the measure in question involves numbers, shapes, stars, sounds, or any other object whatever » (Macbeth, 2004, p. 98)

Descartes cherche à représenter toute sorte de mesure de façon simple, et il ramènera tout problème de géométrie ou d'arithmétique à des calculs sur des longueurs de lignes : Dans son *Discours de la méthode* (Descartes, 1897-1913, p. 20) [49] :

> Puis, ayant pris garde que, pour les connoiſtre, i'aurois quelquefois beſoin de les conſiderer chaſcune en particulier, & quelquefois ſeulement de les retenir, ou de les comprendre pluſieurs enſemble, ie penſay que, pour les conſiderer mieux en particulier, ie les deuois ſuppoſer en des lignes, a cauſe que ie ne trouuois rien de plus ſimple, ny que ie pûſſe plus diſtinctement repreſenter a mon imagination & a mes ſens ; mais que, pour les retenir, ou les comprendre pluſieurs enſemble, il falloit que ie les expliquaſſe par quelques chiffres, les plus courts qu'il ſeroit poſſible ; et que, par ce moyen, i'emprunterois tout le meilleur de l'Analyſe Geometrique & de l'Algebre, & corrigerois tous les defaus de l'vne par l'autre.

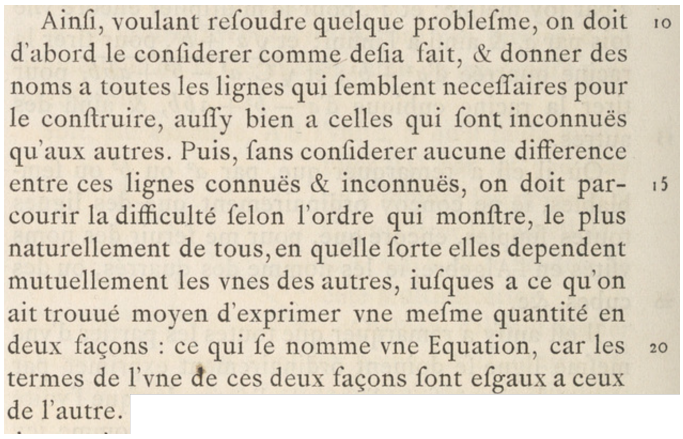
49. Transcription https://fr.wikisource.org/wiki/Discours_de_la_mÃ©thode/Ã©dition_Adam_et_Tannery



Mais ſouuent on n'a pas beſoin de tracer ainſi ces lignes ſur le papier, & il ſuffiſt de les deſigner par quelques lettres, chaſcune par vne ſeule. Comme, pour adiouſter la ligne BD a GH, ie nomme l'vne a & l'autre b, & eſcris $a+b$; et $a-b$, pour ſouſtraire b d'a; et ab, pour les multiplier l'vne par l'autre; et $\frac{a}{b}$, pour diuiſer a par b; et aa ou a^2, pour multiplier a par ſoy meſme; et a^3, pour le multiplier encore vne fois par a, & ainſi a l'infini; et $\sqrt{a^2+b^2}$, pour tirer la racine quarrée d'a^2+b^2; et $\sqrt{C.\, a^3-b^3+abb}$, pour tirer la racine cubique d'a^3-b^3+abb, & ainſi des autres.

Figure 1.10 – « Comment on peut user de chiffres en géométrie » (Descartes, 1897-1913, p. 371)
Source gallica.bnf.fr / Bibliothèque nationale de France

Représenter les objets de la géométrie et de l'arithmétique Descartes représente les lignes et leur longueur par une lettre, et définit les écritures des opérations (figure 1.10, p. 62) : $a+b$, $a-b$, $\frac{a}{b}$ et ab pour la multiplication. Il précise en outre l'écriture de l'exposant : « aa ou a^2, pour multiplier a par soi-même ; et a^3, pour le multiplier encore une fois par a, et ainsi de suite à l'infini » (Descartes, 1897-1913, p. 272). Comme Viète, il nomme par des



Ainſi, voulant reſoudre quelque probleſme, on doit d'abord le conſiderer comme deſia fait, & donner des noms a toutes les lignes qui ſemblent neceſſaires pour le conſtruire, auſſy bien a celles qui ſont inconnuës qu'aux autres. Puis, ſans conſiderer aucune difference entre ces lignes connuës & inconnuës, on doit parcourir la difficulté ſelon l'ordre qui monſtre, le plus naturellement de tous, en quelle ſorte elles dependent mutuellement les vnes des autres, iuſques a ce qu'on ait trouué moyen d'exprimer vne meſme quantité en deux façons : ce qui ſe nomme vne Equation, car les termes de l'vne de ces deux façons ſont eſgaux a ceux de l'autre.

Figure 1.11 – « Qu'est-ce qu'une équation ? » (Descartes, 1897-1913, p. 372)
Source gallica.bnf.fr / Bibliothèque nationale de France

lettres autant les connus que les inconnus : dans la méthode qu'il donne pour aboutir à ce qu'il nomme une « Equation », il prescrit ainsi :

> Ainsi pour résoudre quelque problème, on doit [...] donner des noms à toutes les lignes qui semblent nécessaires pour le construire, aussi bien à celles qui sont inconnues qu'aux autres. (ibid., p. 372)

Descartes précisera dans la phase suivante, qu'il ne faut « considérer aucune différence entre ces lignes connues et inconnues ».

Ainsi, comme le reformule Macbeth (2004, p. 99), « As Descartes employs them, letters and combinations of letters signify something in particular, namely, line lengths (themselves conceived as representing arbitrary quantities), either those that are given or those that are sought ». Cependant, il ne précise pas explicitement l'usage des premières lettres de l'alphabet pour les données et les dernières lettres de l'alphabet pour les inconnues. Il ne semble d'ailleurs pas y

avoir toujours un même ordre concernant la représentation des inconnues : dans un problème la première inconnue est nommée z et les suivantes y puis x (Descartes, 1897-1913, p. 375), mais dans un autre ce sera x puis y puis z (ibid., p. 383). Dans ce même problème une différence sera nommée k. Il est seulement indiqué au détours d'un problème du livre II (ibid., p. 394) :

> [...] pour ce que CB et BA font deux quantités indéterminées et inconnues, je les nomme l'une y et l'autre x. Mais, afin de trouver le rapport de l'un à l'autre, je considère aussi les quantités connues qui déterminent la description de cette ligne courbe : comme GA que je nomme a, KL que je nomme b, et NL que je nomme c.

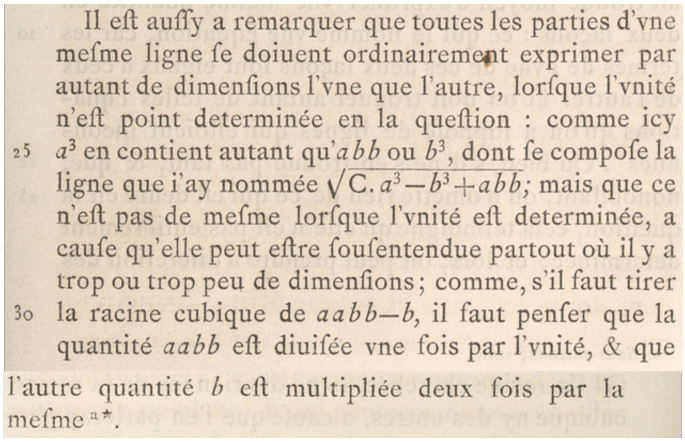
En revanche, une dizaine d'années avant *La Géométrie*, Descartes écrivait dans ses *Règles pour la direction de l'esprit*, règle XVI (Descartes et Salgues, 1824-1826, p. 314) :

> Mais pour plus de facilité, nous nous servirons des caractères a, b, c etc. pour exprimer des grandeurs déjà connues, et A, B, C, pour les grandeurs inconnues [...]

On voit que Descartes ne voulait pas traiter différemment connues et inconnues, mais il souhaitait malgré tout marquer leur différence.

En outre, la représentation des puissances de l'inconnue marquant à la fois le signe de l'inconnue et un nombre définissant une relation, Serfati (1998, p. 262) conclut :

> Conjuguant donc les avantages de ceux de Viète et de Bombelli, le système cartésien se dispensait aussi de leurs inconvénients respectifs. Son avènement signa la disparition de la symbolique diophanto-cossique qui, des siècles durant, avait gouverné la pensée mathématique sur la question des puissances. Avec lui disparurent ses principales limitations. Se substituèrent en effet aux comptines, des formules additives simples sur les nombres entiers, déployées à longueur de pages dans la Géométrie. Ainsi, la seule considération de la formule : $x^1 \cdot x^1 = x^2$ dispensa désormais du « Res in rem.... »



Il eſt auſſy a remarquer que toutes les parties d'vne meſme ligne ſe doiuent ordinairement exprimer par autant de dimenſions l'vne que l'autre, lorſque l'vnité n'eſt point determinée en la queſtion : comme icy a^3 en contient autant qu'abb ou b^3, dont ſe compoſe la ligne que i'ay nommée $\sqrt{C.\, a^3 - b^3 + abb}$; mais que ce n'eſt pas de meſme lorſque l'vnité eſt determinée, a cauſe qu'elle peut eſtre ſouſentendue partout où il y a trop ou trop peu de dimenſions; comme, s'il faut tirer la racine cubique de $aabb - b$, il faut penſer que la quantité $aabb$ eſt diuiſée vne fois par l'vnité, & que l'autre quantité b eſt multipliée deux fois par la meſme [a]*.

Figure 1.12 – « Gérer l'homogénéité » (Descartes, 1897-1913, p. 371)
Source gallica.bnf.fr / Bibliothèque nationale de France

Dépouillement ontologique La relation entre quantités, plutôt que la géométrie, est un point important pour Descartes : Macbeth (2004, p. 100) soutient ainsi que « Descartes' concern is not with geometrical figures but with the relations among line segments that can be "read off" such figures, relations that can be expressed algebraically ». C'est peut-être parce qu'il s'intéresse aux relations et non à la géométrie, que Descartes en vient à régler les problèmes d'homogénéité

rencontrés par ces prédecesseurs, y compris Viète. Il explique ainsi (figure 1.12, p. 63), que « toutes les parties d'une même ligne se doivent ordinairement exprimer par autant de dimensions l'une que l'autre » : par exemple a^3 et abb sont de même dimension. Descartes explique alors qu'il n'y a pas de problème d'homogénéité dans une expression telle que $aabb - b$, car « l'unité [lorsqu'elle est déterminée] est sous-entendue partout où il y a trop ou trop peu de dimensions » : si on veut une dimension 3 pour extraire une racine cubique (cela définit l'unité), il suffit de diviser $aabb$ par une fois l'unité, et multiplier b deux fois. L'expression est donc compatible avec la dimension 3 :

$$aabb - b = \frac{aabb}{1} - b \times 1 \times 1$$

Descartes ne donne pas la règle générale de façon explicite (l'unité est « sous-entendue »), mais son propos est générique, et étend la solution déjà vue chez les babyloniens — « projection » d'un côté c sur une longueur de 1, de sorte que cela forme une surface d'aire $c \times 1$ — à *toutes* les dimensions. Il se débarrasse de l'ontologie de l'expression grâce aux calculs sur des formules.

L'unité sous-entendue n'est jamais explicitement précisée : puisque le problème de l'homogénéité est réglé,

> that is why there is in Descartes' symbolism nothing corresponding to Viète's annotations 'plano', 'solido', and so on. In Descartes' Geometry a letter such as 'a' or a combination of signs such as '(a+b)2' is not an uninterpreted expression that can be interpreted either geometrically or arithmetically ; it is a representation of (an indeterminate) line length where a line length is to be distinguished both from a Euclidean line segment and from a number classically conceived as a collection of units. (Macbeth, 2004, p. 101)

Serfati (1998, p. 271) ne dit pas autre chose :

> [...]jamais Descartes n'interpréta exhaustivement les équations qu'il obtint. Le fait cependant que l'interprétation demeurât virtuellement possible fut — au moins au début du XVII^{e} siècle — le garant ontologique de l'écriture symbolique, savoir la traduction du symbolique en termes communicables à tous, au moyen d'une liste d'instructions séquentiellement ordonnée.

S'il existe un moyen systématique et ordonné, un algorithme, permettant de (re)passer du langage de la théorie θ au langage naturel, nul besoin d'expliciter le sens des équations ou des étapes et transformations de l'équation.

1.6.4 Après Descartes, l'ouverture des possibles

Les bases symboliques apportées par Descartes furent fondamentales pour le développement des mathématiques. Nous n'en donnerons ici qu'un exemple temporellement proche, puisque nous nous intéressons à l'algèbre élémentaire et que les éléments que nous allons présenter vous nous en éloigner.

Si Descartes s'est débarrassé de l'ontologie des formules, il ne va pas cependant aller jusqu'à envisager la possibilité de substitution d'un symbole par une formule. Ainsi Serfati (ibid., p. 259) remarque que :

> D'une façon digne d'être notée, la *Géométrie* ne porte cependant aucun exemple comme : $(y-3)^2$. Cette absence est significative des limitations que Descartes apporta à l'emploi de son propre système pour les puissances. Descartes se refusait ainsi à la substitution $x \curvearrowright (y-3)$ au lieu de la lettre x, dans l'assemblage x^2.

En effet, « l'écriture de la simple puissance d'un binôme, comme $(x+3)^2$, requérait des délimitants (parenthèses), ce à quoi Descartes répugna toujours, tout comme Viète. » (ibid., p. 259). Serfati donne alors l'exemple proposé par Descartes dans son livre III (Descartes, 1897-1913, p. 447). Descartes se propose d'augmenter de 3 la racine de l'équation :

$$x^4 + 4x^3 - 19xx - 106x - 120 = 0$$

Il indique qu'il faut « prendre y au lieu d'x, et penser que cette quantité y est plus grande d'x de 3, en sorte que $y - 3$ est égal à x ». Ainsi, on pourrait substituer x partout où elle apparait, par $y - 3$. Mais au lieu d'écrire par exemple $(y - 3)^3$ au lieu de x^3, Descartes indique que « au lieu d'x^3, il faut mettre son cube, qui est $y^3 - 9yy + 27y - 27$ ». Faisant de même pour x^2 et x^4, Serfati (1998, p. 260) considère qu'on trouve là « de surprenantes considérations rhétoriques, quatre lignes bien embarrassées, auxquelles Descartes s'était lui-même contraint ».

Cette substitution, qui est un outil utile de transformation d'expressions, sera utilisée notamment par Leibniz et Newton. Résumons en quelques lignes ce que Serfati (1997, p. 237-249) détaille :

— substituer 3 par $(3 + 5)$ permet d'obtenir une écriture telle que a^{3+5}, avec un constat d'égalité $a^{3+5} = a^3 \cdot a^5$;

— on peut aussi de même substituer le nombre 3 dans a^3 par l'expression $2 \cdot 5$, pour aboutir aux expressions et à l'égalité $a^{2\cdot5} = (a^2)^5$;

— Newton substituera une lettre p au lieu du nombre, permettant d'établir des égalités génériques telles que $a^p \cdot a^q = q^{p+q}$. Notons qu'ici les symboles passent d'un statut de paramètre au statut d'indéterminée (ou de *nombre généralisé*, voir par exemple (Malisani et Spagnolo, 2009, p. 20)) ;

— la lettre étant elle-même substituable par une formule contenant des lettres, on a pu obtenir des écritures telles que a^{2x+y} ou $a^{x+\frac{1}{x}}$;

— ces écritures possibles amènent à donner des définitions de ce que sont ces nouveaux nombres, tels que $a^{\frac{p}{q}}$ où p et q sont des entiers, comme par exemple dans $a^{\frac{1}{2}}$ qui est une autre écriture du nombre $\sqrt{a}$;

— la lettre, dans a^x, est finalement substituable par un nombre quelconque, même irrationnel...

Ainsi on voit que les écritures proposées par Descartes, une fois la possibilité de substitution actée, vont étendre les écritures et le domaine des nombres. De même, avec ces nouvelles écritures, pour « le concept de polynôme, en place d'un lot d'exemples spécifiés, concept qui se clivera à son tour, chez Euler et Riemann, par le jeu d'une nouvelle différenciation, la distinction entre polynôme et fonction quelconque, toutes choses qui avaient été tout simplement impossibles en fait, sinon en droit, avant l'exponentielle cartésienne. » (Serfati, 1997, p. 249).

1.6.5 Bilan

Avec Viète et Descartes débute une révolution fondamentale pour l'algèbre. Les écritures ainsi que leurs transformations et substitutions permettront d'explorer d'autres domaines des mathématiques, de poser de nouveaux problèmes et de nouvelles preuves. Cela était impossible avant qu'on ne dispose d'une représentation dans θ du donné, du non connu, de ce qu'on nomme couramment le *paramètre*. Cette nouvelle écriture appelle une nouvelle épistémologie, et nous verrons dans la partie suivante que cela sera un des éléments tout aussi fondamental — voire fondateur — de l'algorithmique et de l'informatique.

1.7 Brève histoire de l'informatique

Dans cette partie, nous allons laisser de côté l'évolution de l'algèbre — qui dépasse nos préoccupations en matière d'algèbre élémentaire dans le cadre de son apprentissage — pour nous intéresser à l'informatique qui, selon nous, a réellement pu débuter son histoire grâce à la capacité à envisager la généralisation et la formaliser. Il ne s'agit pas de faire une histoire exhaustive de l'informatique, mais de souligner, dans l'histoire de l'informatique, ses évolutions fondamentales et les concepts associés, en relation avec l'algèbre. Nous aborderons aussi l'histoire de l'enseignement de l'informatique en France.

Pour un panorama plus complet de cette histoire mêlant mathématiques, technologie et même horlogerie, voir (Lazard et Mounier-Kuhn, 2022), (Piguet et Hügli, 2004), (Laguës, Beaudouin et Chapouthier, 2017) pour les aspects plus spécifiquement liés à la mémoire et au stockage, ou encore, concernant l'évolution des langages informatiques, les sites du Software Preservation Group [50] du Computer History Museum (Mountain View) ou le « Online Historical Encyclopaedia of Programming Languages » [51] de D. Pigott. Le travail considérable de Ricquebourg (2008) dans son « Archéologie de l'informatique » constitue aussi une référence.

1.7.1 Pascal, algorithme et machine

Avant Pascal, différents algorithmes existaient afin de permettre de déterminer la divisibilité d'un nombre par un nombre sans effectuer la division (Chabert, 2010, p. 278-280). Le plus connu est sans doute celui permettant de déterminer la divisibilité par 9 d'un nombre quelconque en additionnant les chiffres qui les composent : 2745 est divisible par 9 car $2 + 8 + 4 + 5 = 18$ et 18 est divisible par 9. Pascal précisera :

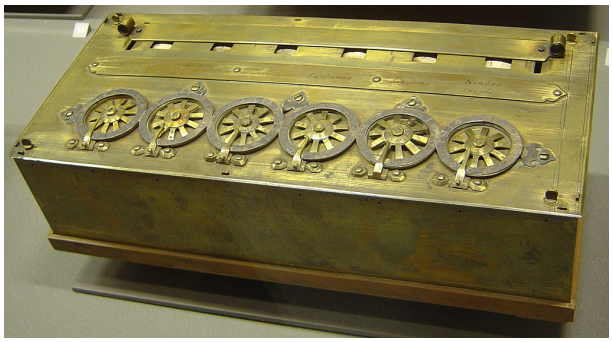
> Bien que cette règle soit communément employée, je ne crois pas que personne jusqu'à présent en ait donné une démonstration ni ait cherché à en généraliser le principe. Dans ce petit traité, je justifierai le caractère de divisibilité par 9 et plusieurs autres analogues ; j'exposerai aussi une méthode générale qui permet de reconnaître, à la seule inspection de la somme de ses chiffres, si un nombre donné est divisible par un nombre quelconque ; cette méthode ne s'applique pas seulement à notre système décimal de numération (système qui repose sur une convention, d'ailleurs assez malheureuse, et non sur une nécessité naturelle, comme le pense le vulgaire), mais elle s'applique encore sans défaut à tout système de numération ayant pour base tel nombre qu'on voudra, ainsi qu'on le verra dans les pages qui suivent (Pascal, 1665, p. 84-86)

Ainsi, Pascal veut non seulement prouver, mais aussi *généraliser* une méthode : il cherche un algorithme résolvant le problème *pour une famille d'instances de ce problème*, pour reprendre les termes utilisés dans la définition de l'algorithme selon (Modeste, 2012). Il indique ensuite l'objectif de son algorithme et son domaine de validité. On peut noter d'ailleurs une généralisation ambitieuse : il ne s'agit pas seulement de généraliser la méthode, il s'agit de généraliser la méthode et son domaine de validité, en l'étendant à tout système de numération. Cette volonté d'étendre la validité de l'algorithme au plus de cas possible est typique de la pensée informatique : il s'agit ici de penser à plusieurs niveaux d'abstraction, comme le dit Wing (2006, p. 34).

50. https://www.softwarepreservation.org/
51. http://hopl.info/

Dans sa méthode, Pascal mobilise des lettres pour représenter les différents chiffres : « un diviseur quelconque que nous représenterons par la lettre A », et « un dividende TVNM dans lequel les lettres M, N, V, T représentent respectivement les chiffres des unités simples, des dizaines, des centaines, des unités de mille, et ainsi de suite ». Il indique aussi comment procéder à une instanciation par une substitution : « pour passer des quantités littérales aux quantités numériques, il suffirait de remplacer chacune des lettres par l'un des 9 premiers nombres » [52]. Il explicite ensuite sa méthode, en la justifiant, en ne se basant que sur les lettres, avant de donner un exemple instancié. L'utilisation de symboles désignant des objets non connus *a priori* semble permettre ainsi non seulement de formaliser une méthode générique, mais sans doute ainsi d'*envisager la possibilité de généraliser* une méthode.

Pascal dispose ainsi des outils de pensée permettant d'envisager la généralisation et de la formuler, en manipulant des symboles désignant des objets (ici des chiffres) non connus *a priori*, et il considère les chiffres eux-mêmes comme les composantes d'une écriture d'un nombre dans une certaine base. Cela a sans doute rendu possible une des réalisations majeures de Pascal : la Pascaline (figure 1.13, p. 67). La Pascaline est reconnue comme étant la première « machine



(a) Pascaline (CC-BY-SA D. Monniaux)

* ARITHMÉTIQUE, (*machine.*) c'eſt un aſſemblage ou ſyſtème de roues & d'autres pieces, à l'aide deſquelles des chiffres où imprimés ou gravés ſe meuvent ; & exécutent dans leur mouvement les principales regles de l'*Arithmétique*.

La premiere *machine arithmétique* qui ait paru, eſt de Blaiſe Paſcal, né à Clermont en Auverne le 19 Juin 1623 ; il l'inventa à l'âge de dix-neuf ans. On en

(b) Définition de la Pascaline dans l'*Encyclopédie*(Diderot, 1751-1765, p. 680) source Gallica/BnF

Figure 1.13 – La machine Arithmétique de Pascal, ou Pascaline

arithmétique » construite : « assemblage d'un système de roues et d'autres pièces, à l'aide desquelles des chiffres ou imprimés ou gravés se meuvent et exécutent dans leur mouvement les principales règles de l'arithmétique » (figure 1.13b, p. 67). Autrement dit, c'est une machine capable de réaliser automatiquement des additions, soustractions, multiplications et divisions. Elle dispose de roues permettant d'entrer les différents chiffres, d'un afficheur du résultat, et d'une languette mobile permettant de basculer l'affichage de l'addition à la soustraction et inversement. Elle est aussi la seule qui soit décrite dans l'*Encylopédie* de Diderot, sur quatre pages complètes (Diderot, 1751-1765, p. 681-684) [53]. Cette création est donc la première machine à calculer mécanique réalisée. Wihelm Schikard (1592-1635) en aurait construite une en 1623, à base d'engrenages et de batonnets de Neper, mais il n'existe pas d'exemplaire connu (Piguet et Hügli, 2004, p. 33). En outre, la propagation de la retenue par des engrenages implique des limites physiques telles que le report multiple de retenues risquait de casser les engrenages. Pascal résoudra ce problème en utilisant un sautoir (figure 1.14, p. 68).

L'objectif de Pascal était en premier lieu de l'aider, ainsi que son père, à faire les nombreux calculs nécessaires à l'administration d'un état moderne :

52. Ici, le « 9 » est un nombre générique définissant le système de numération.

53. Contrairement à ce que l'on peut lire couramment, elle n'est cependant pas la seule à être nommée dans cette Encyclopédie, qui évoque les constructions plus tardives des machines arithmétiques de M. de l'Epine (1725) et de M. Boitissendeau « dont l'Académie fait aussi l'éloge » (Diderot, 1751-1765, p. 684).

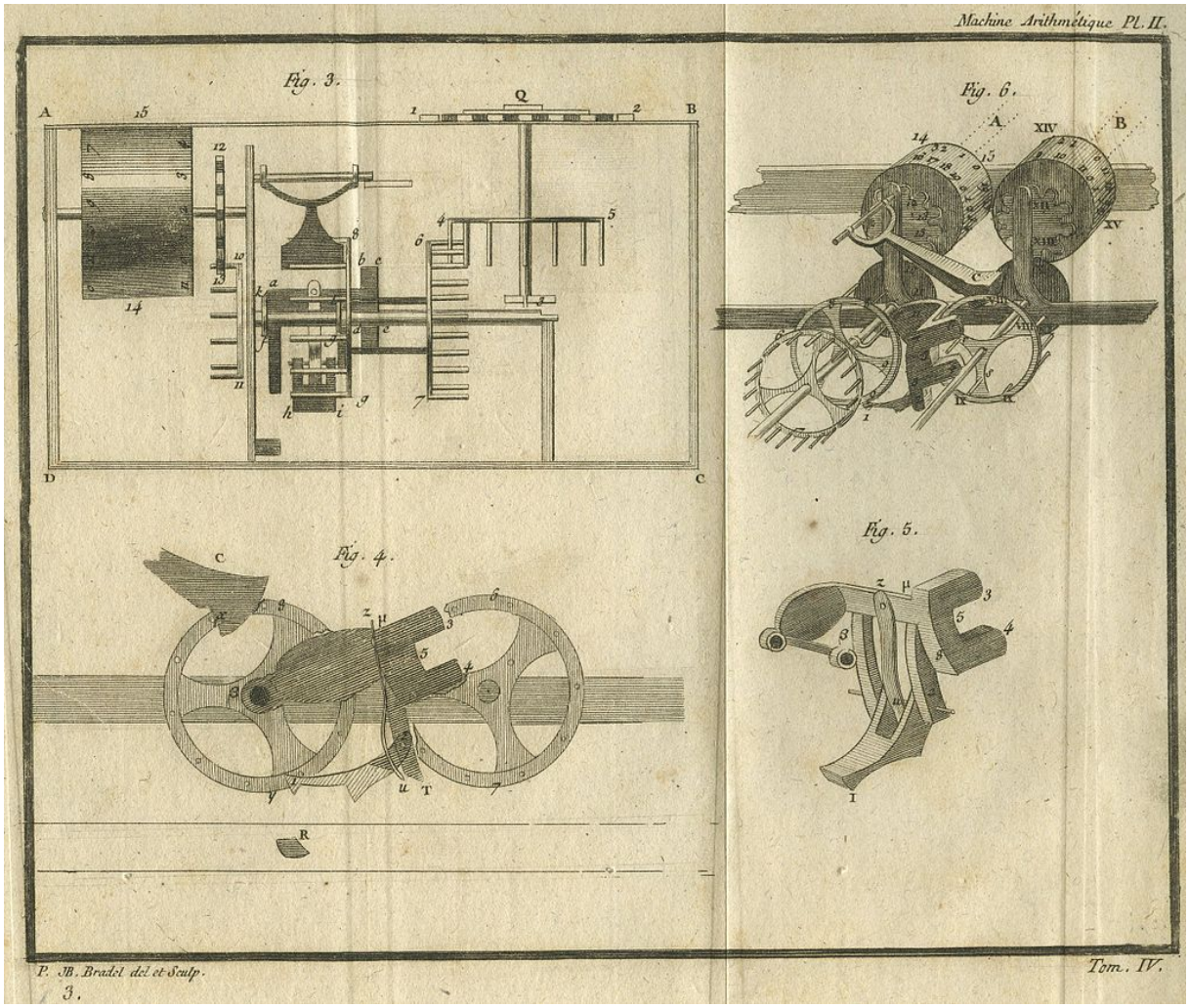


Figure 1.14 – Détail du mécanisme et du sautoir de la Pascaline (Wikicommons)

> Les longueurs et les difficultés des moyens ordinaires dont on se sert m'ayant fait penser à quelque secours plus prompt et plus facile pour me soulager dans les grands calculs où j'ai été occupé depuis quelques années en plusieurs affaires qui dépendent des emplois dont il vous a plu, honorer mon père pour le service de sa majesté en la Haute Normandie (Pascal, 2016a, p. 1000)

Pascal veut aussi palier les difficultés liées au calcul « au jeton ou à la plume » :

> [...] j'expose au public une petite machine de mon invention, par le moyen de laquelle seule tu pourras, sans peine quelconque, faire toutes les opérations de l'arithmétique, et te soulager du travail qui t'a souventes fois fatigué l'esprit, lorsque tu as opéré par le jeton ou par la plume (Pascal, 2016b, p.1003)

Les calculs au jeton et à la plume font référence aux deux modes de calculs concurrents, entre abacistes et algébristes : le calcul en utilisant des abaques (ou des jetons), et le calcul posé. Pascal veut ainsi « [suppléer] au défaut de l'ignorance ou du peu d'habitude » et supprimer les erreurs relevant d'une mauvaise application des procédures ou du « défaut de la mémoire » de ces méthodes (ibid., p.1005) : les erreurs sont fréquentes, notamment car il faut mémoriser certains résultats, et

il faut avoir une bonne connaissance de la procédure de calcul. En somme, Pascal veut débarrasser l'utilisateur de toute charge cognitive liée aux calculs, il veut *supprimer la pensée*[54] : une démarche logique et réfléchie, dépendant de connaissances et capacités du dispositif d'exécution humain, devient une démarche automatisée et non pensée, donc exécutable par une machine.

De plus, Pascal, toujours dans sa préoccupation de l'utilisateur (et non plus dans un objectif d'apprentissage), souhaite réaliser une machine « commode » à utiliser, tout en étant efficace et rapide, même si pour cela elle doit être plus complexe à concevoir :

> Ainsi que je n'ignore pas que la machine ne peut être moins composée, et particulièrement si j'eusse voulu instituer le mouvement de l'opération par la face antérieure, ce qui ne pouvait être qu'avec une incommodité ennuyeuse et insupportable ; au lieu que maintenant il se fait par la face supérieure avec toute la commodité qu'on saurait souhaiter, et même avec plaisir : tu leur diras aussi que mon dessein n'ayant jamais visé qu'à réduire en mouvement réglé toutes les opérations de l'arithmétique, je me suis en même temps persuadé que mon dessein ne réussirait qu'à ma propre confusion, si ce mouvement n'était simple, facile, commode et prompt à l'exécution, et que la machine ne fût durable, solide, et même capable de souffrir sans altération la fatigue du transport ; (ibid., p.1004-1005)

Enfin, il considère que sa machine ne doit pas nécessiter d'explications complexes sur sa conception et son fonctionnement, et que son usage doit s'approprier facilement :

> Oui, j'espère que tu approuveras que je me sois abstenu de ce discours, si tu prends la peine de faire réflexion d'une part sur la facilité qu'il y a d'expliquer de bouche, et d'entendre par une brève conférence, la construction et l'usage de cette machine ; et d'autre part, sur l'embarras et la difficulté qu'il y eût eu d'exprimer par écrit les mesures, les formes, les propositions, les situations et le surplus des propriétés de tant de pièces différentes. (ibid., p.1004)

On peut par ailleurs noter, que cette machine est une des premières machines produisant de l'information en fonction de données entrées par l'utilisateur. Comme le soulignent Lazard et Mounier-Kuhn (2022, p. 30), d'autres machines produisent de l'information, les horloges :

> Les horloges sont des mécanismes automatiques où pratiquement toute l'énergie est transformée en information — information sur l'écoulement du temps et, dans le cas des horloges astronomiques, sur des phénomènes périodiques. Les mécaniciens du Moyen Âge ont inventé des automates produisant de l'information. Leurs successeurs des temps modernes construiront les premières « horloges à calcul »

Mais ces horloges ne produisent pas de l'information en fonction d'une certaine valeur non définie *a priori* : ce sont des automates. En revanche, l'odomètre d'Archimède (figure 1.15, p. 70), par exemple, est une machine produisant de l'information dépendant du contexte d'exécution. Le principe de cet odomètre et de ses variantes est, grâce à un système d'engrenages, de faire tomber une bille ou un caillou dans un réceptacle, lorsqu'une certaine distance fixe est parcourue, c'est-à-dire à chaque fois que la roue fait un certain nombre de tours — la version la plus simple étant de faire tomber un caillou pour chaque tour, mais cela limite les distances mesurables. Cette version de l'odomètre met en œuvre l'algorithme 1.7.1 (p. 70). On a ici

54. Notons néanmoins que pour les multiplications et divisions, il était nécessaire d'appliquer une procédure particulière, nécessitant un ajout d'informations sur la machine avec des bandes de papier, ainsi que l'utilisation de tables de multiplication. Concernant la multiplication, l'utilisateur devait en outre appliquer une procédure particulière reprenant des éléments de la multiplication posée.

une utilisation concrète d'une mémoire (le réceptacle des billes ou cailloux) dans un rôle de compteur ou d'accumulateur (*stepper* ou *gatherer* chez Sajaniemi (2005, p. 7)). Ceci correspond à une implémentation pratique d'une définition classique de la variable informatique [55].

```
Données : le périmètre de la roue P_ROUE
Résultat : La distance parcourue
début
    compteur ← 0;
    répéter
        si tour de roue complet effectué alors
            compteur ← compteur + 1;
        fin
    jusqu'à mesure terminée;
    retourner compteur × P_ROUE
fin
```

Algorithme 1.7.1 : Algorithme implicite de l'odomètre d'Archimède(version simple : 1 caillou par tour de roue)

Figure 1.15 – Odomètre d'Archimède, III[e] av. J.-C. (source kotsanas.com)

Pascal a donc conçu une machine produisant de l'information de façon automatisée, en fonction de données entrées par l'utilisateur. Cette machine devait suppléer une activité cognitive, de façon efficace et dénuée d'erreur. Pascal ne cherchait pas à concevoir une machine *simple*, mais une machine *simple d'utilisation*, en considérant la machine comme une « boîte noire » dont la description du fonctionnement est inutile, seule la description de sa fonction l'étant.

On peut par ailleurs souligner que Pascal a mis en œuvre des techniques algorithmiques encore utilisées, techniques algorithmiques issues de la conception même de la machine. Ainsi, pour la soustraction, Pascal conserve la même mécanique, mais en modifiant la représentation des données. En effet, Pascal utilise une propriété particulière des systèmes de numération : dans une base n, $a - b = \overline{\overline{a} + b}$, avec $\overline{i}$ le complément à $(n-1)$ de i. Les roues de l'afficheur disposant de deux lignes de nombres complémentaires, si l'on fait afficher le nombre a avec la languette en position « soustraction », puis que l'on entre b, la machine ajoute $\overline{a}$ et b, et l'affichage avec la languette en position soustraction indique donc $\overline{\overline{a} + b}$, soit le résultat de $a - b$.

55. Bien entendu, l'algorithme proposé ici est une conception actuelle, il n'était pas formalisé.

On retrouve ainsi, dans la Pascaline et la démarche de Pascal, des questions ou principes spécifiques à l'informatique :

— le programme s'exécutant sur la machine est défini par sa fonction et le domaine de validité de ses entrées (des données) ;

— des questions qui relèvent de ce que l'on nomme à présent les « Interactions Homme Machine » sont déjà posées ;

— des questions d'efficacité et d'exactitude sont présentes ;

— des solutions innovantes sont créées pour concevoir un algorithme exécutable par la machine (par exemple la soustraction) ;

— la propagation d'un drapeau (*flag* dans les rôles des variables informatiques de Sajaniemi (ibid., p. 7), la retenue) en cascade se retrouve dans les circuits additionneurs ;

— des solutions mécaniques permettant l'entrée et la sortie des données et résultat sont créées :

  — l'entrée des valeurs se fait en introduisant un stylet dans un certain trou d'une certaine roue numérique, et en tournant cette roue jusqu'à ce que le stylet se trouve bloqué par un butoir, à la manière des cadrans de téléphones analogiques (figure 1.16, p. 71) ;

  — l'affichage prend en charge deux codages des nombres (le nombre et son complément) ;

(a) Roue d'inscription de la Pascaline

(b) Cadran téléphonique (source wikicommons CC-BY-SA C. Rizzo)

Figure 1.16 – Dispositif d'entrée de chiffres

La Pascaline, finalement, est une implémentation concrète de l'algorithme très simplifié 1.7.2 (p. 72) : la machine est une traduction opérationnelle de cet algorithme générique, elle résout une famille d'instances du problème « additionner ou soustraire une suite de nombres », les nombres valides dépendant du nombre de barillets présents. En revanche, elle ne résout pas directement les multiplications ou divisions, elle est impliquée dans l'algorithme de la multiplication ou de la division, tel un sous-programme.

Données : Le type d'opération Op ('+' ou '-'), une suite de nombres en chiffres $a, b_1, b_2, ..., b_n$
Résultat : Le résultat de a Op $\sum_{i=1}^{n} b_i$

début
  lire a ;
  si Op = *'+'* **alors**
    $v \leftarrow$ a
  fin
  sinon
    $v \leftarrow \bar{a}$
  fin
  répéter
    lire B ;
    $v \leftarrow v +$ B ;
    afficher v ;
  jusqu'à *entrée terminée*;
  retourner v ;
fin

Algorithme 1.7.2 : Algorithme implicite de la Pascaline(version simplifiée)

La machine arithmétique de Pascal va être suivie, jusqu'à l'avènement des machines électroniques, de nombreuses versions et évolutions, dont par exemple :

— Leibniz (1694) qui concevra une machine permettant d'effectuer les quatre opérations (Lazard et Mounier-Kuhn, 2022, p. 42). Il percevra aussi l'utilité du système binaire pour les calculs effectués par les machines, principe en vigueur de nos jours ;

— Thomas de Colmar (1851) créera « une machine à calculer à addition et à soustraction directes, qui permet de multiplier et de diviser rapidement par additions ou soustractions partielles, grâce à une platine mobile portant le résultat » (ibid., p. 47). Cet « Arithmomètre » sera vendu à plusieurs milliers d'exemplaires (figure 1.17, p. 73) ;

— Schwilgué (1844) présentera un modèle « que l'on considère comme la première calculatrice à touches opérationnelle » (ibid., p. 53) ;

— en 1876 Tchebychev créera un additionneur ne nécessitant pas de report de la retenue, procédé qui permettra d'accélérer les traitements et qui sera utilisé jusqu'à la fin des années 1960 (ibid., p. 60) ;

— pour l'Exposition Universelle de 1889, Léon Bollée présentera une machine à calculer disposant de plaques représentant la table de Pythagore, ce qui peut s'apparenter « sous une forme matérielle, [à] ce qu'un informaticien d'aujourd'hui appellerait un "sous-programme" » (ibid., p. 63)

— ...

Toutes ces machines permettent d'effectuer une suite de calculs, mais en entrant chaque nouvelle valeur après la production du résultat, et non en entrant l'ensemble des valeurs avant le début de l'exécution. En outre, refaire la même suite de calcul nécessite d'entrer de nouveau chacune des valeurs. Jacquard, Babbage et Hollerith notamment utiliseront un système permettant de reproduire de façon économique une séquence d'instructions ou de données.

Figure 1.17 – *Arithmomètre* de Colmar (Science Museum, London, photo personnelle)

1.7.2 Jacquard et les cartes perforées

Au début du XIX^{e} siècle, Joseph-Marie Jacquard (1752-1834) construit « le premier métier à tisser automatique programmable à l'aide de cartes perforées » (ibid., p. 46). Il reprend ainsi des idées de B. Bouchon, J.-B. Falcon et J. Vaucanson. Les deux premiers ont travaillé à « la mise au point des cartes puis des cartons perforés »(Ricquebourg, 2008, p. 265) : ils ont permis d'étendre les possibilités des cylindres à picots utilisés dans les automates — puisqu'en permettant virtuellement une séquence infinie, contrairement au cyclindre —, tout en facilitant la production et en la rendant plus économique (ibid., p. 265). Vaucanson, quant à lui, est célèbre par ses automates qui reproduisent le vivant : *le joueur de flûte*, ou le *canard* par exemple. Il cherchait à améliorer les métiers à tisser de Bouchon, et aurait promis à des ouvriers mécontents, selon Laguës, Beaudouin et Chapouthier (2017, p. 196), « Vous prétendez être les seuls à pouvoir exécuter ce dessin... Eh bien, je le ferai faire par un âne ! » [56], illustrant tout à la fois cette suppression de la pensée que nous évoquions pour la Pascaline, et, déjà, les problèmes sociaux créés par l'automatisation. Notons que parmi les automates célèbres, *L'écrivain* (1772) de Jacquet-Droz se distingue par sa capacité à utiliser des données produites par l'utilisateur. Cet automate trace des lettres et symboles (40) grâce à des cames, et dispose d'un mécanisme permettant d'entrer le texte que l'on veut faire écrire (Piguet et Hügli, 2004, p. 28-29). *Le Dessinateur* (1774), quant à lui, était limité au tracé de quatre dessins. En ce sens, *L'Écrivain* est un automate bien plus générique.

Jacquard va améliorer les systèmes de Bouchon et Vaucanson : son métier à tisser est considéré comme étant « programmable », puisque les cartes perforées traduisent un motif en un « programme », la « suite d'enfilage des aiguilles correspondant aux différentes couleurs » (Lazard et Mounier-Kuhn, 2022, p. 46). Sur ces cartons perforés, un trou permettait laisser monter une tige reliée aux fils que l'on souhaitait passer au-dessus, une absence de trou laissant en dessous les fils reliés à cette tige [57]. Cet artefact permet ainsi de conserver, de mémoriser, la suite d'instructions nécessaires à la création d'un certain motif. Si le « programme » est exact, s'il permet la conception du motif attendu, alors tous les motifs issus des bandes perforées représentant ce programme seront identiques, et exacts. La mémorisation des instructions permet une réutilisation ultérieure en limitant le risque d'erreur. De plus, cette automatisation et cette capacité à réitérer un même motif améliorent l'efficacité de la production.

56. Ricquebourg (2008, p. 266), quant à lui, affirme que Vaucanson aurait affirmé que grâce à ses machines « un cheval, un bœuf, un âne [faisaient] des étoffes bien plus belles et plus parfaites que les ouvriers en soye. »

57. Une description de ce mécanisme est donnée dans (Menabrea, 1842, p. 677-678), texte célèbre décrivant la Machine Analytique (Analytical Engine) de C. Babbage, dont nous parlerons amplement plus loin.

Les bandes ou cartons perforés, tout comme les cylindres à picots, sont constitués de mots ou d'instructions « interprétables » par la machine : chaque ligne est une instruction précisant les actions effectuées par la machine. On est en train de passer de la recherche de langage exprimant les algorithmes pour qu'ils soient exécutables par des humains, à la recherche de langage permettant l'exécution par une machine. Recherche, nous le verrons, qui se poursuivra jusqu'à aboutir aux langages de programmations modernes et aux ordinateurs.

La machine de Jacquard serait ainsi la première machine programmable, avec un stockage de programme externe. On peut néanmoins relativiser ces affirmations.

Considérons en premier lieu le « langage » utilisé ici. Pour le métier Jacquard, il s'agit de transformer une partie d'un motif en une instruction permettant à la machine de tisser cette partie. Si l'on prend l'exemple plus simple de la boite à musique, chaque ligne de picots indique quels sons doit produire la machine. Pour une boite à musique disposant d'un peigne de quatre lames, par exemple, le mot « picot ; pas picot ; pas picot ; picot », lorsqu'il sera « lu », produira la vibration des lames 1 et 4. De même, pour un métier à tisser de quatre fils, « trou ; pas trou ; pas trou ; trou » indique que les fils regroupés sur les tiges 1 et 4 doivent être tirés, les autres non. Il s'agit finalement d'une programmation binaire : ce mot pourrait ainsi s'écrire 1001 (ou 0110). En notation BNF [58] :

<Action> : := 0|1
<Instruction> : := <Action><Action><Action><Action>
<Programme> : := <Instruction>|<Instruction><Instruction>

Pour une boite à musique dont le peigne est constitué de lames produisant les sons Do, Ré, Mi, Fa : « Jouer les notes Ré et Fa » se traduirait en « 0101 », ce qui pour la machine aboutira à l'action « faire vibrer la lame 2 et la lame 4 ». Il s'agit donc ici d'une forme de langage très simple. On peut d'ailleurs se demander si la qualification de « langage » n'est pas abusive, notamment concernant le métier Jacquard : il s'agit finalement d'un dispositif mécanique, laissant passer ou non une tige.

Ricquebourg (2008, p. 266) se demande d'ailleurs si ces dispositifs de contrôle (arbres à cames, cylindres à picots ou cartes perforées) sont réellement des programmes, soit des « dispositifs de contrôle par programme ». Il considère finalement que ces dispositifs sont des « dispositifs de contrôle par séquence » :

> De façon générale, un automate contrôlé par séquence peut se voir défini de la sorte : 1) c'est un dispositif spécialisé dans l'effectuation d'un type unique de processus : la tâche qu'il est en mesure de réaliser est donc toujours la même ; 2) la sortie de la machine ne peut servir à influencer l'ordre dans lequel se présentent ses instructions ni non plus leur nature. (ibid., p. 326)

Autrement dit, le dispositif de contrôle par séquence ne met pas en œuvre les concepts algorithmiques de variable ou de condition. Outre le fait que *L'Écrivain* ne réalise pas toujours la même chose, puisque son action dépend de l'entrée faite par l'utilisateur, nous considérons que cette différence entre contrôle par séquence et contrôle par programme, malgré son intérêt, est peu pertinente d'un point de vue didactique. Notamment, le mélange d'un terme issu de l'algorithmique (séquence) avec un terme lié à l'informatique (programme, mise en mot d'un algorithme dans un certain langage pour une certaine machine) introduit une confusion fâcheuse. Un algorithme peut n'être constitué que d'une séquence d'instructions, la mise en mot de cette séquence d'instructions consiste donc bien en un programme.

58. Forme de Backus-Naur, voir https://fr.wikipedia.org/wiki/Forme_de_Backus-Naur

Cependant, le codage inscrit sur ces dispositifs est-il le codage d'un programme, ou de données alimentant un programme ? Si l'on voit la boîte à musique comme Piguet et Hügli (2004, p. 32), pour qui le peigne est une unité de traitement et le cylindre la mémoire, alors l'unité de traitement exécute les instructions contenues en mémoire, c'est donc un programme. Mais si l'on considère que ces machines visent à résoudre une famille d'instances de problèmes (tracer des motifs par tissage, produire des mélodies, produire un message écrit...), elles sont des réalisations concrètes de l'algorithme résolvant ce problème et le codage mémorisé est en fait l'entrée du programme, ce qui définit l'instance du problème résolu. Le codage peut être vu comme une séquence d'instructions (tirer tel ou tel fil, faire vibrer telle ou telle lame), ou comme une séquence de données (quel fil tirer, quelle lame faire vibrer). La spécialisation de ces machines engendre sans doute cette confusion, mais cela pointe aussi un élément caractéristique de l'informatique, tout du moins dans la version d'une architecture de Von Neumann : programme et données peuvent se confondre en mémoire, et un programme peut être une donnée de lui-même. Cet aspect fondamental est souligné par Wing (2006, p. 33) :

> Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code.

Le stockage externe, sur bande, du programme ou des données, sera mobilisé plus tard par Charles Babbage pour sa Machine Analytique, calculateur mécanique souvent considéré comme le premier ordinateur.

1.7.3 Babbage, Lovelace et les premières écritures de programme

Machine à différence

En 1822, Charles Babbage (1791-1871) « présenta à la Royal Astronomical Society les plans d'une machine à différences finies » (Lazard et Mounier-Kuhn, 2022, p. 48). Son objectif était notamment de supprimer les multiples erreurs présentes dans les tables mathématiques. Ces tables de résultats mathématiques étaient très utilisées : comme Lazard et Mounier-Kuhn (ibid.), Ricquebourg (2008, p. 275), souligne la nécessité, notamment pour les puissances maritimes telles que l'Angleterre, de disposer de tables de résultats mathématiques exacts, qui « constituaient par conséquent un enjeu réellement crucial (économique, militaire, scientifique), tant pour les Nations que les individus ». Or, les erreurs — de calcul ou de copie — étaient très nombreuses : Ricquebourg (ibid., p. 275) rapporte ainsi que dans des tables de multiplications très utilisées (pour des produits allant jusqu'à 100×1000), on pouvait trouver « jusqu'à quarante erreurs par page ». Babbage chercha donc un moyen d'automatiser cette tâche de productions de résultats, et il conçut les plans de sa première *machine à différence* capable d'évaluer les valeurs de polynômes jusqu'au degré 6. Le principe, dont Babbage n'est pas l'inventeur (Piguet et Hügli, 2004, p. 36), permettait de ramener une progression polynomiale à une progression arithmétique. Par exemple, pour le polynôme $F(x) = 2x+3$, la différence entre deux valeurs successives ($x \in \mathbb{N}$) est toujours de 2 : $F(x+1) - F(x) = 2(x+1) + 3 - (2x+3) = 2$. On pouvait ainsi aisément calculer les valeurs successives de $F(x)$ au moyen d'une simple addition. Pour un polynôme de degré 2, il faut aller à la deuxième différence, etc [59]. La machine à différence implémentait directement l'algorithme permettant de déterminer n'importe quelle valeur d'un des polynômes représentables, en décomposant une opération complexe en une suite d'opération simples, là encore procédé typique de la pensée informatique :

59. Voir l'exemple donné dans (Piguet et Hügli, 2004, p. 37) ou celui de (Ricquebourg, 2008, p. 286-287).

> Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively (Wing, 2006, p. 33)

Cette nécessité typiquement informatique de devoir abstraire et décomposer un problème est notamment liée aux contraintes liées aux machines. Comme le dit Wing (2011, p. 20), la pensée informatique relève et de la pensée mathématique et de la pensée technologique, puisque les systèmes informatiques sont contraints par les lois de la physique :

> Computational thinking draws on both mathematical thinking and engineering thinking. Unlike mathematics, however, our computing systems are constrained by the physics of the underlying information-processing agent and its operating environment.

Il n'existe pas à cette époque de moyen technologique permettant de calculer directement $2x + 3$ pour une certaine valeur de x, mais il existe des moyens (mécaniques) permettant de procéder à des opérations arithmétiques simples.

Cette première machine fût vite abandonnée par Babbage, avant même sa mise en fabrication. Ce dernier s'attela à une version évoluée, la *machine à différence numéro 2* : celle-ci devait introduire une nouveauté fondamentale, la possibilité « de faire en sorte que les valeurs représentées sur l'arbre des résultats puissent être automatiquement réutilisées pour modifier celles représentées par les roues numériques du dernier axe des différences. » (Ricquebourg, 2008, p. 300). Elle disposait en outre d'un dispositif d'impression des résultats plus évoluée (voir figure 1.18, p. 76). Cette seconde machine à différence, elle non plus, ne fût pas construite du vivant de

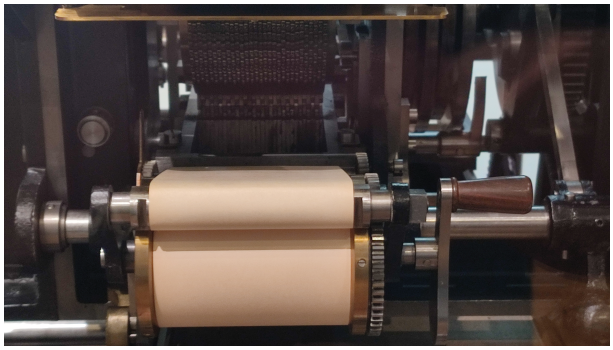


Figure 1.18 – Dispositif d'impression de la *Machine à différence n°2* Science Museum, London, photo personnelle

Babbage. Scheutz père et fils (1843) en construisirent une version simplifiée (de 5 digits et trois différences, puis de 15 digits et quatre différences, voir figure 1.19, p. 77) Le Science Museum de Londres, pour le centenaire de la mort de Babbage, fit construire une version fonctionnelle de la *machine à différence n°2* (figure 1.20, p. 78).

Machine Analytique

Vers 1840, Babbage, poursuivant ses recherches, proposera une machine plus générique (et théoriquement plus économique). En effet, sa *Machine à différence* était finalement un calculateur simple, capable de résoudre une famille de problèmes issue d'un théorème particulier de l'analyse, et non un calculateur générique. C'est ce que rappelle Menabrea (1842, p. 674) dans son texte relatant les informations données par Babbage lors d'une conférence en 1840 :

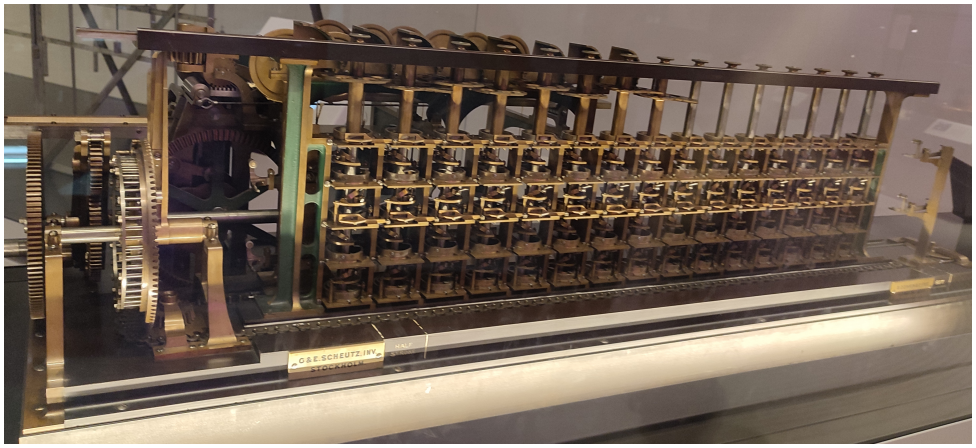


Figure 1.19 – Machine à différence de Georg et Edvard Scheutz, 1859 (Science Museum, London, photo personnelle)

> We see that its use is confined to cases where the numbers required are such as can be obtained by means of simple additions or subtractions ; that the machine is, so to speak, merely the expression of one particular theorem of analysis ; and that, in short, its operations cannot be extended so as to embrace the solution of an infinity of other questions included within the domain of mathematical analysis.

Babbage cherchera donc à concevoir une machine « dont les opérations devraient elles-mêmes posséder toute la généralité de la notation algébrique, et qu'il nomme, à ce titre, la machine analytique » [60]. Cette *Machine Analytique* (*Analytical Engine*, A.E. par la suite) a donc pour ambition de rendre automatisable des expressions ou suites de calculs algébriques. La généricité, pilier de l'algèbre, devient un objectif technologique.

Menabrea (ibid., p. 675) illustre le fonctionnement en indiquant que si l'on veut calculer le produit de deux binômes $(a+bx)(m+nx)$, soit $am+(an+bm)x+bnx^2$, il faut calculer d'abord am, an, bm, bn puis $an+bm$ et enfin distribuer les coefficients obtenus suivant les puissances de la variable. Il précise ainsi l'algorithme de calcul effectif. Menabrea précisera ensuite [61] :

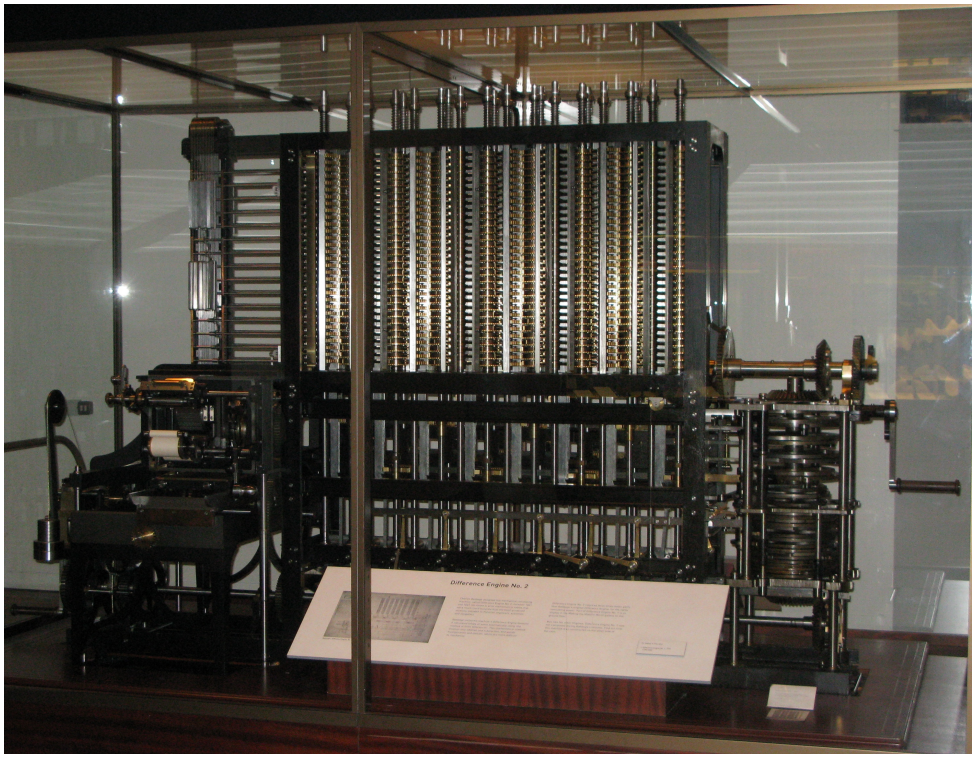
> But if human intervention were necessary for directing each of these partial operations, *nothing would be gained under the heads of correctness and oeconomy of time* ; the machine must therefore have the additional requisite of *executing by itself all the successive operations required* for the solution of a problem proposed to it, when once the primitive numerical data for this same problem have been introduced. Therefore, since from the moment that the nature of the calculation to be executed or of the problem to be resolved have been indicated to it, the machine is, by its own intrinsic power, of itself *to go through all the intermediate operations* which lead to the proposed result, *it must exclude all methods of trial and guess-work*, and can only admit the direct processes of calculation. (ibid., p. 675)

Cet extrait souligne et rappelle plusieurs points essentiels concernant l'informatique :

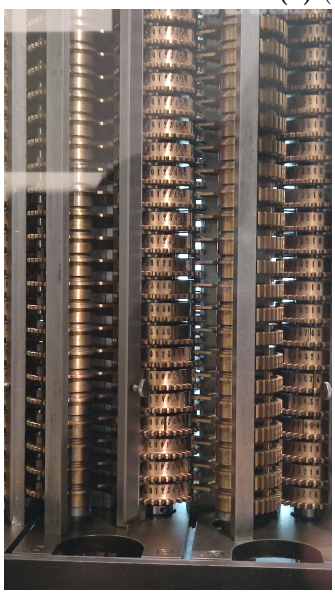
— la recherche d'économie et d'exactitude ;

60. « Mr.Babbage, renouncing his original essays, conceived the plan of another system of mechanism whose operations should themselves possess all the generality of algebraical notation, and which, on this account, he denominates the Analytical Engine » (Menabrea, 1842, p. 674).

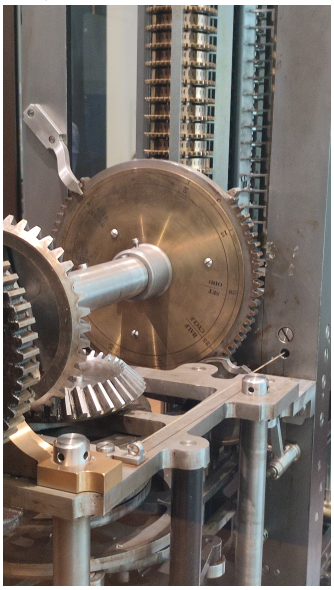
61. Nous soulignons.



(a) (source Wikicommons, CC-BY-SA geni)



(b) (Science Museum, London, photo personnelle)



(c) (Science Museum, London, photo personnelle)

Figure 1.20 – *Machine à différence n°2*, Science Museum, London

— la nécessité de supprimer les interventions humaines : il s'agit ainsi de *programmation* dans le sens « anticipation des actions nécessaires à la résolution d'un problème », et non de *pilotage*, qui consisterait, comme avec une simple calculatrice, à faire contrôler les actions et mémoriser les résultats par un humain ;
— le besoin de supprimer l'implicite, « de passer par toutes les opérations intermédiaires » ;

— il ne doit y avoir aucune forme de prise de décision non programmée (d'essais-erreurs) : on supprime la pensée du dispositif d'exécution.

D'autre part, l'organisation de la machine elle-même est proche de nos ordinateurs actuels, séparant mémoire et unité de calcul, et disposant de périphérique d'entrée/sortie. En effet, elle dispose :

1. d'une *mémoire* (*Storehouse*), capable de mémoriser 16 nombres de 33 chiffres digitaux. Ricquebourg (2008, p. 305) indique que plusieurs textes attestent de la volonté de Babbage de pouvoir étendre cette mémoire, peut-être même jusqu'à mille valeurs !
2. d'une *unité de traitement arithmétique* (*Mill*, le *moulin*), permettant les calculs avec les quatres opérations arithmétiques ;
3. un lecteur de cartes perforées, cartes inspirées des travaux de Jacquard. Ces cartes peuvent être soit des *Operations Cards*, soit des *Variables Cards*, soit des *Number Cards* (par exemple pour le nombre π) ;
4. un dispositif permettant d'imprimer les résultats ou de perforer des cartons, pour une réutilisation ultérieure de ces résultats comme données ;
5. des *Control Barrels*, qui implémentent physiquement les instructions lues — donc qui configurent le mécanisme de façon à ce qu'il opère effectivement l'opération demandée — et s'incrémentent automatiquement pour passer à l'instruction suivante. Ricquebourg (ibid., p. 307) y voit « une forme primitive de pointage permettant de réaliser, sur les modes matériel et positionnel, un type très élémentaire d'adressage direct »

A.E. et algèbre Menabrea, tout comme Ada Lovelace — qui traduira et commentera son texte —, considère en particulier que ce sont ces cartes perforées qui « donnent une généralité équivalente à celle d'une formule algébrique ». Menabrea affirme ainsi :

> the use of te cards offers a generality equal to that of algebraical formulæ, since such a formula simply indicates the nature and order of the operations requisite for arriving at a certain definite result, and similarly the cards merely command the engine to perform these same operations ; but in order that the mechanisms may be able to act to any purpose, the numerical data of the problem must in every particular case be introduced. Thus the same series of cards will serve for all questions whose sameness of nature is such as to require nothing altered excepting the numerical data. In this light the cards are merely a translation of algebraical formulæ, or, to express it better, another form of analytical notation. (Menabrea, 1842, p. 688)

Ada Lovelace, qui a accompagné de commentaires très pointus sa traduction du texte de Menabrea, précisera à ce propos dans sa note A :

> The distinctive characteristic of the Analytical Engine, and that which has rendered it possible to endow mechanism with such extensive faculties as bid fair to make this engine the executive right-hand of abstract algebra, is the introduction into it of the principle which Jacquard devised for regulating, by means of punched cards, the most complicated patterns in the fabrication of brocaded stuffs ... We may say most aptly, that the Analytical Engine *weaves algebraical patterns* just as the Jacquard-loom weaves flowers and leaves (ibid., p. 696)

Elle ajoutera un peu plus loin que :

> The bounds of arithmetic were however outstepped the moment the idea of applying the cards had occurred ; and the Analytical Engine does not occupy common ground with mere "calculating machines." It holds a position wholly its own ; and the considerations it suggests are most interesting in their nature. In enabling mechanism to combine together general

> symbols in successions of unlimited variety and extent, a uniting link is established between the operations of matter and the abstract mental processes of the most abstract branch of mathematical science. A new, a vast, and a powerful language is developed for the future use of analysis, in which to wield its truths so that these may become of more speedy and accurate practical application for the purposes of mankind than the means hitherto in our possession have rendered possible. Thus not only the mental and the material, but the theoretical and the practical in the mathematical world, are brought into more intimate and effective connexion with each other [62]. (Menabrea, 1842, p. 697)

Cette généricité viendrait de la séparation entre les opérations et les opérandes : les *Operation-cards* déterminent l'opération (ou plus généralement l'action), et les *Variable-cards* (ainsi que les *Number-cards*) définissent les valeurs sur lesquelles opérer. Les *Operation-cards* devaient permettre aussi d'avancer ou de reculer dans la lecture des cartes d'instruction ou de valeurs — donc de mettre en place une structure de contrôle. Ada Lovelace évoque la possiblité de réaliser des boucles permettant de réutiliser des cartes ou des groupes de cartes. Dans l'exemple de sa note F, elle explique ainsi que trois *operation-cards* remplaceraient 330 cartes en cas de non-répétition, et des milliers ou des millions si on augmente le nombre d'inconnues. Ada Lovelace se préoccupait déjà d'efficacité ou d'ergonomie, et y compris pour les variables temporaires, dont elle estime le nombre moyen à trois par opération (ibid., p. 720-721). En outre, l'utilisation de ces cartes revient à produire un code contenant d'une part la représentation de l'opération, d'autre part la représentation du ou des opérandes, en différenciant variable — donc en codant la colonne du *Storehouse*, ce qu'on nommerait maintenant l'adresse —, et littéraux — donc en codant la valeur, techniques de codages utilisées un siècle plus tard pour représenter les instructions en langage machine.

D'un autre côté, les *Variable-cards* sont représentées de deux façons par Ada Lovelace, utilisées figure 1.21 (p. 85) et 1.22 [63] (p. 86).

Une première version, proche d'une représentation de la machine, permet à Lovelace d'expliciter le fonctionnement de la machine, du point de vue des *Variable-cards*, tout en marquant le lien, au travers du langage, avec l'algèbre. Dans la note B, on voit ainsi une première représentation, « pour représenter convenablement les colonnes de disques sur le papier » [64] (ibid., p. 702) :

V_1	V_2	V_3	V_4	&c.
◯	◯	◯	◯	&c.
0	0	0	0	
0	0	0	0	
0	0	0	0	&c.
0	0	0	0	
□	□	□	□	&c.

62. « Les limites de l'arithmétique ont cependant été dépassées dès que l'idée d'appliquer les cartes est apparue ; et la machine analytique n'occupe pas le même terrain que les simples "machines à calculer". Elle occupe une position qui lui est propre et les considérations qu'elle suggère sont très intéressantes de par leur nature. En permettant au mécanisme de combiner des symboles généraux dans des successions d'une variété et d'une étendue illimitées, un lien est établi entre les opérations du problème et les processus mentaux abstraits de la branche la plus abstraite de la science mathématique. Un langage nouveau, vaste et puissant est développé pour l'usage futur de l'analyse, dans lequel il sera possible de manier ses vérités de manière à ce qu'elles puissent être appliquées plus rapidement et plus précisément aux besoins de l'humanité que les moyens dont nous disposons jusqu'à présent ne l'ont permis. Ainsi, non seulement le mental et le matériel, mais aussi le théorique et le pratique dans le monde mathématique, sont mis en relation plus intime et plus efficace l'un avec l'autre. »

63. Versions postcript issues du site de John Walker — par ailleurs co-auteur d'AutoCAD—, https://www.fourmilab.ch/babbage/

64. Le *Storehouse* est constitué de colonnes, chacune représentant une valeur grâce aux disques (portant les chiffres de 0 à 9) représentant chacun un chiffre.

Elle explique alors sa notation. En premier lieu, les « V » permettent de faire référence simplement à n'importe quelle colonne, à l'écrit ou à l'oral, et sont numérotés pour distinguer les colonnes. La lettre « V » fait référence à ce qui, dans le texte de Menabrea, est appelé *Variable*, ou *Variable column* ou encore *column of Variable*. Lovelace précise alors :

> The origin of this appellation is, that the values on the columns are destined to change, that is to *vary*, in every conceivable manner. But it is necessary to guard against the natural misapprehension that the columns are only intended to receive the values of the *variables* in an analytical formula, and not of the *constants*. The columns are called Variables on a ground wholly unconnected with the *analytical* distinction between constants and variables. In order to prevent the possibility of confusion, we have, both in the translation and in the notes, written Variable with a capital letter when we use the word to signify a *column of the engine*, and variable with a small letter when we mean the *variable of a formula*. Similarly, Variable-cards signify any cards that belong to a column of the engine. (ibid., p. 704)

Ainsi, Lovelace fait d'une part la distinction entre la *variable* et la *constante*, d'un point de vue algébrique — notons qu'elle ne parle pas du rôle de la variable, inconnue, paramètre ou indéterminée —, et d'autre part la distinction entre la *Variable*, qui désigne la mécanique représentant une valeur qui est variable, que l'on peut changer — c'est-à-dire qu'une même *Variable*, une même colonne, peut représenter des valeurs différentes —, et la *variable* algébrique. Lovelace distingue donc ce qui relève du langage de ce qui relève de l'artefact.

La *Variable* peut tout aussi bien contenir la représentation d'une *variable* que d'une *constante*. Lovelace précise aussi que les *Variable-cards* ou les *Variables* sont divisées en trois classes :

1. celles dont les données sont inscrites (les Variables de données, *Data*) ;
2. celles qui doivent recevoir les résultats finaux (*Result Variable*) ;
3. et celles qui vont recevoir des données intermédiaires et temporaires, nécessaires pour travailler afin d'atteindre les résultats finaux (*Working Variable*)

En outre, les *Variable-cards* permettaient soit de représenter dans une *Variable-Column* une quantité numérique codée par la carte (initialisation) ou stockée dans un des registres du *Mill* (affectation), soit mettre à zéro une *Variable*, soit transférer la valeur d'un *Variable* pour être utilisée par le *Mill*. On retrouve ainsi ce qui caractérise les variables informatiques : variables d'entrée, de sortie, ou temporaire (de travail), et les opérations qui leur sont associées. En outre, Ada Lovelace souligne dans sa note D que ces *Variables* peuvent changer de statut : une variable d'entrée peut devenir à un moment donnée une variable de travail, de même qu'une variable de travail peut finalement devenir variable de sortie (par exemple une variable ayant un rôle de *most-recent holder* pour Sajaniemi (2002, p. 2)). Elle précise aussi que la ré-utilisation de certaines variables est nécessaire, car le nombre de variables de travail nécessaires pour expliciter un calcul devient rapidement supérieur au nombre de *Variables* disponibles.

Les cercles doivent contenir le signe (+ ou −) du nombre, et les 0 seront substitués par les différents chiffres du nombre (ici seuls quatre disques, quatre *digits*, sont représentés). Les carrés quant à eux sont prévus pour y inscrire un symbole générique ou une combinaison de symboles. Lovelace illustre tout ceci avec l'exemple de l'écriture de $a = 5$, $n = 7$, $x = 98$ dans ce langage écrit qu'elle définit pour la A.E. :

V_1	V_2	V_3	V_4	&c.
$\oplus$	$\oplus$	$\oplus$	$\oplus$	&c.
0	0	0	0	
0	0	0	0	
0	0	9	0	&c.
5	7	8	0	
$\boxed{a}$	$\boxed{n}$	$\boxed{x}$	$\square$	&c.

À partir de cet exemple, toute expression mobilisant a, x, n est représentable dans une *Result Variable*, par exemple :

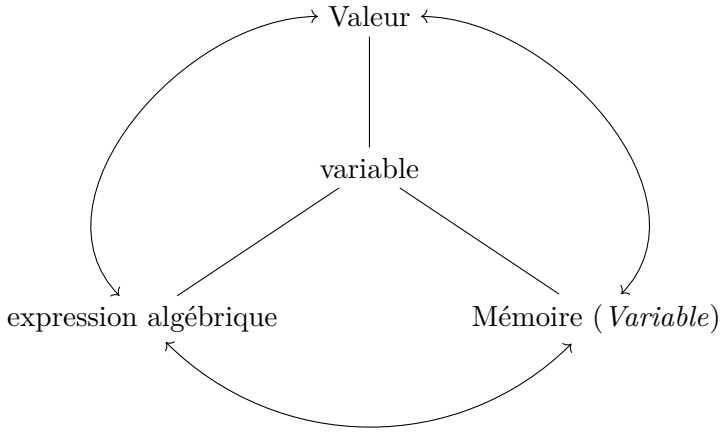
V_1	V_2	V_3	V_4	&c.
$\oplus$	$\oplus$	$\oplus$	$\oplus$	&c.
0	0	0	0	
0	0	0	0	
0	0	9	0	&c.
5	7	8	0	
$\boxed{a}$	$\boxed{n}$	$\boxed{x}$	$\boxed{ax^n}$	&c.

On voit ainsi apparaitre trois éléments liés aux variables :

1. un terme, un symbole du langage de description du problème pour la machine (V_3), qui réfère à un espace concret mémorisant une représentation de la valeur ;
2. la représentation codée de la valeur dénotée par ce symbole ($\begin{matrix}\oplus\\0\\0\\9\\8\end{matrix}$, soit la valeur 98) ;
3. le symbole ou l'expression algébrique qui est rendu effectif dans la machine (x).

Elle crée ainsi une chaine de référence : $x \longrightarrow V_3 \longrightarrow \begin{matrix}\oplus\\0\\0\\9\\8\end{matrix} \longrightarrow 98$

On pourrait représenter ces relations de la façon suivante :



C'est cette relation entre l'expression algébrique (éventuellement réduite à une lettre) et la *Variable* qui lui permet d'affirmer plus loin, dans sa note E :

> Many persons who are not conversant with mathematical studies, imagine that because the business of the engine is to give its results in *numerical notation*, the nature of its processes must consequently be *arithmetical* and *numerical*, rather than *algebraical* and *analytical*. This is an error. The engine can arrange and combine its numerical quantities exactly as if they were *letters* or any other *general* symbols ; and in fact it might bring out its results in algebraical *notation*, were provisions made accordingly. It might develope three sets of results simultaneously, viz. *symbolic* results (as already alluded to in Notes A. and B.), *numerical* results (its chief and primary object) ; and *algebraical* results in *literal* notation. This latter however has not been deemed a necessary or desirable addition to its powers, partly because the necessary arrangements for effecting it would increase the complexity and extent of the mechanism to a degree that would not be commensurate with the advantages, where the main object of the invention is to translate into *numerical* language general formulæ of analysis already known to us, or whose laws of formation are known to us. But it would be a mistake to suppose that because its *results* are given in the *notation* of a more restricted science, its *processes* are therefore restricted to those of that science. The object of the engine is in fact to give the *utmost practical efficiency* to the resources of *numerical interpretations* of the higher science of analysis, while it uses the processes and combinations of this latter. (Menabrea, 1842, p. 713)

Ainsi pour Lovelace, cette machine manipule des valeurs comme s'il s'agissait de symboles, et en algèbre on manipule des symboles comme s'ils étaient des valeurs, la boucle est bouclée. Elle souligne en fait que même si les résultats sont présentés sous forme numérique, les procédés (les algorithmes) sont d'ordre algébriques. Elle émet en outre l'hypothèse, avec beaucoup d'avance sur son temps, qu'il serait possible de produire des résultats sous forme algébrique.

Terminons cette rapide exploration du texte qui permet de dire que le premier informaticien était une informaticienne, la comtesse Ada Augusta de Lovelace (1815-1852), en abordant un point, là encore remarquable : la conception d'un algorithme comme un automate à états. En effet, dans sa note G, Ada Lovelace représente l'algorithme de calcul des nombres de Bernouilli sous la forme du diagramme de la figure 1.22 (p. 86). Ce diagramme est une écriture d'un algorithme, dans un langage spécifiquement conçu pour cette machine : c'est un programme. On remarque les dix-sept dernières colonnes, indiquant les différents rôles des *Variables* ainsi que leur sens algébrique. Juste avant se trouve la colonne indiquant l'expression algébrique du calcul en cours. Les trois colonnes précédentes (soit les colonnes 3 à 5, suivant les colonnes indiquant le nombre d'opérations exécutées et la nature des opérations) indiquent :

— l'opération et les *Variables* concernées : par exemple pour l'opération 10, ${}^{1}V_{21} \times {}^{3}V_{11}$;

— la ou les variables recevant le résultat : pour l'opération 10, ${}^{1}V_{12}$;

— les changements de valeurs des variables ;

La notation introduite par Lovelace, ${}^{m}V_{n}$ est explicitée dans sa note D (ibid., p. 708-709). V_n, comme on l'a déjà vu, représente la colonne des variables numéro n : « The *lower* indices are obviously indices of *locality* only, and are wholly independent of the operations performed or of the results obtained, their value continuing unchanged during the performance of calculations ». L'indice supérieur, m, indique « any *alteration* in the value which a Variable represents ». En cela, Lovelace indique un *changement d'état* : ${}^{m}V_{n}$ indique la valeur de V_n après sa $m^{\text{ième}}$ modification [65]. Faisant de nouveau la relation entre écriture mobilisant des variables algébriques et écriture

65. En réalité, cela est vrai seulement si la variable n'a pas été réinitialisée. En effet Ada Lovelace, dans la description de ses conventions d'écriture, précise « Whenever a value again gives place to zero, the Variable again becomes ${}^{0}V$, even if it have been ${}^{n}V$ the moment before ». En cas de nouveau changement, elle deviendra ${}^{n+1}V$.

mobilisant les *Variables* de la machine, elle précise qu'à la place de l'expression $V_n = V_p + V_n$, il est plus cohérent et plus clair d'écrire ${}^{m+1}V_n = {}^{q}V_p + {}^{m}V_n$ [66]. Elle différencie ainsi clairement l'objet concret, la mémoire et la représentation symbolique dénotant la valeur représentée dans cette mémoire. En outre cette question d'*état* est évoquée dans sa note C :

> The *Operation-cards* merely determine the succession of operations in a general manner. They in fact throw all that portion of the mechanism included in the *mill* into a series of different *states*, which we may call the *adding state*, or the *multiplying state*, &c. respectively. In each of these states the mechanism is ready to act in the way peculiar to that state, on any pair of numbers which may be permitted to come within its sphere of action. Only *one* of these operating states of the mill can exist at a time ; and the nature of the mechanism is also such that only *one* pair of numbers can be received and acted on at a time . (Menabrea, 1842, p. 704)

Les états sont ici des états physiques, concrets : le *additing state* indique que, suite à la lecture de l'opération, le *mill* est reconfiguré, mécaniquement, afin de pouvoir réaliser une addition. De même, les états des *Variables*, sont des états concrets. Cependant, on voit poindre ici une conception d'un algorithme comme étant un automate à états, avec l'ensemble des états et les règles permettant de passer d'un état à un autre, ce qui sera exploré ensuite par Turing. Mais Ada Lovelace va plus loin et écrit dans sa note A :

> In fact the engine may be described as being the material expression of any indefinite function of any degree of generality and complexity, such as for instance,
> $F(x, y, z, logx, siny, x^p, \&c.)$
> which is, it will be observed, a function of all other possible functions of any number of quantities. (ibid., p. 691)

Cette conception d'un programme (puisque ici Lovelace parle de la machine) comme étant une fonction sera reprise par Church un siècle plus tard, avec le λ-calcul.

Ainsi, Charles Babbage cherchait à concevoir un calculateur « universel » potentiel, et Ada Lovelace a construit pour cette machine potentielle un langage de description de l'algorithme, dénotant les cartes à entrer pour implémenter le programme : dans la description de l'algorithme de calcul des nombres de Bernouilli (figure 1.22, p. 86), la colonne « Nature of Operation » permet de définir l'*Operation-card* nécessaire, et les colonnes « Variables acted upon », « Variables receiving results » et « Indication of change in the value of any Variable » permettent de définir les *Variable-cards* ainsi que leur type (transfert vers le *mill*, transfert du *Mill* vers la colonne *Variable*, ou remize à zéro de la colonne *Variable*) [67]. Cette forme de langage permet aussi de décrire l'algorithme d'un point de vue algébrique, donc lisible par un humain, en considérant ${}^{m}V_n$ comme équivalent à une variable algébrique [68]. On a donc ici trois niveaux de langage : celui de l'algèbre, spécifique à l'humain, celui de la définition des cartes à utiliser, interface entre l'humain et la machine, et celui des cartes (le codage éffectif des opérations ou des *Variables*), spécifique à la machine.

66. Knuth et Pardo (1976, p. 8) reprendront d'ailleurs ces arguments en affirmant, en parlant des représentations d'algorithme pour la machine de Turing ou du λ-calcul de Church, « There was no concept of assignment (i.e., of replacing the value of some variable by a new value) ; instead of writing "$s \leftarrow -s$" one could write $s_{n+1} = -s_n$,, giving a new name to each quantity that would arise during a sequence of calculation. »

67. Par exemple pour l'opération 6, l'*operation-card* sera celle de la soustraction, les *Variable-cards* concerneront V_{13} (en lecture et en écriture), et V_{11}, en lecture avec remize à zéro. Notons que le signe « = » dans la colonne « Indication of change... » ne dénote pas une égalité entre le membre de gauche et le membre de droite, il s'agit plutôt d'une séparation marquant le changement d'état de la *Variable*, l'état final étant à droite

68. Pour l'opération 6, ${}^{1}V_{13} = -\frac{1}{2} \cdot \frac{2n-1}{2n+1}$. Lovelace crée alors la variable notée A_0, qui, si elle n'est pas une simple lettre, est une variable algébrique.

Number of Operations	Nature of Operations	Variables for Data						Working Variables									Variables for Results	
		${}^{1}V_0$	${}^{1}V_1$	${}^{1}V_2$	${}^{1}V_3$	${}^{1}V_4$	${}^{1}V_5$	${}^{0}V_6$	${}^{0}V_7$	${}^{0}V_8$	${}^{0}V_9$	${}^{0}V_{10}$	${}^{0}V_{11}$	${}^{0}V_{12}$	${}^{0}V_{13}$	${}^{0}V_{14}$	${}^{0}V_{15}$	${}^{0}V_{16}$
		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		m	n	d	m'	n'	d'										$\frac{dn'-d'n}{mn'-m'n} = x$	$\frac{d'm-dm}{mn'-m'n} = y$
1	×	m	....	....	....	n'	....	mn'										
2	×	....	n	....	m'	....	....	....	$m'n$									
3	×	....	....	d	....	....	....	....	....	dn'								
4	×	....	0	....	....	....	d'	....	....	....	$d'n$							
5	×	0	....	....	....	....	0	....	....	....	....	$d'm$						
6	×	....	....	0	0	....	....	....	....	....	....	.....	dm'					
7	−	....	....	....	....	....	....	0	0	....	....	.....	.....	$(mn' - m'n)$				
8	−	....	....	....	....	....	....	....	....	0	0	.....	.....	.....	$(dn' - d'n)$			
9	−	....	....	....	....	....	....	....	....	....	....	0	0	.....	.....	$(d'm - dm')$		
10	÷	....	....	....	....	....	....	....	....	....	....	.....	.....	$(mn' - m'n)$	0	.....	$\frac{dn'-d'n}{mn'-m'n} = x$	
11	÷	....	....	....	....	....	....	....	....	....	....	.....	.....	0	.....	0	.....	$\frac{d'm-dm'}{mn'-m'n} = y$

Figure 1.21 – Diagramme pour la note D, A. Lovelace (Menabrea, 1842, p. 711)

Number of Operation	Nature of Operation	Variables acted upon	Variables receiving results	Indication of change in the value on any Variable	Statement of Results	Data			Working Variables									Result Variables			
						$^1V_1$	$^1V_2$	$^1V_3$	$^0V_4$	$^0V_5$	$^0V_6$	$^0V_7$	$^0V_8$	$^0V_9$	$^0V_{10}$	$^0V_{11}$	$^0V_{12}$	$^0V_{13}\cdots$	$^1V_{21}$	$^1V_{22}$	$^1V_{23}$	$^0V_{24}\cdots$
						○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
						0	0	0	0	0	0	0	0	0	0	0	0	0	B in a dec. fract.	B in a dec. fract.	B in a dec. fract.	0
						0	0	0	0	0	0	0	0	0	0	0	0	0				0
						1	2	4	0	0	0	0	0	0	0	0	0	0				0
						1	2	n											B_1	B_3	B_5	B_7
1	×	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$\left\{\begin{matrix} {}^1V_2 = {}^1V_2 \\ {}^1V_3 = {}^1V_3 \end{matrix}\right\}$	$= 2n$		2	n	$2n$	$2n$	$2n$											
2	−	$^1V_4 - {}^1V_1$	$^2V_4$	$\left\{\begin{matrix} {}^1V_4 = {}^2V_4 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n - 1$	1			$2n-1$													
3	+	$^1V_5 + {}^1V_1$	$^2V_5$	$\left\{\begin{matrix} {}^1V_5 = {}^2V_5 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n + 1$	1				$2n+1$												
4	÷	$^2V_5 \div {}^2V_4$	$^1V_{11}$	$\left\{\begin{matrix} {}^2V_5 = {}^0V_5 \\ {}^2V_4 = {}^0V_4 \end{matrix}\right\}$	$= \frac{2n-1}{2n+1}$				0	0						$\frac{2n-1}{2n+1}$						
5	÷	$^1V_{11} \div {}^1V_2$	$^2V_{11}$	$\left\{\begin{matrix} {}^1V_{11} = {}^2V_{11} \\ {}^1V_2 = {}^1V_2 \end{matrix}\right\}$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$		2									$\frac{1}{2} \cdot \frac{2n-1}{2n+1}$						
6	−	$^0V_{13} - {}^2V_{11}$	$^1V_{13}$	$\left\{\begin{matrix} {}^2V_{11} = {}^0V_{11} \\ {}^0V_{13} = {}^1V_{13} \end{matrix}\right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$											0		$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - {}^1V_1$	$^1V_{10}$	$\left\{\begin{matrix} {}^1V_3 = {}^1V_3 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n - 1 (= 3)$	1		n							$n-1$							
8	+	$^1V_2 + {}^0V_7$	$^1V_7$	$\left\{\begin{matrix} {}^1V_2 = {}^1V_2 \\ {}^0V_7 = {}^1V_7 \end{matrix}\right\}$	$= 2 + 0 = 2$		2					2										
9	÷	$^1V_6 \div {}^1V_7$	$^3V_{11}$	$\left\{\begin{matrix} {}^1V_6 = {}^1V_6 \\ {}^0V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} = A_1$						$2n$	2				$\frac{2n}{2} = A_1$						
10	×	$^1V_{21} \times {}^3V_{11}$	$^1V_{12}$	$\left\{\begin{matrix} {}^1V_{21} = {}^1V_{21} \\ {}^3V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$											$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$		B_1			
11	+	$^1V_{12} + {}^1V_{13}$	$^2V_{13}$	$\left\{\begin{matrix} {}^1V_{12} = {}^0V_{12} \\ {}^1V_{13} = {}^2V_{13} \end{matrix}\right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$												0	$\left\{-\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}\right\}$					
12	−	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$\left\{\begin{matrix} {}^1V_{10} = {}^2V_{10} \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n - 2 (= 2)$	1									$n-2$							
13	−	$^1V_6 - {}^1V_1$	$^2V_6$	$\left\{\begin{matrix} {}^1V_6 = {}^2V_6 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n - 1$	1					$2n-1$											
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$\left\{\begin{matrix} {}^1V_1 = {}^1V_1 \\ {}^1V_7 = {}^2V_7 \end{matrix}\right\}$	$= 2 + 1 = 3$	1						3										
15	÷	$^2V_6 \div {}^2V_7$	$^1V_8$	$\left\{\begin{matrix} {}^2V_6 = {}^2V_6 \\ {}^2V_7 = {}^2V_7 \end{matrix}\right\}$	$= \frac{2n-1}{3}$						$2n-1$	3	$\frac{2n-1}{3}$									
16	×	$^1V_8 \times {}^3V_{11}$	$^4V_{11}$	$\left\{\begin{matrix} {}^1V_8 = {}^0V_8 \\ {}^3V_{11} = {}^4V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$								0			$\frac{2n}{2} \cdot \frac{2n-1}{3}$						
17	−	$^2V_6 - {}^1V_1$	$^3V_6$	$\left\{\begin{matrix} {}^2V_6 = {}^3V_6 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n - 2$	1					$2n-2$											
18	+	$^1V_1 + {}^2V_7$	$^3V_7$	$\left\{\begin{matrix} {}^2V_7 = {}^3V_7 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 3 + 1 = 4$	1						4										
19	÷	$^3V_6 \div {}^3V_7$	$^1V_9$	$\left\{\begin{matrix} {}^3V_6 = {}^3V_6 \\ {}^3V_7 = {}^3V_7 \end{matrix}\right\}$	$= \frac{2n-2}{4}$						$2n-2$	4		$\frac{2n-2}{4}$								
20	×	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$\left\{\begin{matrix} {}^1V_9 = {}^0V_9 \\ {}^4V_{11} = {}^5V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$									0		$\left\{\frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4}\right\} = A_3$						
21	×	$^1V_{22} \times {}^5V_{11}$	$^0V_{12}$	$\left\{\begin{matrix} {}^1V_{22} = {}^1V_{22} \\ {}^0V_{12} = {}^2V_{12} \end{matrix}\right\}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = B_3 A_3$											0	$B_3 A_3$			B_3		
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$\left\{\begin{matrix} {}^2V_{12} = {}^0V_{12} \\ {}^2V_{13} = {}^3V_{13} \end{matrix}\right\}$	$= A_0 + B_1 A_1 + B_3 A_3$												0	$\{A_0 + B_1 A_1 + B_3 A_3\}$				
23	−	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$\left\{\begin{matrix} {}^2V_{10} = {}^3V_{10} \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n - 3 (= 1)$	1									$n-3$							
Here follows a repetition of Operations thirteen to twenty-three																						
24	+	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$\left\{\begin{matrix} {}^4V_{13} = {}^0V_{13} \\ {}^0V_{24} = {}^1V_{24} \end{matrix}\right\}$	$= B_7$																	B_7
25	+	$^1V_1 + {}^1V_3$	$^1V_3$	$\left\{\begin{matrix} {}^1V_1 = {}^1V_1 \\ {}^1V_3 = {}^1V_3 \\ {}^5V_6 = {}^0V_6 \\ {}^5V_7 = {}^0V_7 \end{matrix}\right\}$	$= n + 1 = 4 + 1 = 5$ by a Variable-card. by a Variable-card.	1		$n+1$			0	0										

Figure 1.22 – Diagramme pour la note G, A. Lovelace (Menabrea, 1842, p. 730)

1.7.4 Zuse, binaire, logique et premier langage

Ces deux aspects de l'informatique, le côté « machine », matériel et concret, avec son lot de défis technologiques, et le côté « langage » permettant de décrire l'algorithme pour le faire-faire par un dispositif d'exécution (Samurçay et Rouchier, 1985), vont évoluer jusqu'à atteindre une forme de maturité que Knuth et Pardo (1976, p. 3) situent en 1957, « when the practical importance of algebraic compilers was first being demonstrated, and when computers just beginning to be available in large numbers. » Les ordinateurs sont à cette époque devenus des objets commerciaux depuis 1951 (Lazard et Mounier-Kuhn, 2022, p. 123-142), les technologies fondamentales sont en place (disque dur, mémoire, transistors), et la théorie aussi (langages « de haut niveau » — Algol, Fortran... —, *théorie des algorithmes* de A. Markov — une des fondations des théories des langages formels et des compilateurs [69]—, système d'exploitation...)

Avant d'atteindre cette maturité, le développement des machines a longtemps pris le pas sur le développement des langages : de nombreux calculateurs mécaniques, puis électro-mécaniques, verront le jour, la plupart sans stockage, destinés parfois à être utilisés dans des « fermes » de calculs, avec des opérateurs humains. Des calculateurs spécialisés verront aussi le jour, par exemple pour calculer les gains aux courses de chevaux (1913, ibid., p. 71), ou pour jouer aux échecs (Leonard Torres-Quavedo, 1920, ibid., p. 72). Des machines avec stockage externe se développeront néanmoins : notamment Herman Hollerith (1860-1929), qui concevra en 1890 une machine à compter, en partie électrique, en codant des informations sur des cartes perforées. Hollerith fondera plus tard l'entreprise Tabulating Machine Company, qui deviendra International Business Machine Inc. (IBM). Les cartes perforées continueront d'être utilisées, notamment à des fins statistiques, jusqu'à la fin des années 1970, et le standard introduit par IBM en 1928 (24 lignes de 80 caractères) sera utilisé pour les premiers terminaux textuels (ibid., p. 63-65).

Ce fort développement est lié selon Lazard et Mounier-Kuhn (ibid., p. 67) à la deuxième révolution industrielle. De nouvelles nécessités de calculs et de traitements apparaissent, pour l'économie, les transports, les « organisations spécifiques du travail », la gestion des flux et des stocks, ou encore dans le domaine militaire.

À côté du commerce de calculateurs mécaniques qui se développe, la recherche de calculateurs « universels » se poursuit, avec une forte accélération entre la fin des années 1930 et les années 1950, dopée par les exigences militaires [70]. Si des machines telles que le *Colossus* (1943) — conçut par Alan Turing et qui servira à déchiffrer les codes de la machine Enigma —, le *Mark I* (1944) ou le *ENIAC* (1945) sont célèbres, elles ne sont pas programmables par l'intermédiaire d'un langage, il faut modifier physiquement la machine. Konrad Zuse — dont les travaux ont été redécouverts après la seconde guerre mondiale —, travaillera lui aussi de son côté à l'élaboration de calculateurs universels, mais il verra les potentialités d'une évolution vers un ordinateur générique — et non plus seulement algébrique —, puis vers un ordinateur générique programmable :

> [...] it now turns out, been thought of earlier (i.e., binary mechanical arithmetic (Leibniz), program control (Babbage), instruction formats with numerical storage addresses (Ludgate) and floating point number representations (Torres y Quevedo)), Zuse's great achievement was to turn these ideas into reality (Randell, 1976, p. 7)

69. Rappelons qu'un compilateur est un programme permettant de transcrire automatiquement un langage « de haut niveau » en un langage exécutable par la machine.

70. Lazard et Mounier-Kuhn (2022, p. 67) donnent l'exemple des progrès de l'artillerie, ce qui nécessitera des calculs de trajectoire et de correction automatique complexes, par exemple pour des navires mouvants visant des cibles mouvantes à plusieurs dizaines de kilomètres.

Zuse utilisera aussi un stockage externe du programme (sur un film 35mm percé, qui facilite le contrôle de l'avancée des instructions), et il fut aussi un des premiers, en parallèle de Georges Stibitz (1904-1995), a concevoir l'utilité de calculs en binaire, s'inspirant ainsi du travail de Leibnitz :

> We know Konrad Zuse as an engineer who started developing program-controlled binary calculators from 1936 onward and completed the first fully operational digital computer in 1941. Zuse received the inspiration to use the binary number system from the Dyadik of G.W. Leibniz (1646–1716), realizing that the functions his binary machines had to perform could be described by logical expressions. This triggered in him a strong interest not only in propositional calculus but in mathematical logic in general (Giloi, 1997, p. 17)

Une de ses avancées fondamentales fut ainsi de concevoir le calcul arithmétique, en binaire, comme un calcul propositionnel. Son Z3, en 1941, est ainsi considéré comme le premier ordinateur numérique programmable opérationnel (Giloi, 1997, p. 18 ; Ricquebourg, 2008, p. 202 note 316). Il sépare mémoire (avec un adressage numérique), stockage et unité de traitement. Cette unité de traitement ne gère pas que les calculs arithmétiques, mais aussi logiques, c'est ce qu'on appelera plus tard une unité de traitement arithmétique et logique (UAL). La construction du Z4 amènera en effet Zuse à considérer la possibilité de construire une « machine logique » plus générique que ses machines algébriques : il s'est rendu compte que les éléments logiques mis en œuvre dans ses machines (ce qu'il appelait sa « combinatoire des conditionnels ») étaient identiques au calcul des prédicats :

> By his own account, Zuse very soon realized that his "combinatorics of conditionals" (has he called it) was identical to predicate calculus and he conceived a much more powerful machine, the "logic machine" which would be more general and surpersede the algebraic machine he had already built. (Rojas et al., 2000, p. 3)

Zuse cherchera alors, à partir de 1944, à construire non seulement un langage de description des algorithmes, comme l'avait fait Ada Lovelace, mais aussi un langage de description formel — et donc automatisable — qui permet aux programmeurs de travailler sur un algorithme écrit dans un langage lisible et (davantage) compréhensible. Il écrivit ainsi un mémo nommé *Plankalkül* :

> He [Zuse] used *Angaben* for data and *Forschrift* for algorithm. Not having at his disposition the word *Programm*, he called a program *Rechenplan*. The notational and conceptual system of expressing a *Rechenplan* he called *Plankalkül*. (Bauer et Wössner, 1972, p. 678)

On voit ainsi que Zuse mobilise des concepts informatiques essentiels : l'algorithme, les données, le langage, et le programme vu comme une description dans un langage formel d'un algorithme écrit pour une certaine machine et manipulant certaines données. Son objectif, selon la traduction qu'en font Knuth et Pardo (1976, p. 11) était le suivant :

> The mission of Plancalculus is to provide a purely formal description for any computational procedure.

Pour atteindre cet objectif, il se focalisera sur les problèmes logiques plutôt que sur les problèmes de calculs (Giloi, 1997, p. 18). Il aboutira à un langage qui met en œuvre ce qui deviendront les principes et fonctionnalités des langages de haut niveau (Rojas et al., 2000, p. 3-4) [71] :

— c'est un langage de programmation impératif de haut niveau ;
— les programmes sont des fonctions réutilisables ;
— les fonctions ne sont pas récurisives ;

71. Traduction et notes personnelles.

— les appels de fonctions ne se font qu'avec un passage par valeur [72]

— les variables sont locales aux fonctions ;

— c'est un langage typé ;

— les types fondamentaux de données sont les tableaux et les t-uplets de tableaux ;

— le type des variables n'a pas besoin d'être déclaré dans un entête particulier [73] ;

— les conditions sont exprimées en utilisant des commandes protégées [74]

— il y a une instruction TANT QUE... pour l'itération ;

— il n'y a pas de GOTO [75].

Mis à part les instructions permettant le contrôle du programme, les instructions sont essentiellement composées d'expressions, mêlant variables et opérateurs arithmétiques ($+, -, \times, /$) ou logiques ($\wedge, \vee, \neg, =, \neq, \leq$), des quantificateurs (les équivalents de $\exists$ et de $\forall$), ainsi qu'un opérateur spécifique μ (Bauer et Wössner, 1972, p. 682). Comme chez Lovelace, les variables sont classées en trois familles (données, variables temporaires, résultats) :

> In a *Rechenplan* $\mathscr{P}$, i.e. in a program or a subroutine [...], an identifier is a letter followed by a number. The letter is V, Z, R, or C, depending on whether the object in question is used as an input parameter (*Variable*), intermediate value (*Zwischenwert*), result parameter (*Resultatwert*), or as a constant in $\mathscr{P}$. The distinguishing number (*Nummer*) is attached to the letter in the line below. The letter classifies the objects. (ibid., p. 679)

Les constantes sont en outre dénotées avec la lettre C, et les variables (indices) de boucles, *loop variables,* sont notées i0, i1, i2... en fonction de la profondeur de la boucle (Rojas et al., 2000, p. 5).

L'affectation, selon Bauer et Wössner (1972, p. 681), est la plus importante fonctionnalité pour la construction d'un programme. Zuse utilise le signe $\Longrightarrow$ pour l'assignement, ce qui donne par exemple une écriture telle que :

	Z	$+$	1	$\Longrightarrow$	Z
V	1				1
S	$1 \cdot n$		$1 \cdot n$		$1 \cdot n$

Cette écriture signifie que l'entier (de n bits) de la variable temporaire $Z1$ est augmenté de 1. L'affectation peut aussi impliquer une initialisation (ou une déclaration) de variable :

	Z	$+$	1	$\Longrightarrow$	R
V	1				0
S	$1 \cdot n$		$1 \cdot n$		$1 \cdot n$

72. Ce n'est pas l'adresse d'un éventuel paramètre qui est passée à la fonction, qui ne peut donc modifier que les variables déclarées au sein de la fonction. D'une façon plus générale, la mémoire est adressée mais non accessible au programmeur : Bauer et Wössner (1972, p. 8) soulignent que, comme pour le (futur) langage ALGOL60, « the intention was to make the address calculation not accessible to the programmer, and this was motivated by the desire for error-free programming as well as by awareness of the frequent malfunction of machines in those years ».

73. Le type est défini à l'assignation.

74. Ces commandes, équivalentes aux SI...ALORS..., sont de la forme *expr* $\longrightarrow$ [*blocdecommandes*].

75. Ce qui est plutôt positif, car comme l'a souligné Disjkstra (1968) dans une lettre devenue célèbre « Go-to Statement Considered Harmful », l'instruction GOTO (un branchement sur une instruction quelconque) rend la recherche d'une description de la progression de l'algorithme très difficile. Finalement, cela déstructure l'algorithme.

Ainsi, s'il y a plus d'une assignation pour une variable, la première est considérée comme une déclaration (et initialisation) de la variable, les autres étant des assignations. C'est, selon Bauer et Wössner (1972, p. 5), ce qui définit le concept de variable (informatique). Contrairement à Lovelace, Zuse utilise ainsi un même symbole pour désigner une variable informatique ou un objet algébrique : en fait, il utilise la variable de la logique propositionnelle, des systèmes formels. En outre, selon Knuth et Pardo (1976, p. 16), Zuse a tenu à préciser les relations mathématiques entre les variables, ce qu'on appellera plus tard les invariants.

Si le langage développé par Zuse dans son *Plankalkül* n'a pas été concrétisé [76], il contient les germes de ce que seront les langages informatiques ultérieurs : des langages formels et compilables.

1.7.5 Hopper, Backus, Rutishauser... et premiers langages compilés

Zuse, parmi les nombreux exemples d'algorithmes qu'il a joints à son *Plankalkül* [77], a présenté des algorithmes permettant de tester si une expression logique était bien formée, en tenant compte des parenthèses. C'est peut-être ce qui amènera Rutishauser à envisager la possibilité d'automatiser la transcription du langage de programmation au langage machine :

> C'est en effectuant des travaux sur le Z4 que le mathématicien suisse Heinz Rutishauser eut une importante intuition : l'ordinateur étant par essence un automate logique universel capable de manipuler des signes alphanumériques qui en raison dernière étaient toujours représentés en binaire, il était sans nul doute possible de le programmer de manière à ce qu'il vérifie et traduise lui-même en langage-machine les séquences d'instructions que l'utilisateur pouvait avoir précédemment écrites en un quelconque langage symbolique. (Ricquebourg, 2008, p. 956)

Cette approche permet de gérer un problème majeur sur les machines rencontrés par les programmeurs à cette époque : la machine étant pilotée par des instructions codées numériquement, la procédure d'implémentation d'un programme est complexe, source d'erreurs, et peu lisible, ce qui rend l'identification des erreurs très difficile. Rutishauser (1951, p. 1) écrit ainsi :

> For the numerical treatment of a mathematical problem on program directed digital computers, the machine code, that is, the collection of commands for the execution of the individual calculation steps, must either be fed to the machine through a punched tape, or else already be available in core memory.
>
> It is known that machine code generation, that is, the determination of such machine code for a particular problem, is often a considerable task, and in extreme cases a significant part of the whole expenditure can fall to the preparation task. Thus tools have been constructed for the simplification of code generation, such as the "Coding Machine" for the Mark III [...] K. Zuse made far-reaching beginnings in the automation of code generation in his "Allgemeinen Plankalkül" (General code-calculator), which, however, also required the construction of special hardware.
>
> On the other hand, the author of this work acquired the conviction long ago that it must be possible, owing to their versatility, to utilize program-directed computers themselves as code generators. This would therefore mean that one would not only solve numerical problems with these calculating machines, but also “calculate" machine code.

Il s'agit finalement de créer un programme traduisant des programmes d'un certain langage vers un langage machine : l'idée du compilateur était née. Pour sa machine, Rutishauser envisage un codage sur un nombre à 6 digits (figure 1.23, p. 91). Les premières propositions de Rutishauser

76. Mise à part l'implémentation, en février 2000, réalisée par Rojas et al. (2000).

77. Algorithmes de tri, traitant des graphes, d'arithmétique entière ou en virgule flottante, et 49 pages d'algorithmes pour jouer aux échecs ! (Knuth et Pardo, 1976, p. 9-10).

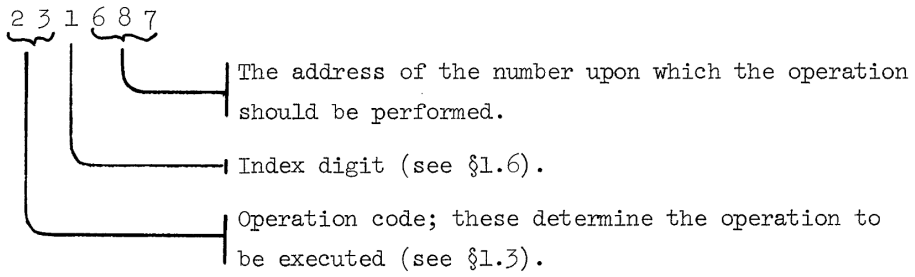


Figure 1.23 – Codage d'une instruction (Rutishauser, 1951, p. 4)

consistent néanmoins en un langage peu lisible, fait de nombreuses abbréviations dont le sens n'est pas accessible sans apprentissage. Ainsi, une instruction équivalente à $sin(ax + b) \implies y$ (soit l'affectation à la variable y du résultat de $sin(ax + b)$) s'écrit :

```
A     x
x     b     } Calculation of the argument
+     a
BZ    4     Storage of 1+program counter in IR4
C     sin   Jump to the sin subroutine
--------    } Empty instructions
--------    }
S     y     Storage of the result.
```

(ibid., p. 23)

Les idées de Rutishauser vont être la base des premières propositions pour un langage « universel » de description des algorithmes (Bauer, Bottenbruch et al., 1958, p. 356), langage qui débutera sous le nom IAL (pour *International Algebraic Language*), et s'appellera finalement Algol (*ALGOrithmic Language*). Cependant, dès 1954, dans le rapport préliminaire consacré au langage FORTRAN, *FORmula TRANSlator*, destiné en premier lieu à l'IBM 704, la nécessité de s'intéresser non plus seulement aux progrès des machines, mais aussi à l'aspect programmation, notamment pour des raisons économiques, est soulignée :

> The time and cost required for the solution of a problem on a high speed calculator fall roughly into 4 categories :
>
> 1. Analysis and Programming
> 2. Coding
> 3. Debugging
> 4. Machine Solution
>
> Faster and more capacious machines will considerably reduce the cost and time required for item 4 but so far the advent of new machines seems to have done little to reduce either the cost or time required for items 1, 2, and 3. (Backus, Herrick et Ziller, 1954, p. 1)

Pour ces nouveaux langages, de nouvelles contraintes seront aussi fixées, par exemple par Bauer, Bottenbruch et al. (1958, p. 355) pour Algol :

> The symbols necessary for representation of the elements of the language are chosen as far as possible in conformity with usages of printing. Furthermore the language has been adapted as far as possible to the following postulates : I) The language should coincide wherever possible with standard mathematical formula notation, and should, for a mathematician, be readable without further explanations. II) It should be possible to use the language in publications, for the description of computing processes. III) The language should describe any computing process as accurately as to allow automatic translation into machine programs.

La question de la lisibilité et du partage écrit est ainsi posée, tout en maintenant la possibilité de compilation.

Les dénominations des variables « as designations for numbers as in elementary arithmetics » (Bauer, Bottenbruch et al., 1958, p. 358) se font dans la version 1954 du FORTRAN sous la forme d'une séquence de deux caractères alphanumériques, dont le premier doit être parmi les lettres i, j, k, l, m, n pour les entiers, et une autre lettre pour les flottants (Backus, Herrick et Ziller, 1954, p. 4). On retrouve ici des conventions issues des mathématiques, avec en outre l'idée de marquer le type de la variable par son symbole, et non par sa déclaration, contrairement à Zuse. Pour Bauer, Bottenbruch et al. (1958, p. 357), toute variable pourra s'écrire sous la forme d'une lettre quelconque éventuellement suivie de lettres ou chiffres (ibid., p. 357), préfigurant les normes d'écritures de nombreux langages.

Le symbole d'affectation est pour la version 1954 de FORTRAN le signe « = » [78], en précisant :

> It should be noted that the equals sign in an arithmetic formula has the significance of « replace ». In effect, therefore, the meaning of an arithmetic formula is as follows : Evaluate the expression on the right and substitute this value as the value of the variable on the left. (Backus, Herrick et Ziller, 1954, p. 9)

Pour Algol (1958), c'est le symbole utilisé par Zuse qui est évoqué, « => », et deux alternatives sont proposées pour les fonctions, « => » et « := » [79], avec dans l'idée, comme Zuse, de ne pas utiliser le signe « = » :

> The main principle was to write to the left of the "ergibt"-symbol "=>" the arithmetic expression to be evaluated and to the right the designation of the new quantity defined by the calculation. This "ergibt" symbol corresponds better to the dynamic process of computing than the usual equality symbol. Especially it can be used in situations where the latter may lead to contradictions, e.g., s + 2 => s ("old" s plus 2 yields "new" s) (Bauer, Bottenbruch et al., 1958, p. 356)

Les définitions de fonctions nommées sont aussi prévues, ainsi bien sûr que toutes les structures de contrôle (boucles et conditions), de même que la prise en compte de la lecture et écriture sur des bandes perforées.

Dans les années 1950 se développent ainsi de nombreux langages compilés :

> On pourrait mentionner ici aussi les travaux réalisés par Alan M. Turing à Manchester (où il rédigera un manuel de programmation pour le Mark I), et le Short Code, un interpréteur de pseudo code conçu en 1949 par John W. Mauchly à destination des ordinateurs B.I.N.A.C. et U.N.I.V.A.C. I et II. Mais le virage le plus important sera assurément amorcé entre 1951 et 1958 avec l'apparition successive des langages A-O (langage de programmation mathématique

78. Berry (2020) rapporte que, selon ses souvenirs, ce choix d'utiliser « = » pour l'affectation et « == » pour tester l'égalité était dû à une recherche d'économie de mémoire, l'affectation étant plus fréquente que le test d'égalité.

79. c'est ce dernier symbole qui sera finalement retenu.

> à compilateur intégré que Grâce Hopper créa en 1951 à la Remington Rand pour l'U.N.I.V.A.C. I), Autocode9, FORTRAN (pour FORmula TRANslator, langage spécialement développé par John Backus pour l'I.B.M. modèle 704, un ordinateur à virgule flottante), et International Algebraic Language (ou ALGOL 58, un langage de programmation défini par un comité d'informaticiens européens et américains au rang desquels figurait également John Backus). La multiplication et la diffusion des langages de ce genre, assorties à un effort croissant réalisé en direction d'une formulation de plus en plus symbolique des instructions informatiques – on se dirigera ainsi progressivement vers une expression des ordres proche du langage naturel comme on peut le voir par exemple en PASCAL ou en BASIC – permettront désormais aux utilisateurs d'ordinateurs d'écrire, de modifier et de corriger leurs programmes (pilotes de système ou applications professionnelles) avec une facilité bien plus importante qu'ils ne pouvaient le faire auparavant. (Ricquebourg, 2008, p. 949)

Les bases sont posées, de (très) nombreux langages fleuriront et d'autres continuent à être inventés : la figure 1.24 (p. 96) est un extrait du travail réalisé par Éric Lévénez [80], qui renvoie aussi notamment au site « Online Historical Encyclopaedia of Programming Languages » [81] référençant quelques 8945 langages de programmation !

1.7.6 Church, Turing et la calculabité

Parallèlement à cette recherche de langages et de machines, des recherches théoriques issues de mathématiques auront une grande influence sur l'informatique. Outre Gödel et son théorème d'incomplétude, Von Neumann et ses prescriptions sur l'architecture d'un ordinateur, nous ne pouvons omettre de rappeler le rôle et l'importance de Alan Turing (1912-1954) et de celui qui fut aussi le directeur de thèse de Turing, Alonzo Church (1903-1995). Berry (2020, p. 71) dit d'eux :

> La paternité de l'informatique revient incontestablement à Alan Turing et Alonzo Church en 1936.

Leur apport majeur a été la réponse au troisième problème de Hilbert concernant la décidabilité : existe-t-il ou non une « une méthode de calcul, encore appelée procédure effective ou algorithme, permettant de démontrer ou d'infirmer une proposition quelconque en un nombre fini d'étapes. » (Ricquebourg, 2008, p. 150). Church et Turing y ont répondu à la même époque (1936-1937), mais de façon indépendante. Turing a défini une machine universelle abstraite, que l'on nomme depuis « machine de Turing », machine disposant d'un nombre d'états fini et d'une table de transition associant état et symbole :

> Turing a montré comment effectuer tous les calculs connus par ses machines et a construit une « machine universelle » U capable de simuler toute autre machine M : U simule le calcul de M sur des données D en prenant comme données d'une part D et, d'autre part, un programme P écrit de la même façon que les données, mais spécifiant par son texte ce que M doit faire. C'est cette idée absolument géniale qui a rendu possible la construction des ordinateurs modernes : du superordinateur de la météo au microcontrôleur d'un lave-linge, ils sont logiquement équivalents, se programment de la même façon et ne diffèrent que par leurs performances en vitesse et taille de mémoire ! (Berry, 2020, p. 71)

Ainsi, tout ce qui est calculable (par un mathématicien) peut être représenté et calculé par une machine de Turing.

80. Lévénez, 2023.
81. http://hopl.info/

De son côté, Church, pour le même problème, a établi que tout ce qui est calculable (par un mathématicien) peut être représenté par une fonction, représentable dans « un nouveau formalisme logique, le "lambda-calcul", apparemment tout aussi simpliste [que la machine de Turing] car sa définition ne prend que quelques lignes. » (Berry, 2020, p. 71).

Finalement, ces deux systèmes sont équivalents :

> En s'appuyant sur Kleene[82], Turing prouve que sa machine logique est aussi expressive que les systèmes formels imaginés dans les années 30 (Herbrand, Gödel, Kleene et le lambda-calcul de Church), au sens où elle définit la même classe de fonctions. (Longo et Lassègue, 2020, p. 5)

Ainsi, un algorithme peut être vu (et construit, ou exécuté) de façon équivalente comme une machine à état ou comme une fonction. L'aspect fonction est resté longtemps en retrait, avant la création de langages de programmation fonctionnelle inspirée du λ-calcul. Ainsi Backus (1978, p. 614) décrira « a new class of computing systems uses the functional programming style both in its programming language and in its state transition rules. ». En fait, Backus oppose les deux paradigmes de programmation, ou modes de programmation, impératif et fonctionnel, comme le suggère le titre de son article, « Can Programming Be Liberated from the von Neumann Style ? » Il souligne notamment que, si pour une système de « Fonctionnal Programming » (FP), les changement d'états sont simples, ils sont multiples et sans rapport direct avec l'objectif du programme pour les langages impératifs, car toutes les étapes doivent être détaillées, alors qu'en FP ce qui se passe entre la déclaration de la fonction et son résultat est une boite noire. Cela se traduit entre autres par le fait que « Von Neumann programming languages use variables to imitate the computer's storage cells » (ibid., p. 616)(comme les colonnes de Variables de Babbage), et donc que l'instruction d'affectation est l'opération fondamentale des langages impératifs. Ainsi, les opérations sont atomiques :

> Consider a typical program ; at its center are a number of assignment statements containing some subscripted variables. Each assignment statement produces a oneword result. The program must cause these statements to be executed many times, while altering subscript values, in order to make the desired overall change in the store, since it must be done one word at a time. The programmer is thus concerned with the flow of words through the assignment bottleneck as he designs the nest of control statements to cause the necessary repetitions. (ibid., p. 616)

Chaque opération doit être détaillée en programmation impérative — ce qui rend la compréhension du programme produit difficile, à moins de l'exécuter mentalement —, alors qu'en FP il suffit de combiner des fonctions, — dont le sens est directement accessible. Backus donne l'exemple (ibid., p. 616) du produit scalaire. Celui-ci s'écrirait en langage impératif :

c `:=` `0`
`For` i `:=` `1` **`step`** `1` **`do`**
    c `:=` $c + a[i] \times b[i]$

Alors qu'en FP ce serait :

```
Def Innerproduct
          ≡ (Insert +)∘(ApplyToAll ×)∘Transpose
```

82. Lui aussi élève de Church.

Backus veut ainsi construire une « algèbre des programmes », dans laquelle les programmes sont manipulables comme des variables, et dans laquelle se feraient les preuves de programmes, alors que les preuves de programmes sont forcément dans un langage externe au langage de programmation pour les langages impératifs.

Cette opposition programmation impérative - programmation fonctionnelle est cependant de moins en moins d'actualité, les langages impératifs proposant des aspects de programmation fonctionnelle (fonctions λ ou anonymes, en Javascript ou Python par exemple), et des langages fonctionnels peuvent proposer des instructions et procédés de langages impératifs (en Rust par exemple, on peut notamment rendre une variable « mutable »).

La programmation fonctionnelle dépassant le cadre de notre recherche, nous n'approfondirons pas plus le sujet.

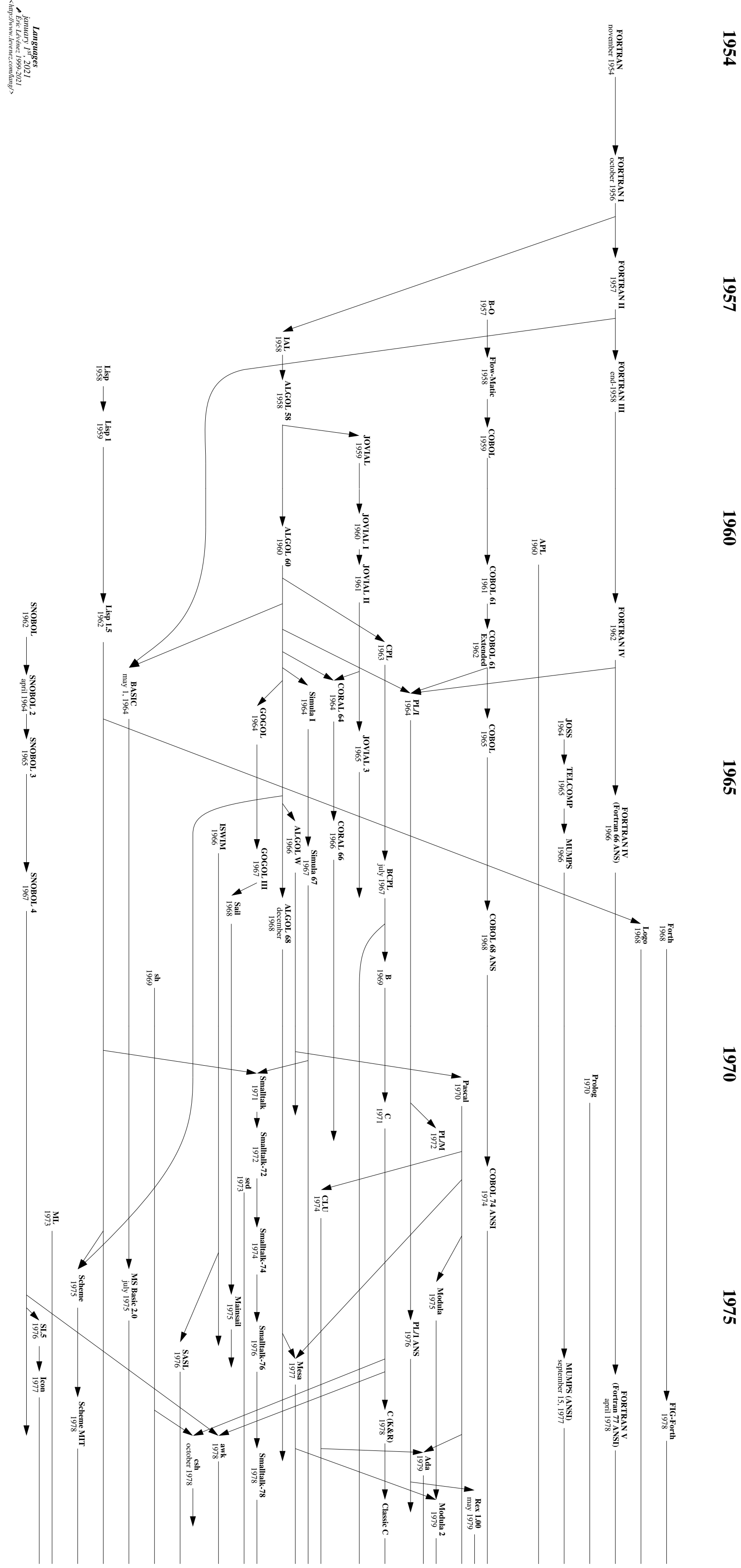


Figure 1.24 – Explosion de langages... Extrait de (Lévénez, 2023)

1.7.7 Informatique et enseignement en France

Cette explosion des langages accompagne une diffusion de plus en plus massive des machines. En France, l'Éducation Nationale se penchera très tôt sur le sujet. En effet, dès 1967, alors que le mot « informatique » rentre tout juste dans le dictionnaire de l'Académie Française, et avant même la découverte du microprocesseur (1971), la première Licence informatique est créée. En 1970 a lieu le Séminaire du Centre d'Études et de Recherches pour l'Innovation dans l'enseignement (CERI-OCDE) pour l'enseignement de l'informatique à l'école secondaire, ainsi que la création, au minsitère de l'Éducation Nationale, d'une Mission à l'informatique. En 1971, l'association EPI (Enseignement Public et Informatique) est créée. L'essentiel des éléments abordés dans cette partie est issu notamment d'un texte de Baudé (2019) écrit pour les quarante ans de cette association, complété par les nombreuses informations et articles disponible sur le site de l'association [83]. Dès sa création, l'EPI défendra trois aspects de l'informatique à l'école :

> L'introduction de l'informatique se présente actuellement, semble-t-il, sous trois aspects :
>
> — comme l'enseignement d'une matière nouvelle[...]
>
> — comme l'enseignement d'une méthode de pensée à l'intérieur des matières existantes, chaque professeur retrouvant dans sa discipline les notions fondamentales de modèle, d'algorithme, d'information ;
>
> — comme l'utilisation d'un moyen nouveau, comparable à ce que fut le premier livre imprimé, aidant le professeur dans la partie répétitive de son travail.
>
> (ibid.)

On y voit ainsi apparaitre l'informatique en tant que science, l'informatique en tant que mode de pensée, et l'informatique comme outil pour enseigner : l'informatique est à la fois un *objet* d'étude et un *outil* pour enseigner/apprendre [84]. Ces préoccupations seront plus ou moins suivies par l'institution. Cette décennie verra des expérimentations fleurir : l'expérience des 58 lycées par exemple, les premières recherches sur la robotique pédagogique, avec les tortues Logo (voir figure 1.25, p. 97) étudiées notamment par Seymour Papert (ancien collaborateur de Piaget).

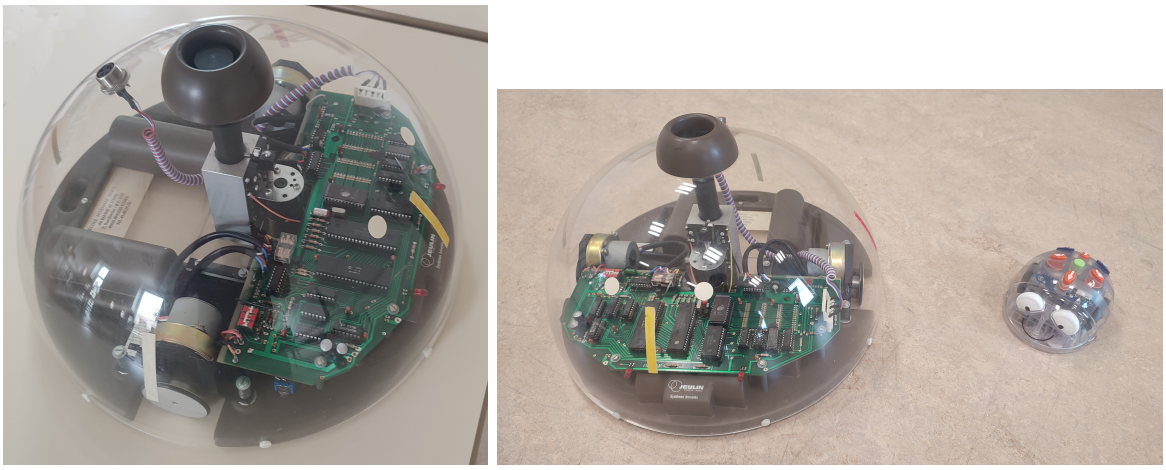


Figure 1.25 – Tortue Jeulin et une version moderne, le Blue-bot, Inspé de Nantes, site d'Angers

83. https://www.epi.asso.fr
84. Voir (Douady, 1984) concernant la dialectique outil-objet.

Au début des années 1980 les Écoles Normales commencent à s'équiper en ordinateurs. Dans les lycées, l'option informatique est créée. La Direction de Écoles anticipe la nécessité de formation à l'informatique du fait de l'informatisation anticipée de la société (circulaire du 24 mars 1983). Des éléments « informatisés » font leur entrée dans l'école maternelle :

> Progressivement, l'enfant découvre et construit des relations spatiales. Il saisit sa propre position dans l'espace ; il perçoit et représente la position d'un objet par rapport à un ou plusieurs autres ; il appréhende, nomme, représente des formes et des itinéraires. Des objets informatisés tels que robots pédagogiques et automates (« tortue », jouets programmables, etc.) peuvent rendre des services à l'école maternelle (Circulaire n°86-046 du 30 janvier 1986, « orientation pour l'école maternelle »)

En 1985 est lancé le plan Informatique Pour Tous, et on trouve dans les programmes et instructions pour l'école élémentaire :

> [En mathématiques]Enfin, l'utilisation de l'informatique, à propos de la résolution d'un problème numérique ou géométrique, en particulier au cours moyen, permet d'initier l'élève à la recherche d'algorithmes et de développer ses capacités logistiques.
>
> [...En Science] L'importance de l'informatique justifie qu'au cours moyen cinquante heures au moins lui soient consacrées.
>
> [...] Objets et systèmes informatiques
>
> — Le développement de l'informatique dans la société (transformation de l'activité professionnelle et de la vie quotidienne par la télématique, la bureautique et la productique ; problèmes sociaux et éthiques).
>
> — La technologie informatique (le micro-ordinateur ; automates programmables et robots).
>
> — Le logiciel (analyse et modification de logiciels simples ; début de programmation dans une perspective logistique)

Baudé (2023) rappelle aussi l'entrée de l'enseignement de l'informatique au collège en 1985 (au côté de l'usage des ordinateurs) :

> C'est lorsque l'élève « programme un ordinateur pour un traitement voulu, que l'obligation de précision doit lui apparaître comme une évidente nécessité. » (Méthodes, p. 81 et 82). En classe de quatrième, le travail doit permettre « d'utiliser rationnellement des calculatrices de poche... L'utilisation d'un ordinateur peut accompagner utilement ces activités. Son usage permettra de dégager progressivement les notions de codage et d'algorithme » et en troisième : « Analyse (et construction) d'algorithmes comme suite d'instructions aboutissant à la résolution d'un problème donné. Application numérique à l'aide d'un ordinateur. » (Programmes p. 87 à 89).

Dans la lettre du ministre de l'éducation du 29 octobre 1985 (B.O. n° 39 du 7 novembre 1985, pages 2778 à 2780)[85], J.-P. Chevènement souligne que l'informatique doit être la fois vue comme une matière d'enseignement et comme un outil d'enseignement. Il précise en outre, concernant les objectifs de connaissance :

> Concepts, structures et méthodes de base de l'informatique : l'enseignement dégagera l'importance fondamentale des méthodes d'analyse et de programmation qui constituent l'apport le plus spécifique et le plus fécond de l'informatique à la démarche scientifique.

À la fin des années 1980 et jusqu'à la fin des années 2000, l'informatique outil prend le pas sur l'informatique objet — par exemple, les options informatiques sont supprimées —, malgré le manque d'informaticiens qui commence à poindre.

85. Reproduite ici : https://www.epi.asso.fr/revue/40/b40p039.htm.

Au début des années 2010, plusieurs rapports insistent sur la nécessité de l'enseignement de l'informatique objet. Un rapport de 2009 du SNRI (Stratégie Nationale de Recherche et d'Innovation) concernant le groupe « Numérique, calcul intensif et mathématiques » indique :

> *Dès les cycles primaire et secondaire, un enseignement d'informatique* doit viser l'acquisition de bases solides en matière de programmation et d'algorithmique, de représentations numériques des textes, des images et des sons, d'architecture des machines, de réseaux, de bases de données, etc. Pour être efficaces, ces cursus doivent développer la continuité du raisonnement depuis la conception de l'expression algorithmique jusqu'à sa réalisation effective et prendre au lycée la forme d'une discipline scolaire en tant que telle. [86]

En 2012, la spécialité Informatique et sciences du numérique fait son retour pour les élèves de terminales, mais le rapport de 2013 de l'Académie de Sciences « L'enseignement de l'informatique en France » [87] marque une forte inquiétude : son sous-titre est « Il est urgent de ne plus attendre ». Ses préconisations concernant l'enseignement de l'informatique (p.4) sont, outre « un soin particulier » qui doit être accordé aux questions de genre :

> — Il peut et doit être commencé dès le primaire, par une sensibilisation aux notions d'information et d'algorithme, possible à partir d'exemples très variés dans le style de La main à la pâte. Il doit être approfondi au collège et au lycée.
>
> — On pourra y distinguer trois phases principales :
>
> 1. La sensibilisation, principalement au primaire, qui peut se faire de façon complémentaire en utilisant des ordinateurs ou de façon « débranchée » ; un matériau didactique abondant et de qualité est d'ores et déjà disponible.
> 2. L'acquisition de l'autonomie, qui doit commencer au collège et approfondir la structuration de données et l'algorithmique. Une initiation à la programmation est un point de passage obligé d'activités créatrices, et donc d'autonomie.
> 3. Le perfectionnement, qui doit se faire principalement au lycée, avec un approfondissement accru des notions de base et des expérimentations les plus variées possibles.

Concernant plus spécifiquement le collège (p.5), il est proposé d'« Introduire un véritable enseignement d'informatique, qui ne soit pas noyé dans les autres enseignements scientifiques et techniques, mais développe des coopérations avec ceux-ci dans une volonté d'interdisciplinarité. ». On vise ainsi à la fois les concepts de l'informatique mais aussi les aspects de la pensée informatique mobilisables dans d'autres disciplines.

Les arguments évoqués dans ce rapport sont également des préoccupations présentes dans d'autres pays :

> L'argumentaire développé semble ériger la pensée informatique en un savoir-faire fondamental, nécessaire à tout individu, et non plus réservé aux seuls informaticiens. Cette dynamique va de pair avec une focale s'orientant vers la SI [Science Informatique]. De nombreux pays, notamment au niveau des décideurs politiques, apparaissent sensibilisés à un tel argumentaire : la Grande-Bretagne, les États-Unis, Israël, la Nouvelle-Zélande (Brown et al., 2014), (Gal-Ezer and Stephenson, 2014), (Bell et al., 2014) et plus récemment, la France... (Chiprianov, Coulange et Train, 2018, p. 20)

86. Disponible en ligne : http://media.enseignementsup-recherche.gouv.fr/file/Defi_de_connaissance_pluridisciplinaire/97/5/SNRI2009_rapport_groupe_de_travail_Nummath_65975.pdf

87. https://www.academie-sciences.fr/fr/Rapports-ouvrages-avis-et-recommandations-de-l-Academie/l-enseignement-de-l-informatique-en-france-il-est-urgent-de-ne-plus-attendre.html

Depuis 2016, des aspects concernant l'enseignement de l'informatique objet ont fait leur retour dans les programmes du collège et de l'école primaire :

> Ce sont des dimensions, concernant à la fois l'AN [Alphabétisation Numérique, le versant *outil*]et la SI [Science Informatique, le versant *objet*], qui sont déclinées dans un programme ambitieux en place depuis la rentrée 2016 dans le cadre de la scolarité obligatoire [...] (Chiprianov, Coulange et Train, 2018, p. 21)

Dans les programmes de l'école primaire et du collège en France, à partir de 2016, une initiation à l'algorithmique et la programmation se formalise, non comme une discipline à part entière mais comme un enseignement conjoint aux mathématiques et à la technologie :

> En outre, un enseignement de l'informatique (algorithmique et programmation) est dispensé conjointement en mathématiques et en technologie. Il n'a pas pour objectif de former des élèves experts, mais de leur apporter des clés de décryptage d'un monde numérique en évolution constante. Il permet d'acquérir des méthodes qui construisent la pensée algorithmique et développe des compétences dans la représentation de l'information et de son traitement, la résolution de problèmes, le contrôle des résultats. [88]

Dans le document d'accompagnement « Algorithmes et Programmation » (Recherche (MENSR), 2016), il est indiqué :

> Le logiciel Scratch offre un environnement d'édition et d'exécution des programmes. Il s'agit d'un logiciel gratuit et disponible sur toutes les plates-formes usuelles, choisi pour sa simplicité, sa fiabilité et sa robustesse dans la mise en œuvre. Il permet de travailler tous les concepts figurant au programme [...]

Cet environnement de programmation fait partie des nouveaux langages de programmation graphique, que nous appelons Environnement de Programmation Graphique par Blocs (EPGB) (Legrand, 2019), qui vont clore cette courte histoire de l'informatique.

1.7.8 Environnement de Programmation Graphique par Blocs (EPGB), des langages pour apprendre

Scratch [89], ou sa version adaptée aux plus jeunes enfants ScratchJr [90], fait partie de ces nouveaux outils [91] permettant de construire des situation enseignement/apprentissage des concepts informatiques :

> [...] plusieurs outils technologiques innovants ont vu le jour ces dernières années, pouvant s'inscrire sous cette perspective : tels sont les environnements informatiques (p.e. ScratchJr), les jouets et les robots programmables (comme les Bee-Bot et les Probot) et les kits de constructions robotiques (comme les Lego MindStorms) (Komis, Touloupaki et Baron, 2017, p. 2)

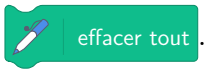
88. Eduscol, https://www.education.gouv.fr/les-programmes-du-college-3203
89. https://scratch.mit.edu/
90. https://www.scratchjr.org/
91. « Renouvellement » dans le cas de la robotique pédagogique.

Scratch ou ses dérivés (ScratchJr, mBlock [92], Phratch [93], Snap ! [94]), tout comme blockly [95] et les langages utilisant cette biblitothèque (par exemple pour blockyGames [96] ou les langages permettant de programmer les cartes micro:bit [97], ou arduino) de même que certains robots comme le Thymio ou l'Ozobot, tous sont des Environnements de Programmation Graphique par Blocs (EPGB). Un EPGB est un logiciel permettant à la fois de développer et d'exécuter un programme conçu dans un « langage entièrement visuel et fonctionnant donc par manipulation gestuelle » (ibid.). La construction d'un programme se fait par « glisser-déposer » des instructions disponibles. En reprenant le vocabulaire utilisé par Komis, Touloupaki et Baron (ibid.), une instruction est un *bloc*, et l'assemblage (la juxtaposition verticale) de blocs forme un *script* exécutable. Plusieurs scripts peuvent coexister, formant ainsi un *programme*. Les blocs ne peuvent s'emboîter que de façon logique, en raison de leurs formes, ce qui élimine les messages d'erreurs(Sáez-López, Román-González et Vázquez-Cano, 2016), ou du moins limite les risques d'erreur lexicale, syntaxique ou sémantique.

> This visual environment enables an intuitive drag and drop method of programming which allows users to explore and create in educational settings at several levels in primary school. The aforementioned application is aimed at engaging young learners to provide an accessible starting point for learning with limited or no programming background (Good, 2011) (ibid., p. 130)

Ces EPGB disposent tous d'un environnement de programmation similaire inclus dans une fenêtre divisée en différentes zones (figure 1.26, p. 102), avec notamment : la palette des commandes — avec éventuellement une sélection des catégories d'instructions —, un espace de programmation sur lequel on glisse les blocs pour construire les scripts, et un espace d'exécution fournissant un retour visuel des actions. L'espace d'éxécution est un espace similaire à un écran, sur lequel seront visualisées certaines actions : déplacement des lutins (ou personnages), formes dessinées (à la manière de LOGO), images insérées etc. C'est aussi un espace permettant une interaction avec l'utilisateur, en captant par exemple les évènements tel le clic de souris, ou en proposant des demandes à l'utilisateur afin qu'il entre une valeur. Sur cet espace peuvent aussi être affichés des messages à destination de l'utilisateur. Enfin, le contenu des variables existant peut être affiché et éventuellement modifié (ce qui n'est pas sans conséquences didactiques).

Les scripts sont des assemblages de blocs. Ces blocs, toujours en utilisant les formulations de Komis, Touloupaki et Baron (2017), peuvent être par exemple :

- des instructions générales : 

...
- des instructions spécifiques : 

...
- des opérateurs :  , 

...
- des variables 

ou des instructions modifiant une variable 

...
- des blocs de contrôle 

, 

...

92. https://www.mblock.cc
93. https://github.com/janniklaval/phratch, projet semble-t-il abandonné.
94. https://snap.berkeley.edu/
95. https://developers.google.com/blockly
96. https://blockly.games/
97. https://microbit.org/

Figure 1.26 – Organisation spatiale d'un EPGB (ici Snap !)

— etc.

Les exécutions des scripts sont généralement déclenchées soit par appui sur une icône spécifique, soit en réaction à un évènement — par exemple un script commençant par 

se déclenchera lors de l'appui sur la touche « a », soit encore en cliquant directement un script.

Parmi ces EPGB, le plus connu est Scratch :

> Scratch is a programming language created by the Lifelong Kindergarten group at the MIT Media Lab. The Scratch programming language offers more than 100 programming blocks, grouped into eight different categories (motion, looks, sound, pen, control, sensing, operators, and variables). This programming environment enables young people to create their own interactive stories, games, and simulations, and then share these creations in an online community with other young programmers from around the world. Pupils can program and share interactive media such as stories, games, and animation. Children learn to think creatively and collaboratively using Scratch. Coding in this interface is easier than traditional programming languages due to kids playing and interacting with the colorful blocks to create scripts. Scratch is based on the ideas of the constructivist learning and "logo" project (Papert, 1980). This versatile application can be used to create projects containing media scripts. Images and sounds can be imported or created in Scratch using a built-in paint tool and sound recorder (Maloney, Resnick, Rusk, Silverman, & Eastmong, 2010) (Sáez-López, Román-González et Vázquez-Cano, 2016, p. 132)

Scratch, inspiré des travaux de Papert sur le Logo, a été créé avant tout pour des ateliers, et non pour l'enseignement, et nous verrons que cela a des conséquences didactiques non négligeables.

> Originally inspired by Papert's work (Papert, 1980), Scratch was intended by Resnick to support creative work with multimedia (Maloney, Peppler, Kafai, Resnick, & Rusk, 2008) in "computer clubhouses" or after-school learning centres for children from deprived communities, and was first deployed in 2005. Those children enjoyed multimedia "mash-ups," like the sampling techniques used in the pop music they liked, which is why the new system came to be called "Scratch." (Wilson et Moffat, 2010, p. 2)

Pour notre expérimentation, nous utiliserons un dérivé de Scratch, Snap !, conçu par l'Université de Berkeley, et davantage axé sur l'enseignement/apprentissage :

> Snap ! (formerly BYOB) is an extended reimplementation of Scratch (https://scratch.mit.edu) that allows you to Build Your Own Blocks. It also features first class lists, first class procedures, first class sprites, first class costumes, first class sounds, and first class continuations. These added capabilities make it suitable for a serious introduction to computer science for high school or college students. (Harvey et Mönig, 2020, p. 5)

1.8 Bilan : concepts à l'interface algèbre-algorithmique-informatique

Si l'on reprend les éléments essentiels dégagés tout au long de cette enquête épistomologique croisant algèbre et informatique (et donc algorithmique), on peut constater la grande proximité entre algèbre et informatique — mais l'algèbre n'est pas réductible à l'informatique, pas plus que l'inverse —.

1.8.1 Algèbre et algorithme

Ainsi, si l'on s'intéresse aux concepts algorithmiques présents dans l'algèbre, on a pu constater que les quatre concepts de l'algorithme (séquence, itération, condition, variable) sont mobilisés dans les démarches étudiées. La séquence d'instruction, ordonnée et la moins ambigüe possible, est clairement présente. Les itérations (les boucles), sont aussi localement présentes, mais souvent de façon implicite. L'explicitation des boucles est rendue nécessaire dès lors qu'on veut automatiser, ou algorithmiser, un processus. Mais la boucle est-elle toujours nécessaire ? Si l'on regarde les activités d'initiation à la programmation, la boucle est généralement amenée commme une contrainte, ou éventuellement comme un gain de temps et de place (aspect économique). Cependant, la construction de cette nécessité économique semble être rarement abordée. Nous avons fait une rapide expérimentation avec des élèves de quatrième qui met en question l'intérêt économique de la boucle comparé au coût cognitif qu'implique sa mobilisation : un groupe notamment n'a pas construit cette nécessité malgré les 683 instructions nécessaires à une répétition de motif ! (Legrand, 2019, p. 6). L'intérêt ergonomique de la boucle (qui facilite la lecture et la correction) est quelque fois cité, sa nécessité lors d'une généralisation n'étant pas évoquée [98]. Rappelons aussi que la boucle n'est nécessaire qu'en programmation impérative, et non en programmation fonctionnelle. Les fonctions récursives suffisent, mais on peut penser que leur étude est peu acessible à des élèves de cycle 4 (voir les travaux à ce sujet de León (2019) et León et Modeste (2020)). Concernant les conditions, elles sont généralement présentes lors de disjonction des cas. Si l'on s'intéresse à la question de la preuve, elle est présente en algèbre comme en informatique. Le preuve en algèbre est passée d'une justification externe (généralement géométrique) à une justification interne, basée sur le langage algébrique et ses règles — la preuve par l'expression pour al-Khwārizmī—. Pour l'informatique, la preuve est externe pour ce qui concerne les langages impératifs : il faut s'appuyer sur l'algèbre.

98. En 2020, sur un échantillon de trois cahiers d'activités d'initiation à la programmation.

La question de la « variable », la représentation d'un objet dénotant une valeur non forcément déterminée *a priori*, traverse l'algèbre. On a pu voir notamment que le paramètre est fondamental pour l'algèbre, mais aussi fondateur pour l'algorithme et l'informatique. Ces paramètres, « soit les variables du système dont les valeurs sont supposées connues » (Chevallard, 1989, p. 65) constituent, notamment, les données manipulées par l'algorithme, un autre des quatre concepts de l'informatique (Dowek, 2011).

1.8.2 Algèbre et machine

Un troisième concept de l'informatique, la machine, ou plus généralement ce que Samurçay et Rouchier (1985) appellent le « dispositif d'exécution », traverse aussi l'histoire : d'abord humaines, car nécessitant des capacités d'interprétation d'éléments implicites, elles sont devenues automatiques grâce à l'existence de procédure automatisables, c'est-à-dire formulées dans un langage monofonctionnel, algorithmisable (Duval, 2002, p. 11). Ces machines seront d'abord spécifiques, avant de gagner en généricité, notamment avec Babbage, pour finir par être non plus des machines algébriques mais des machines logiques, ce qui leur permet de ne pas manipuler que des nombres.

1.8.3 Algèbre et langage

Cette recherche d'un langage monofonctionnel, le moins ambigu possible, a été constante. Il a ainsi fallu déterminer des symboles pour les opérateurs ou quantificateurs, mais aussi pour ce qui est flou, non encore déterminé dans un procédé algèbrique : l'inconnue d'abord, le paramètre tardivement. Comme rappelé ci-dessus, langage et machine ont progressivement gagné en généricité. Mais l'on a aussi vu apparaitre la nécessité d'un langage de description de l'algorithme qui soit plus lisible, plus compréhensible, qu'un langage de description de l'algorithme destiné à la machine. Ce langage devait donc être différent de celui permettant de définir les états successifs adéquats de la machine (tourner tel engrenage, brancher tel circuit...). Ces langages ont d'abord été proches de la machine (Lovelace, ou les différents assembleurs), et ont gagné en généricité en s'abstenant de plus en plus de la machine, grâce aux compilateurs. Ces compilateurs étant eux-mêmes des programmes, faisant la transcription d'un langage évolué à un langage exécutable par une certaine machine, soit un programme qui prend en donnée un programme pour produire un programme.

1.8.4 Informatique et algèbre

L'algèbre mobilise donc les concepts informatiques, et pas seulement algorithmiques. Mais l'informatique s'est aussi construite en partie sur l'algèbre : on part de problèmes algébriques, on a construit des machines algébriques, puis des langages algébriques destinés aux machines. Ces langages permettent de manipuler et opèrer sur des symboles représentant des valeurs non déterminées *a priori* : ils permettent la manipulation d'expressions algébriques. Mais l'informatique ne se limite pas, ou ne se limite plus, à l'algèbre. Ada Lovelace l'avait déjà entrevu :

> Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent. (Menabrea, 1842, p. 694)

Zuse puis Rutishauser l'ont ensuite précisé : les machines n'étant plus seulement algébriques ou arithmétiques, mais logiques, elles permettent de représenter et d'opérer sur toute relation établie entre des objets d'un domaine particulier — ou plus précisément sur leur représentation codée —, domaine non forcément mathématique. On peut donc opérer non seulement sur des nombres, mais aussi sur des chaines des caractères, des musiques etc.

Finalement, on peut dire que l'algèbre est algorithmique, voire informatique. Mais que l'informatique n'est pas qu'algébrique. Un point fondamental est commun à ces deux domaines : un langage symbolique permettant de manipuler (conceptualiser) les paramètres fut nécessaire à leur évolution. Ainsi, en rapport avec notre problématique, la question se précise : peut-on faire construire aux élèves le concept de paramètre en mathématique en s'appuyant sur la construction du concept de paramètre en informatique ?

L'apparition tardive du paramètre et de sa représentation constitue une rupture épistémologique, source possible d'obstacles épistémologiques. Mais le paramètre en algèbre est-il le même concept que le paramètre en informatique ? Nous avons déjà abordé les questions liés au concept plus général de variable, et il semble opportun de définir de façon plus précise les termes de variable, de paramètre ou encore de constante.

1.9 Paramètre, variable et rôles

Le caractère générique de l'algèbre et des différents langages de description d'algorithmes, vient de leur capacité à exprimer la généralité, c'est-à-dire à utiliser des termes du langage pouvant désigner des objets différents. En informatique, on nomme ces termes des « variables », tout comme le faisait déjà Ada Lovelace. Mais le paramètre en informatique, qui permet de représenter une famille d'instances de problèmes, est-il une variable au sens mathématique du terme ? En fait, il ne semble pas y avoir de définition générale consensuelle de la variable mathématique. Nous allons nous attacher à présent à préciser ce que nous entendons par *paramètre*. À la suite de cela nous nous poserons la question du paramètre comme étant ou non une forme de variable, et donc nous chercherons une définition de ce qu'est la variable en général, ce qui a un intérêt didactique évident. Nous clorons cette section par un retour sur le paramètre, en étudiant la rupture épistémologique qui accompagne son apparition en tant que symbole mathématique, et en déduisant des principes didactiques pour construire ce concept.

1.9.1 le paramètre

Le paramètre, nous l'avons déjà signalé, est ce qui permet la généralisation. La représentation de l'inconnue n'est en effet pas l'élément fondamental :

> Mais la clé du succès ne tient pas seulement dans ce petit x qui figure l'inconnue du problème. D'emblée la puissance algébrique est mise en relation avec le fait de désigner par des lettres, à côté des quantités inconnues, que l'on recherche, les quantités connues elles-mêmes - les données. (Chevallard, 1989, p. 65)

Le paramètre s'oppose donc à l'inconnue. Descartes, dans sa règle XVI des « Règles pour la direction de l'esprit », propose « nous nous servirons des caractères a, b, c, etc. pour exprimer les grandeurs déjà connues, et A, B, C, pour les grandeurs inconnues » (Descartes et Salgues, 1824-1826, p. 314). Il précisera dans « La Géométrie » que l'on doit « donner des noms à toutes les lignes qui semblent nécessaires pour le construire [le problème] aussi bien à celles qui sont inconnues qu'aux autres » (Descartes, 1897-1913, p. 372), mais tout en considérant

dans la phrase qui suit qu'on ne doit « considérer aucune différence entre ces lignes connues et inconnues ». Pour al-Khwārizmī, il n'y a que trois sortes de nombre : l'inconnue (la racine), le carré de l'inconnue, et le nombre simple. Ce nombre simple « est un nombre qu'on exprime sans qu'il soit rapporté ni à une racine, ni à un carré ». D'un autre côté, comme le rappelle Radford (1991, p. 3) « Diophante élabore une théorie mathématique sur deux classes d'objets : les nombres, en tant que nombres invariants déterminés [...], et l'arithme ». En cela, le paramètre, qui n'est pas l'inconnue, serait donc un nombre (invariant) déterminé, l'inconnue étant un nombre indéterminé. Chevallard (1989, p. 65) précise quant à lui :

> Ce qui fait la force de l'algèbre, donc, c'est ce que nous nommerions aujourd'hui l'emploi de paramètres, soit les variables du système dont les valeurs sont supposées connues.

On retrouve ainsi le paramètre comme étant « variable ». Ce n'est pour autant pas incompatible avec le monde de Diophante — qui, rappelons-le, ne disposait pas du concept de paramètre ni de sa représentation —. En effet, le paramètre représente une propriété d'un système mathématisé qui peut changer, mais il est susceptible de devenir déterminé dans un problème spécifique : il a un statut de nombre déterminé. Cependant, si l'on utilise un symbole pour représenter le paramètre, cela ne fait pas de ce symbole la représentation d'un nombre : c'est un élément du langage dénotant un nombre possible, c'est un signe ayant un rôle de paramètre.

On retrouve cette conception du paramètre sous le terme de *placeholder* dans le monde anglophone. Par exemple, Ely et Adams (2012, p. 21) le définissent ainsi :

> We use the word placeholder to mean a letter standing for a number that will be provided in a particular problem or context. A placeholder is often called a given or a constant ; in specific instances it is a parameter or a coefficient.

De même, on a vu chez al-Khwārizmī (p. 54) une identification des « variables du système » dans son « Chapitre sur les transactions » (Rashed, 2007, p. 198) : quatre « nombres » déterminant la situation sont isolés et mis en relation, et la solution du problème consistant à chercher la valeur d'un nombre est présentée de façon rhétorique, mais générique, en se basant sur les relations entre le « nombre inconnu cherché par le demandeur » et les « nombres évidents ». Il s'agit donc bien ici de mettre en relation l'inconnue et des « nombres qui seront fournis pour un problème ou contexte particulier » : ces nombres ont bien un statut de paramètre. Ils sont cependant évoqués, du point de vue du langage, non en identifiant chacun des nombres [99], mais de façon globale, par la catégorie de nombre à laquelle ils appartiennent dans le cadre du problème, les « nombres évidents ». On a bien une opposition entre nombre inconnu et nombres connus, mais il n'y a pas de mot ou de symbole spécifique de la théorie désignant ces valeurs et permettant de les manipuler. Le *rôle* du paramètre est présent, mais sa représentation dans un langage est absente.

Regardons à présent les définitions du paramètre données par quelques dictionnaires. Dans le CNRTL [100] :

> A. - MATHÉMATIQUES
> 1. *ALG.* [Dans une expr. math., une fonction ou une équation en x et y p.ex. et p.oppos. à *inconnue*] Variable susceptible de recevoir une valeur constante pour un cas déterminé et qui désigne certains coefficients ou certaines quantités en fonction desquels on veut exprimer une proposition ou les solutions d'un système d'équations (d'apr. Bouvier-George Math.

99. Dans la présentation du problème, des noms désignant une propriété du système sont utilisés (« taux », « prix », « quantité évaluée »...), mais il n'y a pas association nom et valeur (comme le suggère Chevallard) : dans la suite du problème, al-Khwārizmī évoque « le nombre de... ». Le « taux » n'est pas un nombre pour al-Khwārizmī.
100. https://www.cnrtl.fr/definition/param%C3%A8tre

> 1979) ; en partic., troisième variable m par exemple, en fonction de laquelle peut s'exprimer chacune des variables indépendantes x et y (m étant un réel fixe, c'est-à-dire qu'il est donné ou encore connu bien que non spécifié numériquement). *Étudier pour quelles valeurs du/des paramètre(s) l'équation est impossible ou indéterminée ; discuter l'équation selon les valeurs du/des paramètre(s) ; résoudre et discuter un problème dépendant d'un/de plusieurs paramètre(s). Une fonction à 1 paramètre [...]* (Ging.-Lauret 1973). [...] B. *P. anal.* [En raison de l'ambivalence de ce concept suggérant soit la notion de variable soit celle de constante] Élément de base variable (selon le cas : caractéristique ou donnée variable) entrant dans l'élaboration d'un ensemble qui constitue un tout.[...] — *INFORMAT.* Variable de type connu mais dont la valeur, l'adresse ou le nom ne sont précisés qu'au moment de l'exécution. Un programme admet des paramètres lorsqu'il est rédigé en fonction de plusieurs cas d'emploi, chaque cas d'emploi conduisant à fixer la valeur des paramètres (Ging.-Lauret 1973).

L'opposition connu-inconnu est bien présente, de même que le caractère déterminant du paramètre : lorsqu'il est fixé, il détermine un problème spécifique, une instance de la famille du problème représentée par les « variables ». Cette « ambivalence » entre variable et constante est soulignée — nous y reviendrons plus loin, car c'est une des raisons de la difficulté de construire le concept de paramètre (Serfati, 1997 ; Ely et Adams, 2012).

Le Robert [101] reprend le terme de variable, sans en noter l'ambivalence :

1. Mathématiques : Variable dont dépendent les coefficients de certaines équations → variable.
2. Informatique : Valeur, option dont le choix permet d'adapter une application, un périphérique à l'environnement de travail.

Le paramètre mathématique serait une variable, mais le paramètre informatique désignerait une valeur.

Le Larousse [102] quant à lui, reprend le paramètre comme variable du système mathématisé, comme une valeur, comme un nom ... :

> nom masculin (de grec *metron*, mesure)
>
> 1. Grandeur mesurable permettant de présenter de façon plus simple et plus abrégée les caractéristiques principales d'un ensemble statistique.
> 2. Élément en fonction duquel on explicite les caractéristiques essentielles d'un phénomène, d'une question : La pluie, l'obscurité sont des paramètres dont il faut tenir compte.
> 3. Nom donné à certains coefficients, à certaines quantités, autres que la variable ou l'inconnue, en fonction desquels on peut exprimer une proposition ou les solutions d'un problème.
> 4. Pour un arc paramétré (I, f), nombre réel variable qui décrit I ; pour une surface paramétrée (A, g), chacune des deux variables réelles de la fonction g.
> 5. En informatique, variable dont la valeur, l'adresse ou le nom ne sont précisés qu'à l'exécution du programme.

Là encore, le paramètre est défini comme étant une variable, mais aussi comme n'étant ni la variable ni l'inconnue... Du côté de la didactique, Chevallard (1989, p. 65) précise que les paramètres sont « les variables du système dont les valeurs sont supposées connues ». Alors, peut-on dire qu'un paramètre est une variable ? Pour y répondre, il faut au préalable définir ce qu'est une variable.

101. Robert, 2023b.
102. Larousse, 2023.

1.9.2 Variable

La variable est un élément essentiel du passage de l'arithmétique à l'algèbre :

> The introduction of the concept of the variable represents a critical point in the arithmetic-algebraic transition. This concept is complex because it is used with different meanings in different situations. Its management depends on the particular way of using it in problem-solving. (Malisani et Spagnolo, 2009, p. 19)

Mais qu'est-ce qu'une variable ? Suivant les domaines, les disciplines, les contextes, il semble que le concept de variable soit lui-même variable (Schoenfeld et Arcavi, 1988, p. 425).

Le Robert [103] reste très général :

> 1. Qui est susceptible de se modifier, de changer souvent (opposé à invariable). [...]
>   — **Sciences** : Qui prend, peut prendre plusieurs valeurs distinctes. Grandeur, quantité variable.
>   — nom féminin : Une variable : symbole ou terme auquel on peut attribuer plusieurs valeurs numériques différentes.
> 2. Qui prend plusieurs valeurs, plusieurs aspects (selon les cas individuels, les circonstances).
> 3. Conçu, fabriqué pour subir des variations.

Cette définition est davantage centrée sur l'adjectif que sur le nom, mais celle du nom « variable » dans le cadre très général des « Sciences » laisse entrevoir la possibilité d'un concept commun, liant symbole et valeur — ou valeurs —.

Le Larousse [104] donne les définitions suivantes, en se focalisant sur le nom et non l'adjectif :

> Élément qui peut prendre des valeurs différentes à l'intérieur d'un ensemble, d'un système, d'une relation.
> **Informatique** : Information identifiée par un nom ou par une adresse, et pouvant prendre une ou plusieurs valeurs numériques, logiques ou alphanumériques, au cours du déroulement d'un programme.
> **Logique et mathématiques** : Terme indéterminé qui, dans une relation ou une fonction, peut être remplacé alternativement par divers termes déterminés (constantes) qui en sont les valeurs.
> **Mathématiques** : Nom qui désigne un élément quelconque d'un ensemble. **Statistique** : Grandeur susceptible de varier dans un ensemble donné, et telle qu'à chaque valeur prise par cette grandeur puisse correspondre, au moins théoriquement, un effectif de personnes ou une fréquence en pourcentage.

Là encore, nous retrouvons différents éléments qui semblent constitutifs de la variable : d'un côté, une valeur, une grandeur, un élément ou une information ; d'un autre côté, un nom, un terme ou une adresse, qui dénote cette valeur.

Sur Wikipédia [105], on retrouve cette disjonction de définitions suivant les domaines ou disciplines :

> En mathématiques et en logique, une variable marque un rôle dans une formule, un prédicat ou un algorithme.

103. Robert, 2023c.
104. Larousse, 2020a.
105. *Variable* 2019.

> En statistique(s), une variable peut aussi représenter une qualité. Elle peut être quantitative ou qualitative.
> En probabilité, la variable aléatoire est une fonction.
> En physique, en biologie, en mécanique et en chimie, la variable représente un paramètre mesurable comme la température, le temps, la vitesse ou l'intensité.
> En informatique, une variable est un symbole (habituellement un nom) qui renvoie à une position de mémoire dont le contenu peut prendre successivement différentes valeurs pendant l'exécution d'un programme.
> En sociolinguistique, une variable est un mot dont la forme varie selon le genre, le nombre ou la fonction.

Ici la variable peut « marquer un rôle », représenter une valeur ou un paramètre, ou désigner un emplacement en mémoire.

Comme nous nous intéressons à la construction des concepts de variable ou de paramètre, il nous faut aussi chercher comment les didacticiens des mathématiques définissent ces concepts, et ce qu'ils ont établi concernant leur construction. Or, cette idée de *rôle* ou de *statut* se retrouve en didactique comme le rappellent Ely et Adams (2012, p. 21) :

> We use the term variable in keeping with the usage of Collis (1975), Küchemann (1978), and a number of other researchers, such as Philipp (1992), who terms it a varying quantity. When a letter is used as a variable, "the letter is seen as representing a range of unspecified values, and a systematic relationship is seen to exist between two such sets of values" (Küchemann 1981, p. 104).

En reprenant les travaux de Kücheman, on s'intéresse ainsi au « statut des lettres » (et non plus des symboles ou des termes). Ces éléments sont repris et détaillés par exemple dans (Briant, 2013, p. 33-38), et sont aussi présent dans les documents d'accompagnement sur le calcul littéral ou même dans les programmes :

> Les élèves sont progressivement familiarisés aux différents statuts de la lettre (indéterminée, variable, inconnue, paramètre) et du signe égal (pour fournir le résultat d'une opération, pour traduire l'égalité de deux représentations d'un même nombre, dans une équation, dans une identité). (Recherche (MENSR), 2015, p. 35)

Mais Küchemann (1981) ne cherchait pas à définir ce qu'était une variable ou un paramètre, il cherchait à comprendre comment les élèves interprétaient un symbole particulier, la lettre, qu'ils ont déjà rencontrée dans des contextes autres qu'algébriques. Mais si la lettre, en algèbre, a différents rôles ou statuts, quel est l'objet mathématique représenté par une lettre ? Comme le rappelle Briant (2013, p. 38), la lettre change de statut suivant le problème, et au cours du problème, passant dans l'exemple donné du statut de paramètre au statut d'inconnue puis à celui de nombre déterminé. Ely et Adams (2012, p. 21) donnent un autre exemple dans leur définition de la variable :

> There are two properties here, both of which are crucial in our historical story and for student learning. First, a variable is indeterminate, rather than determinate, meaning that it does not represent just one (or a few) specific numerical value(s) that can be determined from the information provided. Rather it is capable of assuming any of a (large) set of values, and it stands for a generic element of that set. Thus, for instance, in the equation $y = -\frac{1}{2}x + 6$, both x and y are variables. If one becomes specified with a particular value, then the other letter becomes an unknown that can be found. The second property that distinguishes a variable is that it is a part of a "systematic relationship"— a variable quantity can vary, and when it does there is some other quantity that varies with it. These two quantities are said to co-vary (Carlson et al. 2002).

Toute l'ambiguïté de la notion de variable se retrouve dans l'expression « both x and y are variables » :

— les auteurs parlent-ils de la *lettre* x ? Oui, puisqu'ils précisent ensuite « the other letter » ;
— les auteurs parlent-ils de la *variable* x ? Oui, puisque cela illustre la définition donnée, et c'est une des traductions possibles de « x and y are variables » : ce sont des variables ;
— les auteurs précisent-ils le *rôle* de l'objet représenté par le symbôle « x » ? Oui, puisque c'est une autre traduction de l'expression précédente : x et y sont variables. De plus l'autre lettre *devient* une inconnue. Mais l'inconnue est-elle un statut ou un objet mathématique ?

On devrait sans doute préciser que l'*objet* représenté par le *symbole* littéral « x » change de *rôle* suivant le problème. C'est ce que dit aussi Wagner (1983, p. 476), dans son article « What Are These Things Called Variables ? » : « The meaning of a literal symbol is derived from its role [as name, unknown, indeterminate, parameter, etc.], its domain of values, and its associated truth set [if used in an open sentence]. » Comme l'écrivent Ely et Adams (2012) ci-dessus, elle précise :

> Mathematically speaking, the most significant difference between letters and numerals is the one just alluded to — that numerals represent a single number but letters can represent, simultaneously yet individually, many different numbers, as in $0 < x < 20$ or $y = 3x + 2$. It is this property of simultaneous representation that we refer to when we call literal symbols variables, knowing full well that some of these symbols may, depending on the context, represent single unknown numbers or even constants !

L'objet *variable* serait ainsi un objet représenté par une lettre représentant (ou dénotant) elle-même plusieurs valeurs possibles, éventuellement une seule (pour le rôle d'inconnue ou de constante — de paramètre pris dans un singleton —). C'est cette vision de la variable qui est reprise dans (Moss, Czocher et Lamberg, 2018), dans laquelle les autrices différencient la lettre de la variable. Elles utilisent ainsi dans la première phrase de leur article la proposition « the use of letters in algebraic expressions and equations — variables — » (ibid., p. 10). Le tableau représenté figure 1.27 (p. 111) résume leur conception de la variable dans un sens général, avec des rôles particuliers suivant les problèmes.

C'est aussi le point de vue retenu par Malisani et Spagnolo (2009, p. 20), reprenant ce qu'en disaient Bardini et Radford :

> A variable is an algebraic object that can be replaced by a number (Bardini, Radford, & Sabena 2005). But the notion of a variable has a plurality of conceptions : generalised number (an indeterminate number that appears in generalisations and in general methods) ; unknown (a specific number but unidentified, its value could be calculated with consideration of the restrictions of the problem) ; "in functional relation" (relation of the variation to other variables) ; totally arbitrary sign (it appears in the study of algebraic structures) ; and register of memory (in informatics ; Usiskin 1988). The variable in functional relation is simply named "variable" by Kuchemann (1981), who defines it as follows : "the letter is seen as representing a range of unspecified values and a systematic relationship is seen to exist between two such sets of values". We shall define a variable in functional relation as a "thing that varies". This last expression chiefly points out that the letter can represent a range of unspecified values, even if the systematic relation is not rendered explicitly yet. From a historical point of view this notion precedes the conception of variable in functional relation.

La variable, objet algébrique couramment représenté par une lettre, a ainsi plusieurs rôles possibles — dont celui de variable, ou celui de constante — et dénote une valeur non déterminée *a priori*.

Table 1 The five meanings of variables, the problem context that elicited them, and a sample of student work are shown.

Student Meanings of Letters and Variables	Description	Example of a Task with Context	Expression or Equation from Student Work
Letter as a label	The letter is used as a label to identify a category and keep track of the number of items in that category. The letter itself represents one unit. The coefficient represents the number of items in that category.	Write an expression for the number of hexagons and pentagons on a soccer ball.	$20h + 12p$, which means 20 hexagons + 12 pentagons
Variable as a changing quantity	The variable is used to represent different or changing values.	Write an expression to determine the total cost of purchasing 4 packs of 6 cupcakes and a single cupcake. The packs of cupcakes and the single cupcake have different prices depending on different stores.	$4s + c$, which means the cost of 4 packages of 6 cupcakes + the cost of a single cupcake
Variable as a known value	In a known-quantity variable, the value of the variable is given.	The cost of 4 packages of 6 cupcakes is \$6 and the cost of a single cupcake is \$1. Using the expression for the cost of cupcakes ($4p + s$), substitute the given price for each variable to determine the total cost.	$s = 6$ $c = 1$ $4s + c = 4(6) + 1$ $= 24 + 1$ $= 25$ The total cost is \$25.
Variable as an unknown quantity	The variable is the unknown value in an equation.	Max has a certain number of soccer balls, and David gives him 4 more soccer balls. Now Max has 8 soccer balls. How many did Max have in the first place?	b is the number of soccer balls that Max has. $b + 4 = 8$ $-4 \quad -4$ $b = 4$
Variables as independent or dependent	One variable in an equation is the independent variable, and the other variable is the dependent variable. There is a relationship between the independent and dependent variable.	Write an equation for the perimeter of a square.	p is perimeter s is length of a side of the square $p = 4s$

Figure 1.27 – Cinq significations de la variable — ou de la lettre (Moss, Czocher et Lamberg, 2018, p. 12)

La définition donnée par le CNRTL concernant la variable en mathématique, informatique ou sciences physiques, s'oppose en partie à cette conception, :

> Symbole, terme, phénomène observable auquel on peut attribuer différentes valeurs prises dans un ensemble. Anton. *constante*.[106]

106. CNRTL, 2023b.

Le paramètre serait donc une variable dans ce sens, mais pas la constante. Or, qu'est-ce qu'une constante si ce n'est un paramètre déterminé dans un contexte particulier, ou un paramètre dont le domaine est un singleton ? La constante gravitationnelle G a généralement un rôle de constante, mais on peut imaginer un univers dans lequel cette valeur serait différente, et ce que cela induirait par exemple sur les processus de naissance des étoiles ou des galaxies : G serait dans ce cas le symbole d'une variable dans un rôle de paramètre.

La formulation de la variable mathématique sur Wikipédia, fruit d'une longue discussion débutée en 2006 [107], reprend ces divers éléments [108] :

> Dans les mathématiques supérieures et en logique, une variable est un symbole représentant, a priori, un objet indéterminé. On peut cependant ajouter des conditions sur cet objet, tel que l'ensemble ou la collection le contenant. On peut alors utiliser une variable pour marquer un rôle dans un prédicat, une formule ou un algorithme, ou bien résoudre des équations et d'autres problèmes. Il peut s'agir d'une simple valeur, ou d'un objet mathématique tel qu'un vecteur, une matrice ou même une fonction. Dans un polynôme, une fraction rationnelle ou une série formelle, la variable est remplacée par une indéterminée notée X.
>
> Il est d'usage d'utiliser un certain type de symbole pour l'objet que l'on souhaite représenter, par exemple les lettres de i à n pour les indices, les lettres de la fin de l'alphabet pour les vecteurs, ou bien ε pour un réel strictement positif ayant pour but de tendre vers 0.

Une autre formulation illustre cette vision de la variable :

> Chaque langue connaît des noms propres et des noms communs. Vercingétorix, la France, Sirius, trois, la révolution de février, le Soldat inconnu désignent chacun un seul objet, tandis que des mots tels que rat, pierre, je, hier, là-bas sont des termes ambigus. En mathématique, les termes ambigus sont appelés des variables. (Freudenthal, 2020)

Finalement, on pourrait dire qu'une variable, dans un certain langage, est ce qui est flou dans une expression écrite dans ce langage.

1.9.3 Variables mathématiques et variables informatiques

Dans les textes évoquant les variables d'un point de vue mathématique et d'un point de vue informatique, deux différences considérées comme importantes sont mises en exergue :

— la variable informatique représente une adresse en mémoire, un emplacement physique tandis que la variable mathématique est un objet algébrique abstrait ;

— la variable informatique peut changer de valeur en cours d'exécution, pas la variable mathématique.

C'est par exemple ce qu'on peut relever dans la partie suivante de l'article sur la variable mathématique de Wikipédia :

> Dans les langages de programmation impératifs, ce que les informaticiens appellent des variables sont des repères de valeurs qui évoluent au cours du temps, on parle aussi de références. Il s'agit donc plutôt de l'identification d'emplacements en mémoire. Si une variable informatique n'est pas initialisée, sa valeur est non définie. Quand on doit utiliser dans le même cadre le concept de variable mathématique et le concept de variable informatique, comme c'est le cas en sémantique des langages de programmation, on appelle la variable informatique un « emplacement » (« location » en anglais). [109]

107. https://fr.wikipedia.org/wiki/Discussion:Variable_(mathÃ©matiques)
108. *Variable (mathématiques)* 2020.
109. Ibid.

De même, Briant (2013, p. 86) commence ainsi une partie consacrée aux variables informatiques et mathématiques :

> La nécessité de stocker des informations (nombres, textes, etc.) au cours de l'exécution d'un programme informatique mène à la notion de variable. Les variables informatiques sont des emplacements de mémoire physiques de l'ordinateur, repérés par des adresses binaires. Tout programme commence par la déclaration des variables [110], où l'on précise leur nom, leur contenu et leur type (numérique, alphanumérique, booléen, etc.). La première catégorie d'instructions qu'un ordinateur comprend est l'affectation d'une valeur à une variable, selon le type défini.

La variable informatique est donc une version numérique [111] de la *Variable* de Ada Lovelace, qui repérait la colonne du *Storehouse* stockant une certaine valeur (voir p. 81).

Mais Ada Lovelace avait déjà montré qu'une *Variable* est mise en relation non seulement avec la valeur qu'elle dénote, mais aussi avec la variable mathématique qui dénote cette valeur. On a donné l'exemple suivant p. 82 :

$$\begin{array}{c} V_3 \\ \hline \oplus \\ 0 \\ 0 \\ 9 \\ 8 \\ \hline x \end{array}$$

Cela pourrait être une illustration concrète de ce que Briant (ibid., p. 85) appelle « Double transposition de la résolution d'un problème mathématique en vue de sa programmation » (figure 1.28, p. 114) : un algorithme mathématique est transposé en un « algorithme informatisé, écrit en pseudo-code, où la structure de l'algorithme tient compte des actions élémentaires réalisables par une machine. » ; la seconde transposition consistant en l'écriture du programme dans un langage spécifique. Briant donne comme exemple l'algorithme permettant de de simplifier la racine carrée d'un entier naturel donné N, soit écrire $\sqrt{N} = a\sqrt{b}$. Une possibilité est la suivante, pour la version « mathématique » de l'algorithme (ibid., p. 84) :

> Pour chaque entier I compris entre 1 et $\text{Ent}(\sqrt{N})$:
>
> — Tester si la division de N par I^2 donne un reste nul ;
>
> — Si c'est le cas, affecter à a la valeur de I ; (1)
>
> — Si ce n'est pas le cas, passer à la valeur suivante de I ; (2)
>
> Calculer la valeur de $b = \frac{N}{a^2}$

On voit ici poindre une nouvelle ambivalence [112] :

110. En fait, la plupart des langages modernes n'imposent pas une déclaration en tête de programme ou de bloc de code.

111. Dans le sens de « traitée par un système électronique ».

112. On peut aussi noter l'ambiguïté d'un algorithme rhétorique :

— les instructions (1) et (2) laissent entendre que si le reste est nul, on affecte a et on sort de la boucle, puisque si ce n'est pas le cas on doit passer à la valeur suivante. Dans ce cas l'algorithme renverra toujours $N = N\sqrt{1}$;

— sinon, cela signifie que l'instruction (2) n'est pas nécessaire : on passe de toute façon à la valeur suivante de I, puisqu'implicitement on recherche la plus grande valeur de a.

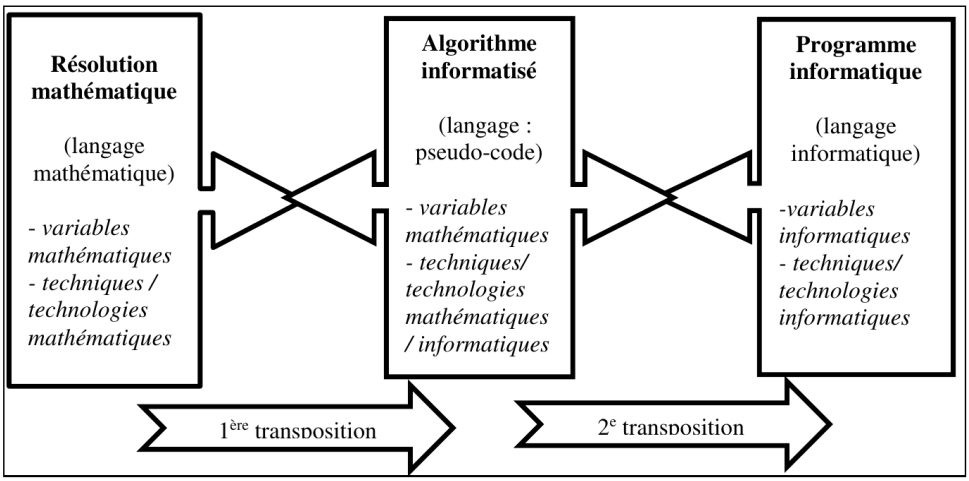


Figure 1.28 – Double transposition de la résolution d'un problème mathématique en vue de sa programmation (Briant, 2013, p. 85)

— N, I, b et a sont ici des variables mathématiques,
— aucune référence n'est faite à la mémoire de l'éventuelle machine sur laquelle on pourrait implanter cet algorithme,
— mais I et a changent de valeur lorsqu'on exécute l'algorithme.

I et a sont-elles alors des variables mathématiques ou informatiques ? On voit que la caractéristique « Une variable qui peut changer de valeur en cours de problème » n'est pas suffisamment discriminante. On pourrait objecter qu'il s'agit là d'une écriture déjà « informatique » d'une résolution mathématique. Mais comment en faire un énoncé dans lequel chaque variable est non mutable ? On pourrait considérer les variables $I_1 = 1$, $I_2 = 2$ etc., mais dans ce cas il faudrait une formulation de l'indice, donc exprimer I_i, i changeant de valeur... Pour supprimer la boucle, on pourrait écrire cet algorithme sous forme d'une fonction récursive, et dans ce cas on se retrouve dans le paradigme de programmation fonctionnelle dans lequel les variables sont des variables mathématiques (transparence référentielle). De plus, N et b ne vont pas changer de valeur au cours de l'exécution, y compris lorsque ces variables seront concrétisées dans un certain langage. Par exemple en Javascript ou pourra écrire :

```
const N=parseInt(prompt()); // ou const N=28;
...
const b=N/(a*a)
```

La variable b, d'un point de vue informatique désigne tout à la fois un espace mémoire et une valeur (c'est le côté $V_3 \to [0,0,9,8]$ de l'exemple Lovelace), et d'un point de vue mathématique désigne une valeur (le côté $x = 98$ de l'exemple de Lovelace). Considérer que b n'est pas une variable mathématique, c'est ne considérer que l'objet « matériel », et non l'objet abstrait : une variable informatique est une variable disposant d'une représentation dans un certain langage pour une certaine machine, et une variable mathématique est une variable disposant d'une représentation dans le langage algébrique. De plus, ces représentations de variables n'ont de sens que si elles sont manipulées dans des expressions exprimant des relations entre d'autres variables du système mathématisé. a n'a pas de sens en soit, mais a un sens dans le cadre du problème considéré et par sa relation $N = b\sqrt{a}$.

Samurçay (1985, p. 144) précisait déjà :

> La formule "une variable c'est une adresse", ne rend que très partiellement compte du caractère "invariant" que présente, paradoxalement, le concept de variable : en effet, si la valeur d'une variable varie, comme il se doit, non seulement sa désignation mais également sa relation fonctionnelle avec les instructions et les autres variables du programme sont, elles, invariantes. Le caractère invariant se manifeste évidemment par la donnée d'un nom à la variable, mais aussi par le contrôle exercé sur l'ensemble des valeurs qu'elle peut prendre et, dans certains types de langages, par la précision d'un type qui détermine les opérations qu'on peut effectuer sur les variables. De ce point de vue, l'étude du concept de variable rejoint des études plus générales de la formation des connaissances et notamment des invariants opératoires.

Concernant la variable informatique, on pourrait dire que c'est un objet représenté par un mot (ou un symbole) du langage de description de l'algorithme, représentant (ou dénotant) plusieurs valeurs possibles, éventuellement une seule, manipulable dans des expressions du langage comme si cette valeur était connue : lorsque j'écris dans un programme l'affection $y \leftarrow 4x+3$, x et y existent comme objets abstraits, et seront concrétisés matériellement à l'exécution. En effet, d'un point de vue effectif, lorsque l'algorithme, écrit dans ce langage pour une certaine machine, est exécuté — donc lors du calcul effectif —, la variable informatique (dans le cas du paradigme de programmation impérative) désigne aussi la représentation concrète de la valeur : dans une table à poussière, avec un abaque, sur des colonnes d'engrenages ou dans la mémoire électronique d'une machine.

La différence avec la variable mathématique serait donc plutôt à chercher du côté de la représentation concrète. Ainsi par exemple, un codage sur huit bits d'un entier positif ne permettra de représenter uniquement les entiers de 0 à 255. De même, le codage traditionnel en binaire de nombres décimaux n'est pas toujours cohérent avec la réalité arithmétique : l'expression « $0.1+0.2 = 0.3$ » est ainsi fausse dans la plupart des langages de programmation, la représentation binaire de 0.3 nécessitant une infinité de bits. C'est notamment ici que la deuxième transposition évoquée par Briant intervient : l'implémentation concrète, dans un certain nombre de cas, doit prendre en compte les particularités du couple (machine, langage) [113].

Cette transposition concerne aussi la nécessité, en algorithmique ou informatique, d'expliciter tous les calculs. La variable de travail déjà évoquée par Ada Lovelace est ainsi une variable qui n'est pas nécessaire dans la résolution du problème mathématique, mais qui le devient lorsqu'on passe à une résolution informatique. Il y a donc, en informatique, au moins deux types de variables :

> Dans les situations-problèmes que rencontre effectivement le sujet, les variables interviennent avec des statuts fonctionnels différents. Dans un premier temps on peut en distinguer deux :
>
> — les variables qui sont les données explicites du problème ;
>
> — les variables qui sont rendues nécessaires par la solution informatique.
>
> Par exemple, la solution d'un problème d'échange des valeurs de deux variables A et B fait intervenir ces deux types de variables : A et B sont des données du premier type, la variable intermédiaire C qu'il est nécessaire d'utiliser pour mémoriser le contenu d'une des variables au moment de l'échange est du second type [114]. (ibid., p. 146)

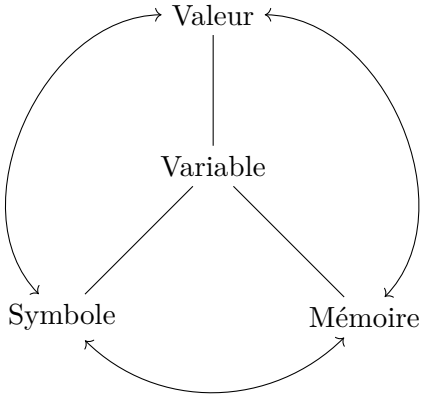
Ainsi, d'une part, la variable, en mathématique comme en informatique, a différents rôles ou statuts, et d'autre part lorsqu'on parle de variable :

113. Notons néanmoins que le calcul effectif d'une procédure algébrique par un mathématicien mobilise aussi des outils manipulant des représentations de nombres, par exemple la calculatrice. Dans certains cas, là aussi, la prise en considération des contraintes de la machine devront être prises en compte.

114. Pour échanger A et B il faut ainsi opérer les assignations suivantes : $C \leftarrow A$; $A \leftarrow B$; $B \leftarrow C$; Dans certains cas, notamment si A et B sont des entiers, on peut se passer de variable intermédiaire aux dépens de la lisibilité : $A \leftarrow A + B$; $B \leftarrow A - B$; $A \leftarrow A - B$. C'est aussi une bonne illustration des difficultés engendrées par le changement de valeur d'une variable en cours de programme, que l'on peut illustrer en marquant les états : $A_1 \leftarrow A_0 + B_0$; $B_1 \leftarrow A_1 - B_0$ donc $B_1 = A_0 + B_0 - B_0 = A_0$; $A_2 \leftarrow A_1 - B_1$, donc $A_2 = A_0 + B_0 - A_0 = B_0$.

— on considère un symbole dénotant une certaine valeur (non forcément déterminée ou connue *a priori*), que l'on peut manipuler : c'est le côté mathématique (ou informatique en programmation fonctionnelle), la partie gauche des relations représentée ci-dessous, l'aspect sémiotique (ou celui de calcul formalisé) ;

— on considère un symbole dénotant une localisation spatiale concrète contenant une représentation d'une certaine valeur : c'est le côté informatique (en programmation impérative), à droite dans le schéma ci-dessous, l'aspect concret (ou celui du calcul effectif).



1.9.4 Définitions

Suite à ces éléments épistémologiques et didactiques concernant la variable, nous pouvons tenter de formaliser les définitions de la variable et du paramètre. Nous cherchons des définitions exploitables tant dans un registre informatique que mathématique, et d'un point de vue épistémologique tout autant que didactique.

Considérant que, d'un point de vue sémiotique (et non du point de vue de la représentation concrète), la différence entre variable informatique et variable informatique se joue au niveau du langage formel — que nous désignerons par θ [115] — dans lequel elles sont représentées, nous utiliserons la définition suivante de la variable :

Définition 6 (Variable) *Dans un langage θ une variable est un symbole du langage θ représentant un objet non forcément déterminé ou connu* a priori*, manipulable dans des expressions de θ comme si cet objet était connu.*

Cette définition est fonctionnelle pour le langage algébrique comme pour un certain langage informatique. Notons bien que si nous considérons ici la variable du point de vue sémiotique, nous n'omettons pas le fait que la concrétisation de l'usage des variables en informatique n'est pas totalement identique à la concrétisation de l'usage des variables en algèbre.

Que ce soit en informatique ou en algèbre, la variable, telle qu'on l'a définie, peut prendre plusieurs rôles ou statuts, non seulement en fonction du problème, mais aussi en fonction de l'étape de résolution du problème. Briant (2013, p. 38) donne la traduction suivante :

> En outre, le sens de la lettre comme paramètre ou comme inconnue ou variable, pourrait changer au cours du processus de résolution d'un problème [...]. La résolution de ce problème (« Trouver l'équation de la droite de pente 3 passant par (2, 5) ») commence par l'écriture d'une équation $y = ax + b$, où l'on sait communément déterminer que x et y sont des variables

115. En référence au « langage de la théorie de Diophante ».

> alors que a et b sont des paramètres. Le processus se poursuit par la substitution de a par 3, et la résolution d'une équation avec b pour inconnue, et où les constantes 2 et 5 sont substituées à x et à y. Le processus se termine par la substitution des constantes trouvées pour a et b, et en laissant les variables x et y dans la formule : $y = 3x + 1$. (Bloedy-Vinner, 1994)

La variable, en mathématique, peut ainsi avoir un rôle d'inconnue, de paramètre, d'indéterminée... ou de variable. Cette formulation « variable dans un rôle de variable » est problématique, cependant elle reste circonscrite à la mathématisation du système, pour reprendre les termes de Chevallard : il s'agit ici d'établir des relations ou des co-variations entre les différentes propriétés du système, et non de poser un problème. « $3a + 7$ » n'a pas de sens en soit (à part d'être un nombre si a est un nombre). En revanche, écrire $b = 3a + 7$ (ou « le nombre de bidules est $3a + 7$ »), ou $3a + 7 = a + 1$, établit des relations entre les éléments (les variables) du système mathématisé. Dans $b = 3a + 7$, a et b sont des variables dans le sens d'une manifestation d'une covariation. Si le problème, dans ce système, est « déterminer a pour que $3a + 7 = b$ », a prend alors le role d'inconnue, b de paramètre.

Pour la variable informatique, du moins en programmation impérative, Sajaniemi (2002) a, lui aussi, identifié différents rôles possibles (figure 1.29, p. 118), en les organisant selon l'évolution de leur complexité en terme d'apprentissage (figure 1.30, p. 118). Il précise aussi que ces rôles — comme en mathématique — sont amenés à changer au cours de l'exécution du programme, notamment lorsqu'une variable est mobilisée dans deux boucles (Sajaniemi, 2005, p. 6). Dans le même passage, il suggère aussi que, comme pour les variables mathématiques, le rôle perçu pouvait être variable suivant les individus. Il donne ainsi l'exemple de la suite de nombres 1,1,2,3,5,8,13... : un mathématicien la reconnaitra comme la suite de Fibonacci, une variable parcourant ces nombres sera alors un *stepper*, dans le sens d'un parcours organisé de différentes valeurs connues ; un novice y verra un *gatherer*, un accumulateur utilisant des valeurs précédentes pour en produire de nouvelles. Samurçay (1985, p. 146) avait déjà évoqué la différence de perception, mais aussi la différence cognitive de ces deux rôles :

> Le sujet peut par ailleurs, se représenter la fonction de certaines variables mieux que d'autres, suivant leur signification. Par exemple, les variables qui jouent le rôle de compteur, font appel dans l'opération de mise à jour à des activités cognitives plus faciles que celles qui jouent un rôle de "récapitulation" des résultats :
>
> compteur := compteur + 1
>
> se réalise par l'ajout d'une constante. La fonction sous-jacente est une fonction de succession et non de sommation comme dans le cas de
>
> somme := somme + nombre
>
> qui indique l'addition à la valeur de la variable d'accumulation de la valeur d'une autre variable.

Le rôle de *fixed value*, quant à lui, est équivalent au rôle de paramètre. Sajaniemi rajoute le littéral et la constante, non comme des rôles, mais comme des constructions du langage. En effet, littéral et constante ont un rôle de paramètre, mais du point de vue concret les valeurs seront non pas mémorisées mais directement inscrites dans le code produit par la compilation.

La variable dans un rôle de paramètre apparaît comme étant un concept à l'interface entre algèbre et informatique : c'est une variable nécessaire à la représentation et la résolution mathématique ou informatique du problème, mais ce n'est pas une variable spécifiquement informatique, nécessitant une transposition, notamment car sa valeur — en général [116] — ne change pas. Nous pourrions définir le paramètre et sa nécessité de la façon suivante :

116. Comme l'avait déjà noté Lovelace, une variable d'entrée, ce qui est en fait un paramètre, peut être utilisée comme variable de travail : elle change alors de statut et n'est plus un paramètre.

Role	Informal description
Fixed value	A variable initialized without any calculation and not changed thereafter
Stepper	A variable stepping through a systematic, predictable succession of values
Follower	A variable that gets its new value always from the old value of some other variable
Most-recent holder	A variable holding the latest value encountered in going through a succession of values, or simply the latest value obtained as input
Most-wanted holder	A variable holding the best or otherwise most appropriate value encountered so far
Gatherer	A variable accumulating the effect of individual values
Transformation	A variable that always gets its new value with the same calculation from values of other variables
One-way flag	A two-valued variable that cannot get its initial value once its value has been changed
Temporary	A variable holding some value for a very short time only
Organizer	An array used for rearranging its elements

Figure 1.29 – Rôles des variables à un niveau de programmation novice (Sajaniemi, 2005, p. 7)

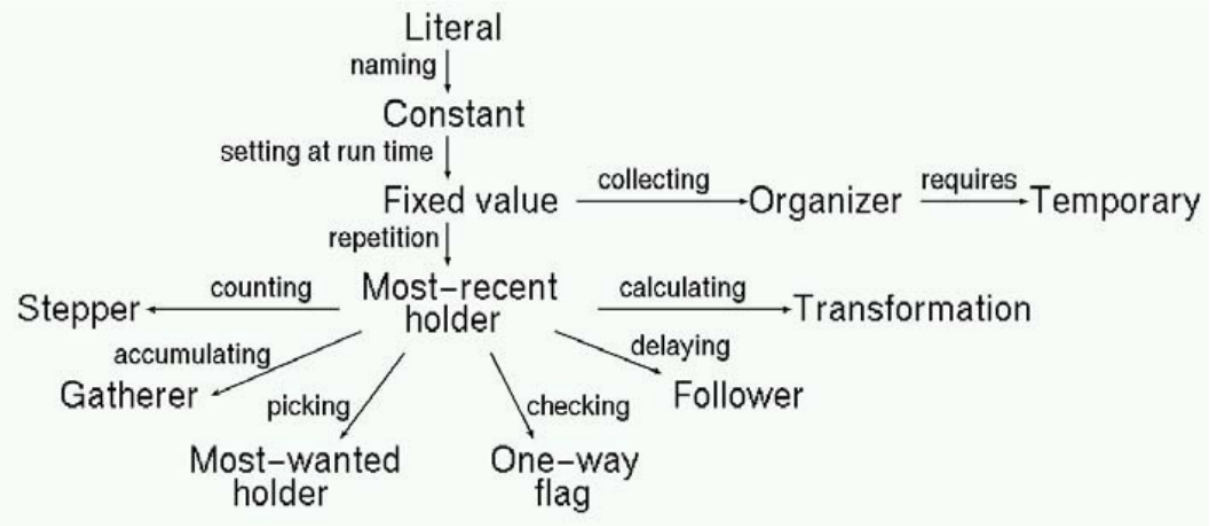


Figure 1.30 – Relations entre les rôles des variables pour un apprentissage incrémental (Sajaniemi, 2005, p. 8)

Définition 7 (paramètre) *Un paramètre est une variable (dans le sens défini plus haut) dont la valeur est supposée connue (déterminée) lors du calcul effectif (exécution de la méthode, du procédé, de l'algorithme) et impliquée dans d'autres calculs (ou instructions).*

On retrouve ici la notion de paramètre au sens de Chevallard (1989) : une variable du système mathématisé dont la valeur est supposée connue.

Ainsi, construire la nécessité de la variable dans un rôle de paramètre reviendrait à la « définition » suivante :

Définition 8 (Nécessité du paramètre) *Il s'agit d'établir la nécessaire existence d'un objet représentant une certaine propriété non définie* a priori*, ainsi que la nécessaire existence, dans le registre de représentation sémiotique de la construction du problème (θ), d'un symbole permettant de manipuler cet objet comme s'il était connu, dans une expression de θ mettant en relation la valeur que dénote l'objet et au moins une autre valeur du système mathématisé.*

L'inconnue est définie par sa mise en relation avec des nombres ou des représentants de nombres ($x = 4 - \frac{7}{3}$ ou $x = b - \frac{7}{3}$). Le paramètre, lui, va définir la relation. Ferdinando Arzarello et Sandra Bogossi nous ont fait la remarque suivante [117] :

> The writing $y = mx$ denotes the bundle of lines with center the origin and adopting a formal notation it can be represented as $\{\{(x, y)|y = mx\}|m \in \mathbb{R}\}$. In it we notice that firstly are quantified the variables x and y and in a second moment the parameter m. **This is exactly the connoting characteristic of the parameter : it is the variable which logically is quantified last, that is the variable on which acts the more external quantifier**. The same situation reoccurs in the case of equations or problems with a parameter : when the text of an exercise says "for each value of m find the values of x such that...", x is for sure the unknown and m is the parameter, not because of the conventional use of the chosen letters, but because m is quantified more externally. This kind of formulation is defined in Bloedy-Vinner (2001) as dynamic or again as an ordered and quantified structure because it corresponds to a potential order of substitutions that is typical of parametric expressions.[...] The two contexts above show an important connection between mathematics and computer science through a logic analysis of language(s).

Dans la situation que nous analyserons (2.1, p. 131), les variables du système mathématisé sont ainsi la valeur de la mesure (que nous noterons n), le nombre d'hexagones total du motif (que nous noterons N), le nombre de répétitions de chacune des boucles (que nous noterons b_i, avec $i \in [1..8]$). Le problème définit alors le rôle de chacune de ces variables. Ainsi, pour généraliser, les élèves doivent trouver l'expression permettant de déterminer le nombre de répétitions de chacune des boucles. On pourrait représenter ce problème ainsi (considérant que l'ensemble des valeurs des boucles détermine le tracé du motif, et que toutes les valeurs sont entières) :

$$\forall n, \texttt{le tracé est valide} = vrai \implies \exists b_1, b_1 = n - 1$$

Le quantificateur le plus externe agit sur le paramètre du problème, ici n. Lorsqu'on cherche à identifier l'instance tracée par un script, ce problème pourrait se formuler de la façon suivante, montrant le rôle de paramètre de b_1 :

$$\forall b_1, \texttt{le tracé est valide} = vrai \implies \exists n, n = b_1 + 1$$

D'un point de vue plus spécifiquement mathématique, les problèmes « trouver le nombre total d'hexagones connaissant la mesure » (1) et « trouver la mesure du motif formé d'un certain nombre d'hexagones » (2) montrent bien quelle variable prend le rôle de paramètre, et quelle variable prend le rôle d'inconnue :

$$(1) : \forall n, \texttt{le tracé est valide} = vrai \implies \exists N, N = 9n - 12$$
$$(2) : \forall N, \texttt{le tracé est valide} = vrai \implies \exists n, N = 9n - 12$$

117. Commentaires à une présentation lors du colloque ETM7, Strasbourg 2022 (Legrand, 2023).

Ainsi, la variable dans un rôle de paramètre est essentielle pour exprimer formellement une relation ou un problème générique, et sa représentation sémiotique est à distinguer de celle de l'inconnue. Mais comme nous l'avons vu, la représentation symbolique du paramètre est assez tardive dans l'histoire de l'algèbre. La construction de ce concept a dû nécessiter une rupture épistémologique, ce qui induit le risque d'un obstacle épistomologique. Nous terminerons cette partie par une réflexion sur ce sujet, et en abordant les implications didactiques qui en résultent.

1.9.5 Variable dans un rôle de paramètre : obstacles et pistes

Entre les premières dénominations de l'inconnue par les babyloniens et la représentation symbolique du paramètre par Viète et Descartes, il s'est écoulé plus de trois millénaires. Comment expliquer cette arrivée tardive du paramètre ? Quelles sont les conditions de son émergence ? Et quelles pistes didactiques cela implique-t-il ?

À la suite de Furinghetti et Paola (1994, p. 2), on peut penser que des obstacles épistomologiques ont empêché cette émergence :

> In line with (Harper, 1987) it is reasonable to think that this fruitful convention has taken such a long time to emerge because of epistemological obstacles.

Cette compréhension de l'histoire du concept et de ses obstacles peut nous donner des indications sur les obstacles que pourraient rencontrer les élèves lors de la construction du concept de paramètre :

> The historical story is quite relevant to our understanding of student learning, but not because we expect students to go through exactly the same development. Some detailed theories account for relationships between the historical development and student conceptual development, such as the theories of epistemological obstacles (Brousseau 1997) and of genetic epistemology (Piaget and Garcia 1989). (Ely et Adams, 2012, p. 23)

C'est ce qu'affirment aussi Bächtold, Durand-Guerrier et Munier (2017, p. 10) :

> L'épistémologie, en s'appuyant sur l'histoire des sciences, permet de comprendre comment ces connaissances ont été constituées : dans quel contexte scientifique, socio-culturel et économique, sous l'impulsion de quelles motivations, suivant quelles méthodes, quelles étapes et en surmontant quels problèmes. Elle permet également de cerner le statut et les fonctions des différents concepts, modèles et théories scientifiques. En mettant ces analyses en regard avec les conceptions des apprenants et avec les pratiques enseignantes, l'épistémologie offre un éclairage précieux pour repérer les obstacles aux apprentissages et déterminer les étapes possibles de l'appropriation par les élèves des connaissances scientifiques (Bachelard, 1938 ; Piaget, Inhelder, 1966 ; Martinand, 1986 ; Brousseau, 1998 ; Viennot, 2008).

Ely et Adams (2012, p. 23) considèrent ainsi deux changements historiques de pratique (par rapport à l'utilisation de l'inconnue), que l'on souhaite aussi voir chez les élèves : l'usage d'une lettre pour une quantité indéterminée, pas uniquement pour l'inconnue, ainsi que la représentation et la détermination de la manière dont une quantité change par rapport à une autre. Ils identifient aussi deux changements de point de vue qui ont été nécessaires chez les mathématiciens, mais qui ne le seraient pas pour les étudiants actuels : avoir une vue plus générale d'un objet mathématique, et considérer les propriétés d'un objet comme étant déterminables, quantifiables.

Plus précisément, ils considèrent qu'un changement de façon de penser a été nécessaire :

> In order to use a symbol as a placeholder rather than as an unknown, a shift in thinking was needed that allowed a symbol to refer to a more general kind of object. For the ancient Greek mathematicians, the manipulations and representations that could be used for a particular mathematical object depended upon the actual existence and properties of the object itself, the ontology of the mathematical object. (ibid., p. 25)

On a vu, par exemple avec al-Khwārizmī ou même avec Viète, que ces mathématiciens éprouvaient des difficultés à se détacher de l'ontologie des objets qu'ils manipulaient. Il s'agit donc de pouvoir *identifier un objet à sa représentation* et *remplacer un objet déterminé par un objet susceptible d'être déterminé*, ce que ces auteurs affirment impensable chez les mathématiciens grecs. Cette rupture de façon de penser, cette rupture épistémologique, est aussi évoquée par Radford (2006b, p. 519) :

> The difference between "ancients" and "moderns" can be explained through an epistemological shift that occurred in the post-feudal period. Referring to 16th Century "modern" epistemology, Hanna Arendt argues that the focus changed from the object to be known to the process of knowing it.

Notons que l'inconnue, si elle n'est pas déterminée dans le sens « connue » en début de problème, est déterminée parce qu'elle est une valeur *déjà là*, une valeur existant indépendamment de l'exécution ou non de la procédure de résolution. L'inconnue est *un nombre* qu'il reste à découvrir, alors que le paramètre dénote *n'importe quel nombre possible* dans un certain ensemble. Cette différence est présente dans la symbolisation algébrique :

> Viète introduisit un nouveau système de signes, uniquement constitué de lettres, et dont la fonction véritable consista, en dernière analyse, à faire prendre en charge par l'écriture symbolique deux concepts jusqu'alors considérés comme opposés : l'arbitraire et le fixé ou, plus significativement, le quelconque et le singulier. (Serfati, 1997, p. 129)

Avant Viète, « le connu de tous se confondait avec le numérique [avec les nombres], et lui-même avec le Donné » (ibid., p. 137) : les données, les paramètres, étaient représentés par des nombres, ayant éventuellement un caractère d'exemple générique. Mais comme le soulignent Bardini, Radford et Sabena (2005, p. 136) :

> A variable is not a number in the arithmetic sense. A number, e.g. the number 3, does not vary. A variable is an algebraic object.

Mais comment faire lorsque le « donné » n'est pas donné ? Comment le connu peut-il être arbitraire tandis que l'inconnu est fixé ? Cette tension fixe-variable ou unique-multiple a dû être dépassée pour aboutir à la symbolisation du paramètre. Il est ainsi probable que confronter les élèves à cette tension et leur permettre de la dépasser soit une étape nécessaire pour construire le concept de paramètre : c'est un seul symbole, mais qui peut dénoter de nombreux nombres possibles. C'est d'ailleurs une difficulté déjà notée chez les élèves, qui ont des difficultés à considérer qu'une même lettre puissent prendre différentes valeurs, comme dans les problèmes de type « calculer $3x + 7$ pour $x = 2$ et $x = 7$ »[118].

Comment alors faire vivre cette tension ? Où, pour poser la question autrement, comment en est-on venu à dépasser cette tension à l'époque de Viète et Descartes ?

118. Notons que ce problème est un exemple de situation dans laquelle la variable mathématique x change de valeur.

Radford (2006b) considère que les éléments culturels et économiques ont eu une influence essentielle. Sans oublier l'influence de l'imprimerie, il souligne deux aspects selon lui « cristallisants » : la *valeur* et l'*efficacité* (ibid., p. 520). La *valeur*, c'est-à-dire la recherche d'une équivalence entre des tâches ou des objets non comparables *a priori*, est ce qui permet d'additionner un travail et des chevaux. Celle ci serait « one of the greatest mathematical conceptual categories of the Renaissance » (ibid., p. 513), et permet d'établir des relations abstraites et calculables entre différentes choses :

> In terms of **representations**, value made it possible to see that one thing could take the place of another, or, in other terms, that one thing (a money coin, e.g.) could be used to *represent* something else. And this is the key concept of algebraic representation. [...] Without a doubt, value has shown that representation is arbitrary in the sense that the value of a thing does not reside in the thing itself but in a series of contextual usage values, and we know that the arbitrariness of the signifier is one of the key ideas of algebraic representation. (ibid., p. 513-514)

D'un autre côté, même si l'imprimerie et la diffusion imprimée des savoirs algébriques ont rendu nécessaire une formalisation du langage utilisé, ce n'est pas la seule raison de l'émergence du symbolisme algébrique, concrétisé par la représentation du paramètre. Comme le signale Serfati (1997, p. 137) :

> Dans les problèmes "en nombres", la représentation partiellement symbolique du XVI^{e} siècle pouvait donc apparaitre comme techniquement satisfaisante — par l'automatisme induit dans le calcul — en même temps que largement insuffisante méthodologiquement, par le défaut d'universalité qu'elle véhiculait.

Or, au XVI^{e} siècle, rappelle Radford en citant Klein, on passe de questions sur les objets aux questions sur les méthodes (Radford, 2006b, p. 519), et on est notamment à la recherche de méthodes claires et efficientes. Nous avons déjà évoqué cette recherche d'efficacité (facilité d'application, rapidité, absence d'erreur) présente pour des raisons économiques ou militaires, et tout particulièrement depuis le début du XVII^{e} : non seulement elle a produit la nécessité de machines, mais aussi la nécessité de méthodes automatiques, algorithmisables. Radford (ibid., p. 519-520) le souligne aussi :

> The use of letters in algebra, I want to suggest, was related to the idea of rendering the algebraic methods efficient in the previous sense, that is to say, in accordance to the general 16th century understanding of what it means for a method to be clear and systematic, an understanding that rested on the idea of efficiency in the technological sense. You write down your unknowns, and then you translate your word-problem. Now you no longer have words with meanings in front of you. What you have is a series of signs that you can manipulate, in a machine-like manner, in an efficient way.

1.10 Conséquences didactiques et questions de recherche

Le concept de variable est donc un concept essentiel, tant pour l'algèbre que pour l'informatique, mais il n'est pas toujours construit par les élèves de collège. D'une part parce que, selon Schoenfeld et Arcavi (1988, p. 420), « Despite the importance of the concept [of variable], however, most mathematics curricula seem to treat variables as primitives terms that — after some practice of course — will be understood and used in a straightforward way by most students. ». Présenter la lettre et son usage, tout comme présenter la boucle et son usage, ne sont sans doute pas suffisant pour permettre de construire le concept de variable ou d'itération. D'ailleurs, Coppé et Grugeon (2009, p. 4) constatent que :

— Les élèves éprouvent des difficultés à mettre en œuvre un calcul littéral (respectivement algébrique) fonctionnel au collège (respectivement au lycée).

— Les élèves ont du mal à mobiliser une lettre pour résoudre un problème si on ne la leur donne pas.

— Les approches visant la généralisation ou la modélisation sont encore peu mises en avant dans l'enseignement actuel de l'algèbre.

— L'algèbre est encore enseignée en privilégiant la dimension objet plutôt que la dimension outil.

Suivant les travaux de Kieran entre autres, elles soulèvent l'intérêt de résoudre des problèmes nécessitant l'outil algébrique, afin de construire les concepts algébriques :

> Les différents emplois de l'algèbre permettent l'émergence des nouveaux objets de l'algèbre (expressions algébriques, formules, équations et systèmes d'équations, identités). Ils peuvent engager les élèves à donner du sens aux différents statuts des lettres (nombres généralisés, variables, inconnues, indéterminées, paramètres) en relation avec ces emplois de l'outil algébrique mais aussi avec l'évolution du raisonnement algébrique (expression de méthodes générales pour résoudre des classes de problèmes) (Bell 1996). (ibid., p. 6)

On retrouve ici, une fois de plus, le lien avec les algorithmes (méthodes organisées et effectives de résolution d'une famille d'instance de problèmes). Nous allons ainsi *chercher une situation, à l'interface entre algèbre et informatique, qui permette d'étudier comment et à quelles conditions les élèves peuvent construire le concept de variable dans un rôle de paramètre.*

D'autre part, les problèmes de généralisation de motifs (*patterns*), comme par exemple le « carré bordé », sont souvent évoquées comme favorisant potentiellement l'entrée dans l'algèbre (Grugeon-Allys et Pilet, 2017, p. 14 ; Kieran et al., 2016, p. 5). Ils s'appuient « sur la coordination de plusieurs registres de représentations sémiotiques : programmes de calcul exprimé en français, représentations figurées [...], écritures numériques et littérales » (Coppé et Grugeon, 2009, p. 9), ce qui permet de donner « du sens aux lettres » et d'« associer à plusieurs expressions une seule dénotation ». Cependant, si le raisonnement analytique est présent même s'il n'y a pas de recherche de symbolisme formel, cette recherche est nécessaire si l'on veut faire vivre le critère d'analycité de Radford (2014, p. 260). Ainsi « cette situation ne peut vivre que si le professeur organise un milieu adapté avec un appui sur des formulations, des gestes et des artefacts pour favoriser l'accès au sens des expressions » (Coppé et Grugeon, 2009, p. 9). Selon Radford (2006a, p. 5) :

> Generalizing a pattern algebraically rests on the capability of grasping a commonality noticed on some elements of a sequence S, being aware that this commonality applies to all the terms of S and being able to use it to provide a direct expression of whatever term of S.

Mais comment amener les élèves à construire la nécessité d'une représentation symbolique de l'expression ? L'utilisation du langage naturel peut être suffisant pour exprimer une méthode systématique. Ainsi, dans le problème des « chaines du joaillier », Bronner et Squalli (2021, p. 17), une fois que les élèves ont établi une méthode permettant de calculer le nombre de maillons de n'importe quelle chaine (généralisation algébrique naïve), « Le professeur demande ensuite de trouver : "une phrase mathématique à envoyer au bijoutier qui permettra de calculer le nombre de tiges pour n'importe quelle chaîne". ». Cependant, le seul garant de ce qu'est « une phrase mathématique » est ici l'enseignant : il est difficile pour un élève de s'imaginer à la place du bijoutier et de constater les imprécisions de la méthode qu'il lui transmet.

Or, nous avons vu que les contraintes économiques ont lié efficacité de la méthode et machine. Pour être exécutée par une machine, une méthode doit être formulée dans le langage de la machine. « faire-faire » la généralisation par une machine implique donc la nécessaire représentation, dans le langage de la machine, d'une propriété non connue *a priori*, et manipulable dans le langage de la machine : une situation de généralisation de motif informatisée pourrait ainsi contribuer à construire la nécessité de la variable dans un rôle de paramètre, du point de vue informatique et algébrique. C'est ce qu'évoquaient déjà Samurçay et Rouchier (1985, p. 242) :

> À la différence d'une situation habituelle de résolution de problème, la situation de programmation fait intervenir un élément essentiel qui est le dispositif d'exécution, défini par un ensemble d'opérations permises et les conditions de validité qui leur sont associées. La solution d'un problème doit alors être analysée à un double niveau :
>
> — les résultats qu'on veut obtenir par l'application du programme à des données ;
>
> — le programme qui est l'expression dans un langage de programmation d'une procédure définie par référence au dispositif d'exécution.
>
> Un premier aspect permet d'opposer comme le fait Hoc (1982), les situations de programmation aux situations de « production de résultat » dans lesquelles l'objectif d'élaboration de procédure s'identifie de fait à l'exécution. Un deuxième aspect concerne l'explicitation par le sujet des procédures qu'il utilise pour résoudre un problème. Cette explicitation est rendue nécessaire par le fait que l'exécution de la procédure se fait et doit se faire dans un temps distinct de celui de la construction du programme : l'exécution est différée. Le troisième aspect concerne le passage d'une planification des actions exécutables par le sujet lui-même à celle d'un plan d'actions dont l'exécution n'est pas contrôlée par l'opérateur humain. Le sujet n'aura pas l'occasion d'intervenir en cours d'exécution pour modifier le déroulement de la procédure.

La recherche de concepts à l'interface entre algèbre et algorithmique ou informatique nous amène à souligner à la fois l'importance fondamentale du paramètre dans une perspective de généralisation, et les difficultés de la construction de ce concept. Une situation de généralisation de motif pourrait permettre d'établir la nécessaire existence d'un objet représentant une certaine propriété non définie *a priori*. Cependant, la nécessaire existence, dans le registre de représentation sémiotique (informatique ou algébrique) de la construction du problème, d'un symbole permettant de manipuler cet objet comme s'il était connu, ne peut s'établir que si le dispositif d'exécution contraint le langage. On voit alors l'intérêt de travailler ce concept dans un cadre informatique. De plus, un EPGB, comme Scratch ou Snap !, permet une exploration du langage sans contraintes d'écriture syntaxique des instructions, tout en donnant accès aux différents éléments disponibles dans ce langage. Ceci rend ainsi plus aisé les explorations du langage, ainsi que les manipulations et transformations de programme.

1.10.1 Nouvelles questions de recherche

Il nous faut donc *chercher une situation de généralisation informatisée de motif, à l'interface entre algèbre et informatique, qui permette d'étudier comment et à quelles conditions les élèves peuvent construire le concept de variable dans un rôle de paramètre.* Nous nous interrogeons ainsi aux aspects sémiotiques : il s'agit notamment d'*établir les conditions de possibilité de la construction de la nécessité du paramètre, de sa représentation dans un certain registre sémiotique, et de l'utilisation de cette représentation dans une expression permettant de manipuler le paramètre comme si sa valeur était connue.* Ce qui implique d'étudier *en quoi une situation informatisée contraignant l'utilisation d'un langage formel permet-elle de participer à la construction de la nécessité du paramètre.* Nous nous intéressons ici à la construction de connaissances apodictiques,

nous chercherons donc plus précisément *comment et à quelles conditions les élèves construisent les faits et les nécessités du problème de généralisation informatisée d'un motif, dans le Cadre de l'Apprentissage par Problématisation (CAP)* (Doussot et al., 2022). Il s'agit d'effectuer une étude dynamique de l'activité de l'élève dans un contexte de programmation, il nous faudra ainsi récolter de façon efficace les traces de l'activité de programmation des élèves : nous chercherons donc à *concevoir un dispositif de captation automatisée permettant récolter les traces des actions de programmation des élèves.* Afin d'analyser ces traces — et de questionner l'utilité et la pertinence du dispositif de captation —, cela nécessitera d' *établir une méthodologie permettant d'identifier les faits et nécessités construits par les élèves, ainsi que leur mise en tension, à partir des traces d'actions de programmation.*

Dans le prochain chapitre, nous entamerons ainsi l'étude didactique d'une situation de généralisation informatisée de motif dans un EPGB, censée permettre la construction de la nécessité du paramètre, pour des élèves de cycle 4.

Chapitre 2

Méthodologie et cadre théorique

Sommaire du chapitre

Liste des figures

Liste des tables

Suite aux résultats de notre enquête épistémologique, nous cherchons à savoir *en quoi et à quelles conditions une situation de généralisation de motif informatisée pourrait permettre aux élèves de construire la nécessité du paramètre et de sa représentation.* Une situation impliquant la généralisation informatisée d'un motif ayant été conçue lors de notre Master, nous allons nous appuyer sur celle-ci pour notre expérimentation. Nous présenterons ainsi dans un premier temps la situation et son analyse *a priori* dans le Cadre de l'Apprentissage par Problématisation (2.1, p. 131).

Considérant selon Vergnaud (2011, p. 37) que « l'activité en situation est un moyen essentiel d'étudier la pensée et son évolution au cours du développement, ainsi que les différences entre individus », nous allons mettre en place une méthodologie permettant d'analyser l'activité des élèves lors de la partie programmation de la situation. L'analyse de l'activité de programmation des élèves pourra ainsi nous permettre d'étudier leur construction du problème. Le résultat (le script construit), même s'il est le produit d'un ensemble d'actions, n'est pas suffisant pour identifier cette activité et encore moins les raisons de cette activité. Les traces de l'activité de programmation, dont notamment les traces des actions de programmation — ou traces de programmation —, sont ainsi essentielles pour étudier la pensée de l'élève (la pensée est action selon C. S. Peirce, Houser et Kloesel (1992, p. 129)). Nous avons ainsi instrumenté l'EPGB utilisé afin que non seulement il soit le support de la situation et le fournisseur de rétroaction, mais aussi pour qu'il permette de capter, mémoriser et représenter les actions de programmations des élèves.

Ce dispositif expérimental sera présenté dans la partie 2.2 (p. 155). Après avoir fixé quelques conventions d'écriture (2.3, p. 162), nous convoquerons des cadres supplémentaires afin de décrire et analyser l'activité des élèves dans une situation de programmation impliquant des rétroactions du dispositif (2.4, p. 171). Enfin, les aspects sémiotiques liés à la nécessité de représentation du paramètre impliquant de déterminer comment nous allons identifier et analyser les procédés sémiotiques à l'œuvre dans l'activité de l'élève, nous expliciterons notre méthodologie de construction et de représentation des données (2.5, p. 196).

2.1 Présentation de la situation

La généralisation est fondamentale pour l'algèbre comme pour l'informatique, et le paramètre en est l'élément essentiel. Les situations de généralisation de motifs sont régulièrement mises en avant pour développer la généralisation algébrique, mais lorsque le dispositif d'exécution est humain, le paramètre n'est pas nécessaire : une généralisation algébrique naïve suffit (Radford, 2008, p. 3). Nous nous proposons donc d'utiliser une situation de généralisation de motif informatisée afin de rendre nécessaire le paramètre, soit l'existence d'un objet dénotant une certaine propriété, disposant d'une représentation (univoque) dans un certain langage θ, et dont cette représentation est manipulable dans des expressions de θ comme si l'objet était connu.

En Master — avant d'avoir établi l'importance du paramètre comme concept à l'interface entre algèbre et informatique —, nous avions conçu une situation permettant d'explorer les liens entre informatique et algèbre. Cette situation étant basée sur une généralisation de motif informatisée, nous avons fait le choix de nous appuyer sur elle afin d'étudier la construction du concept de paramètre chez les élèves.

Après avoir rappelé ce qu'on entend par « généralisation de motif » et explicité le choix du motif, nous présenterons succinctement la situation expérimentée. Nous préciserons les choix de mise en œuvre, puis, après avoir exposé le Cadre de l'Apprentissage par Problématisation (CAP), nous présenterons une analyse *a priori* de la situation, en nous intéressant plus particulièrement à la partie concernée par le concept de variable dans un rôle de paramètre.

2.1.1 Généralisation de motifs

Selon Radford (2008, p. 2) (figure 2.1, p. 132) :

> Generalizing a pattern *algebraically* rests on the capability of *grasping* a commonality noticed on some particulars (say $p_1, p_2, p_3, ..., p_k$) ; extending or generalizing this commonality to all subsequent terms ($p_{k+1}, p_{k+2}, p_{k+3}, ...$), and being able to use the commonality to provide a direct *expression* of any term of the sequence.

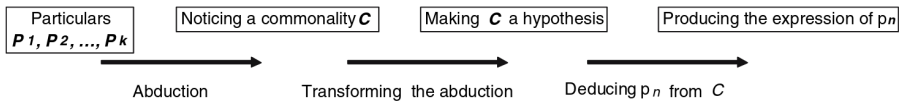


Figure 2.1 – Architecture des généralisations algébriques de motifs (Radford, 2008, p. 3)

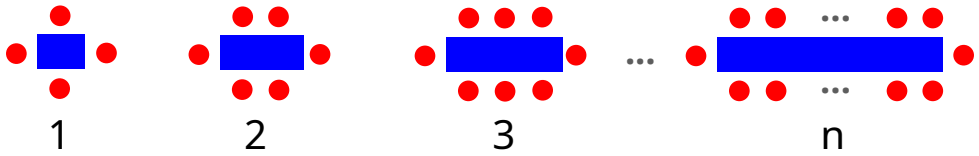


Figure 2.2 – Exemple de motif à généraliser

En reprenant un exemple dérivé des « tables de la cafétéria » donné par Bronner et Squalli (2021, p. 9) (figure 2.2, p. 132), il s'agit, si l'on détaille en reprenant les niveaux de généralisation proposés par Radford (2006a) :

— de trouver une régularité commune à différentes instances du motif (et reconnaître que ces régularités s'appliquent à toutes les instances) : ici par exemple, une chaise à chaque extrémité de la table, autant de chaises de chaque côté, et autant de chaises sur un côté que la valeur du nombre désignant l'instance ;

— de l'étendre aux instances suivantes et proches : par exemple pour l'instance 4, une chaise à chaque extrémité, une chaise de plus que l'instance 3 de chaque côté soit 10 chaises. Ou pour l'instance 5, une chaise de plus que l'instance 3 de chaque côté, et encore une chaise de plus de chaque côté, soit 12 chaises. C'est ce que Radford définit comme étant la *généralisation arithmétique*.

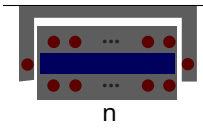
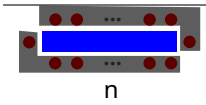
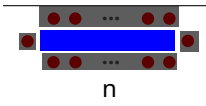
— de déterminer une certaine instance, à partir d'un ou deux cas : par exemple penser que, comme pour l'instance 2 on a 6 chaises, on en aura 12 pour l'instance 4. C'est ce que Radford nomme l'*l'induction naïve*, l'application de schémas, généralement variables, par essai-erreur ;

— de déterminer un schema, « a rule providing one with an expression of whatever term of the sequence (arithmetic generalizations would be those which fail to meet the third component) » (ibid., p. 15). Radford distingue alors trois niveaux de *généralisation algébrique* :

1. être capable de trouver 22 chaises pour l'instance 10 ou 104 chaises pour l'instance 51, mais en s'appuyant sur des nombres, non sur des indéterminés. Radford considère cette forme de généralisation comme étant une *généralisation algébrique factuelle*, basée sur l'action, que nous avons nommée aussi *généralisation algébrique naïve* ;

2. être capable d'exprimer cette généralisation en nommant l'indéterminée : par exemple « les chaises du haut », « les chaises des extrémités » [1] ;
3. être capable de formaliser cette généralisation avec des symboles (pas forcément des lettres), par exemple $2n + 2$. C'est la *généralisation algébrique symbolique*

La généralisation algébrique symbolique est liée à l'idée de « sens et dénotation » de Frege, déjà évoquée page 27 : le dénoté de $2n + 2$, de $2(n + 1)$ ou de $1 + n + 1 + 2$ est le même (ici le nombre de chaises pour une instance n), mais leur sens est différent, et traduit le schéma identifié.

Schéma	expression
	$2n + 2$
	$2(n + 1)$
	$1 + n + 1 + n$

Une généralisation de motifs implique donc en général plusieurs registres de représentation sémiotiques (Duval, 1993) : la représentation figurée du dessin, l'expression algébrique, la formulation de la méthode en langue naturelle, ou sous forme de calcul ($2 \times 3 + 1$ par exemple). C'est cette coordination entre différents registres qui « permet de donner du sens aux lettres, aux expressions littérales, d'associer à plusieurs expressions une seule dénotation via les procédés de calcul et leurs représentations, d'illustrer un système de règles de formation et de transformation d'expressions littérales. » (Coppé et Grugeon, 2009, p. 9).

Dans notre cas, deux autres registres seront présents :

— une représentation figurée construite par un programme de tracé du motif ;
— une représentation formelle constituée de l'algorithme traçant le motif.

2.1.2 Situation « les Triangles de Sierpinski »

Afin de créer la nécessité d'une représentation formelle du paramètre et d'expression le mobilisant, tout en évitant les problèmes de syntaxes et de connaissance des mots clefs d'un certain langage informatique, nous avons fait le choix de faire construire par les élèves une partie d'algorithme de tracé d'un motif spécifique, dans un EPGB. L'EPGB permet en outre de proposer une rétroaction qui permettra aux élèves de contrôler les actions (voir Rétroaction et milieu, p. 176).

Motif choisi et scripts associés

Le choix du motif se base sur plusieurs contraintes que nous avons fixées en fonction de nos objectifs :

— Le motif doit être représentable dans un EPGB tel que Scratch : son tracé dans l'EPGB doit « donner à voir » ce que fait le programme de construction, afin de permettre aux élèves de faire un lien entre programme et rétroaction.
— Le motif doit s'appuyer sur des propriétés géométriques connues des élèves, qu'ils pourront mobiliser dans la construction du paramètre.

1. Voir aussi al-Khwārizmī et son « chapitre sur les transactions », p. 54.

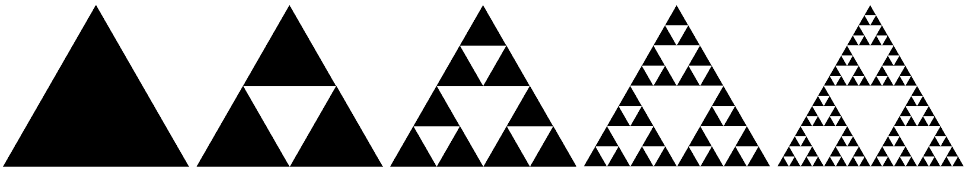


Figure 2.3 – Les triangles de Sierpinski

— L'algorithme du tracé doit être simple (une séquence de boucles, sans condition ou branchement) et continu, c'est-à-dire sans « trou » (sans déplacement du « lutin » non accompagné d'un tracé), ni chevauchement (puisque nous désirons connaître le nombre d'hexagones nécessaire à la construction d'un certain TS). C'est un élément important pour permettre l'interprétation de la rétroaction.

— Les expressions permettant de déterminer les différentes parties du tracé ne doivent pas être toutes identiques et l'une au moins doit être non triviale. Par « triviale » nous entendons une expression immédiatement identifiable. Par exemple, dans l'activité du « carré bordé », l'algorithme consisterait à répéter quatre fois un tracé (une boucle) de n carrés, suivi d'une rotation à 90° : il n'y a pas à proprement de relation entre le nombre d'hexagones tracés par une boucle, mais il y a identification du nombre d'itérations et du paramètre n. Nous souhaitions proposer une figure dont l'algorithme mobilise des expressions (ou des relations entre le paramètre *mesure* et le nombre d'hexagones tracés) différentes, dont au moins une est résistante.

Le motif choisi est la première itération d'un triangle de Sierpinski (voir figure 2.3, p. 134) formé par des hexagones jointifs : c'est ce qui est nommé dans la séquence un « Triangle de Sierpinski » (TS). Pour ces figures, nous définissons la *mesure* comme étant le nombre d'hexagones sur un côté des triangles de base. Ainsi dans la figure 2.4a (p. 135), $mesure = 6$. La figure tracée dans ce cas sera un TS6, et le script traçant cette figure s'écrira $TS(6)$. La figure 2.4b (p. 135) précise la séquence des huit boucles traçant de façon continue un TS. Les hexagones représentés avec une marque verte sont des hexagones uniques, tracés en dehors d'une boucle. Le tracé est en fait une suite de « tampons » d'un hexagone d'une taille calculée pour que le TS ne sorte pas des limites de l'espace d'exécution, suivi d'un déplacement équivalent à la taille de ce tampon. La figure 2.5 (p. 136) montre le script traçant un TS4 pour la séance 2, ainsi que celui traçant ce même TS4 pour les séances suivantes. Pour ces deux derniers, on change de niveau d'abstraction : les instructions permettant de déplacer le lutin et tamponner sont rassemblées dans un « bloc » 

(une procédure). Pour une certaine mesure n, le nombre d'hexagones nécessaires est $N = 9n - 12$.

On peut rassembler ces boucles suivant leur nombre d'itérations, ce nous amène à considérer trois « familles » de boucles :

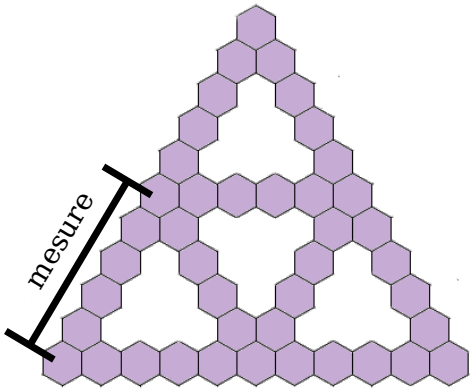
— $\mathbf{B_1}$, les boucles 1 et 6, qui tracent $mesure - 1$ hexagones.

— $\mathbf{B_2}$, les boucles 2, 3, 4, 5 et 8, qui tracent $mesure - 2$ hexagones.

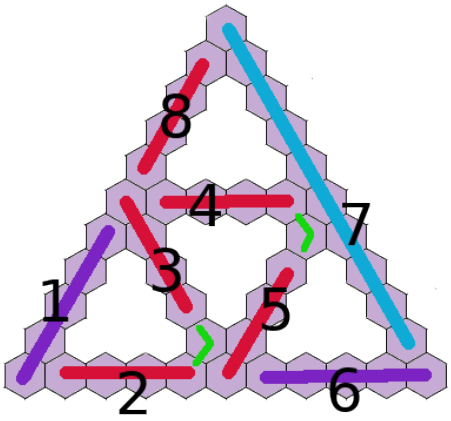
— $\mathbf{B_7}$, la septième boucle, qui trace $2 \times mesure - 2$ hexagones.

Chaque boucle sera désignée par son numéro d'ordre, b_2 désigne ainsi la deuxième boucle du script (voir les Conventions d'écriture, p. 162).

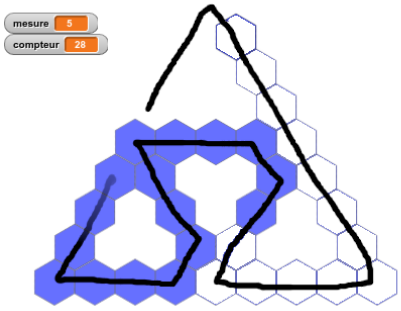
2. Le trait noir est ici rajouté pour visualiser le tracé virtuel, de même que les contours d'hexagones non encore tracés. Sur l'espace d'exécution n'apparaissent que les hexagones effectivement tracés.



(a) « Triangle de Sierpinski » de mesure 6 (TS6)



(b) Séquence de boucles traçant un TS6



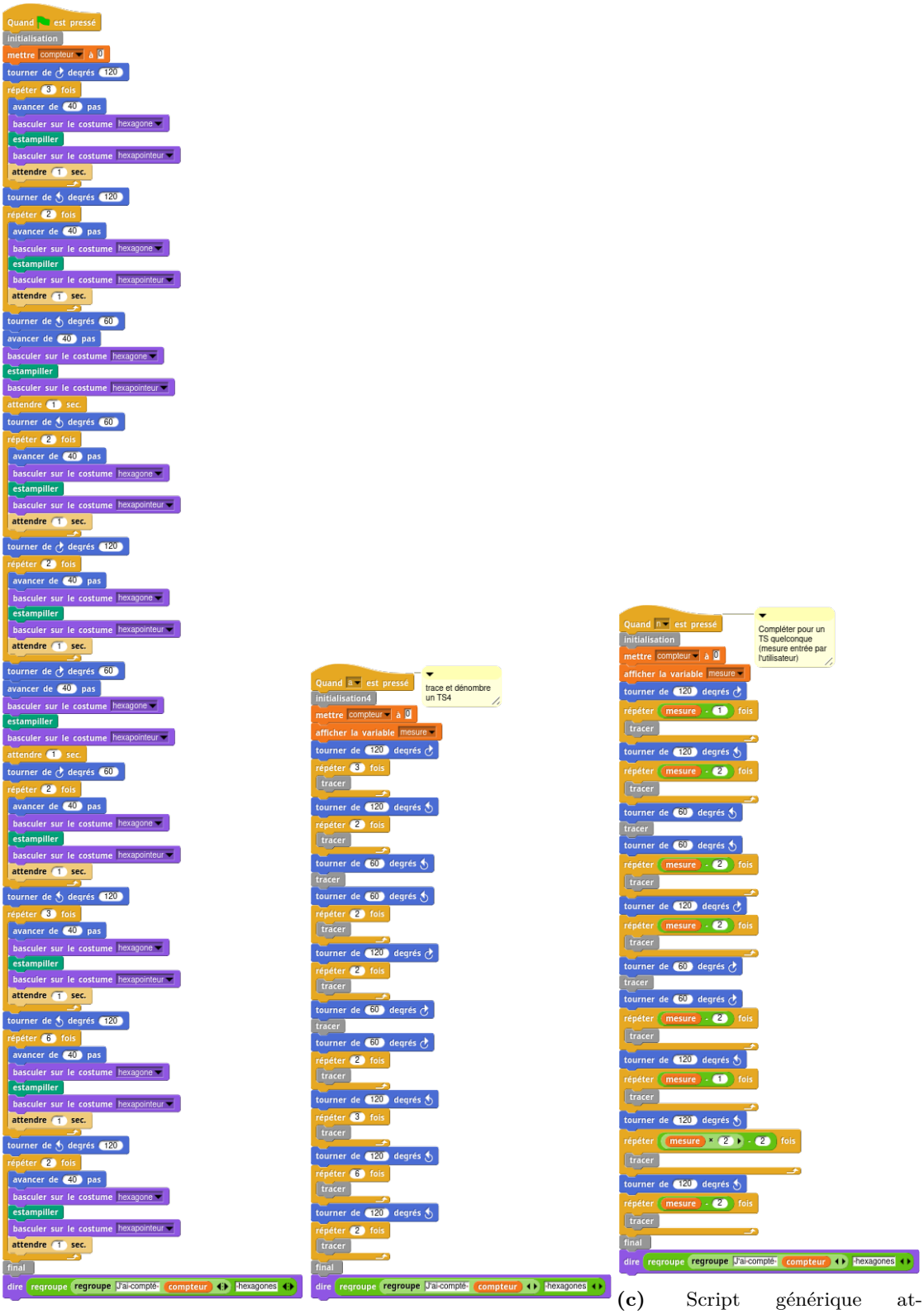
(c) Tracé d'un TS5 en cours (boucle b_5) [2]

Figure 2.4 – Les TS et leur tracé

Organisation de la séquence La séquence servant de support à notre expérimentation s'est déroulée sur sept séances et se divise en cinq parties :

1. Partie 1 : Découverte du motif et de ses propriétés, et énoncé du problème « quel est le plus grand TS constructible avec [un certain nombre] d'hexagones » (séance sur le motif non informatisé).
2. Partie 2 : Construction du script $TS(4)$, première rencontre avec une variable informatique (motif informatisé).
3. Partie 3 : Généralisation du script et résolution du problème (motif informatisé)
4. Partie 4 : Établissement d'une méthode permettant de calculer directement le nombre d'hexagones d'un TS (motif informatisé *vs* expression algébrique).
5. Partie 5 : Implémentation de la solution dans un programme (expression algébrique dans un programme)

Il s'agit pour l'essentiel de la séquence utilisée en support de notre travail de Master — nous préciserons dans l'analyse *a priori* les ajustements effectués.



(a) Script n°1 **(b)** Script n°2 :$TS(4)$ **(c)** Script générique attendu :$TS(n)$

Figure 2.5 – Scripts proposés et script générique attendu

2.1.3 Choix de mise en œuvre

Concernant la mise en œuvre, en classe ordinaire, de cette situation, différents choix ont été faits : les modalités d'organisation de la recherche et de la situation d'apprentissage, l'organisation pédagogique, et le choix du dispositif support (l'EPGB). Ce dispositif support de l'expérimentation fait partie d'un dispositif expérimental — incluant notamment un système de captation des actions de programmation — qui serait détaillé dans la partie suivante.

Ingénierie et séquence forcée

La situation que nous avons construite est une ingénierie didactique, dans le sens où :

> l'ingénierie didactique se trouve au croisement de deux préoccupations fondamentales de la didactique des mathématiques comme science : - l'étude didactique des mathématiques, c'est-à-dire l'étude d'une organisation des mathématiques compatible avec le projet de les enseigner en lien avec leur usage pour résoudre des problèmes ; - l'observation de l'activité mathématique des élèves et didactique des professeurs face à ces problèmes. (Perrin-Glorian et Bellemain, 2019, p. 48)

La mise en œuvre de cette situation est faite non par le chercheur mais par une enseignante de classe ordinaire. Si l'organisation générale est contrainte (les 5 phases et leurs objectifs), la mise en pratique pédagogique est négociée avec l'enseignante, qui a en charge l'enseignement des mathématiques pour les classes visées. Il s'agit à la fois de produire des savoirs pour les élèves et d'analyser les conditions de production de ces savoirs.

Cependant, il ne s'agit pas de produire une situation type qui pourrait être reprise telle quelle par les enseignants. Entre chaque séance, un point est fait l'enseignante afin d'ajuster les séances suivantes, en fonction des éléments issus de l'analyse *a priori*, des objectifs d'apprentissages et des objectifs de recherche. En cela, notre expérimentation se rapproche de ce que Orange (2010, p. 77) nomme une « situation ou séquence forcée » :

> Les situations forcées sont des situations d'enseignement qui ne sont généralement pas isolées, mais organisées en une séquence d'enseignement (d'où la notion de séquences forcées) ; elles sont construites au sein d'un groupe de recherche comportant des chercheurs en didactique et des enseignants experts. L'ensemble du groupe connaît le cadre théorique et la problématique de la recherche.

Si l'enseignante avec laquelle nous avons collaboré est bien une « experte », elle n'a profité que d'une approche succincte du CAP, centré sur la notion de nécessité. Cependant,

> Cette co-conception se matérialise en particulier dans les moments d'échange organisés entre chaque séance au cours desquels l'organisation initiale est ajustée en fonction du déroulement effectif de la séance qui vient de se terminer. C'est cet ajustement constant qui permet l'objectivation de la co-conception dans des instruments tels que les caricatures qui sont des objets organisés selon les énoncés produits dans la classe sous le contrôle de l'enseignant et les principes actualisés du cadre de la problématisation. (Doussot et al., 2022, p. 189)

Ainsi, si la conception de la séquence a été faite essentiellement en aval de la collaboration avec l'enseignante, celle-ci a pris une part active dans son ajustement pour sa classe et sa pratique, et dans les ajustements opérés entre chaque séance. Notre expérimentation est finalement à mi-chemin entre ingénierie didactique et séquence forcée.

2.1.4 Le Cadre de l'Apprentissage par Problématisation

Nous nous situons dans le Cadre de l'Apprentissage par Problématisation, dans lequel on s'intéresse à la construction de savoirs raisonnés, par la mise en tension de faits et de nécessités ou de conditions (Fabre et Orange,1997). Dans une première approche,

> *Les données correspondent à des faits constatés* [...] *Les conditions expriment la règle à suivre pour mettre en relation les données et obtenir le résultat* [...] Enfin, traiter un problème exige de reconnaître le cadre dans lequel il se pose. (Fabre, 2017, p. 16)

Cela amène Fabre (ibid., p. 18-29) à considérer cinq critères caractérisant le processus de problématisation : 1) l'examen d'une question ; 2) l'articulation du doute et de la certitude ; 3) l'articulation des données et des conditions (ou des nécessités) dans un cadre ; 4) avec une pensée qui se surveille et 5) dans une perspective heuristique.

Pour qu'il y ait un problème, il doit y avoir une question issue d'une « désadaptation, rupture dans notre expérience » (ibid., p. 18). Le problème peut ainsi provenir d'une énigme, d'une controverse, ou d'un échec. L'énigme « survient à l'occasion d'un événement inhabituel et quand nous nous demandons ce dont il s'agit, ce qui l'a provoqué » (ibid., p. 18) ou lorsqu'on se demande comment s'y prendre pour réussir un projet conséquent. Il s'agit ici de la position du problème. Construire le problème, « c'est le déterminer (Fabre, 1999, p. 181), "chercher les données et les conditions pertinentes" (Fabre, 1999, p. 198) pour le circonscrire et le résoudre » (Hersant, 2020, p. 206), avant de pouvoir le résoudre. Ces trois phases, position, construction et résolution du problème, ne sont cependant ni linéaires ni chronologiques, mais s'articulent dans le processus de problématisation. Ces trois phases représentent la facette cognitive du CAP, et la construction du problème en est la facette épistémologique. Dans le CAP, on s'intéresse non seulement à la solution mais surtout la construction de la solution : « le savoir visé ne peut résider uniquement dans la solution de la question posée ; il doit prendre en compte la construction de problème » (Orange, 2012, p. 35). Cette dimension épistémologique,

> concerne la mise en tension des faits et d'idées si on s'intéresse aux problématisations scientifiques (Orange, 2000). Les conditions du problème sont des contraintes non contingentes, des impératifs du problème, ce qu'il faut respecter et ce sur quoi on n'a pas prise ; elles sont de l'ordre de l'apodictique (Bachelard, 1938) et expriment des nécessités. (Choquet et al., 2023).

Il s'agit de construire des savoirs raisonnés, c'est-à-dire des savoirs non pas *assertoriques*, de l'ordre du « savoir que », mais des savoirs *apodictiques*, soit « savoir que ça ne peut pas être autrement ». Ainsi, en mathématique, « nous considérons que les élèves problématisent lorsqu'ils construisent les nécessités d'un problème en référence à une exploration empirique, c'est-à-dire lorsqu'ils ont une compréhension du problème qui est de l'ordre du "pourquoi ça ne peut pas être autrement" qui émane d'un travail mobilisant des "faits" et une construction du problème » (Hersant, 2020, p. 60). Cependant, lors de la problématisation, « on suspend, tout au long de l'examen, tout jugement définitif de type *apodictique* ou *assertorique* » (Fabre, 2017, p. 19) : les faits ou les données, tout comme les conditions ou les nécessités, ne sont justement pas donnés, ils sont construits. Orange (2012, p. 17) rappelle ainsi les mots de Bachelard dans « La formation de l'esprit scientifique » : « Rien ne va de soi. Rien n'est donné. Tout est construit. », ce que Doussot et al. (2022, p. 16) reprennent ainsi :

> Ce qui nous semble important à souligner c'est que dans cette perspective, les faits ou les données malgré leur appellation ne sont pas donnés ; ils sont construits, tout comme les conditions, les idées, les expériences ou les théories. Tout est construit dans le cadre de l'enquête. En effet, comme le précise Dewey, l'idée « provoque et dirige de nouvelles observations qui fournissent une nouvelle matière factuelle » (Dewey, 1993 [1967], p. 183-184) et « l'accumulation de ces faits suggère des hypothèses : c'est la dialectique des indices et des preuves » (Fabre, 2006, p. 19).

Cependant, si rien n'est donné, tout ne peut pas être mis en doute : Fabre (2017, p. 20) rappelle ainsi que pour Dewey ou Bachelard, « on ne peut douter qu'en s'appuyant sur des certitudes ». Il considère ainsi trois exemples type de *hors question* : « a) ce qui tombe en dehors du questionnement, ce dont on ne se préoccupe pas ; b) ce qui est présupposé par ce questionnement ; c) ce sur quoi il se fonde » (ibid., p. 20). Ainsi, par exemple, dans notre situation, les élèves doivent présupposer qu'il existe un algorithme générique traçant et dénombrant n'importe quel TS ; le mettre en doute ne leur permettrait plus d'avancer dans le problème attendu. Cette dialectique entre le doute et la certitude est pour Fabre (ibid., p. 20) « l'âme même de la problématisation »

La construction du problème se fait en articulant données ou faits, et conditions ou nécessités. Les données sont des faits construits ou établis, des constats, et relèvent de l'assertorique, tandis que les nécessités relèvent de l'apodictique (ibid., p. 24), et cette articulation se fait dans un certain cadre. Orange (2012) évoque quant à lui, dans le cadre de l'épistémologie des sciences, trois registres : le registre empirique, celui des faits et des données ; le registre des modèles ou registre des nécessités ; et le registre explicatif (REX), composé des principes explicatifs qui permettent d'articuler les tensions entre données et nécessités :

> Comme registre donnant sens aux modèles et nécessités qui sont mis en jeu dans une problématisation, il indique une manière habituelle ou établie de confronter les faits et les idées explicatives pour identifier les conditions de possibilité d'une explication. Sous cette définition perce la profondeur qui caractérise ce type de registre, la stabilité qu'il représente dans une activité qui réclame du mouvement entre faits et idées. Le registre explicatif est composé de principes structurants, de modes de raisonnement, des références à des mondes d'expérience. Il est ce qui cadre les opérations de recherche, d'exploration des possibles et d'identification et de caractérisation des conditions. (Doussot et al., 2022, p. 14)

Problématiser, c'est ainsi articuler données et nécessités dans un certain REX. Cette articulation doit s'accompagner d'un contrôle, d'une « pensée qui se surveille », ou d'une « surveillance de soi par soi ». Donnant l'exemple du calcul, Fabre (2017, p. 25) dit ainsi : « Calculer, c'est effectuer plus ou moins mécaniquement une opération, mais également surveiller la procédure utilisée et contrôler la vraisemblance du résultat ». Cette surveillance de soi nécessite aussi « une perspective heuristique dans une logique d'enquête » : pour problématiser, il fait avoir une idée de la forme de la solution, un « savoir d'avance », un « savoir qui n'est pas encore réponse, mais qui permet d'en anticiper les caractéristiques formelles et matérielles. » (ibid., p. 28)

Dans notre expérimentation, nous cherchons à établir les conditions de construction de la nécessité du paramètre. Dans la partie suivante, nous procéderons à une première analyse *a priori*, globale, de la situation, avant de nous intéresser plus particulièrement à la phase 3 de généralisation.

2.1.5 Analyse *a priori*

La situation que nous avons expérimentée a été construite lors de notre Master, et elle ne s'appuyait pas fondamentalement sur le CAP. Pour l'essentiel, il s'agissait de faire jouer les cadres algébriques, algorithmiques et graphiques (Douady, 1984), en proposant un milieu à potentialité a-didactique :

> c'est-à-dire une situation mathématique apparemment « dénuée d'intention didactique » dans laquelle le maître n'intervient pas au niveau des connaissances et des savoirs. Dans cette situation adidactique, un système antagoniste de l'élève est nécessaire : c'est le milieu (Brousseau, 1986, 1990) qui doit, d'une part, fournir des rétroactions interprétables par l'élève et, d'autre part, permettre la reconnaissance dans la réponse de l'élève de la connaissance visée [...] (Hersant, 2010, p. 8-9)

Ainsi, le milieu, au sens de la Théorie des Situations Didactiques (TSD) (Brousseau, 2011), doit pouvoir permettre de dévoluer la validation aux élèves, de leur permettre d'entrer dans l'activité avec leurs connaissances disponibles, tout en permettant « l'apparition de la connaissance souhaitée tout en contraignant l'activité des élèves de façon à éviter la dispersion des connaissances (correctes ou erronées) produites et ainsi faciliter la validation. » : c'est un milieu *rétroactif*, *proactif* et *contraignant* (Hersant, 2010, p. 43).

Afin de satisfaire à la caractéristique proactive, la situation a été pensée afin de construire, notamment, la *nécessité* du paramètre. Ainsi, afin d'analyser les potentialités de cette situation, nous nous intéressons dans ce travail non pas à la solution, mais à la position et la construction du problème, et plus particulièrement à la construction des nécessités et à leurs tensions avec les faits construits : le CAP est adapté pour cette analyse, tant *a priori* que *a posteriori*.

Partie 1 (Séance 1)

Les deux premières questions de la partie 1 (figure 2.6, p. 142) s'appuient sur le registre graphique uniquement [3]. Elles visent à permettre aux élèves d'identifier les éléments déterminant la validité d'une figure. En effet, cette connaissance est nécessaire pour interpréter les rétroactions produites par le milieu (au sens de Brousseau (2011)) qui leur permettront de contrôler et ajuster leur activité (voir 2.4.2, p. 176). L'essentiel des difficultés provient du tracé d'une figure géométrique par des éléments discrets et possédant une dimension : on représente un segment par une suite d'hexagones. Il faut voir les hexagones à l'extrémité des « segments » comme étant des sommets, et les triangles de base doivent avoir leurs sommets confondus (figure 2.7a, p. 143). Comme la représentation des segments a une forme d'épaisseur, le risque est de voir le segment comme étant le bord de la figure tracée par les hexagones. Sur la figure 2.7b (p. 143), on voit ainsi le tracer qui résulterait d'une conception du segment comme étant le bord « externe » tracé par les hexagones, la figure 2.7c (p. 143) montrant l'inverse.

Le tracé du « plus grand TS possible » vise aussi à définir ce qu'est « le plus grand TS », et notamment aboutir au fait que « le plus grand TS » est équivalent à « le TS qui a le plus grand nombre d'hexagones » et à « le TS qui a la plus grande mesure ». Ces deux questions, enfin, permettent d'établir des premiers faits reliant la mesure à un nombre d'hexagones du TS, ce qui permettra de valider ou non le dénombrement des hexagones fait par le script dans les séances suivantes.

3. Du matériel est également à disposition pour les élèves les plus en difficulté, ce qui sera utilisé avec des élèves de Segpa (Section d'Enseignement Général et Professionnel Adapté).

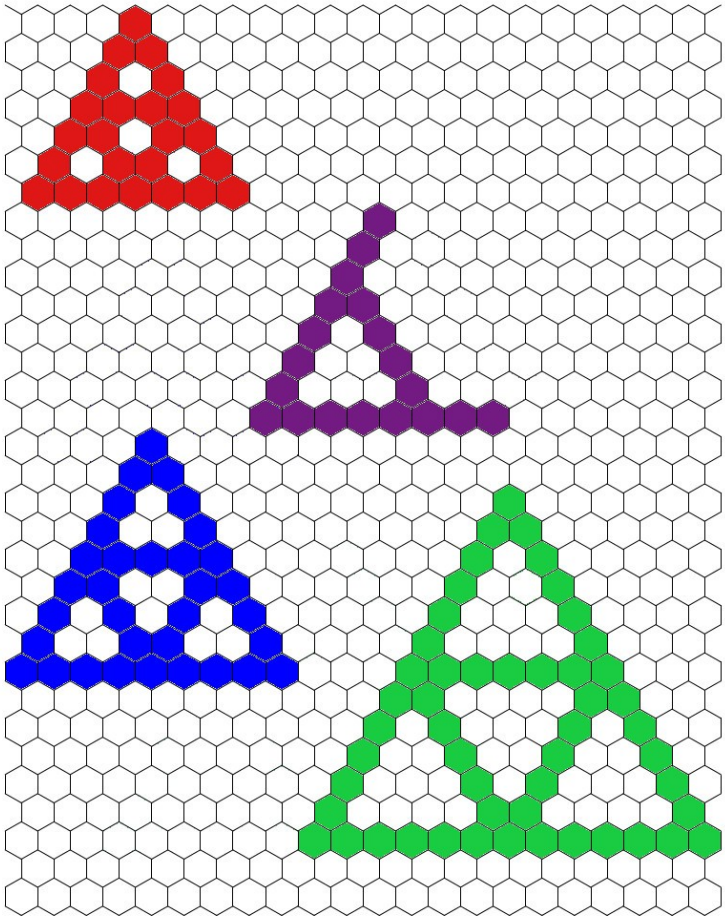
La troisième question permet de préciser le problème que l'on posera aux élèves. La recherche du plus grand TS constructible avec 35 hexagones permettra notamment de bien définir les hexagones que l'on compte (les hexagones colorés). Il s'agit de résoudre une instance du problème en se basant sur le registre graphique et numérique. Une erreur attendue chez les élèves est de considérer l'ensemble des hexagones, c'est-à-dire à la fois les hexagones colorés, et les hexagones « blancs ». Le matériel peut permettre d'expliciter la construction, en ajoutant des pions hexagonaux sur la grille hexagonale. Dans l'activité du carré bordé, cette distinction n'est pas faite, notamment parce qu'on peut déterminer le nombre de carrés colorés en soustrayant l'aire du carré blanc de l'aire du carré complet (Coppé et Grugeon, 2009, p. 9). Dans notre cas, cette distinction est nécessaire car le tracé produit par le script ne rend visible que les hexagones effectivement tracés, la grille n'étant pas visible. En outre, l'utilisation de la surface des triangles pour déterminer le nombre total d'hexagones n'est pas accessible à des élèves de ce niveau. L'utilisation de formules de calcul de l'aire du triangle n'est pas possible, et un calcul effectif partant de l'aire totale d'un TS de mesure n serait $\sum_{i=1}^{2n-1} i$, somme non calculable directement par des élèves de cycle 4 [4].

L'autre difficulté attendue est qu'il n'existe pas de TS utilisant 35 hexagones (le TS5 nécessite 33 hexagones, le TS6 en a besoin de 42) : il s'agit bien de trouver le plus grand *possible* et non d'utiliser tous les hexagones.

La quatrième question, qui constitue le problème principal posé aux élèves sur cette séquence, va amener l'insuffisance d'un traitement dans le registre graphique. On incitera les élèves à proposer la recherche d'une solution informatique. Cette question ne trouvera sa solution qu'en séance 6. Les valeurs choisies pour le nombre d'hexagones disponibles sont des variables didactiques fondamentales : un TS de mesure suffisamment grande n'est pas traçable à la main, cela invalide la résolution graphique et impose d'en trouver une nouvelle. Les deux solutions possibles sont le calcul algébrique ou la construction et le dénombrement par un programme de n'importe quel TS. La séance ayant lieu en salle d'informatique, on attend des élèves qu'ils envisagent une solution informatique, même s'ils n'ont que peu de connaissances en ce domaine. La solution algébrique serait l'objet de la Partie 4. Les valeurs choisies (1400, 1654, 1710) ont pour objectif de traiter plusieurs cas permettant d'invalider l'approximation attendue $N = \mathrm{E}\frac{n}{9} + 1$, où N est le nombre total d'hexagones d'un TS de mesure n. En posant $n = 9q + r$, et sachant que pour un TS de mesure n, $N = 9n - 12$, cette approximation est valide pour $r < 6$. Il nous faut donc proposer plusieurs valeurs de n avec $r < 6$ (par exemple 1400, $r = 5$), $r = 6$ (par exemple 1608 et il ne reste plus d'hexagones), $r > 6$ (par exemple 1654 et $r = 7$) ou $r = 0$ (par exemple 1710). Ces valeurs élevées ont aussi pour objectif de rendre le tracé suffisamment long pour faire émerger la possibilité d'une recherche de méthode plus efficace, plus rapide, plus économique : c'est ce qui conduira à la partie 5, dont l'objectif sera de déterminer une méthode de calcul direct du nombre d'hexagones d'un TS quelconque.

4. Avec cette méthode, le nombre N d'hexagones d'un TS de mesure n serait $N = n(2n-1) - 2(n-2)(n-3)$.

Les triangles de Sierpinski



1) Compléter le triangle de Sierpinski de mesure 6.

2) Tracer sur le pavage vierge, le plus grand triangle de Sierpinski possible.

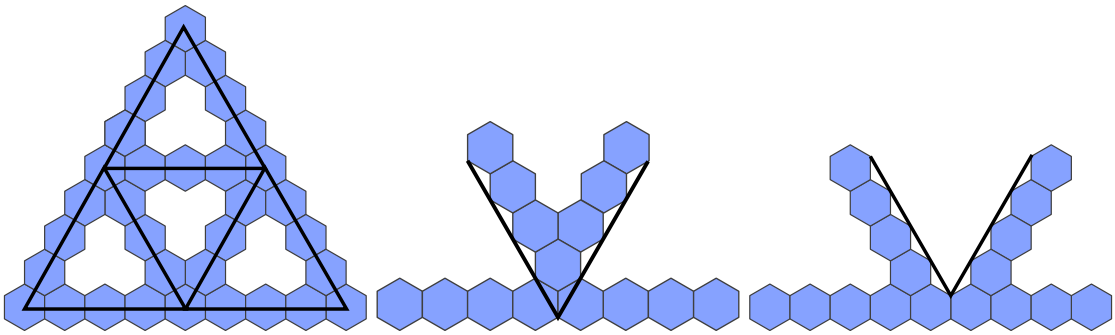
3) Si on dispose de 35 hexagones, quel est le plus grand triangle de Sierpinski que l'on peut obtenir ?

.....

4) Si on dispose de 1400 hexagones, quel est le plus grand triangle de Sierpinski que l'on peut obtenir ? Si on dispose de 1654 hexagones ? Si on dispose de 1710 hexagones ?

.....

Figure 2.6 – Partie 1, document élève



(a) Les segments des triangles tracés **(b)** Une vision erronée : segments « externes » **(c)** Une vision erronée : segments « internes »

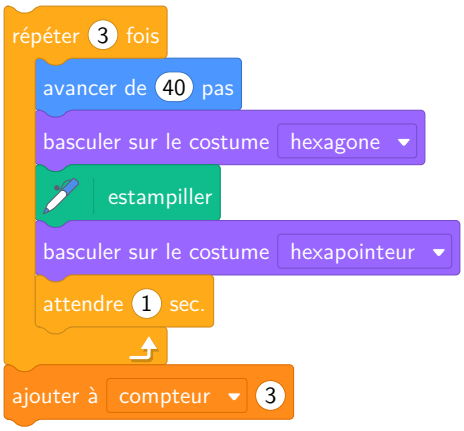
Figure 2.7 – Perception des segments

Partie 2 (Séance 2) : découverte de la structure de l'algorithme et première rencontre avec la variable informatique

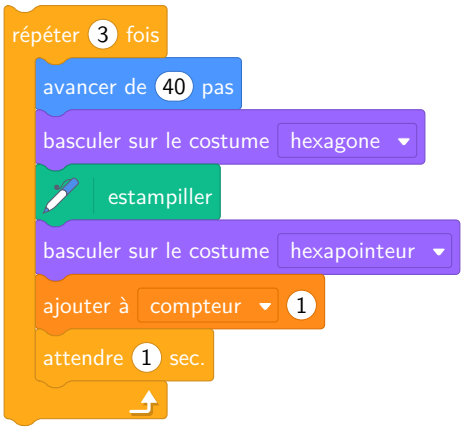
Une fois les éléments constitutifs du problème identifiés, les élèves ont pour consigne de transformer un script traçant un TS (figure 2.5a, p. 136), en un script traçant *et dénombrant* un TS4. Le premier objectif est ici d'amener les élèves à explorer un script long mais de complexité modérée — séquences et boucles uniquement, pas de boucle imbriquée ni de conditions —, afin qu'ils identifient la structure de l'algorithme de tracé (schématisé figure 2.4b et 2.4c, p. 135). Le second objectif est d'amener les élèves à rencontrer une première variable informatique : la variable dans un rôle de paramètre. L'enseignante précise l'utilisation du mode pas-à-pas, puis explicite les attendus :

> Le programme est malheureusement incomplet car il dit qu'il a compté 0 hexagones. Votre travail aujourd'hui est de corriger ce programme pour qu'il donne à la fin le nombre total d'hexagones, ici c'est 24 mais aussi pour qu'il indique le nombre d'hexagones dessinés au fur et à mesure du programme. Ce nombre-là va apparaître dans la case compteur et on doit le voir évoluer au cours du programme, 1 2 3 4 ... comme dans les jeux vidéo où vous gagnez des pièces d'or par exemple. (Annexe A.1, p. 505)

La contrainte d'avoir le « compteur » affichant à tout moment le nombre d'hexagones effectivement tracés vise à amener les élèves à ne pas ajouter au compteur le nombre de répétitions d'une boucle à chaque fin de boucle :

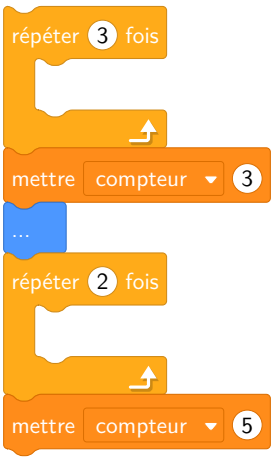


Il faut en revanche incrémenter le compteur à chaque itération [5] :



5. L'incrémentation devrait avoir lieu *après* le 

Une autre solution, encore moins généralisable, serait d'affecter une valeur actualisée du compteur après chaque itération ou tracé indépendant :



La première solution nécessiterait, lors de la phase de généralisation, de généraliser aussi l'incrémentation du compteur, ce que nous souhaitons éviter. En outre, la mise à jour du compteur après un certain nombre d'itérations ne permet pas d'utiliser l'affichage de la valeur du compteur comme critère de validation.

Cependant, les élèves ne peuvent construire cette nécessité : il s'agit ici d'une contrainte didactique. Une autre solution, permettant de construire la nécessité de l'incrémentation du compteur à chaque tracé effectif d'un hexagone, aurait été de repousser cette modification du script pour dénombrer à la phase suivant la généralisation. Nous avons décidé, avec l'enseignante, de ne pas choisir cette solution pour trois raisons : d'une part, comme nous l'avons dit, parce que l'affichage du nombre d'hexagones tracés fait partie des éléments que les élèves devraient pouvoir prendre en compte pour valider ou invalider leur action, d'autre part parce que nous estimions nécessaire d'amener les élèves à explorer la structure de l'algorithme, et enfin parce que la variable dans un rôle de compteur est plus accessible à un néophyte que la variable dans un rôle de paramètre[6]. Notons que la variable dans un rôle de compteur, si elle est explicite en informatique, est souvent implicite en mathématique — par exemple lorsqu'on doit calculer u_1 puis u_2, etc. —, et parfois explicite, comme dans l'expression $\sum_{i=1}^{p} u_i$.

À issue de cette phase, l'institutionnalisation de la variable de compteur est ainsi faite :

Programmation d'un compteur :

est appelée une variable informatique. C'est **un espace de stockage, une mémoire, permettant de conserver une donnée**, ici c'est le nombre d'hexagones tracés.

Au début du programme, cette variable est **initialisée** avec l'instruction

. Il y a aucun hexagone tracé au départ.

Elle va changer au cours de l'exécution du programme avec l'instruction

. A chaque fois qu'un hexagone est tracé, le compteur augmente de 1.

6. Nous avons testé avec une autre classe (3ième Segpa) de commencer par la phase de généralisation, et nous avons dû revenir sur cette phase d'exploration du script : les élèves cherchaient à modifier la structure de l'algorithme, cette structure n'était pas suffisamment maîtrisée.

Partie 3 (Séance 3, 4 et 5) : phase de généralisation

Lors de la phase de généralisation, les élèves doivent tout d'abord produire (par duplication puis modification) des scripts traçant et dénombrant des instances successives de TS : TS5, T6, TS7. C'est une phase de constructions de faits du registre empirique, où les élèves mettent en œuvre une forme de généralisation arithmétique. Ils doivent ensuite construire, de la même façon, deux instances plus lointaines : le TS11, puis le TS19 ou TS20 ou TS21. Il s'agit ici d'amener les élèves à mettre en œuvre une généralisation algébrique naïve. La dernière et ultime étape consiste à produire le script générique, traçant et dénombrant n'importe quel TS, en fonction de la valeur de la mesure entrée par l'utilisateur. C'est lors de cette phase que les élèves devront mettre en œuvre la généralisation algébrique symbolique. Cette phase est celle qui est censée permettre la construction du paramètre, elle sera donc la phase soumise à l'analyse *a posteriori* dans le cadre de notre recherche.

Les élèves peuvent déterminer les schémas reliant une instance à la suivante, ou une instance à une autre instance en s'appuyant le registre de la représentation figurée du TS ou/et sur le registre numérique du nombre de répétitions des boucles, en le mettant en relation avec le registre algorithmique de formulation du tracé par des boucles.

Traitement intra, inter ou trans-objectal On s'attend à ce que les élèves établissent ou se basent sur des traitements de type intra-objectal, inter-objectal puis trans-objectal (Piaget et García, 1983). Le traitement intra-objectal se limite à considérer les propriétés d'un objet indépendamment d'autres objets proches ou similaires. Resta-Schweitzer (2011, p. 74) souligne que :

> Dans ce type d'argumentation, il existe une confusion entre ce qui est général et ce qui est nécessaire, une sorte de croyance de l'ordre de « cela doit être ainsi ». Il s'agit d'un traitement argumentatif qui impose des limitations et font obstacle au développement de la conceptualisation du réel mais qui constitue tout de même une expérience nécessaire susceptible de permettre la découverte des propriétés des objets ou des événements et la construction d'explications à caractère local, particulières voire subjectives.

Par exemple, pour une instance de script, les élèves vont pouvoir observer que les boucles b_2 et b_3 ont le même nombre de répétitions, ou que b_2 est répétée une fois de moins que b_1 [7]. Notons que certains constats locaux sont valides globalement :

— $\forall p, b_1^p = b_6^p$
— $\forall p, b_2^p = b_3^p = b_4^p = b_5^p = b_8^p$
— $\forall p, b_2^p = b_1^p - 1$
— $\forall p, b_7^p = 2b_6^p = 2b_1^p$

Les relations intra-objectales concernant la boucle b_7 (le « grand côté ») ne sont pas toutes générales. En fait, seule la relation $b_7 = 2b_6$ est vraie pour toute instance. En revanche, si on a par exemple $b_1^4 = 3$ et $b_7^4 = 6$, la relation $b_7 = b_1 + 3$ n'est vraie que pour l'instance 4.

7. Nous représentons l'intitulé de la i-ème boucle du script — telle que numérotée dans le schéma de la figure 2.4b, p. 135 — par l'intitulé b_i. Cet intitulé dénote aussi le nombre d'itérations de la boucle. Ainsi, si la première boucle est répétée trois fois, b_1 désigne cette boucle et dénote la valeur 3. Pour une instance de script traçant un TS de mesure p, nous noterons si besoin l'instance en exposant : b_1^p désigne la première boucle du script $TS(p)$. Voir la partie Conventions d'écriture p. 162.

Les relations inter-objectales sont établies entre des propriétés d'objets distincts, elles concernent aussi les transformations de ces propriétés : il s'agit du « passage des attributs aux relations » (ibid., p. 74). Il s'agira ici de mettre en relation le nombre d'itérations des boucles et les instances, cela concerne la généralisation arithmétique. Les principales relations valides que les élèves peuvent établir sont ainsi les suivantes :

— $\forall p, i \in \{1,2,3,4,5,6,8\} \implies b_i^{p+1} = b_i^p + 1$: à part pour la boucle b_7, pour passer d'une instance à la suivante on augmente de 1 le nombre de répétitions des boucles.
— $\forall p, b_7^{p+1} = b_7^p + 2$
— $\forall p, a, i \in \{1,2,3,4,5,6,8\} \implies b_i^{p+a} = b_i^p + a$ (ou $b_{i\neq 7}^p = b_i^a + (p-a)$)
— $\forall p, a, b_7^{p+a} = b_7^p + 2a$ (ou $b_7^p = b_7^a + 2(p-a)$)

Une erreur attendue est de considérer que ces relations entre boucles sont proportionnelles, ce qui n'est pas le cas : $\forall p, a, i \in [1..8] \implies b_i^{p\times a} \neq b_i^p \times a$.

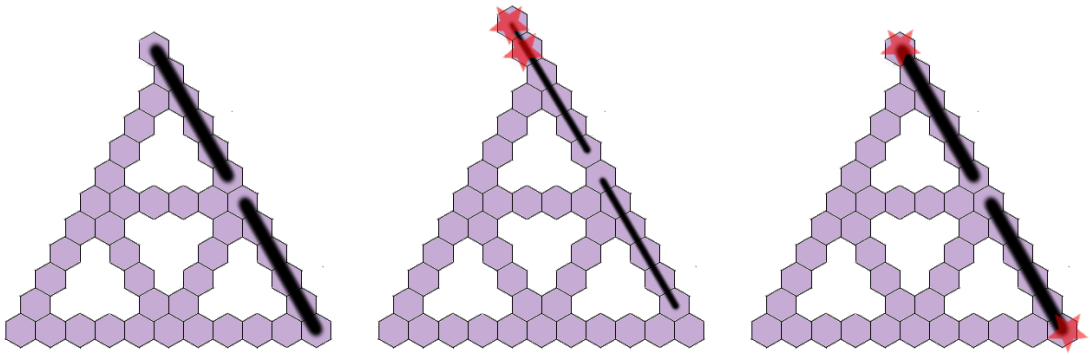
Lorsque des relations trans-objectales seront établies, les « transformations seront mises en relation afin d'aboutir à la construction de structures ». Il s'agira de non plus mettre en relation une instance avec une autre instance, mais de déterminer la structure d'une instance de façon générique : cela concerne la généralisation algébrique (naïve ou symbolique). C'est ce qui permettra aux élèves de concevoir un script générique traçant et dénombrant n'importe quelle instance en fonction de la valeur de la mesure entrée par l'utilisateur. Les relations valides à construire sont les suivantes :

— $\forall n, i \in \{1,6\} \implies b_i^n = n - 1$
— $\forall n, i \in \{2,3,4,5,8\} \implies b_i^n = n - 2$
— $\forall n, b_7^n = 2n - 2 = 2(n-1)$

Afin d'établir ces relations génériques, trois méthodes sont possibles :

1. Déterminer le n-ième terme d'une suite arithmétique de raison 1 ou 2 : comme $b_{i\neq 7}^{p+1} = b_i^p + 1$ et $b_7^{p+1} = b_7^p + 2$, connaissant b_i^4 — $b_1^4 = 3$, $b_2^4 = 2$, $b_7^4 = 3$ — :
   — $\forall n > 4, i \in \{1,6\} \implies b_i^n = b_i^4 + (n-4) \times 1 = n - 1$
   — $\forall n > 4, i \in \{2,3,4,5,8\} \implies b_i^n = b_i^4 + (n-4) \times 1 = n - 2$
   — $\forall n > 4, b_7^n = b_7^4 + (n-4) \times 2 = 2n - 2$
2. Identifier la relation entre des nombres, la transformation permettant de passer d'un nombre à l'autre :
   — Pour l'instance 4, la valeur de b_1 est 3, on a $4 \to 3$. Avec les différentes instances créées, on a aussi les faits $5 \to 4$, $6 \to 5$, etc. Pour passer de la valeur de l'instance à la valeur du nombre de répétitions des boucles, on enlève un. De même pour les autres boucles, sauf b_7.
   — Pour b_7, les faits construits devraient être les suivants : $4 \to 6$, $5 \to 8$, $6 \to 10$ etc. Pour passer de la valeur de l'instance à la valeur du nombre de répétitions des boucles, on prend le double et on enlève 2.
3. Mettre en relation les hexagones tracés, les boucles, et le nombre de répétitions des boucles. Il faut ainsi pour b_1 constater que si l'on trace un côté du triangle de base, on a un hexagone de trop : $b_1 = n - 1$. Pour b_7, on peut voir la boucle comme devant tracer $2n$ hexagones, ce qui serait deux de trop (figure 2.8, p. 148), ou comme devant tracer deux côtés de triangles de base, ce qui serait à chaque fois un hexagone de trop (figure 2.8b, p. 148)

La première méthode ne parait pas accessible pour des élèves de cycle 4, les suites n'étant pas au programme de ce cycle. Les deux autres méthodes sont en revanche envisageables.



(a) $2n$ **(b)** $2n-2$ **(c)** $2n-2$ ou $2(n-1)$

Figure 2.8 – Déterminer b_7 en fonction de la mesure n

Afin de pouvoir représenter l'intégralité d'un TS de mesure donnée dans l'espace d'exécution, un bloc d'initialisation spécifique à chaque script attendu est créé (voir les exemples pour les scripts traçant un TS4, un TS 20,21 ou 22, et le script générique, figure 2.9, p. 149). Le contenu de ces blocs (de ces procédures) n'est pas accessible aux élèves, pour trois raisons essentiellement : d'une part parce que nous ne souhaitions pas permettre leur modification, vu les implications que cela aurait sur le résultat final, d'autre part parce que l'initialisation utilise une expression mobilisant la variable *mesure* et que nous craignions que cela soit modélisant, et enfin parce que cela entraine des difficultés techniques de captation et de reconstitution du programme. De plus, nous souhaitions rendre transparent l'usage du bloc demandant à l'utilisateur d'entrer une valeur demander ◯ et attendre. Ce bloc est selon nous problématique parce qu'il s'agit à la fois d'une opération d'écriture (un affichage à l'écran), d'une opération de lecture (de la valeur entrée) et d'une opération de stockage implicite dans une variable prédéfinie dans l'EPGB, réponse. Nous avons donc fait le choix de ne pas faire apparaitre ce bloc pour les élèves, l'enseignante faisant, lors du début de la phase de généralisation générique, une démonstration et explicitation du bloc initialisation, précisant que sa fonction est d'ajuster le « zoom » et d'affecter la valeur lue à la variable mesure[8].

Nous verrons que cette solution n'est pas totalement satisfaisante, notamment parce que la modification de la *mesure* a une influence directe sur le niveau de zoom de l'affichage, ce qui amène les élèves — au moins dans un premier temps — à considérer que mesure, qui dénote la valeur entrée par l'utilisateur, dénote en fait le niveau de zoom. Ceci s'explique par une confusion entre le système mathématisé (pour reprendre les termes de Chevallard (1989)) et le système informatisé. En effet, outre les variables du système que sont la mesure (n), le nombre total d'hexagones (N), les valeurs du nombre de répétitions de chacune des boucles (b_i), on ajoute ici la dimension en pixels de chaque hexagone, T. Or cette variable ne devrait exister que par rapport au problème informatisé, et non par rapport au problème mathématisé. Or,

8. Le nom de la variable désignant le nombre d'hexagones d'un côté du triangle de base était, pour la version étudiée en Master, *taille*. Outre le fait que cela ne soit pas défini du point de vue mathématique, et donc que le terme de *mesure* était plus adapté, cela a introduit un problème de polysémie : taille désigne le nombre d'hexagones d'un côté des triangles de base, une variable taille_hexagone a été créée pour dénoter la dimension des hexagones, et des blocs existent dans l'EPGB, par exemple choisir ◯ % de la taille initiale, qui permettent de modifier la *taille* des lutins.

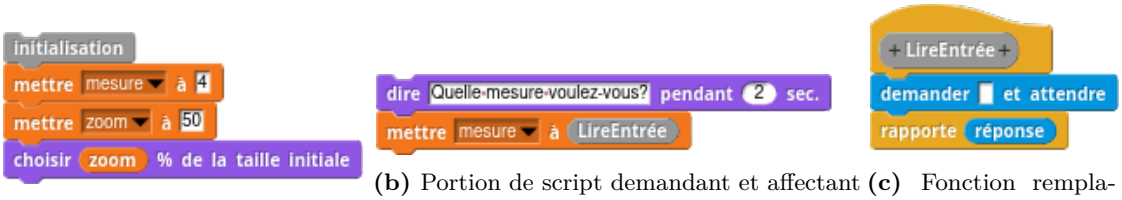
(a) TS4 **(b)** TS20, TS21 ou TS22 **(c)** TSn

Figure 2.9 – Blocs d'initialisation

nous nous sommes finalement posé le problème suivant :

$$\forall n, \left(\begin{array}{l} \texttt{le tracé est valide} = \mathit{vrai} \\ \wedge \\ \texttt{le tracé est entièrement visible} = \mathit{vrai} \end{array} \right) \implies \exists T, T \leq \frac{n}{200}$$

En reprenant les remarques d'Arzarello et Bardini (1.9.4, p. 119), n, quantifié de façon la plus externe, prend le rôle d'un paramètre définissant une relation avec *l'affichage*, ce qui risque de ne pas permettre de constater que n est *aussi* un paramètre définissant une relation avec *le tracé* (le motif, le nombre de répétitions des boucles). L'expérimentation avec les 3ième Segpa visait à tester la suppression de la relation entre la variable mesure et le niveau de zoom, tout en proposant une réécriture du mécanisme d'entrée d'une valeur (voir figure 2.10, p. 149). L'interruption de cette expérimentation a empêché de récolter des données sur les effets de ces modifications.

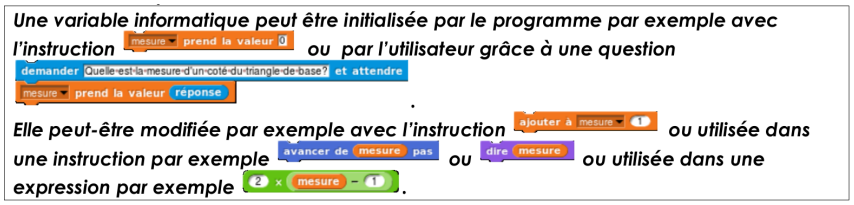


(a) Début de script instancié **(b)** Portion de script demandant et affectant une valeur lue à la variable *mesure* **(c)** Fonction remplaçant le bloc prédéfini

Figure 2.10 – Modifications envisagées de l'initialisation

Nous préciserons par la suite l'espace des contraintes *a priori* que l'on peut construire pour cette phase de la situation.

À la fin de cette partie, il est prévu une institutionnalisation de la variable informatique, une utilisation du programme générique pour répondre au problème, et un constat sur la durée d'exécution du programme. L'institutionnalisation prévue par l'enseignante est la suivante (Annexe A.1, p. 505) :



L'utilisation du programme pour résoudre le problème principal (trouver le plus grand TS possible constructible avec 1400 hexagones) aboutit à la solution du TS156, et les lancements de tests permettent d'établir un nouveau fait : plus la mesure est grande, plus le temps de tracé est long, devenant presque rédhibitoire.

Partie 4 et 5 (séances 6 et 7) : formulation algébrique et implémentation de formules

Suite à ce constat de lenteur [9], les élèves doivent rechercher une méthode permettant « d'aller plus vite que l'ordinateur ». La validation se faisant, pour chaque groupe pensant avoir trouvé une méthode efficace, de la façon suivante :

1. L'enseignante ou le chercheur propose une mesure quelconque.
2. Le programme est lancé, et dès que la valeur est entrée les élèves commencent à appliquer leur méthode.
3. La validation se fait à la fois sur la durée et sur le résultat.

Il s'agit ici d'amener les élèves à modéliser la situation et la formuler, en sortant du cadre algorithmique. La consigne étant « Par groupe de 4, trouver le moyen de calculer le nombre d'hexagones dans n'importe quel triangle de Sierpinski. », il n'y a pas de nécessité pour les élèves d'établir une formulation symbolique, mais il leur est demandé de justifier leur méthode. Du point de vue du chercheur, on va pouvoir observer ici si le travail sur le paramètre dans un registre de langage informatique est transposée dans un langage algébrique ou pseudo-algébrique. La séance suivante rendra nécessaire l'expression du nombre d'hexagones d'un certain TS dans un langage informatique, puisqu'il s'agira d'entrer cette formule dans un programme, qui en testera la validité [10]. L'implémentation de la solution impose en outre de traiter des problèmes de la représentation de la structure profonde de l'expression trouvée qui est sous une forme *superficielle* :

> [...]la grammaire [générative] engendre des formules unidimensionnelles telles que « `{1+2} over {3+4}` » qui forment ce qu'on appelle la « structure profonde » et où toutes les informations pertinentes (tant pour la détermination de la structure syntaxique que pour le calcul de la dénotation) figurent explicitement. [...] La forme *superficielle* des formules est donc caractérisée par l'*implicite* et les informations *indirectes*, par opposition à la forme profonde qui présente explicitement l'ensemble des informations contenues dans le *sens* des formules. Dans « $2 + 3 \times 4^{2^3}$ » par exemple, l'ordre des opérations n'est pas *explicitement* marqué, ni, ce qui revient au même, la portée des opérateurs (ce qui est net pour l'élévation au cube). (Drouhard et Panizza, 2012, p. 219)

Ainsi, une expression telle que « la mesure, on fait -2, la mesure $\times 9$, $+6$ », s'écrirait en algèbre, en prenant n pour la mesure, « $(n - 2) \times 9 + 6$ », et devrait s'implémenter dans l'EPGB sous une forme explicite 

.

Suite à ces séances, l'enseignante s'appuiera sur la situation pour faire produire aux élèves diverses écritures possibles du nombre d'hexagones total d'un TS de mesure n, ce qui permet de mettre au travail les équivalences de formules et les règles de réécriture ou de transformation.

9. Ce qui pourrait être l'occasion d'aborder les notions de complexité d'un algorithme.
10. Sur des instances choisies par l'utilisateur, il ne s'agit pas ici de construire une preuve de programme.

2.1.6 Problèmes posés

Poser le problème dans le Cadre de l'Apprentissage par Problématisation, ce n'est pas seulement formuler une question, c'est aussi avoir une idée des formes de la solution (Fabre, 2017, p. 28). Dans la situation présentée, plusieurs problèmes coexistent et sont imbriqués :

- Quel est le plus grand TS donné que l'on peut construire avec p hexagones ? C'est la question de départ, le problème principal qui rend nécessaire la résolution d'autres problèmes. Forme de la solution : une mesure n telle que le nombre d'hexagones nécessaires pour tracer le TS de mesure n soit inférieur ou égal à p et que le nombre d'hexagones nécessaires pour tracer le TS de mesure $n+1$ soit supérieur à p.
- Comment trouver la mesure n la plus grande, telle que le TS de mesure n utilise moins de p hexagones ? Forme de la réponse : une méthode permettant de trouver la mesure.
- Comment savoir combien d'hexagones sont nécessaires pour construire un TS de mesure n ? Forme de la solution : une méthode permettant de tracer et dénombrer un TS de mesure n quelconque.
- Comment construire un script dans Snap ! traçant et dénombrant tout TS, sa mesure n étant donnée ? Forme de la solution : un script traçant et dénombrant un TS dont la mesure n est entrée par l'utilisateur lors de l'exécution.
- Comment modifier un script traçant et dénombrant un certain TS de mesure n pour qu'il trace un TS de mesure m ? Forme de la solution : une méthode permettant de construire un script traçant et dénombrant un TS d'une mesure donnée m.
- Comment modifier un script traçant et dénombrant un certain TS de mesure n pour qu'il trace un TS de mesure $n+1$? Forme de la solution : une méthode permettant transformer un script traçant un TS de mesure n en un script traçant un TS de mesure $n+1$.
- Comment modifier un script traçant et dénombrant une certaine instance d'un TS pour qu'il trace et dénombre une autre instance ? Forme de la solution : une méthode permettant de trouver les changements à apporter au script.
- Quels sont les éléments déterminant le tracé du TS dans le script traçant et dénombrant un certain TS ? Forme de la solution : les instructions du script qu'il faut modifier / les instructions du script qu'il faut ajouter.
- Comment modifier les valeurs du nombre d'itérations des boucles d'un script traçant et dénombrant un certain TS pour qu'il trace et dénombre un autre TS ? Forme de la solution : des expressions du langage précisant le nombre d'itérations des boucles.
- Comment modifier un script traçant un certain TS pour qu'il dénombre les hexagones nécessaires au tracé de ce TS ? Forme de la solution : les instructions du script qu'il faut modifier / les instructions du script qu'il faut ajouter.

Puisque nous focalisons notre recherche sur la construction de la variable dans un rôle de paramètre, nous nous attacherons à étudier notamment la position, par les élèves des problèmes liés à la construction du script générique. Nous n'étudierons pas ici comment les élèves posent et construisent le premier problème, ni celui concernant la modification du script de tracé pour qu'il dénombre les hexagones.

2.1.7 Espace des contraintes *a priori*

Nous nous intéressons plus spécifiquement ici à la construction du concept de variable dans un rôle de paramètre, soit la partie 3 de notre situation (séances 3 à 5). En complétant la présentation faite plus haut, nous allons construire l'espace des contraintes *a priori* de cette phase de généralisation. Nous nous appuierons ensuite sur cet espace pour analyser l'activité des élèves.

> Pour cela nous utiliserons des schémas synoptiques de type espaces de contraintes proposés par Orange (2000). Ces schémas, élaborés par les chercheures didacticiennes que nous sommes, donnent à voir les types de registres co-construits, autrement dit ce qui se joue dans le travail d'un problème en termes de construction et de mise en tension dynamique de contraintes empiriques et théoriques, jusqu'à la mise au jour de nécessités auxquelles les solutions du problème doivent se conformer. (Hersant et Orange Ravachol, 2015, p. 99)

Ces schémas ne représentent pas l'organisation chronologique du processus de problématisation, mais plutôt l'organisation logique.

> Ces espaces de contraintes sont des outils à la fois pour l'analyse *a priori* ou *a posteriori* de situations de classes et pour l'analyse de recherches réalisées par des chercheurs (Hersant, 2010 ; Orange Ravachol, 2005) (ibid., p. 100)

Pour cet espace des contraintes *a priori*, comme pour les espaces de contraintes *a posteriori*, nous avons adopté le codage suivant :

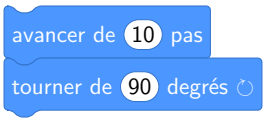
— les éléments grisés sont les faits (qui devraient être) construits ou mobilisés par les élèves : ils constituent le registre des données. Nous ne pouvons bien entendu pas *a priori* connaître les faits qui seront effectivement construits, nous indiquons donc seulement ceux qui « réduisent le champ des possibles » ;

— Les éléments du registre des nécessités sont en bleu. Elles représentent ici les objectifs de construction de savoirs de cette phase. Les nécessités locales, spécifiques au problème, sont représentées avec un bleu plus clair.

— Dans le registre explicatif, nous indiquons ici les registres sémiotiques qui sont susceptibles d'être mobilisés par les élèves, ainsi que leur mouvement.

Les relations logiques entre ces éléments sont représentés par des flèches. Dans un espace des contraintes classique du CAP, on ne fait généralement pas apparaitre les relations de causalités, mais dans notre situation ces enchainements logiques nous paraissent essentiels pour comprendre ce qui amène les élèves à construire (ou non), les nécessités ciblées. Afin de différencier nos représentations des espaces des contraintes tels que les conçoit Christian Orange, nous nommerons ces espaces « espaces des faits-contraintes ».

Nous nous intéressons ici uniquement à la construction du script générique. Une première nécessité locale essentielle est que, quelle que soit la mesure n donnée, le script trace le TS de mesure n. Traduit plus localement, cela signifie qu'un seul script doit tracer plusieurs instances de TS, et que si la mesure change, le TS tracé doit lui aussi changer. Notons que cette nécessité a été ici didactisée : les élèves pourraient très bien changer les valeurs du nombre de répétitions des boucles pour construire une certaine instance, le coût cognitif étant moindre que de passer à la construction d'un script générique. C'est ce qu'on peut observer aussi par exemple lorsqu'on demande aux élèves de faire un script qui trace un carré. Il n'est pas rare qu'un élève déplace seulement deux blocs sur l'espace de programmation, éventuellement en les joignant, pour obtenir le script suivant :



Ce script n'est évidemment pas un programme traçant un carré, mais, comme l'utilisateur peut déclencher l'exécution des scripts en cliquant dessus, il suffit de cliquer quatre fois de suite sur ce script pour tracer un carré : ici, les élèves ne sont pas dans une posture de programmeur, ils ne font pas de la programmation mais du pilotage, ils sont dans une posture d'utilisateur. Afin d'éviter cette posture, une contrainte est imposée : les tests doivent se faire avec l'espace d'exécution en plein écran — ce qui rend inaccessible l'espace de programmation. Comme le

signalent Samurçay et Rouchier (1985, p. 242), « l'exécution de la procédure se fait et doit se faire dans un temps distinct de celui de la construction du programme : l'exécution est différée ». Une autre solution serait de transmettre le programme à un autre groupe ou une autre classe, mais dans ce cas, il n'y aurait pas de rétroaction directe permettant aux élèves d'ajuster les actions.

Pour ce script générique, changer le nombre de répétitions des boucles suffit à déterminer une instance particulière : les b_i suffisent à définir le script. C'est donc les expressions dénotant le nombre de répétitions des b_i qu'il est nécessaire de changer. Or, si une expression est affectée au nombre de répétitions d'une boucle ($b_i = expr$), cette expression est fixe, alors que son dénoté doit changer en fonction de la mesure. Il doit donc exister un bloc, une expression de Snap !, qui résout le problème.

Si l'on détermine le nombre de répétitions d'une boucle par une représentation d'un entier ou une expression arithmétique ne concernant que des entiers, alors il n'y a qu'un seul TS qui peut être tracé, et si on veut tracer un TS différent, il faudrait modifier la valeur ou l'expression : les b_i ne peuvent donc pas être des nombres, ni des expressions arithmétiques.

Il doit donc exister une expression qui permet de représenter les valeurs des b_i pour plusieurs mesures : il y a nécessairement une expression E_θ du langage qui permet d'exprimer b_i en fonction de la mesure. Il y donc nécessairement une relation entre les b_i et la mesure. Comme on a dû construire différents scripts pour différentes mesures, cette nécessité a pu être déjà construite dans les séances précédentes.

Mais si cette expression ne contient que des opérateurs et des nombres, une seule instance est traitée, E_θ doit donc contenir, dans le langage, une représentation R_θ qui dénote la mesure, et qui n'est pas l'écriture d'un nombre. Comme R_θ ne peut pas être un nombre, il doit y avoir un autre élément du langage qui permette de dénoter la mesure. Cet élément est (ou du moins peut être) mesure : c'est lorsque le contenu de cet objet change que le script doit changer.

Si les relations entre les boucles et la mesure sont établies, alors les expressions dans θ de ces relations en découlent.

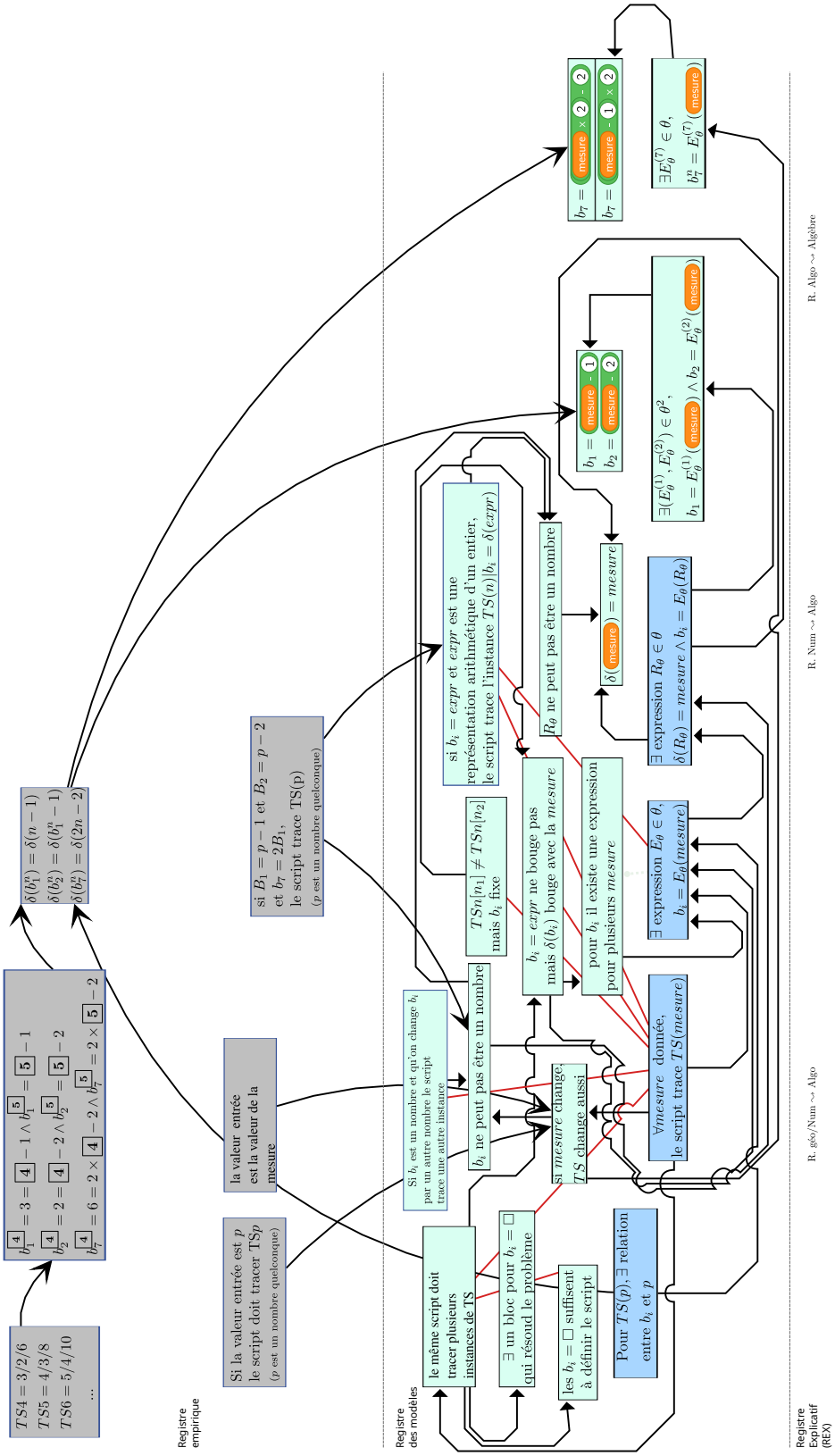


Figure 2.11 – Espace des faits-contraintes *a priori*

2.2 Dispositif expérimental

Analyser la construction d'un concept, ou la problématisation chez les élèves, c'est travailler sur un processus dynamique : le résultat final, ici un programme, n'est pas suffisant pour comprendre ce qui se joue. Afin de construire les espaces de contraintes, il est possible de s'appuyer sur les interactions entre élèves, mais ce n'est pas suffisant. Ceux-ci sont en situation d'action, et ils ne verbalisent ni toutes leurs actions, ni toutes leurs hypothèses, ni la validation ou invalidation d'hypothèses. Cependant, nous pensons que ces éléments argumentatifs peuvent être identifiés en utilisant la transcription des actions et évènements de programmation. Afin de pouvoir mettre en relation action, rétroaction, et ré-action des élèves, il nous faut ainsi avoir accès à l'activité des élèves, c'est-à-dire notamment aux modifications qu'ils opèrent et aux tests qu'ils font.

En Master, nous avions utilisé une captation vidéo de l'écran des ordinateurs des élèves (*screencast*). Cependant, si cela permettait bien une captation dynamique, elle était continue. Se pose alors dans ce cas la question de la granularité : que transcrire, jusqu'où détailler, comment rendre compte de la dynamique et du lien action-rétroaction-réaction ? Il s'agit de s'interroger sur le choix des données à transcrire, comment les transcrire, et comment les représenter. L'appui sur une vidéo est possible, mais il est parfois difficile d'envisager *a priori* les données qui seront nécessaire à l'analyse, et toute nouvelle analyse nécessitera de nouvelles données. Ceci impliquerai donc de refaire la transcription, ce qui peut s'avérer complexe et chronophage, surtout si l'on s'intéresse à de nombreux groupes.

Nous avons donc fait le choix, au début de cette recherche, d'automatiser la captation et la transcription des actions des élèves lorsqu'ils programment, avec un grain fin. Cela permettra plusieurs représentations, et granularité diverse, suivant les besoins (voir la partie Construction et analyse des données p. 196 pour cette situation)

Dans les sections suivantes, nous allons présenter les fonctions attendues du dispositif expérimental, ainsi que sa structure. Ce dispositif intègre notamment un système de captation automatisé que l'on présentera, de même que le système de visualisation et représentation des données, sans rentrer dans les détails techniques. Les éléments suivants sont pour l'essentiel repris d'une intervention lors d'un atelier du colloque EIAH 2019 (Legrand, 2019).

2.2.1 Fonctions recherchées

Nous cherchons à concevoir un dispositif permettant de capter avec une granularité très fine les *événements de programmation*. Dans le cadre de notre dispositif, nous définissons un *événement de programmation* de la façon suivante :

Définition 9 (Événement de programmation) *Nous considérons comme évènement de programmation $e(g,t)$ tout changement d'état de l'EPGB — l'interface de programmation dans un sens très général —, lié au groupe (utilisateur) g au temps t.*

Nous avons défini trois catégories d'évènements, que nous détaillerons ultérieurement :

1. modification de l'environnement de programmation,
2. modification de l'état du programme,
3. modification de la structure du programme.

En effet, les évènements et actions récoltés doivent être d'une granularité suffisamment assez fine, puisque nous ne savons pas encore ce qui sera pertinent ou non, mais doivent aussi être catégorisés pour un traitement plus global. Ils doivent aussi bien entendu être stockés pour des manipulations et analyses ultérieures. Afin de couvrir l'ensemble des éléments hypothétiquement pertinents, ces données doivent nous permettre de :

— visualiser les changements, les modifications apportées au programme,
— visualiser les essais testés par les élèves : le moment des essais, les valeurs testées, les réactions (arrêt, modifications) aux rétroactions,
— visualiser les résultats obtenus lors de l'exécution des programmes,
— reconstruire l'état complet du programme (dans le sens « ensemble de scripts ») d'un groupe g à un instant t quelconque.

L'ensemble de ces trois éléments nous permet de reconstruire de ce que nous nommons *l'histoire du programme*, une « histoire » étant vue comme « connaissance et récit des évènements du passé jugés dignes de mémoire ; les faits ainsi relatés » [11], ou encore ensemble d'événements, évolution concernant une personne ou une chose [12]. Il s'agit, en se basant sur des événements passés et décrits, de reconstruire les différentes versions du programme (et donc de chacun de ses scripts), ses exécutions et ses résultats.

D'autres représentations des données captées sont aussi envisagées, nous les présenterons succinctement.

En outre, nous souhaitons ne pas ajouter de contraintes techniques aux expérimentateurs, voire même mettre en place un dispositif qui leur facilite la mise en œuvre technique des séances proposées. Enfin, l'environnement de programmation utilisé doit répondre aux exigences institutionnelles et ne pas entraîner de changements problématiques pour les élèves comme pour les enseignants. C'est entre autre ce qui nous a amené à choisir d'utiliser l'EPGB Snap ! et non Scratch.

2.2.2 Choix de l'EPGB

Nous avons choisi comme support informatique de l'activité l'EPGB Snap ! [13] de l'Université de Berkeley. Celui-ci se base sur Scratch, ce qui permet de ne pas mettre en difficulté les élèves ou enseignants qui ont l'habitude de cet EPGB. Cependant, Snap ! dispose d'avantages didactiques et techniques qui ont orienté notre choix.

Ainsi, un mode pas-à-pas est disponible, permettant de ralentir l'exécution ou de la faire avancer instruction par instruction, en mettant en valeur les instructions en cours d'exécution. Ceci permet d'explorer un algorithme inconnu, ou de le debugger, de façon plus simple et rapide que la méthode traditionnellement utilisée avec Scratch : ajouter des blocs 

.

Les différents blocs sont organisés par famille (ou catégories), et non disposés les uns à la suite des autres. En outre, et cela est didactiquement fondamental pour nous, il est aisé de décider quels blocs sont disponibles pour les élèves : il suffit, lors de la construction du programme qui servira de base à la séance, de sélectionner quels blocs seront visibles ou invisibles. Cela permet ainsi de définir de quelles instructions les élèves disposent pour résoudre un problème, et donc de poser des contraintes diverses si nécessaire. De plus, cela évite aux élèves de se perdre dans l'exploration de la centaine de blocs existant sous Scratch (et plus nombreux encore sous Snap !).

Enfin, ce logiciel est écrit en javascript, ce qui le rend aisément modifiable et permet de faire une version spécifique que l'on peut distribuer aux différents postes : il n'est pas nécessaire d'installer un logiciel, il s'agit simplement d'une page web.

2.2.3 Structure du dispositif

Afin de satisfaire ces objectifs, nous avons conçu un dispositif en deux parties :

11. Robert, 2023a.
12. CNRTL, 2023a.
13. https://snap.berkeley.edu/

1. la partie *serveur* permet de distribuer une version instrumentée de Snap !. Cette version modifiée par nos soins permet de capter les événements de programmation et de les envoyer au serveur, qui stocke ces événements dans une base de données (en l'occurrence `MySQL`) ;
2. la partie visualisation, qui permet de représenter les données stockées.

La partie serveur permet d'identifier l'utilisateur (en différenciant enseignant et groupe d'élèves). Après identification, l'utilisateur reçoit notre version modifiée de Snap !. La version « enseignant » de Snap ! n'inclut pas la captation, mais ajoute la possibilité d'accéder aux programmes sauvegardés par les élèves (automatiquement ou non).

Du côté de l'enseignant, une interface permet d'organiser la diffusion (la distribution) d'un programme de base, à chacun des groupes. Chaque groupe peut éventuellement recevoir un programme différent. L'intérêt est ici de mettre à disposition des élèves un environnement de programmation préparé, avec d'éventuels morceaux de scripts mais aussi en ne rendant visible que les instructions choisies pour des raisons didactiques. L'enseignant peut aussi consulter les programmes des élèves, sauvegardés par eux, ou sauvegardés automatiquement en fin de séance [14].

Du côté élèves, les menus ont été simplifiés, et la sauvegarde se fait automatiquement sur le serveur, avec ajout de la date et de l'heure. Les élèves ont bien entendu la possibilité de charger un de leurs anciens programmes. Ceux-ci étant stockés sur le serveur, les élèves n'ont pas à rester sur le même poste informatique. La version modifiée de Snap ! capte les événements de programmation et les envoie au serveur pour stockage, en incluant dans cet envoi les détails de l'événement, le groupe, et le moment d'occurrence de l'événement. Enfin, une sauvegarde automatique est effectuée à la fin de la session, avec une capture d'écran de la fenêtre de l'EPGB, notamment afin de pouvoir vérifier la cohérence entre la reconstitution de l'histoire faite par le dispositif, et les états initiaux et finaux du programme.

Une autre partie concerne la représentation des événements de programmation. Pour la reconstitution de l'histoire du programme, elle nécessite une transformation préalable des données (dont le résultat est stocké dans une base de données `Mongodb`, plus adaptée au format choisi). Un autre logiciel est conçu pour représenter les données concernant spécifiquement la situation, en se basant sur les données stockées sur le serveur (voir le chapitre Analyse, p. 217).

La séparation de l'envoi des données d'une part et de la sauvegarde et du traitement d'autre part peut permettre d'adapter ce système de recueil à d'autres interfaces de programmation par blocs, par exemple `Pixel'Art` [15], développé par Christophe Declercq, et qui dispose déjà d'une captation, en mode local, de certains événements.

2.2.4 Système de captation

Les événements captés sont organisés en trois catégories. Les événements de modification de l'environnement sont ceux qui modifient l'environnement global de l'interface. Sont concernés les lancements de l'interface, les créations, sauvegardes ou chargements de programme, la navigation dans les différentes catégories de blocs. Les modifications de l'affichage sont aussi prises en compte : changement de la taille de l'espace d'exécution, passage en plein écran, activation ou désactivation de l'affichage des variables. Certains événements liés à la souris sont aussi captés, notamment les actions sur les menus, ou le fait de cliquer sur un bouton ou un script (ou un bloc).

14. Une évolution future serait de permettre aux enseignants d'avoir accès aux différentes représentations des données, notamment l'histoire du programme, ce qui n'est pas le cas pour le moment, le processus n'étant pas totalement automatisé.

15. https://declercq-c.univ-nantes.io/PixelArt3/

Les événements de modification de l'état du programme sont ceux qui changent l'état d'exécution d'au moins un script ou bloc le composant. Ainsi, les actions déclenchant l'exécution, la pause, l'arrêt ou le ralentissement de l'exécution du programme font partie de cette catégorie. L'arrêt non manuel d'un script — lorsqu'il se termine normalement ou en produisant une erreur — est aussi un événement de modification de l'état du programme. Tous ces événements sont accompagnés d'un événement créé par la modification de l'EPGB, qui consiste en une copie de l'espace d'exécution au moment du déclenchement de l'événement de changement d'état du programme. Cela permet d'obtenir la rétroaction produite. Les différents changements de l'espace d'exécution ne sont cependant pas pris en compte : cela augmenterait de façon conséquente les données envoyées au serveur, et nous pouvons simuler les modifications de l'espace d'exécution d'un script en reconstituant ce script et en l'exécutant — puisque nous disposons de chaque version différente des scripts. L'invite destinée à l'utilisateur ainsi que l'entrée d'une valeur concernant cette invite font aussi partie de cette catégorie.

Enfin, la partie la plus importante concerne les modifications de la structure du programme. Toute modification d'un script fait partie de cette catégorie :

- le chargement ou la création d'un nouveau programme — événement accompagné d'un événement de modification de l'état de l'environnement, mais ici accompagné de la description du programme —
- la création, le déplacement, la suppression, la copie des blocs,
- la navigation dans l'historique (revenant à des déplacements ou suppressions de blocs),
- la création, la suppression, le renommage de variables locales ou globales,
- le changement d'un paramètre d'une instruction, que ce soit la modification d'une valeur ou d'une expression,
- le déplacement, l'insertion ou la suppression de blocs renvoyant une valeur.

Un certain nombre d'événements ou de détails des événements ne sont pas captés dans la version utilisée du dispositif [16]. Notamment, on ne stocke pas les informations sur la localisation des blocs — ce qui serait utile si on voulait étudier les phénomènes de genèses instrumentales (Rabardel, 1995) chez les élèves avec un EPGB, par exemple l'utilisation d'espaces brouillon, ou la mise en rapport (par voisinage) de scripts —. Les déplacements manuels des lutins sur l'espace d'exécution, de même que les autres actions avec la souris sur cet espace, ne sont pas non plus captées. Les modifications des sons, des personnages ou des scènes ne sont pas non plus considérées. En outre, trois limites (au moins) sont présentes dans la version utilisée du dispositif :

- Les difficultés techniques de prise en compte des blocs créés (les procédures ou fonctions [17]) permettent de détecter une création de bloc, mais le nom du bloc créé n'est accessible qu'à la condition que ce bloc soit déplacé sur l'espace de programmation.
- Un appui sur les touches du clavier n'est capté que lorsqu'il déclenche un événement d'exécution de programme. Lors de l'entrée d'une valeur lors de l'invite ou pour changer la valeur d'un paramètre d'une instruction, seule la valeur finale est envoyée au serveur, et seulement lorsque l'entrée est validée.
- Enfin, dans les EPGB comme Scratch ou Snap !, on peut associer un programme (un ensemble de scripts) à un personnage. Dans notre dispositif, si plusieurs lutins disposent de programmes, il n'y a pas d'association enregistrée entre script et lutin, on considère qu'il n'y a qu'un seul programme.

16. Ce sont des évolutions possibles.
17. Car Snap !, contrairement à Scratch, permet de créer des fonctions.

2.2.5 Visualisations disponibles

Les deux visualisations fondamentales disponibles sont celles de la représentation de l'histoire du programme. Une première représentation, diachronique, montre sur un axe horizontal l'ensemble de l'histoire du programme — les différentes versions de l'ensemble des scripts — ainsi que les invites, leurs réponses, et les captures d'écran lors des exécutions ou arrêt de script (voir figure 2.12, p. 160, ou les annexes informatisées). Les différents scripts à un moment donné sont décrits par un pseudo-langage reprenant le vocabulaire de l'EPGB, et sont empilés verticalement. Pour faciliter la lecture, les éléments modifiés, et le script modifié, sont mis en valeur, et il est possible de n'afficher que les scripts modifiés. La deuxième représentation, synchronique (figure 2.13, p. 161), montre, pour un moment choisi, les différents scripts côte à côte, sur l'axe horizontal. D'autres événements sont précisés, mais les images des captures d'écran ne sont pas rendue visibles. Cette représentation, conçue à l'origine avec un objectif d'aide à la mise au point du dispositif, permet d'avoir une photographie de l'état du programme à un moment donné, et est utilisée pour en extraire une écriture utilisable dans un texte (avec une procédure en partie manuelle).

D'autres représentations des données captées sont disponibles : enchainement des sauvegardes, chargement et création ; évolution du nombre d'instructions, recherche de la première occurrence d'une instruction — ici la boucle — (Legrand, 2019, p. 6).

Toute représentation spécifique est bien entendu possible à partir de données captées ou de l'histoire reconstituée.

Figure 2.12 – Exemples d'affichage diachronique

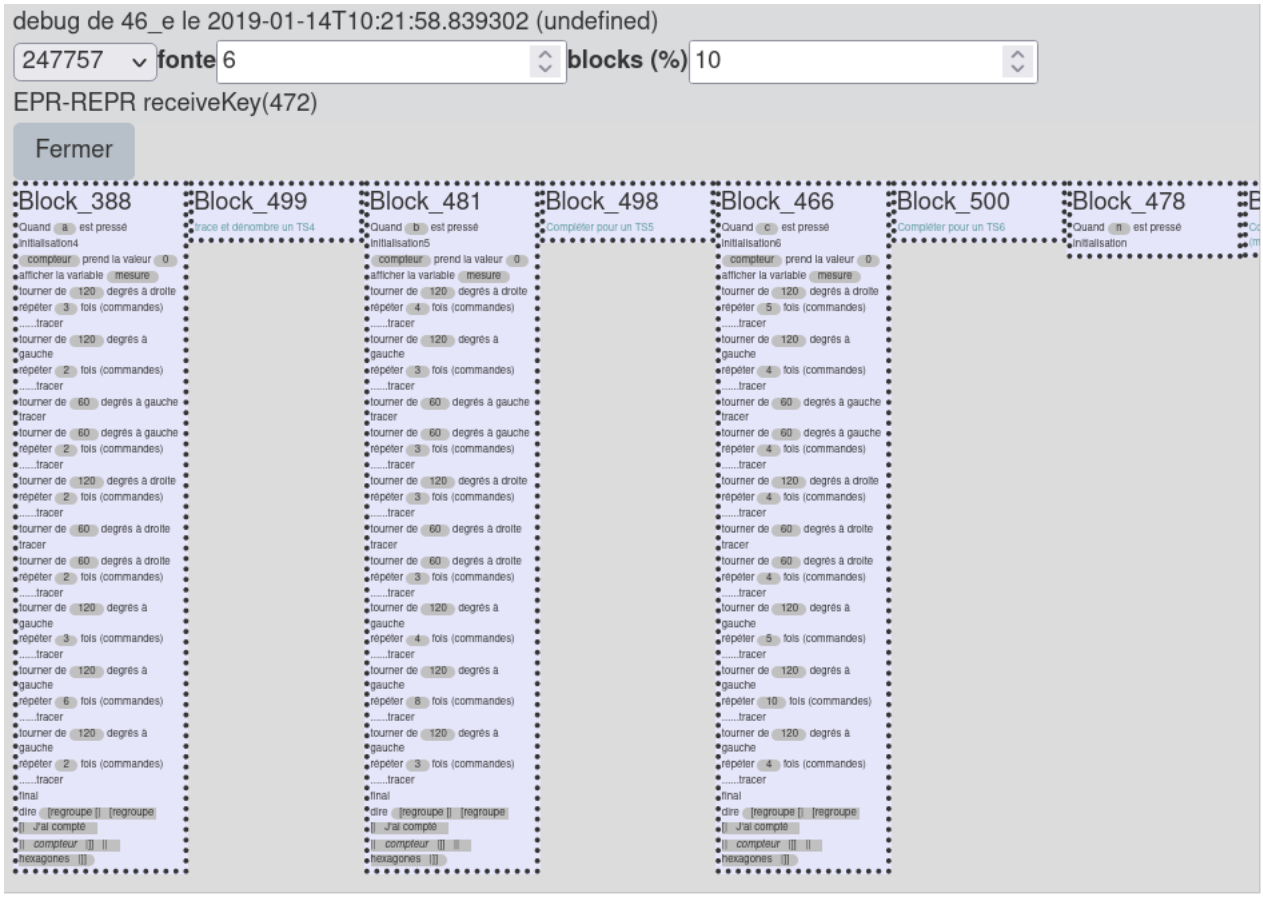


Figure 2.13 – Exemple d'affichage synchronique

2.2.6 Mise en œuvre de la situation

L'expérimentation a été faite dans un collège ordinaire, par une enseignante de mathématique expérimentée. Cette enseignante avait déjà participé à l'expérimentation en Master. L'expérimentation a concerné, en janvier 2019, deux classes de 4ème dont l'enseignante avait la charge [18]. Les binômes sont dénommés en accolant leur numéro de classe et une lettre différenciant les différents groupes. Cette dénomination est aussi leur identifiant pour se connecter au dispositif. Ainsi, le troisième binôme de la classe de quatrième cinq serait désignés par « 45c ». Les groupes ont été constitués par l'enseignante, et les lettres ont été affectées aléatoirement, excepté pour les groupes dont une captation vidéo et audio était prévue.

Les élèves, avant cette séquence, avaient découvert l'EPGB Scratch dans des situations classiques de tracés. En algèbre, ils ont une pratique du calcul littéral à partir d'une formule donnée, mais pas d'expérience de modélisation ou de mathématisation d'une situation.

Lors de la phase 1 les élèves explorent individuellement, tout en échangeant entre eux. Pour les phases 2 et 3, c'est-à-dire les phases concernant le motif informatisé, ils sont par binômes, les binômes restant figés sur les quatre séances concernées. Ces binômes sont choisis et constitués par l'enseignante, de façon à ce que chaque binôme soit homogène. En phase 4 et 5, les différents binômes sont séparés pour former des groupes hétérogènes de quatre élèves. L'ensemble de la séquence se déroule dans une salle d'informatique. Celle-ci dispose d'un espace central formé de tables et chaises, comme une classe ordinaire, les différents postes informatiques étant réparti contre les murs autour de cette salle. Un tableau interactif est disponible et a été utilisé par l'enseignante tout au long de cette séquence.

2.3 Conventions d'écriture

2.3.1 Représentations des boucles

Dans la partie généralisation de notre situation, les éléments caractéristiques d'un script sont les boucles et les expressions permettant de déterminer le nombre de répétitions de ces boucles, seuls éléments à devoir être modifiés. Ces expressions seront désignées par l'*intitulé* de la boucle, formé de la lettre b avec en indice le numéro d'ordre de la boucle dans le script. Dans l'exemple figure 2.14 (p. 165), la boucle n°1 est :

b_1 désigne alors l'expression ③ de la boucle n°1. On écrira ainsi indifféremment :

— $b_1 =$ 

— $b_1 =$ ③

— $b_1 =$ "3"

— $b_1 = 3$

18. Une expérimentation a débuté la même année avec une classe de troisième Segpa mais a dû être interrompue.

Nous utiliserons les guillemets « "" » pour encadrer une expression en pseudo-langage algorithmique, notamment lorsque la confusion est possible avec une écriture algébrique. Ainsi, $b_1 =$ "$3+1$" est clairement identifiable à $b_1 =$ répéter (3 + 1) fois ou $b_1 =$ (3 + 1). En revanche, $b_1 = 3+1$ pourrait tout aussi bien représenter $b_1 =$ (3 + 1) que $b_1 =$ (4). Dans une expression algorithmique, la variable représentant valeur de la mesure d'un TS sera noté « mesure » ou « "*mesure*" » ou « *mesure* ». Dans une expression algébrique, elle sera désignée par la lettre n. Ainsi, on écrira indifféremment :

- $b_1 =$ répéter (mesure + 1) fois ;
- $b_1 =$ (mesure + 1) ;
- $b_1 =$ "$mesure + 1$" ;
- $b_1 = mesure + 1$, écriture algorithmique équivalente à celle ci-dessus ;
- $b_1 = n + 1$, écriture algébrique permettant le calcul du nombre de répétition de la boucle n°1 [19].

Afin de différencier les boucles correspondant à différents scripts, la mesure du TS censé être tracé et dénombré par ce script sera indiquée en exposant : b_1^p représente la boucle n°1 du script censé tracer un TS pour une certaine mesure donnée p. Par convention, n servira à représenter le script permettant de tracer et dénombrer toute instance de TS.

Dénoté de l'expression d'une boucle « $b_i = expr$ » met en relation l'intitulé de la boucle n°i et l'expression (écrite dans un certain langage) permettant de déterminer le nombre de répétitions de cette boucle à l'exécution. Cette valeur, dénotée par l'expression $expr$, sera notée $\delta(b_i)$ ou $\delta(expr)$. Ainsi, si $b_1 = 5-1$, $\delta(b_1) = \delta(5-1) = 4$. Lorsque le script permet à l'utilisateur d'entrer la valeur de la mesure, celle-ci sera précisée entre crochets : $\delta(expr)[p]$ est la valeur de l'expression lorsqu'on remplace la $mesure$ dans l'expression par la valeur de p. Si $b_1 = mesure+1$, $\delta(b_1)[5] = 6$; et $\delta(b_1)[12] = 13$. De même, $\delta(5-1)[5] = 4$ et $\delta(5-1)[12] = 4$ [20].

2.3.2 Représentation des boucles et des actions de modification des boucles

Nous définissons les ensembles d'expressions suivants :

- $\mathfrak{b}$ est l'ensemble des *intitulés* possibles des boucles d'un script, soit l'ensemble des écritures de la forme « b_i » pour $i \in \mathbb{N}^*$. $\mathfrak{b} = \{b_1, b_2, b_3, \dots\}$.
- $\mathfrak{B}$ est l'ensemble des expressions représentant la *relation* entre l'intitulé d'une boucle et l'expression du nombre de répétitions de cette boucle. Une expression de $\mathfrak{B}$ est constituée :
  - d'une expression d'un intitulé de boucle (un élément de $\mathfrak{b}$),
  - du signe « = »,
  - d'une expression dont l'interprétation permet de déterminer à l'exécution le nombre de répétitions de la boucle ; cette expression peut être exprimée dans différents langages, ce langage étant éventuellement précisé si nécessaire.

  « $b_1 = 4$ », « $b_7 =$ "$5-1$" », « $b_2 =$ (mesure - 2) », « $b_{125} = 2n-2$ » sont des éléments de $\mathfrak{B}$.
- $\mathfrak{A}$ est l'ensemble des expressions représentant l'*affectation* de l'expression du nombre de répétitions à une boucle donnée. Une expression de $\mathfrak{A}$ est constituée :

19. C'est ici un exemple erroné, la boucle attendue étant $b_1 = n-1 =$ (mesure - 1)

20. Nous verrons que dans le cadre de la situation étudiée, $\forall i, j \in \mathbb{N}^*, i \neq j \implies \delta((5-1))[i] = \delta((5-1))[j]$ n'est pas une évidence pour les élèves.

— d'une expression d'un intitulé de boucle (un élément de $\mathfrak{b}$),

— du signe de l'affectation « ← »,

— d'une expression dont l'interprétation permet de déterminer à l'exécution le nombre de répétitions de la boucle ; cette expression peut être exprimée dans différents langages, ce langage étant éventuellement précisé si nécessaire.

« $b_1 \leftarrow 4$ », « $b_7 \leftarrow$ "$5-1$" », « $b_2 \leftarrow$  », « $b_{125} \leftarrow 2n-2$ » sont des éléments de $\mathfrak{A}$. Un élément de $\mathfrak{A}$ désigne une *action* modifiant un script, alors qu'un élément de $\mathfrak{B}$ désigne une *relation*.

Si $b_1 =$  et qu'on exécute l'action $b_1 \leftarrow$ ④, on obtiendra $b_1 =$  : on a modifié l'expression permettant de déterminer à l'exécution la valeur du nombre de répétitions de la boucle n°1.

2.3.3 Représentations des scripts et des rétroactions

Scripts et boucles

Si $I \subset \mathfrak{B}$ [21], I suffit à définir un script dans notre situation, à la condition que la structure du script ne soit pas modifiée. En effet, normalement, le script de traçage et de dénombrement d'un TS est défini uniquement par les expressions de ses boucles. Nous confondrons, lorsqu'il n'y aura pas d'ambiguïté, un script S et l'ensemble des expressions de ses boucles : $S = \{expr | expr \in \mathfrak{B}\}$. On utilisera la même notation pour une partie de script, comme illustré en figure 2.14 (p. 165). Autrement dit, dans notre situation, la description d'un script peut se limiter, dans la plupart des cas, à la suite des expressions permettant de déterminer à l'exécution le nombre de répétitions de chacune des boucles du script traçant un TS.

Lorsque c'est possible, pour simplifier l'écriture des ensembles des expressions des boucles, les écritures des trois familles de boucles ($\mathbf{B_1} = \{b_1, b_6\}, \mathbf{B_2} = \{b_2, b_3, b_4, b_5, b_8\}$ et $\mathbf{B_7} = \{b_7\}$, voir 2.1.2, p. 133) seront représentées par la suite des expressions de chacune de ces familles, séparées par des « / ». Ainsi, « $\{expr_1/expr_2/expr_3\}$ » représente $\{b_1 = expr_1, b_6 = expr_1\} \cup \{b_2 = expr_2, b_3 = expr_2, b_4 = expr_2, b_5 = expr_2, b_8 = expr_2\} \cup \{b_7 = expr_3\}$. Cette écriture décrit l'état normal du script, et permet de comparer et catégoriser les scripts. Lorsque la représentation du nombre de répétitions d'une boucle est différente de celle des autres boucles de la même famille, l'identifiant de la boucle et cette représentation seront précisées. Par exemple si, $S = \{3/4/7\}$ et que les élèves modifient $b_8 \leftarrow 5$, le script résultant sera noté $\{3/4/7, b_8 = 5\}$.

Si besoin, le langage d'écriture des expressions sera précisé : θ pour le langage de description de l'algorithme (Snap !), a pour une expression algébrique, ρ pour une écriture rhétorique ou syncopée. Par exemple pour rendre compte que la représentation du nombre de répétitions de la boucle b_1 est l'expression « 4-1 », on notera $I_a = \{b_1 = 4-1\}$: $I_\theta = \{b_1 =$ $\}$ et $I_\rho = \{$« pour b_1 on fait moins un »$\}$.

On définira la fonction $\mathbf{b}$ comme étant la projection d'un ensemble d'expressions dans l'ensemble $\mathfrak{B}$: le résultat de cette fonction est l'ensemble des intitulés des boucles des expressions. Ainsi, si $S = \{b_1 = 3, b_2 = 4\}$, $\mathbf{b}(S) = \{b_1, b_2\}$. De même, si $\mathcal{A} = \{b_2 \leftarrow$ ⑤$, b_7 \leftarrow$  $\}$, $\mathbf{b}(\mathcal{A}) = \{b_2, b_7\}$.

21. Et que $\exists j \in \mathbb{N}^*, \forall i \in \mathbb{N}^*, i < j \implies b_i \in \mathbf{b}(I)$.

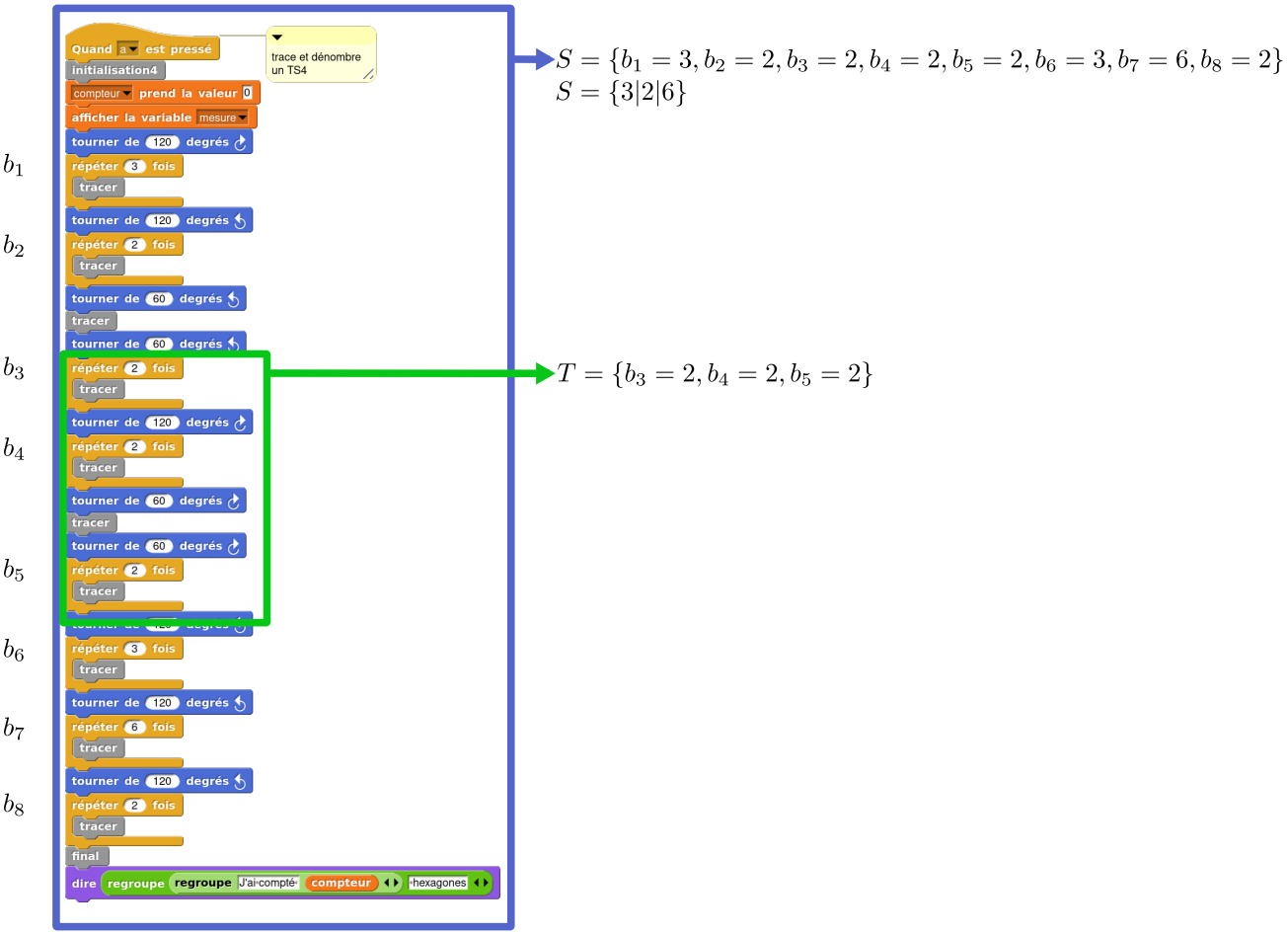


Figure 2.14 – Exemple de notation simplifiée d'un script S et d'une de ses parties T

Rétroaction d'un script

Pour un script $I \subset \mathfrak{B}$, « $I[]$ » représente l'*exécution* du script I. Pour une expression *expr*, $I[expr]$ représente l'exécution du script avec une valeur *expr* entrée par l'utilisateur, qui est censée être la valeur de la mesure du TS devant être tracé. Normalement, *expr* est un nombre entier formé de chiffres.

Le résultat de l'exécution simultanée (avec le pseudo-parallélisme de Snap !) de deux scripts I et J sera noté $(I \wedge J)[]$. Nous parlerons d'« exécution simultanée » ou d'« exécution parallèle » lorsqu'une partie au moins de deux scripts est exécutée en parallèle. Nous ne précisons pas davantage, l'exécution parallèle de deux scripts dans notre situation ne permettant pas aux élèves d'interpréter la rétroaction, hormis l'absence de validité du tracé (voir figure 2.15, p. 166).

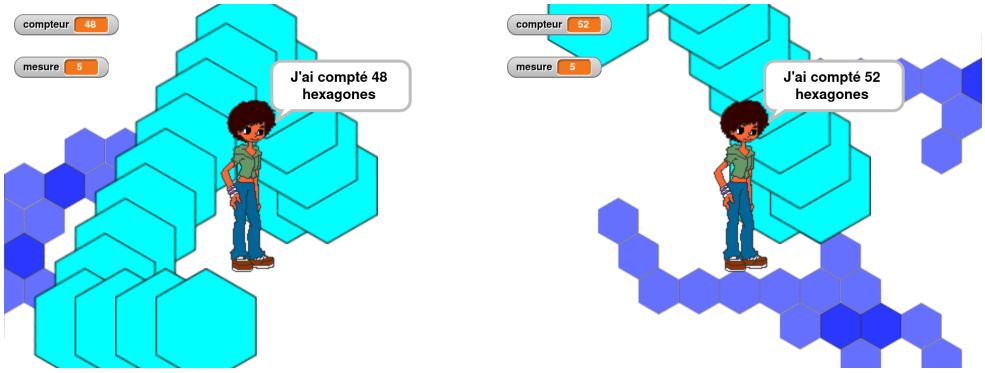


Figure 2.15 – Deux exemples d'exécution partiellement parallèle des scripts $TS(4)$ et$TS(5)$

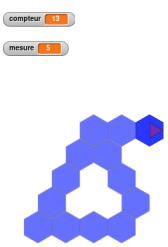
On écrira $I[] \rightsquigarrow (image, cpt, n)$ la rétroaction produite pour l'exécution d'un script I, où *image* est le résultat visuel du rendu, *cpt* le nombre d'hexagones dénombrés par le programme, et n la valeur de mesure lors de l'exécution. Ainsi, si $S = \{4/3/8\} = \{b_1 = 4, b_2 = 3, b_3 = 3, b_4 = 3, b_5 = 3, b_6 = 4, b_7 = 8, b_8 = 3\}$:

$$S[] \rightsquigarrow \left(\text{  } \text{} , 33, 5 \right)$$

Si $T = \{4/3/6\}$:

$$T[] \rightsquigarrow \left(\text{  } \text{} , 31, 5 \right)$$

En cas de rétroaction dynamique (c'est-à-dire de pauses dans l'exécution du script), on indiquera en indice la boucle en cours de traçage. Ainsi, avec le script S précédent :

$$S_4[] \rightsquigarrow \left(\text{  } \text{} , 13, 5 \right)$$

Script solution

Si p est un nombre entier, $TS(p) \subset \mathfrak{B}$ est l'ensemble des expressions des boucles b_i de l'algorithme de traçage et dénombrement qui est censé produire le TS de mesure p. Autrement dit, si p est un nombre, $TS(p)$ désigne une instance particulière de script traçant et dénombrant un TS : $TS(17)$ est l'instance du script permettant de tracer le TS de mesure 17.

Par convention on utilisera la lettre n pour représenter la généricité : $TS(n)$ désignera l'ensemble des expressions des boucles de l'algorithme de traçage et dénombrement produisant le TS pour n'importe quelle instance entrée par l'utilisateur, cette entrée étant référencée par n dans le script.

$TS(p)$ est donc la *solution* pour tracer et dénombrer un TS de mesure p : $TS(p)$ trace et dénombre TS(p). $TS(n)$ est le script permettant de tracer et dénombrer n'importe quelle instance de TS.

Ainsi, $TS(5) = \{4/3/8\} = \{b_1 = 4, b_2 = 3, b_3 = 3, b_4 = 3, b_5 = 3, b_6 = 4, b_7 = 8, b_8 = 3\}$ (figure 2.16a), et $TS(n) = \{n-1/n-2/2(n-1)\} = \{b_1 = n-1, b_2 = n-2, b_3 = n-2, b_4 = n-2, b_5 = n-2, b_6 = n-1, b_7 = 2n-2, b_8 = n-2\}$ (figure 2.16b, p. 168). Dans la figure 2.14 (p. 165), $S = TS(4)$: S est la solution pour tracer un TS4.

Du point de vue rétroaction, on a :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 33, 5 \right)$$

$$TS(n)[11] \rightsquigarrow \left(\text{  } \text{} , 87, 11 \right)$$

Script élève et but

Lorsque c'est nécessaire, le *but* (au sens de Vergnaud) des élèves sera précisé en exposant. Le but est ici assimilé à la mesure du TS que les élèves veulent tracer, ou n si leur objectif est de construire le script générique. Ainsi, si les élèves font des modifications sur le script leur permettant de tracer un TS5 et sur celui permettant de tracer un TS11, on pourra différencier ces deux scripts par S^5 et S^{11}. Cela implique de pouvoir identifier le but des élèves, ce point sera abordé plus loin.

Boucles valides ou erronées

Pour un script S, on définira $V^p(S) \subset \mathfrak{B}$ comme étant l'ensemble des *boucles valides* du script S ayant pour but p. $V^p(S) = S \cap TS(p)$. Les intitulés des boucles valides sont donc $\mathbf{b}(V^p(S))$.

(a) Script $TS(5)$ **(b)** Script $TS(n)$

Figure 2.16 – Deux exemples de scripts valides

De même, $E^p(S)$ est l'ensemble des *boucles erronées* du script S ayant pour but p. $E^p(S) = S \backslash TS(p)$.

$E^p_{\text{élève}}(S)$ pourra préciser l'ensemble des boucles erronées *identifié* par « *élève* » pour un script S. Potentiellement, $E^p_{\text{élève}}(S) \neq E^p(S)$.

2.3.4 Scripts, actions et modifications

Pour un ensemble d'actions $\mathcal{A} \subset \mathfrak{A}$ et un script (ou une partie de script) S, $S \odot \mathcal{A}$ est le script S auquel on a appliqué les modifications de $\mathcal{A}$. Si $S = \{b_1 = 3, b_2 = 2, b_3 = 2\}$ et $\mathcal{A} = \{b_1 \leftarrow 5\}$, $S \odot \mathcal{A} = \{b_1 = 5, b_2 = 2, b_3 = 2\}$. Cela correspond par exemple à l'étape ③ de la figure 2.17 (p.173) : $\forall b \in B, \texttt{modifier}(b)$.

Les ensembles S et $\mathcal{A}$ sont caractéristiques de l'activité de l'élève : pour le script S, il choisit de modifier une partie des boucles $B \subset S$ en leur appliquant les affectations $\mathcal{A}$, ce qui conduira au script $S \odot \mathcal{A}$, c'est-à-dire au script dont seules les boucles $\mathbf{b}(B)$ (ou, ce qui revient au même, $\mathbf{b}(\mathcal{A})$) ont été modifiées. Si nécessaire, une notation indicielle sera appliquée. Dans l'exemple précédent, seule la boucle b_1 est modifiée.

2.3.5 Exemple

Prenons un exemple pour illustrer ces conventions. Les élèves commencent par dupliquer le script (valide) permettant de tracer et dénombrer un TS5. Ils obtiennent ainsi le script $S = \{4/3/8\} = \{b_1 = 4, b_2 = 3, b_3 = 3, b_4 = 3, b_5 = 3, b_6 = 4, b_7 = 8, b_8 = 3\}$. Ils décident de modifier les boucles b_1 et b_2, donc $\mathbf{b}(B) = \{b_1, b_2\}$, en leur donnant une valeur de répétition de 5 et 4 respectivement : $\mathcal{A} = \{b_1 \leftarrow 5, b_2 \leftarrow 4\}$. En appliquant ces actions à S, ils obtiennent le script $S' = S \odot \mathcal{A} = \{b_1 = 5, b_2 = 4, b_3 = 3, b_4 = 3, b_5 = 3, b_6 = 4, b_7 = 8, b_8 = 3\}$. Dans ce cas, $V^{11}(S') = (S' \odot \mathcal{A}) \cap TS(11) = \varnothing$ et $E^{11}(S') = (S' \odot \mathcal{A}) \backslash TS(11) = S'$, car pour un TS de mesure 11, aucune des boucles du script modifié des élèves $(S \odot \mathcal{A})$ ne correspond au script permettant de tracer un TS(11). En revanche, $V^6(S') = (S \odot \mathcal{A}) \cap TS(6) = \{b_1 = 5, b2 = 4\}$ et $E^6(S') = (S \odot \mathcal{A}) \backslash TS(6) = \{b_3 = 3, b_4 = 3, b_5 = 3, b_6 = 4, b_7 = 8, b_8 = 3\}$: si les élèves veulent tracer un TS6, les boucles b_1 et b_2 sont les seules valides.

On illustre aussi par cet exemple la dépendance des boucles erronées en fonction du but fixé : E^6 indique les boucles erronées si le script visé est un TS6, tandis que ces boucles erronées seront E^{11} si les élèves veulent construire un TS(11).

Nous résumons ces conventions dans le tableau 2.1 (p. 170).

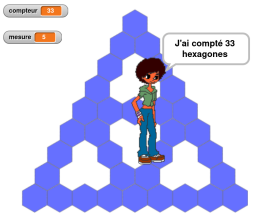
Convention	Signification	Exemple
$\{expr_1/expr_2/expr_3\}$	Représentations des boucles d'un script suivant les trois familles de boucles : $\{b_1 = expr_1, b_6 = expr_1\} \cup \{b_2 = expr_2, b_3 = expr_2, b_4 = expr_2, b_5 = expr_2, b_8 = expr_2\} \cup \{b_7 = expr_3\}$	$\{4/3/8\} = \{b_1 = 4, b_2 = 3, b_3 = 3, b_4 = 3, b_5 = 3, b_6 = 4, b_7 = 8, b_8 = 3\}$
$I[]$, $I \subset \mathfrak{B}$	*exécution* du script I	
$I[expr]$	exécution du script avec une valeur $expr$ entrée par l'utilisateur	
$(I \wedge J)[]$	exécution simultanée (avec le pseudo-parallélisme de Snap !) de deux scripts I et J	
$I[] \rightsquigarrow (image, cpt, n)$	Rétroaction produite pour l'exécution d'un script I, où $image$ est le résultat visuel du rendu, cpt le nombre d'hexagones dénombrés par le programme, et n la valeur de mesure lors de l'exécution.	si $S = \{4/3/8\}$, $S[] \rightsquigarrow$ ( , 33, 5)
$TS(p), p \in \mathbb{N}$	Script *solution* pour tracer et dénombrer un TS(p)	$TS(5) = \{4/3/8\} = \{b_1 = 4, b_2 = 3, b_3 = 3, b_4 = 3, b_5 = 3, b_6 = 4, b_7 = 8, b_8 = 3\}$
$TS(n)$	Script solution pour tracer et dénombrer n'importe quelle instance de TS	$TS(n) = n - 1/n - 2/2n - 2$
S^p	Script ayant pour but de tracer un TS(p)	
$V^p(S) \subset \mathfrak{B}$	ensemble des *boucles valides* du script S ayant pour but p	Si $S = \{b_1 = 3, b_2 = 2, b_3 = 1\}$, $V^4(S) = \{b_1 = 3, b_2 = 2\}$ et $V^5(S) = \varnothing$
$\mathbf{b}(V^p(S))$	Intitulés des boucles valides pour le script S ayant pour but p	Si $S = \{b_1 = 3, b_2 = 2, b_3 = 1\}$, $\mathbf{b}(V^4(S)) = \{b_1, b_2\}$
$E^p(S)$	Ensemble des *boucles erronées* du script S ayant pour but p	Si $S = \{b_1 = 3, b_2 = 2, b_3 = 1\}$, $\mathbf{b}(E^4(S)) = \{b_3\}$
$\mathcal{A} = \{b_i \leftarrow expr\}$	Ensemble d'actions de modifications effectuées sur les boucles d'un script	
$S \odot \mathcal{A}$	Script S auquel on a appliqué les modifications $\mathcal{A}$	$S = \{b_1 = 3, b_2 = 2, b_3 = 2\}$ et $\mathcal{A} = \{b_1 \leftarrow 5\}$, $S \odot \mathcal{A} = \{b_1 = 5, b_2 = 2, b_3 = 2\}$

Tableau 2.1 – Tableau récapitulatif des conventions utilisées

2.4 Schèmes, faits et rétroaction

Les analyses *a priori* et *a posteriori* sont faites dans le CAP. Cependant, comme la position et la construction des problèmes par les élèves se fait tout à la fois dans l'action — la modification des scripts, argumentée ou non, la prise en compte du résultat de leur modification — et dans l'argumentation — les échanges au sein des binômes justifiant, contredisant ou anticipant leurs actions ou leurs résultats —, nous avons besoin d'utiliser la complémentarité apportée par d'autres cadres théoriques. Nous mobiliserons ainsi Vergnaud et Peirce pour nous permettre d'analyser le lien entre activité et connaissance en mobilisant notamment le concept de théorème en acte (2.4.1, p. 171). Nous traiterons la question de la rétroaction et de son lien avec les questions d'information, en nous appuyant notamment sur Shannon (2.4.2, p. 176). Grâce à ces réflexions, nous définirons ce qu'est un fait dans le CAP, pour ce qui concerne notre situation — en différenciant différents types de faits (2.4.3, p.185). Cela nous permettra de définir, dans la section suivante, comment nous construirons et analyserons les données issues de l'expérimentation.

2.4.1 Schèmes et Théorèmes-en-acte

Activité et connaissance

Selon Vergnaud (2011, p. 37) :

> L'activité en situation est un moyen essentiel d'étudier la pensée et son évolution au cours du développement, ainsi que les différences entre individus.

De même, pour C.-S. Peirce (2000, p. 24), « l'idée d'une chose quelconque est l'idée de ses effets sensibles » : la pensée est une action et consiste en une relation [22]. De plus, la pensée en activité vise au repos, à la suppression du doute et à l'établissement d'une « croyance » d'où naitra une habitude, ou des règles d'actions.

> As it appeases the irritation of doubt, which is the motive for thinking, thought relaxes, and comes to rest for a moment when belief is reached. But, since belief is a rule for action, the application of which involves further doubt and further thought, at the same time that it is a stopping-place, it is also a new starting-place for thought. That is why I have permitted myself to call it thought at rest, although thought is essentially an action. The final upshot of thinking is the exercise of volition, and of this thought no longer forms a part ; but belief is only a stadium of mental action, an effect upon our nature due to thought, which will influence future thinking.(C. S. Peirce, Houser et Kloesel, 1992, p. 129)

Ces règles d'action sont alors ce qui distingue « les différentes espèces de croyance »(C.-S. Peirce, 2000, p. 22) permettant d'apaiser un même doute :

> Si les croyances ne diffèrent point sous ce rapport, si elles mettent fin au même doute en créant la même règle d'action, de simples différences dans la façon de les percevoir ne suffisent pas pour en faire des croyances différentes, pas plus que jouer un air avec différentes clefs n'est jouer des airs différents. On établit souvent des distinctions imaginaires entre des croyances qui ne diffèrent que par la façon dont elles sont exprimées. (ibid., p. 22)

Vergnaud et Chartier (1994, p. 7) reprennent d'ailleurs cette idée :

22. Peirce précise en outre « When I just said that thought is an action, and that it consists in a relation, although a person performs an action but not a relation, which can only be the result of an action, yet there was no inconsistency in what I said, but only a grammatical vagueness »(C. S. Peirce, Houser et Kloesel, 1992, p. 130).

> Mais l'étude des apprentissages [...] montre que, au-delà des régularités perceptives, la pensée construit des objets et des prédicats dont il serait aberrant de chercher les raisons dans la seule perception : ils répondent en fait à des questions issues du besoin d'agir, de prévoir, et d'intégrer dans des systèmes cohérents des phénomènes qui contredisent éventuellement l'intuition perceptive ou dont les relations ne sont pas directement observables.

Ces règles d'actions associées à la « croyance » pour un même doute, sont proches de ce que Vergnaud définit comme étant un schème :

> Définition 1 : le schème est une forme invariante d'organisation de l'activité et de la conduite pour une classe de situations déterminée. (Vergnaud, 2007, p. 17)

Ainsi, étudier les apprentissages des élèves, c'est identifier les schèmes (les croyances et les règles d'action) qui organisent l'activité de l'élève. Il faut ainsi s'intéresser aux « objets qu'il considère, leurs propriétés, leurs relations, leurs transformations, et l'effet de son action sur ces objets. » (Vergnaud et Chartier, 1994, p. 7). En effet, C.-S. Peirce (2000, p. 7) affirme :

> L'irritation produite par le doute nous pousse à faire des efforts pour atteindre l'état de croyance. Je nommerai cette série d'efforts recherche, tout en reconnaissant que parfois ce nom n'est pas absolument convenable pour ce qu'il veut désigner.

Cette « recherche » de Peirce est ce que nous nommons problématisation : l'articulation du doute et de la certitude, la recherche des conditions ou des nécessités du problème. Nous nous appuierons donc sur l'identification de l'organisation de l'activité de l'élève pour étudier comment ils posent et construisent le problème.

Actions et schèmes

Dans la phase de généralisation, modifier un script afin de l'adapter au but poursuivi revient à modifier l'ensemble des boucles b_i. Si l'on considère spécifiquement ces modifications, l'organisation de l'activité de l'élève pour un but particulier (passer d'une instance i à une instance j, généraliser le script) peut être représentée par la figure 2.17 (p. 173).

Déroulement de l'activité

Le déroulement attendu est le suivant. Les élèves commencent par choisir le but (le script à concevoir) ①. Ils choisissent ensuite leur sous-but ②, soit $\mathbf{B}$ l'ensemble des boucles qu'ils vont modifier avant toute autre action. On distinguera notamment [23] :

- Modification simultanée de l'ensemble des boucles : $\mathbf{B} = \mathbf{B_a} = \{b_i, i \in [\![1, 8]\!]\}$.
- Modification simultanée de l'ensemble des boucles, excepté la boucle b_7 (le « grand côté », qui est attendu comme étant problématique) : $\mathbf{B} = \mathbf{B_e} = \{b_i, i \in [\![1, 8]\!] \wedge i \neq 7\}$.
- Modification d'une partie des boucles (généralement successives) : $\mathbf{B} \subset \mathbf{B_a}$.
- Modification de la seule boucle b_7 : $\mathbf{B} = \{b_7\}$.

Une fois ces boucles modifiées ③, le script est testé ④. Suivant la rétroaction ⑤, soit le script est considéré comme valide ⑥, soit comme invalide ⑦. Dans le cas où le script est vu comme valide (relativement aux modifications apportées), soit les élèves poursuivent les modifications sur le même but ② s'il reste des boucles à modifier—, soit ils passent à un autre but ① si le but en cours est atteint Dans le cas où la rétroaction permet de considérer le script comme invalide,

23. voir 2.3, p. 162.

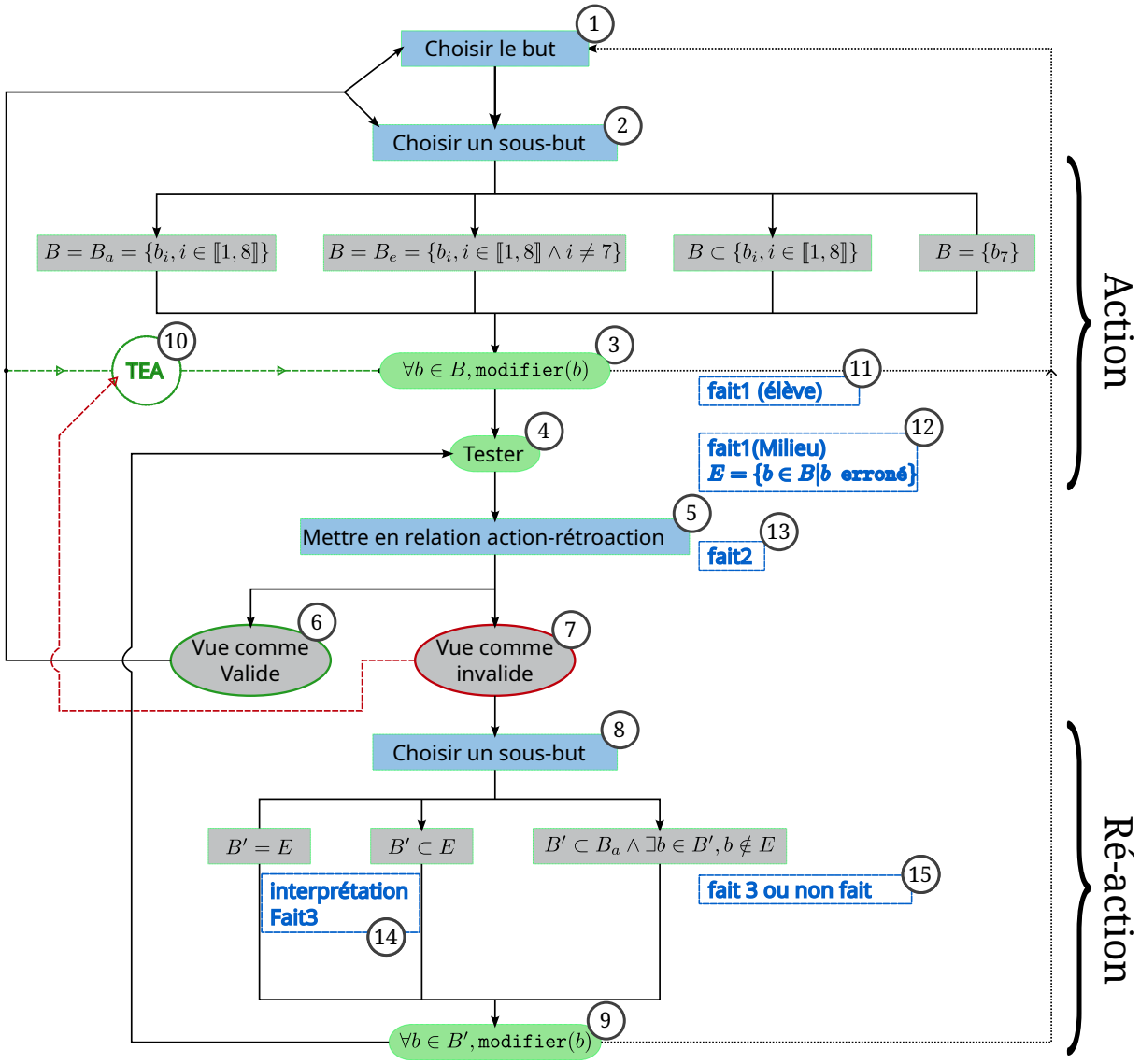


Figure 2.17 – Organisation de l'activité dans la phase de généralisation

alors les élèves devraient définir un nouveau sous-but ⑧ afin de le corriger, donc déterminer un ensemble de boucles $\mathbf{B'}$ à modifier ⑨ puis à tester ④. Idéalement, cet ensemble est confondu avec l'ensemble $\mathbf{E}(\mathbf{p})$ des boucles effectivement erronées pour l'instance p ($\mathbf{B'} = \mathbf{E}(\mathbf{p})$), mais il peut aussi en constituer un sous-ensemble ($\mathbf{B'} \subset \mathbf{E}(\mathbf{p})$), voire contenir des boucles non erronées ($\exists b \in \mathbf{B'} | b \notin \mathbf{E}(\mathbf{p})$ ou $\mathbf{B'} \cap \mathbf{V}(\mathbf{p}) \neq \varnothing$). Notons aussi que les élèves sont susceptibles d'effectuer les modifications (③ et ⑨) sans les tester, en passant à un autre but. Enfin, il est aussi possible que les élèves fassent un test mais ne manifestent pas, par leurs actions, de prise en compte des résultats de ces tests : la relation action-rétroaction n'est pas visible. Ces points seront discutés ultérieurement.

Schèmes et Théorèmes-en-actes

La schématisation de la figure 2.17 (p. 173) est une représentation des règles d'actions, de prise d'information et de contrôle possiblement mises en œuvre par des élèves. Ces règles d'action sont une des composantes du schème selon Vergnaud (2007, p. 17) vu comme « une forme invariante d'organisation de l'activité et de la conduite pour une classe de situations déterminée ». Comme l'algorithme, « le schème s'adresse à une classe de situation »(Vergnaud, 2011), mais le schème ne se réduit pas à un algorithme, ne serait-ce que parce que le critère d'effectivité n'est pas garanti : les algorithmes sont des schèmes mais tous les schèmes ne sont pas des algorithmes.

> Un schème n'est pas en général un algorithme. Certaines formes d'organisation de l'activité mathématique sont effectivement des algorithmes : ils aboutissent, en un nombre fini de pas (effectivité), au traitement de toute situation appartenant à la classe visée. Les algorithmes sont des schèmes, mais tous les schèmes ne sont pas des algorithmes ; on peut même ajouter que certains algorithmes perdent au cours de l'apprentissage ou de l'expérience certaines de leurs caractéristiques, notamment leur propriété d'effectivité : des erreurs et des raccourcis peuvent les priver de la propriété d'aboutir à coup sûr. L'incertitude reste ainsi une propriété des schèmes. (Vergnaud, 2007)

Ainsi, une des caractéristiques essentielles des schèmes est de ne pas être figés, mais plutôt de s'adapter : « Si la connaissance est adaptation, il faut se convaincre que ce qui s'adapte ce sont des schèmes, c'est-à-dire des formes d'organisation de l'activité, et qu'ils s'adaptent à des situations » (Merri, 2007). Ainsi, des élèves peuvent abandonner un algorithme appris en classe au profit d'un schème personnel qui leur paraît mieux adapté à la situation, même si cela n'est pas le cas. « Souvent un sujet individuel dispose de plusieurs schèmes alternatifs entre lesquels il peut choisir en fonction des valeurs des variables de situation et notamment des valeurs numériques. Les recherches montrent cependant que certains individus disposent de toute une panoplie, alors que d'autres n'ont qu'une corde à leur arc, et d'autres aucune » (Vergnaud, 2011, p. 43). Certains schèmes sont fugaces et vite abandonnés, d'autres se trouvent « renforcés à ce point qu'ils chassent les autres du répertoire des possibles »(ibid., p. 43).

Un schème, toujours selon Vergnaud (ibid., p. 43), est formé de quatre composantes :

- un but, des sous-buts et anticipations, déjà évoqués ci-dessus ;
- des règles d'actions, de prise d'information et de contrôle, abordés dans la figure 2.17 (p. 173) et sur lesquels nous reviendrons ;
- des invariants opératoires : concepts-en-acte (CEA) et théorèmes-en-acte (TEA) ;
- et des possibilités d'inférence, puisque « l'activité en situation n'est jamais automatique, mais au contraire régulée par les adaptations locales, les contrôles, les ajustements progressifs. » (Vergnaud, 2007, p. 19)

Dans l'application des règles d'action présentées, ce qui fera la différence entre des élèves, et donc entre schèmes, — outre des « mise en suspens » de l'action lorsque les élèves, en ③ ou ⑨, choisissent de ne pas effectuer de tests mais plutôt de changer de but — se jouera au niveau des modifications des boucles b_i. Ces modifications sont inférées des invariants opératoires « qui permettent au sujet de prélever l'information pertinente et d'en inférer règles d'action et anticipations » (Vergnaud, 1991, p. 83). Ces invariants opératoires forment la partie du schème directement liée aux connaissances. Ils sont constitués de concepts-en-acte ou théorèmes-en-acte, qu'il nous appartiendra donc de tenter d'identifier : ce sont notamment ces TEA ⑩ « qui permettent de formuler ce qui fait la différence entre un moment de développement et un autre, ou encore entre un individu plus expert et un individu moins expert » (Vergnaud, 2007).

Pour Vergnaud (2011, p. 44) :

> Par définition un concept-en-acte est un concept tenu pour pertinent dans l'action en situation ; et un théorème-en-acte est une proposition tenue pour vraie. Ainsi certains concepts-en-acte ont un statut d'objet, d'autres un statut de prédicat à une place, d'autres encore de prédicat à plusieurs places. En outre, les prédicats peuvent devenir des objets : c'est le cas par exemple des couleurs (le ciel est bleu et le bleu du ciel) et de beaucoup de propriétés mathématiques (la cathédrale est symétrique et la symétrie comme objet d'étude en géométrie). Ils entretiennent à leur tour des relations avec d'autres objets.

Le TEA a une valeur de vérité, tandis que le CEA a une valeur de pertinence (Vergnaud, 2007). Le CEA se rapproche ainsi de ce que C.-S. Peirce (2000, p. 6) nomme une « qualité », qui, « prise en elle-même n'est jamais connue par l'observation. On peut voir qu'un objet est bleu ou vert, mais la qualité bleu ou la qualité vert ne sont point choses qu'on voit, ce sont les produits d'une opération de logique. » Pour Peirce comme pour Vergnaud, « l'idée d'une chose quelconque est l'idée de ses effets sensibles »(ibid., p. 24) et « la réalité, comme toute autre qualité, consiste dans les effets perceptibles particuliers produits par les choses qui la possèdent » (ibid., p. 29). Ainsi, on construit de nouveaux concepts, de nouveaux schèmes, de nouvelles « croyances », par l'activité ; de même que l'on peut en partie identifier ces schèmes ou croyances en analysant l'activité d'un individu dans des situations données. Par « croyance », C.-S. Peirce (ibid., p. 22) entend poétiquement « la demi-cadence qui clôt une phrase musicale dans la symphonie de notre vie intellectuelle » : la croyance est une sorte de stabilité (momentanée ou non) de l'esprit, qui « apaise l'irritation donnée par le doute » et « implique l'établissement dans notre esprit d'une règle de conduite, ou, pour parler plus brièvement, d'une habitude. » Cette « habitude particulière d'esprit » nous parait fort semblable au TEA puisqu'elle permet de choisir ou déterminer des inférences, et « peut se formuler en une proposition dont la vérité dépend de la validité des inférences déterminées par cette habitude d'esprit » (ibid.).

Ainsi, TEA et CEA sont indissociables : le CEA permet de choisir ou de former un TEA pertinent, et les TEA permettent de donner un « contenu » au CEA qui sans cela seraient des prédicats ou des objets n'ayant pas d'influence sur la conduite de l'activité.

> Métaphoriquement on peut dire que les concepts-en-acte sont les briques avec lesquelles les théorèmes-en-acte sont fabriqués, et que la seule raison d'existence des concepts-en-acte est justement de permettre la formation de théorèmes-en-acte (propositions tenues pour vraies), à partir desquels sont rendus possibles l'organisation de l'activité et les inférences. Réciproquement, les théorèmes sont constitutifs des concepts puisque, sans propositions tenues pour vraies, les concepts seraient vides de contenu. (Vergnaud, 2007)

Les possibilités d'action en situation « résultent principalement des théorèmes-en-acte spécifiques du domaine et de la classe de situations à laquelle s'adresse le schème considéré, et de théorèmes-en-acte plus généraux, qui couvrent plusieurs domaines d'activité, et qui sont souvent formalisés dans des termes logiques comme la déduction, l'induction, l'abduction »(Vergnaud, 2011) : ce sont ces TEA que nous chercherons à identifier.

Action et ré-action

Les TEA pour ③ (modification d'un ensemble de boucles pour construire un nouveau script) ne sont pas de même nature que pour ⑨ (ajustement, correction des valeurs des boucles) car les actions en jeu et les buts poursuivis sont différents.

En ③, le but est de construire $\{b_i^p, i \in [\![1,8]\!]\}$, pour p fixé ou quelconque ; il s'agit d'exprimer les valeurs du nombre de répétitions des boucles du script permettant de tracer un TS(p) (donc les b_i^p), en fonction de ce qui est déjà connu. L'expression permettant de dénoter b_i^p pourra mettre en relation p, une autre boucle du script en cours, et éventuellement une ou plusieurs boucles d'un autre script déjà construit. On peut ainsi s'attendre à un TEA mettant en relation une boucle b_i pour une instance p et la même boucle d'une autre instance m. Typiquement, on pourra observer $b_i^p = b_i^{p-1} + 1$: pour passer d'une instance $(p-1)$ à la suivante (p), on ajoute un à la valeur des boucles (ce qui est valide pour $\mathbf{B_1}$ et $\mathbf{B_2}$, mais pas pour $\mathbf{B_7}$ ($b_7^p = b_7^{p-1} + 2$)). Ce sera le cas par exemple lors du passage de l'instance 5 à l'instance 6 si les élèves passent de $b_1 = 4$ à $b_1 = 5$. Une autre erreur attendue serait de considérer que lorsqu'on multiplie la valeur l'instance, on multiplie aussi la valeur des boucles : $b_i^{a\times m} = a \times b_i^m$. Par exemple pour passer de l'instance 11 ($b_1 = 10$) à l'instance 22 ($22 = 2 \times 11$), les élèves pourraient doubler le nombre de répétitions de b_1 en passant de $b_1 = 10$ à $b_1 = 20$.

Une autre possibilité est de mettre en relation les boucles d'une même instance : par exemple on a toujours $b_2^p = b_1^p - 1$ ou $b_3^p = b_2^p$, ou encore $b_7^p = 2b_6^p$.

Enfin, le TEA peut exprimer la relation d'une boucle avec la valeur de la *mesure* : $b_1^p = p - 1$, ou $b_7^p = 2p - 2$. Un exemple erroné pourrait être $b_1^5 = 5$ et $b_1^{11} = 11$, ce qui traduirait un TEA de la forme $b_1^p = p$.

Ces trois types de relations sont un indice du type de traitement de la situation opéré par les élèves : intra-objectal pour la relation à l'intérieur d'un même script, inter-objectal pour la relation entre deux instances, et trans-objectale pour la relation avec la valeur de la *mesure* (Piaget et García, 1983). Ici, nous nous intéressons au passage au traitement trans-objectal.

En ⑨ il faut *ajuster* la modification pour adapter le résultat obtenu au résultat attendu, donc en prenant en compte $E(p)$, l'ensemble des boucles erronées. Cependant, les élèves n'identifient pas forcément correctement $E(p)$, ou peuvent choisir délibérément de ne traiter qu'une partie de $E(p)$. Ils agissent en fait sur l'ensemble B', sous-ensemble des boucles *considérées comme erronées* et devant être modifiées.

Nous nous intéresserons ci-dessous plus spécifiquement au TEA⑩ induisant l'action en ③, que nous qualifions d'*action première* : c'est une proposition tenue pour vraie permettant, pour les élèves, de généraliser l'algorithme de tracé d'un (ou des) TS, et donc manifestant des CEA liés à la généralisation. L'interprétation des résultats de cette action — notamment par l'interprétation de la rétroaction — influe sur ce TEA⑩, sans se limiter à une simple validation ou invalidation. L'action en ⑨ est pour nous une *action seconde* : il s'agit d'une *ré-action* mettant en lien action première et effet. Les effets de cette réaction ont une influence moins directe, moins sensible, sur le TEA ⑩, puisque plus spécifique à $E(p)$: il y aura donc production de faits d'une autre nature.

L'identification du TEA en vigueur chez les élèves pour les actions premières consiste en la formulation de a_1 pour un but identifié. Cette formulation se base sur une analyse des actions liées au TEA, mais aussi sur celles d'actions précédentes ou suivantes, y compris pour des buts et sous-buts différents.

2.4.2 Rétroaction et milieu

Comme nous l'avons vu, la conduite de l'activité modifie les schèmes des élèves, mais cette modification ne s'appuie pas sur le seul fait d'agir et de réagir : les prises d'information « nécessaires à la poursuite de l'activité »(Vergnaud, 2011, p. 43) et les contrôles de l'activité « qui permettent au sujet de s'assurer qu'il a bien fait ce qu'il pensait faire et qu'il est toujours sur la voie choisie »(ibid., p. 43) sont des constituants essentiels du schème. Dans notre situation, ceux-ci

se basent en grande partie sur la rétroaction (ou feed-back) produite par le milieu [24]. Comme l'affirme Bosc-Miné (2014, p. 316), « les feed-back ont une importance considérable dans les apprentissages, car les informations reçues à la suite d'une action indiquent à l'individu si celle-ci est en adéquation avec ce qu'il souhaitait réaliser. »

Qu'est-ce qu'un feed-back ?

Bosc-Miné, MCF en psychologie cognitive dans le laboratoire Paragraphe [25], dont G.Vergnaud fut membre, s'intéresse notamment à « la modélisation des stratégies de résolution de problème par l'identification de règles d'actions et de règles d'inférences. » [26] et plus particulièrement au traitement des feed-back. Elle précise que « les feed-back peuvent être définis comme étant les informations issues de l'environnement engendrées par les diverses conséquences de l'activité de l'individu (Balzer, Doherty & O'Connor, 1989 ; George, 1990 ; Mason & Bruning, 2001) et peuvent être codés sous forme de réussite et d'échec ou sous forme d'écart au but (George, 1990) » (ibid., p. 316). Hattie et Timperley (2007, p. 82) précisent que « to take on this instructional purpose, feedback needs to provide information specifically relating to the task or process of learning that fills a gap between what is understood and what is aimed to be understood (Sadler, 1989), and it can do this in a number of different ways ». Les informations produites et communiquées ne devraient pas se limiter à une validation ou invalidation d'une tâche, elles devraient aussi permettre à l'apprenant de se situer par rapport à un but, une norme, un apprentissage... ; de déterminer l'écart avec ce but, cette norme, cet apprentissage... ; et de combler cet écart. La tâche doit donc pouvoir être répétée — avec une nouvelle rétroaction — afin de pouvoir réduire cet écart, et si cet écart diminue, on peut envisager la possibilité d'un apprentissage. Ainsi, pour contribuer aux apprentissages, le feed-back ne peut pas être une simple information transmise. C'est un processus impliquant l'apprenant dans l'interprétation de l'information fournie, cette interprétation dépassant le constat de l'écart pour permettre d'envisager des améliorations possibles :

> Selon Boud et Molloy (2013), le feed-back est un processus par lequel l'apprenant obtient des informations sur son travail afin d'apprécier les similitudes et les différences entre les normes correspondant à cette tâche et les qualités de son propre travail afin de générer des travaux de meilleure qualité. Cette définition met en avant trois points essentiels : 1) le fait que le feed-back soit considéré comme un processus et non pas simplement comme une information reçue, 2) le fait que le rôle de l'apprenant soit mis en avant dans la recherche d'informations, ce qui laisse entendre que la réception de feed-back n'est pas passive et 3) le fait que le feed-back ne soit pas seulement une information sur une performance et une indication d'écart au but mais peut être utile aussi pour s'améliorer dans des tâches ultérieures semblables ou plus complexes. (Bosc-Miné, 2014, p. 317)

Si un feed-back, dans le domaine de l'apprentissage, peut être « une information communiquée à un apprenant par un enseignant »(ibid., p. 316), Hattie et Timperley (2007, p. 81) précisent que « feedback is conceptualized as information provided by an agent (e.g., teacher, peer, book, parent, self, experience) regarding aspects of one's performance or understanding. » L'agent, l'élément de la situation qui produit la rétroaction, peut ainsi être de différentes natures. Dans notre cas, l'agent est l'algorithme exécuté dans l'EPGB, qui produit une restitution graphique de son propre fonctionnement, qui « donne à voir » son fonctionnement.

24. Au sens de Brousseau, *antagoniste à l'élève*.
25. EA 349 Universités Paris 8 et CY Cergy Paris Université.
26. *Le Site de l'équipe Compréhension, Raisonnement et Acquisition de Connaissances - Christelle Bosc-Miné* 2023

Nous nous intéressons ici aux rétroactions produites par le dispositif, en ce qu'elles sont fondamentales pour la construction du problème et puisque nous y avons accès pour tous les groupes suivis. Les rétroactions issues de l'enseignante – normalement exceptionnelles — ou des élèves entre eux, seront abordés lors de l'analyse des groupes dont nous disposons des interactions.

Dans son article, Bosc-Miné (2014) propose une revue de question afin proposer une présentation assez complète des types d'informations véhiculés par les feed-back, ainsi que des caractéristiques de ces rétroactions. Dans la figure 2.18 (p. 178) nous complétons le diagramme proposé par Bosc-Miné en précisant les éléments concernés par notre situation.

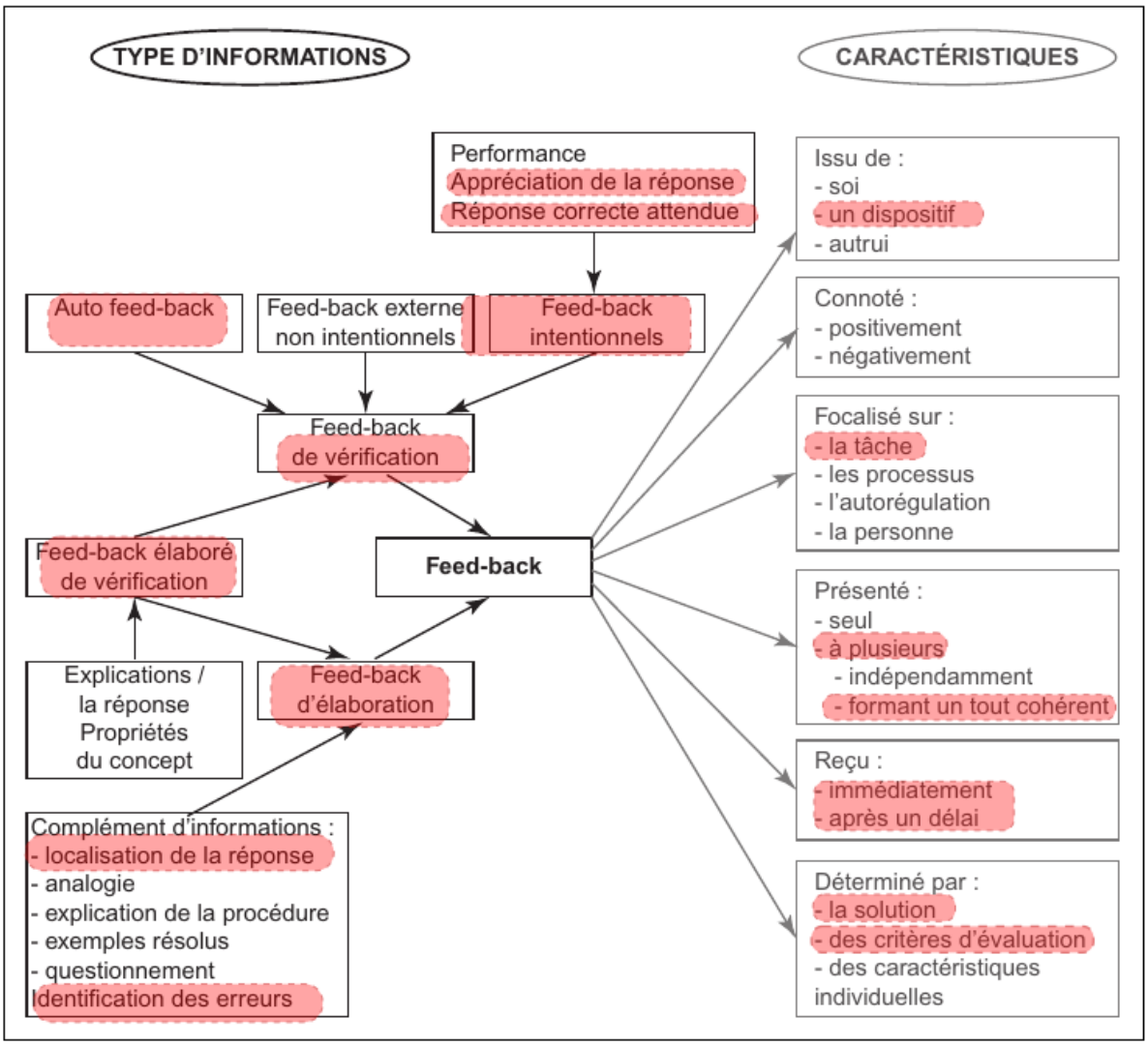


Figure 1. Types d'informations et caractéristiques des feed-back. La flèche noire signifie « est une sorte de ». La flèche grise signifie « a pour propriété d'être ».
Figure 1. Types of information and characteristics of feedback. The black arrow indicate "is a kind of". The grey arrow indicate "has the property of being".

Figure 2.18 – « Types d'informations et caractéristiques des feedbacks » (Bosc-Miné, 2014) Les éléments surlignés par nous correspondent aux caractéristiques de la rétroaction dans la situation *TS*

Caractéristiques du feed-back pour la situation TS

Du point de vue des caractéristiques, comme précisé ci-dessus, la rétroaction est produite essentiellement par « un dispositif ». Elle n'a pas de connotation particulière : il ne s'agit pas ici de féliciter, d'encourager ou de blâmer. L'information présentée est neutre et objective.

Concernant les niveaux de focalisation, présentés par Hattie et Timperley (2007), le feed-back produit est ici focalisé sur la tâche et non sur les processus, l'autorégulation ou la personne. En effet, il s'agit de donner des indications pertinentes sur la validité ou non du tracé, mais aussi des informations complémentaires sur le type d'erreurs et leurs localisations. Cela permet aux élèves de se repérer dans la tâche, mais aussi d'adapter leurs schèmes mobilisés, par la fourniture de données entrant éventuellement en tension avec les TEA à l'œuvre. Ces informations complémentaires tendraient à améliorer l'efficacité des feed-back focalisés sur la tâche (Bosc-Miné, 2014, p. 329). En revanche, il n'y a pas d'information sur la façon de résoudre la tâche ou de proposer des changements de stratégies par exemple. On ne s'appuie pas non plus sur l'autorégulation, « qui concerne la façon dont les individus, contrôlent, dirigent et régulent leurs actions vers l'objectif d'apprentissage » (ibid., p. 329), ni sur les aspects personnels (encouragements ou critiques par exemple).

Rétroaction dynamique et différée D'autre part, le milieu avec lequel l'élève est en interaction permet de produire plusieurs rétroactions simultanées et cohérentes. La validité du script peut-être déterminée par la validité du tracé — sans trou ni chevauchement —, complétée par la valeur du nombre d'hexagones dénombrés mis en relation avec la valeur de la mesure du TS affichée, si cette valeur est comparée à des valeurs connues, trouvées en séance 1 notamment. Ces rétroactions sont aussi multiples dans le temps, les élèves pouvant provoquer le feed-back à tout moment. Ainsi, le feed-back est déterminé par la solution, par des critères de validation, et — dans une faible mesure — par des « caractéristiques individuelles », ici le fait pour les élèves de pouvoir déterminer le moment d'apparition de la rétroaction. Celle-ci est alors reçue à la fois immédiatement et après un (léger) délai : le tracé, dynamique mais non instantané, permet de localiser les instructions en cours et leurs effets — que ce soit en utilisant ou non le mode pas-à-pas ou ralenti —, et une fois ce tracé terminé, le nombre d'hexagones dénombrés est affiché. Le tracé final ainsi que le nombre d'hexagones restent visibles, mais l'information dynamique peut être perdue.

Pour illustrer ces différents aspects, prenons l'exemple d'un script construit sous l'en-tête de script correspondant à la tâche prescrite « Compléter pour un TS6 ». La figure 2.19 (p. 181) montre divers résultats possibles du tracé suivant les erreurs. La figure 2.19a (p. 181) montre un tracé de TS6 valide, pour lequel :

$$\begin{cases} b_1 = b_6 = 5 \\ b_2 = b_3 = b_4 = b_5 = b_8 = 4 \\ b_7 = 10 \end{cases}$$

La figure 2.19b (p. 181) montre un tracé de TS5 valide, pour un TS6 demandé, pour lequel :

$$\begin{cases} b_1 = b_6 = 4 \\ b_2 = b_3 = b_4 = b_5 = b_8 = 3 \\ b_7 = 8 \end{cases}$$

Ces deux motifs ont été construits et dénombrés en séance 1. Dans ces deux cas, le tracé est sans trou ni chevauchement et correspond aux critères de validité établis en séance 1. Si pour 2.19b (p. 181), les élèves s'étaient fixés comme but de tracer un TS6, alors le résultat est erroné, et cette erreur est rendue visible de deux façons simultanées : visuellement, par le nombre d'hexagones des triangles de base que l'on peut aisément dénombrer, et numériquement par l'affichage du nombre d'hexagones (33) qui ne correspond pas à un TS6 (42).

Les figures 2.19c et 2.19d (p. 181) montrent le résultat final d'un $TS(5)$ lorsque $b_1 = 5$ et lorsque $b_8 = 4$ respectivement : les boucles tracent ainsi un hexagone de trop. On peut remarquer que dans ce cas, en ne prenant en compte que le feed-back différé, celui-ci permet bien de déterminer l'invalidité du tracé avec un chevauchement et le nombre d'hexagones dénombré à 34 au lieu de 33. Cependant, l'information fournie n'est pas suffisante pour pouvoir déterminer si l'erreur provient de b_1 ou de b_8, le résultat final étant identique à une translation près [27]. Il faut utiliser le feed-back dynamique (figures 2.19e et 2.19f, p. 181) pour pouvoir localiser la source de l'erreur. Les mêmes erreurs pour un $TS(6)$ produisent deux résultats légèrement différents (figure 2.19g et 2.19h, p. 181) mais sans pour autant permettre la localisation certaine de l'erreur.

Type d'informations du feed-back pour la situation TS

On a vu avec l'exemple précédent que la rétroaction portait deux types d'information [28] différents :

— des informations permettant la validation, la vérification ;

— des informations permettant aux élèves de prélever des indices pour pouvoir progresser.

Bosc-Miné définit ces deux types d'information comme caractéristiques des feed-back de vérification et d'élaboration :

> La vérification est une information concernant la justesse ou l'inexactitude de la réponse et l'élaboration concerne des indices permettant de guider l'individu vers la réponse correcte (Kulhavy & Stock, 1989) ou vers l'amélioration des connaissances. (Bosc-Miné, 2014, p. 318)

Nous considérons ici les feed-back intentionnels, externes, qui sont produit par le milieu. Comme indiqué plus haut, les feed-back informels, non « explicitement fournis par l'environnement »(ibid., p. 318), — c'est-à-dire issus d'échanges entre élèves ou avec l'enseignant, ou encore générés par l'élève lors de l'activité (auto feed-back) — ne seront pas évoqués dans cette partie. Nous soulignerons néanmoins que selon Hattie et Timperley (2007, p. 94) :

> Less effective learners have minimal self-regulation strategies, and they depend much more on external factors (such as the teacher or the task) for feedback. They rarely seek or incorporate feedback in ways that will enhance their future learning or self-regulation strategies.

Ce qui conduit Bosc-Miné (2014, p. 319) à affirmer que, pour favoriser ce contrôle interne, il faut « créer un environnement d'apprentissage dans lequel les élèves développent l'autorégulation et les compétences de détection d'erreurs (Hattie, Biggs, & Purdie, 1996) et leur présenter les critères de réussite, exposer les normes et des modèles attendus de la tâche, les familiariser avec

27. Notons que les élèves ne peuvent comparer le résultat avec un résultat précédent qu'en utilisant leur mémoire : il n'y a pas trace des résultats antérieurs (ce qui est une des limites de la situation).

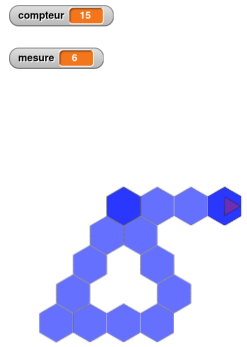
28. Nous reviendrons sur ce terme.

(a) Tracé TS6 valide (compteur=42)

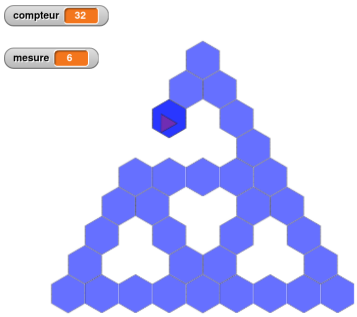
(b) Tracé TS5 valide (compteur=33)

(c) Tracé TS5, $b_1 + 1$ (compteur=34)

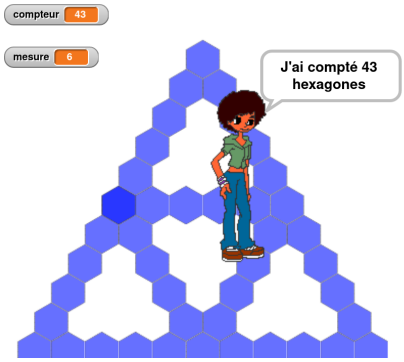
(d) Tracé TS5, $b_8 + 1$, (compteur=34)



(e) Tracé TS5 en cours, $b_1 + 1$



(f) Tracé TS5 en cours, $b_8 + 1$



(g) Tracé TS6, $b_1 + 1$, (compteur=43)

(h) Tracé TS6, $b_8 + 1$, (compteur=43)

Figure 2.19 – Exemples de rétroaction (en cours d'exécution ou finale) pour un TS6 demandé (mesure=6)

les objectifs (Boud & Molloy, 2013). » La situation que nous proposons nous semble correspondre à ces prescriptions : les élèves se sont familiarisés avec les figures afin d'identifier les constructions valides ou erronées, et ils sont confrontés à la recherche et la localisation des erreurs dans leurs scripts. Ces feed-back internes sont susceptibles d'être mis en tension avec les feed-back externes.

Concernant les rétroactions produites par le milieu, les deux types d'information (vérification et élaboration) sont véhiculées, ce que Bosc-Miné nomme un « feed-back élaboré de vérification ». Plus précisément, il s'agit ici d'un feed-back « topic contingent » ou « complément d'informations », qui fournit une information complémentaire à la validation permettant d'identifier l'erreur et sa localisation.

La production d'un feed-back intentionnel externe ne garantit pas que l'élève le prenne en compte ou le comprenne. Cependant, les feed-back élaborés semblent renforcer l'efficacité des feed-back de vérification, et notamment pour les élèves disposant de peu de connaissances dans le domaine concerné par le problème (Bosc-Miné, 2014, p. 323, 324), ce qui est le cas des élèves lors de l'expérimentation.

Pour résumer, la rétroaction fournie à l'élève est un *feed-back intentionnel, externe, de vérification et d'élaboration*, aidant à la localisation et l'identification des erreurs, déterminé par la solution et les critères de validité. Ce feed-back s'appuie sur un dispositif, est non connoté et focalisé sur la tâche. Il est en outre multiple et présenté dynamiquement de façon immédiate et statiquement de façon légèrement différée, le moment du feed-back étant décidé par l'élève. La rétroaction produite, permettant une prise d'information et un contrôle des élèves, est ici un élément essentiel dans le processus d'adaptation des schèmes et de construction du concept de paramètre, puisqu'elle permet des « prises d'information nécessaires à la poursuite de l'activité, et des contrôles qui permettent au sujet de s'assurer qu'il a bien fait ce qu'il pensait faire et qu'il est toujours sur la voie choisie » (Vergnaud, 2011, p. 43). Ces contrôles internes, mis en tension avec les feed-back externes, sont une forme de « surveillance de soi » par soi qui caractérise la problématisation (Fabre, 2017, p. 25). Fabre, p. 25 rappelait dans le même ouvrage que pour Bachelard, « Penser, c'est précisément placer l'objet devant le sujet divisé » il ne s'agit pas seulement d'exécuter mécaniquement une opération, mais aussi de « surveiller la procédure utilisée et contrôler la vraisemblance du résultat ». Ainsi, le milieu antagoniste à l'élève mis en œuvre dans notre situation, produisant des rétroactions permettant à l'élève de contrôler son activité et sa pensée, est susceptible d'amener l'élève à construire le problème, à problématiser.

Mais, si le système de captation automatisé mis en œuvre permet d'identifier la rétroaction produite pour une action des élèves, une question importante (et évoquée plus haut) est de savoir comment déterminer si cette rétroaction est prise en compte ou non par les élèves — donc s'il y a génération d'auto feed-back — et ce que cela implique dans la construction du problème.

Rétroaction et information : Shannon et la théorie mathématique de la communication

La rétroaction n'est pas indépendante de l'interprétation qui en est faite. Les informations fournies par l'environnement sont représentées ou codées, pour être transmises aux élèves, à charge pour ceux-ci d'interpréter le message transmis en faisant le lien entre le codage, leurs actions et leurs buts. Ainsi, dans notre situation, l'artefact informatique, produit une *donnée** permettant d'obtenir une *information* quant à la validité du script et la localisation éventuelle

de l'erreur, cette information étant éventuellement prise en compte (donc *factualisée*) par les élèves. Ce fait, construit, est un des éléments du registre empirique, et il viendra se mettre en tension — ou non — avec les autres faits et nécessités construits par les élèves pour le problème qu'ils posent (voir 2.4.3, p. 185).

Nous ajoutons un astérisque au mot « donnée » afin de le différencier des « données du problème », qui dans le Cadre de l'Apprentissage par Problématisation, sont les éléments du registre empirique construits ou mobilisés par les élèves et qui sont mis en tension avec les conditions ou nécessité du problème. La donnée* est ici une représentation de l'information, indépendamment de toute autre chose [29]. La validité du script est une information, là encore indépendamment de toute autre chose. L'information « le script est valide » est (en partie) codée, représentée, par la donnée* « le tracé est sans trou ni chevauchement », et c'est la mise en relation de l'information et de la donnée* qui va constituer un fait pour les élèves leur permettant d'avancer dans le problème. La donnée* de la théorie de l'information est une représentation effective, tandis que la donnée dans le CAP est une construction ou une mobilisation effective.

Comme le dit Fabre (2019, p. 31) :

> Les faits ne sont pas des choses comme les tables, les chaises ou les arbres, ils concernent des relations, des rapports entre les choses, entre les choses et nous, ou encore des propriétés de choses ou d'évènements.

Ainsi, le « tracé du programme sans trou ni chevauchement » n'est pas un fait, c'est juste une représentation du monde en termes de programme, de tracé, de trou, de couleurs ou de chevauchement. Cette représentation du monde est la représentation d'une information à communiquer. Nous reprenons ici en partie les termes utilisés par Shannon et Weaver (1964), qui prolongent notamment les travaux de Hartley (1928), dans la théorie mathématique de l'information. Ainsi, Shannon utilise une définition large de la communication, incluant « toutes les procédures par lesquelles un certain esprit peut affecter un autre esprit » [30]. Le problème fondamental dans la communication étant finalement de faire en sorte que le message soit correctement transmis (« reproduit ») d'un esprit à l'autre :

> The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. (Shannon, 1948, p. 379)

Shannon (ibid., p. 378) précise que « souvent, les messages ont une signification, c'est-à-dire qu'ils se réfèrent ou sont corrélés, selon un certain système, à certaines entités physiques ou conceptuelles » [31]. Par exemple, le message « il y a un chevauchement dans le tracé » est porteur du sens « le tracé est erroné ». L'information, du point de vue de l'émetteur comme du récepteur, est l'ensemble formé du message (la donnée*) et du sens. Ainsi, « reproduire » un message d'un émetteur vers un récepteur (doté de connaissances), c'est permettre au récepteur de reconstruire le sens du message à partir de sa représentation — c'est-à-dire de le comprendre.

29. « independently of anything else » (C. S. Peirce, Houser et Kloesel, 1992).

30. « The word communication will be used here in a very broad sense to include all of the procedures by which one mind may affect another » (Shannon, 1948, p. 378).

31. « Frequently the messages have meaning ; that is they refer to or are correlated according to some system with certain physical or conceptual entities. »

Considérons deux exemples. Pour le premier, considérons que les élèves ont construit le script S du TS5 avec une erreur sur b_8 dont la valeur est fixée à 2 au lieu de 3. On a donc :

$$S[] \rightsquigarrow \left(\text{ 
} , 32, 5 \right)$$

L'information « pour un TS5, le script S est erroné en b_8, avec un manque d'un hexagone » est donc codée par les données* (ou messages) suivantes :

— il y a un trou dans le tracé du dernier côté tracé
— le programme a dénombré 32 hexagones
— la mesure est 5

Les deux premières données* codent l'information « le tracé est erroné », la localisation en b_8 n'étant codée que par la première. Le tracé avec un trou est une représentation graphique porteuse de sens, un signe. La valeur du dénombrement, tout comme la mesure, est une représentation numérique, elle-aussi porteuse de sens dès lors qu'on la met en relation avec des éléments connus et identifiés du but : construire un TS de mesure 5, le TS5 a 33 hexagones. Dans cet exemple, il est assez probable que les élèves reconstruisent l'information « le script S est erroné ». On peut aussi imaginer qu'ils parviennent à reconstruire « le script S est erroné en b_8 pour un TS5 », voire « le script S est erroné en b_8, qui est trop court, pour un TS5 » ou encore, de manière idéale, « pour un TS5, le script S est erroné en b_8, avec un manque d'un hexagone ». Ces informations réduisent de plus en plus l'incertitude, ou le champ des possibles : c'est ainsi que Shannon et Weaver (1964) définissent l'*information*.

Un deuxième exemple permettra d'illustrer la différence entre l'information représentée et le fait construit. Prenons le cas où les élèves construisent le script S' qui doit tracer un TS5, avec encore une erreur sur b_8, mais cette fois-ci car sa valeur est trop élevée : 4 au lieu de 3. On a :

$$S'[] \rightsquigarrow \left(\text{ 
} , 34, 5 \right)$$

La représentation statique de la rétroaction mène à la représentation des données* suivantes :

— le tracé est bien formé
— il y a un chevauchement sur le sommet commun du premier, troisième et dernier côté tracé
— le programme a dénombré 34 hexagones
— la mesure est 5

On voit ici que le codage de la deuxième donnée*, le chevauchement, est fondamental pour la construction de l'information « le tracé est erroné » car celle-ci doit être encodée dans un certain système de signes sur lesquels il doit y avoir consensus :

> In the first place, there must be a group of physical symbols, such as words, dots and dashes or the like, which by general agreement convey certain meanings to the parties communicating. (ibid., p. 536)

Si le signe « chevauchement » n'est pas assimilé au sens « tracé erroné », l'information reconstruite par les élèves risque de ne pas être reproduite exactement (ni même approximativement) comme l'information codée par le rendu du programme. Il y aura une réduction de l'incertitude, mais en se basant sur une information qui, du point de vue de la situation, est erronée. D'autre part, le signe « tracé » produit par le programme, laisse la possibilité d'une erreur en b_1, en b_8, ou en b_3 [32] : la réduction de l'incertitude n'est pas aussi importante que dans le premier exemple. Shannon (1948) parle de « l'entropie de l'information », et, comme Weaver, il établit une équivalence entre entropie, information et réduction du champ des possibles (« freedom choices »). Cette réduction du champ des possibles est aussi visible chez Hartley (1928) :

> For example, in the sentence, "Apples are red," the first word eliminates other kinds of fruit and all other objects in general. The second directs attention to some property or condition of apples, and the third eliminates other possible colors. It does not, however, eliminate possibilities regarding the size of apples, and this further information may be conveyed by subsequent selections. (ibid., p. 536)

Pour Hartley ou Shannon, la transmission de l'information consiste à sélectionner des symboles qui attirent l'attention du récepteur, chaque nouvelle sélection supprimant des possibilités de suite symboles. Ils s'intéressent plus spécifiquement à la représentation et au codage des informations pour permettre leur meilleure et plus efficace possible reconstruction par le récepteur. L'encodage de l'information est de leur point de vue une technique, et c'est aussi le cas dans la conception d'une situation permettant une rétroaction. La conception d'une rétroaction vise à produire des données* efficacement interprétables. Si l'on cherche à comprendre comment les élèves construisent le problème, il est donc important de les considérer comme des récepteurs afin d'établir les informations qui sont reconstruites par eux. Des récepteurs, mais des récepteurs actifs et créatifs, qui peuvent sélectionner, identifier, mettre en relation de multiples données*. Ces données* ne sont pas obligatoirement celles prévues par la situation, et leur interprétation n'est pas forcément celle attendue : il nous faudra donc préciser les données* disponibles, celles qui seront mobilisées par les élèves et comment ils vont en tirer des faits permettant d'avancer dans le problème. Nous nous intéresserons aussi à cette réduction de l'incertitude qui peut être variable, en fonction de l'entropie de l'information en jeu (voir 3.7.6, p. 453).

2.4.3 Faits

La rétroaction vise à donner une information, à réduire l'incertitude — et à réduire le champ des possibles —, en codant un message dont le sens est l'information à transmettre. Cependant, les élèves ne reconstruisent pas forcément l'information initiale, soit par un décodage erroné, soit lacunaire, ou encore en ne donnant pas le sens attendu au message transmis, et les informations reconstruites ne diminuent pas l'incertitude de façon égale. La reconstruction par les élèves de l'information produite par la rétroaction est donc un élément essentiel afin de déterminer le cheminement des élèves dans leur construction du problème. Ce n'est cependant pas le seul : les

32. Une erreur en b_3 ne pourrait cependant pas aboutir sur la rétroaction donnée en exemple. On a un chevauchement lors de la construction de b_3 seulement dans le cas où le script est $TS(p)$ avec $b_1 = p$.

élèves, l'enseignante, le chercheur sont eux-mêmes producteurs d'éléments empiriques mobilisables. Dans cette partie, nous élargirons la réflexion sur les données et l'information aux faits produits par la situation ou construits par les élèves, en proposant une interprétation de la notion de « fait » dans le Cadre de l'Apprentissage par Problématisation.

Première approche des faits et de la factualisation

Problématiser, dans le CAP, c'est mettre en tension les contraintes ou données du registre empirique et les nécessités ou conditions du registre des modèles. Mais ces données ne sont pas toujours données :

> Ce qui nous semble important à souligner c'est que dans cette perspective, les faits ou les données malgré leur appellation ne sont pas donnés ; ils sont construits tout comme les conditions, les idées les expériences ou les théories. Tout est construit dans le cadre de l'enquête. (Doussot et al., 2022, p. 16)

Ainsi, nous rejoignons C. Orange (séminaire CAP) pour qui « d'un point de vue didactique, il semble intéressant de distinguer les "données" (ce qui est explicitement donné aux élèves) et les faits qu'ils mobilisent (à partir éventuellement de ces données ; mais aussi à partir de leurs connaissances, assurées ou non) », ces faits pouvant aussi être construits à partir d'autres problématisations. Cela nous amène à deux questions : qu'est-ce qu'un fait au regard de la problématisation, et qu'est-ce que la construction d'un fait ?

Qu'est-ce qu'un fait ? Comme le rappelle Fabre (2019), le terme « fait » est porteur de différentes dimensions de la factualité. Il souligne notamment trois aspects dans la question du fait : « 1) l'aspect ontologique ou l'effectivité, la réalité des faits ; 2) l'aspect épistémique ou la connaissance des faits ; 3) le lien entre ontologie et épistémologie ou la question de la vérité ». Pour lui, « les faits sont en tension entre ontologie et épistémologie » :

> Les faits ne sont pas des choses comme les tables, les chaises ou les arbres, ils concernent des relations, des rapports entre les choses, entre les choses et nous, ou encore des propriétés de choses ou d'événements

L'aspect ontologique concerne l'effectivité du fait : « le fait pose une existence, un réel [...], il est le "côté saillant du réel" » : lorsque je dis qu'un élève a modifié telle boucle de telle façon, cette modification existait avant que je ne la mette en mots et même si je ne la mettais pas en mots.

Mais un fait diffère de l'évènement : énoncé, il est lié au langage et à l'interprétation. S. Doussot a d'ailleurs souligné la définition d'un fait selon Ricoeur : « un fait est le contenu d'un énoncé relatant un événement »(Ricœur, 2000, p. 227). M. Fabre le dit autrement : « Nous n'avons accès aux faits qu'à travers le langage, les questionnements ». Si je mets en mots un événement, c'est donc que ça m'intéresse ou me questionne. Ricœur (2020) précise d'ailleurs que « dans un contexte d'action, donc d'intérêt, tout ce qui arrive ne fait pas événement, mais seulement ce qui surprend notre attente, ce qui est intéressant, ce qui est important ». Si j'évoque un fait, c'est donc que ça m'intéresse et que ça me pose question, ce qui illustre l'aspect épistémique du fait : « c'est bien du réel, mais du réel qui me concerne ».

La troisième dimension du fait, la question de la vérité — sujet très actuel — met en exergue la tension entre ontologie et épistémologie du fait. Une même réalité peut déboucher sur questions et donc des faits différents, contradictoires ou non compatibles, suivant par exemple que je sois chercheur ou élève. L'événement d'affection de l'expression numérique « 55 » au nombre de répétitions de la boucle b_3 pourrait permettre ainsi la construction de faits suivants concernant cet événement (et donc le script modifié par l'élève) :

— (a) : « $b_3 \leftarrow 55$ », représentation de cet événement, qui est vrai pour moi en tant chercheur (mais aussi pour le dispositif qui code cet événement), et qui m'intéresse car il me permet de construire le problème de savoir comment l'élève parvient à généraliser un motif informatisé.

— (b) : « $b_3 \leftarrow 5$ », représentation de cet événement, est vrai pour l'élève qui dans l'action pensait modifier les boucles b_2 à b_5 en leur donnant la même valeur (5) pour leur problème consistant à construire le script permettant de tracer un TS6, mais qui, en allant un peu vite, aurait fait une faute de frappe.

(a) et (b) sont contradictoires, mais lequel est la vérité ? D'un point de vue ontologique, il paraitrait légitime de penser que seul (a) est vrai. Mais pour l'élève, (b) est un énoncé tenu pour vrai à un moment donné, du moins jusqu'à ce qu'il le « défactualise » en le remettant en question parce que le tracé n'est pas celui attendu.

Un fait est donc un énoncé tenu pour vrai, construit à partir d'une question. Mais qu'est-ce que « construire un fait » ?

Qu'est-ce que la construction d'un fait ? Comme le disait Bachelard (1938, p. 14), « Rien n'est donné. Tout est construit. » C. Orange (séminaire Problématisation) précise qu'il ne faut pas comprendre cette construction comme étant « un choix parmi les informations disponibles, mais plutôt comme le résultat d'une construction épistémique précédente ». Ainsi dans la phase de généralisation de notre situation les élèves sont par exemple amenés à mobiliser le fait « $14 = 15 - 1$ ». Ce fait, ce résultat arithmétique, est bien entendu disponible chez des élèves de 4e, mais il est ici le résultat d'une construction issue de la tension entre la nécessaire existence d'une relation entre le nombre de répétitions d'une boucle et la valeur de la mesure d'un TS, et le fait précédemment construit que « pour un TS15, la boucle b_1 est répétée 14 fois ». Ce dernier fait est lui-même construit sur un problème précédent (« Modifier le script pour tracer un TS15 ») ; la nécessité étant construite suite au problème « trouver un seul script permettant de tracer toutes les instances », ce qui implique qu'il y a une expression unique pour b_1 représentant différentes valeurs en fonction de la mesure. Dans la même présentation, C. Orange précisait que « ces nouveaux faits (tant qu'on ne les rouvre pas) sont à l'origine de nouveaux problèmes et/ou participent à leur construction » : cette construction de faits, résultats d'une problématisation est ce que dans le CAP nous nommons « factualisation ».

S'il y a factualisation, il peut aussi y avoir « défactualisation », c'est-à-dire remise en cause, ou remise en question, de faits préalablement construits. Il s'agit pour C. Orange de « rouvrir la boîte noire ». En poursuivant l'exemple amorcé ci-dessus, les élèves peuvent avoir construit les faits « $b_1^{15} = 15 - 1$ est valide » et « $b_2^{15} = 14 - 1$ est valide ». Mais ce second fait devra être déconstruit, défactualisé. En effet, s'il contient bien une valeur prédictive et opératoire permettant d'automatiser la généralisation (« pour un TS15 comme pour un TS17 ou un TS231 je fais moins un pour b_1 puis encore moins un pour b_2 »), il est mis en contradiction par la nécessité d'une relation

avec la mesure (et non avec le nombre de répétitions d'un autre boucle) : il faut déconstruire « $b_2^{15} = 14 - 1$ est valide » pour construire « $b_2^{15} = 15 - 2$ est valide » [33]. Les enchainements problématiques (trouver une relation permettant d'exprimer b_2, trouver une relation en fonction de la mesure du TS pour trouver b_2) s'accompagnent de factualisation/défactualisation.

Pour résumer en reprenant l'exemple d'un fait donné par M. Fabre, « le chat est sur le tapis » :

— Un fait n'est pas une chose, ni le chat ni le tapis ne sont des faits

— Un fait est un énoncé, « le fait se donne dans une proposition : un sujet (le chat), un prédicat (sur le tapis), une copule (être) »

— Un fait relève d'un questionnement, d'un problème : si j'évoque ce chat sur tapis à ce moment-là, c'est que j'y vois un intérêt ou un questionnement (pourquoi y-est-il alors que d'habitude il n'y est pas ?, que fait-il dessus ?, comment est-il arrivé là ?). « D'un point de vue épistémique, un fait n'est un fait que relativement à un point de vue sur le monde »

En séminaire CAPMath, nous avons ainsi défini la notion de faits et de factualisation dans notre cadre :

Définition 10 *Les faits sont des énoncés (une relation entre des objets, entre les choses et nous, des propriétés...) tenus pour vrais à un moment donné par un ou des individus donnés.*

Définition 11 *La factualisation est un processus de construction de faits jugés pertinents pour un problème donné, à un niveau donné ou une classe donnée, dans un REX donné, à un moment donné. La factualisation renvoie donc à la position et la construction du problème.*

On distingue ainsi faits et données en ce que tous les faits n'amènent pas systématiquement à la construction de nécessités, mais ils participent à la construction du problème. Cependant, comme le rappelle C. Orange, ces faits sont « des éléments qui ne sont pas toujours visibles quand on s'intéresse seulement aux contraintes et mêmes aux argumentations ». Dans les parties qui suivent, nous reprendrons le schéma de l'organisation de l'activité dans la phase de généralisation (figure 2.17 p. 173) afin d'identifier et de catégoriser les différents faits pouvant être construits, par l'élève ou le chercheur. Ceci nous permettra de préciser, dans le chapitre suivant, comment nous pouvons utiliser les traces de l'activité récoltées pour constituer une représentation des enchainements problématiques dans la phase de généralisation ainsi qu'une variante des espaces de contraintes.

Pour mémoire, l'organisation de l'activité attendue est représentée en trois phases :

— Une phase d'action (② et ③), lors de laquelle les élèves choisissent un sous-but et modifient tout ou partie d'un script en fonction du problème défini par ce sous-but.

— Une phase de test et rétroaction (④, ⑤, ⑥ et ⑦), lors de laquelle les élèves vont potentiellement identifier leurs actions comme étant valide ou non au regard de leur but fixé, ainsi que les origines possibles des erreurs.

— Une phase de ré-action (⑧ et ⑨), lors de laquelle les élèves vont corriger leurs modifications en fonction de l'identification qu'ils auront faite de la localisation et du type des erreurs identifiées.

Selon nous, ces trois phases, très distinctes, permettent de construire des faits eux aussi distincts, que nous qualifions de *faits premiers*, *faits seconds* et *faits tiers*.

33. 15 pourra ainsi prendre un statut de nombre à caractère générique.

Faits premiers

Les actions de modification des boucles (③ et ⑨) sont considérées comme étant des *faits premiers*, ou faits bruts : cela conduit à l'interprétation « le binôme x a fait telle modification y », si on considère le fait du point de vue du chercheur, ou « Nous avons fait telle modification y » du point de vue de l'élève. Rappelons ici que l'interprétation du fait, le fait mis en mot, se distingue du fait « ontologique », le « côté saillant du réel » comme le décrit Fabre (2019). Le *fait premier* est à rapprocher de la catégorie de la « priméité » de Peirce. Nathan Houser résume ainsi la théorie des catégories de Peirce :

> Peirce's universal categories are three : firstness, secondness, and thirdness. Firstness is that which is as it is independently of anything else. Secondness is that which is as it is relative to something else. Thirdness is that which is as it is as mediate between two others. In Peirce's opinion, all conceptions at the most fundamental level can be reduced to these three. (C. S. Peirce, Houser et Kloesel, 1992, p. xxx)

Les actions de modifications d'un script par les élèves sont ainsi des faits, indépendamment de toute autre chose. Ce sont des faits premiers énonçant des événements ayant pour origine les élèves. Chaque script, à un moment donné, est aussi considéré comme un fait premier : c'est un fait pour l'EPGB, considéré comme algorithme mis en mot. De même, la rétroaction est un fait premier, mais ayant pour origine le dispositif ou le milieu ; ce fait étant construit suite au test effectué par les élèves. Cette rétroaction produit des faits premiers sous la forme du tracé et de l'affichage du dénombrement. Ainsi, $\mathcal{A}$ et $S \odot \mathcal{A}[]$ sont des faits premiers.

D'un autre côté, la validité ou l'invalidité des scripts constituent aussi des faits premiers. En fonction du but p, on peut définir les faits premiers de validité d'un script S^p ou d'un script issu d'un ensemble d'actions $S^p \odot \mathcal{A}$ ainsi :

- $\texttt{valide}(S^p) = [S^p = TS(p)] = [E^p(S^p) = \varnothing] = [V^p(S^p) = TS(p)]$
- $\texttt{valide}(S^p \odot \mathcal{A}) = [S^p \odot \mathcal{A} = TS(p)]$

Avec une granularité plus fine, la *validité des actions* effectuées par les élèves sur le script est elle-aussi un fait premier. Ces actions sont considérées comme *totalement* valides si l'ensemble des modifications du script S^p aboutit à (et seulement à) des expressions de $TS(p)$. Elles sont considérées *partiellement* valides si au moins une des modifications de S^p aboutit à une expression du script solution $TS(p)$. Pour un ensemble d'actions $\mathcal{A}$ s'appliquant à un script S^p, les boucles modifiées sont $\mathbf{b}(\mathcal{A})$, et l'ensemble de leurs nouvelles écritures est l'ensemble $S_{modif} \in S^p \odot \mathcal{A}$ tel que $\mathbf{b}(S^p) = \mathbf{b}(\mathcal{A})$. Soit :

- $S_{modif} = \{expr \in S^p \odot \mathcal{A} | \mathbf{b}(expr) \in \mathbf{b}(\mathcal{A})\}$

Les actions effectuées par les élèves sont totalement valides si toutes les expressions des boucles de S_{modif} sont des expressions de $TS(p)$, et partiellement valides si l'une au moins des expressions de S_{modif} est une expression de $TS(p)$. On pourrait ainsi définir :

- $\texttt{total}(\mathcal{A}, S^p) = \texttt{total}(S_{modif}) = [\forall expr \in S_{modif}, \mathbf{b}(expr) \in \mathbf{b}(TS(p))]$.
- $\texttt{partiel}(\mathcal{A}, S^p) = \texttt{partiel}(S_{modif}) = [\exists expr \in S_{modif}, \mathbf{b}(expr) \in \mathbf{b}(TS(p))]$.
- $\texttt{erroné}(\mathcal{A}, S^p) = \texttt{erroné}(S_{modif}) = [\forall expr \in S_{modif}, \mathbf{b}(expr) \notin \mathbf{b}(TS(p))] = \neg(\texttt{total}(S_{modif}) \vee \texttt{partiel}(S_{modif}))$.

Ces faits premiers sont « indépendants de toute chose », sauf du producteur du fait. Au niveau de l'artefact, plusieurs données* [34] particulières sont produites, dont notamment :

- le tracé,

34. L'écriture « donnée* » permet de désigner les faits premiers issus de l'artefact (voir 2.4.2 p. 182).

(a) Un $TS(5)$ pour une *mesure* de 5

(b) Un $TS(5)$ pour une *mesure* de 6

(c) Un $TS(5)$ pour une *mesure* de 3

(d) Un $TS(5)$ pour une *mesure* de 21

(e) Un $TS(5)$ pour une *mesure* de 100

Figure 2.20 – Cinq rétroactions possibles pour un $TS(5)$: valide ou invalide ?

— l'affichage du résultat du dénombrement,
— l'affichage de la valeur de la mesure,
— l'inscription ou non du tracé dans l'espace d'exécution.

Le tracé et l'affichage du résultat sont fondamentaux, pour les élèves, afin déterminer la validité ou non du tracé en fonction du but poursuivi. Concernant le tracé, les faits suivants sont ainsi produits :

— le tracé est bien ou mal formé : la forme du tracé est un TS, ou non ;
— le tracé produit des trous ou des chevauchements, ou non ;
— le nombre d'hexagones d'un côté d'un triangle de base est de j ;
— le nombre d'hexagones tracés par la boucle n°i est j.

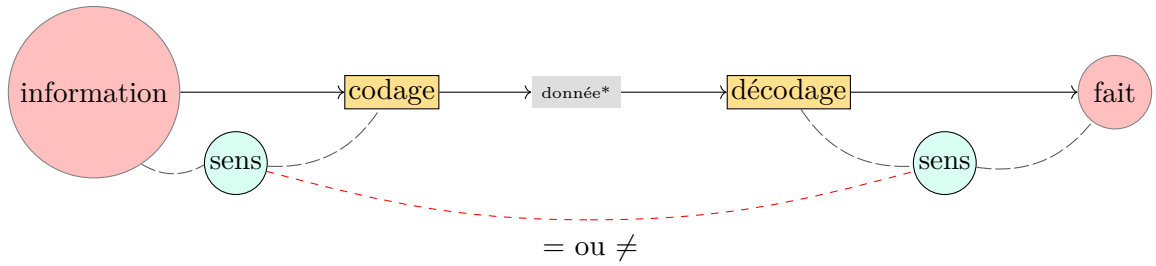
L'affichage du résultat produit le fait :

— l'exécution a permis de dénombrer j hexagones

La valeur de la mesure (affichée ou entrée ou voulue) est aussi un fait constituant de la situation : il peut ainsi y avoir contradiction entre une mesure affichée par le script correspondant au sous-but voulu et la mesure souhaitée. Ce sera le cas par exemple si les élèves veulent construire un TS5 en modifiant le script $TS(6)$ directement, c'est-à-dire en gardant l'instruction initialisation6. Cette procédure initialise notamment la valeur de la *mesure* à 6 et définit la dimension (en nombre de pixels) des hexagones en fonction de cette mesure. Ainsi, dans l'exemple donné par la figure 2.20b (p. 190), le script S des élèves trace et dénombre correctement un TS5, mais ce résultat est non cohérent avec la valeur de la mesure [35].

Enfin, l'inscription ou non du tracé dans l'espace d'exécution, fait auquel on pourrait rajouter l'espace occupé par la figure, sont des éléments visibles dans la rétroaction, mais qui ne sont pas directement en lien avec la validité du script : au plus ils sont le signe d'une incohérence entre la mesure effective (celle qui définit le TS tracé) et la valeur de la mesure affichée (qui définit indirectement la valeur du « zoom »), ou bien ils ne permettent pas de déterminer la validité ou non du tracé. Dans les figures 2.20a et 2.20b (p. 190), le tracé peut être validé, mais ce n'est pas le cas dans les figures 2.20c ou 2.20e (p. 190).

Précisons que ces faits premiers sont des faits *potentiellement* construits par les élèves, mais ce sont aussi en partie des faits *effectivement* construits par le dispositif de captation mis en œuvre (des faits correspondants aux événements qui font bouger l'EPGB, voir 2.2.4, p. 157) et des faits construits par le chercheur. En ce sens, ce sont des faits pour le dispositif et pour le chercheur. Une question méthodologique importante sera de déterminer comment à partir de faits pour le chercheur on peut identifier les faits effectivement construits par les élèves.



Faits seconds

Ces différents faits premiers, axés sur la validation des actions et des scripts, sont ici décrits d'un point de vue ontologique : ce sont des faits indépendamment de toute autre chose, et, notamment, indépendamment de toute interprétation que l'on pourrait faire. Pour paraphraser M. Fabre, lorsque nous disons « tel script est valide », « nous prétendons dire quelque chose du réel, c'est-à-dire quelque chose dont certes nous avons connaissance, mais qui serait le cas même si nous ne le savions pas »(Fabre, 2019). Le fait du point de vue épistémique est « un fait [...] relativement à un point de vue sur le monde » : lorsque les élèves disent (ou pensent) « tel script est valide »,

35. Notons que lorsque l'utilisateur est invité à entrer une valeur pour la mesure, la probabilité d'une contradiction entre la mesure affichée et la mesure entrée est faible : elle nécessiterait que les élèves modifient la structure du script en mêlant initialisation6 et initialisation, ou une entrée, ou une modification manuelle de la mesure mettre mesure à 6.

c'est une manifestation de leur interprétation du monde, en mettant en relation d'autres faits. La validité accordée par les élèves — ou le chercheur — à leurs actions, est une mise en relation de ces actions (faits premiers), du résultat de ces actions (fait premier, rétroaction) et, éventuellement, de leur but (fait premier). Nous considérons alors ces faits comme étant des faits seconds.

La secondéité des faits se retrouve ainsi dans leur mise en relation, notamment en ⑤. La validité ou non de leur action vue par les élèves est un fait second, établissant une relation entre action et rétroaction. Cette relation, en langage naturel, pourrait s'exprimer ainsi :

— « l'ensemble des actions $\mathcal{A}$ produit un résultat valide »
— « l'ensemble des actions $\mathcal{A}$ produit un résultat invalide »
— « l'ensemble des actions $\mathcal{A}$ produit un résultat partiellement valide »
— « le script $S \odot \mathcal{A}$ produit un résultat valide / invalide / indéterminé / partiellement valide »

Le but comme élément essentiel La mise en relation des actions et réactions indépendamment du but poursuivi est-elle possible ? Imaginons le cas où des élèves modifient un script erroné afin de tracer et dénombrer un TS7. Il est possible, s'ils procèdent par essai-erreur et si le fait premier « la forme du tracé est un TS valide » prédomine, que les élèves aboutissent au script $S = \{8/7/16\}$. Celui-ci produit une forme valide et un affichage cohérent du nombre d'hexagones dénombrés (69). Il est alors possible que les élèves considèrent ce script comme valide, ce qui impliquerait que, soit ils n'ont pas établi de relation avec leur but, soit la relation but-tracé valide n'est pas adéquate[36], soit le but des élèves n'est plus le même : on est passé de « but : construire $TS(7)$ » à « but : construire S produisant un tracé valide ». Ce « glissement » de but peut être induit par le changement d'activité de l'élève : dans un premier temps, il s'agit de *modifier l'ensemble du script*[37] en fonction du but poursuivi de tracé d'un TS particulier —ou générique — (étape ③) ; puis, dans un second temps, de *modifier localement le script* (étape ⑨). Le but est ici assimilé au problème (ou sous-problème) que se donnent les élèves : comment modifier un script pour tracer une certaine instance ? Comment corriger un script pour obtenir un tracé valide ? comment modifier un script pour tracer n'importe quelle instance ?...

En tout état de cause, l'identification du but poursuivi par les élèves est un élément essentiel pour établir les faits construits par eux. Cette identification pourra se faire par la tâche prescrite, par la valeur de b_1 ou de la majorité des b_i, par l'entrée de la mesure du TS demandé, ou par les interactions langagières (voir 2.5.1, p. 199).

Faits de validation Parmi les faits seconds susceptibles d'être construits par les élèves en activité dans notre situation, nous pouvons dégager notamment les suivants, qui concernent la validation par la rétroaction :

1. « le tracé est valide/invalide/non déterminé », Rel.1 Tracé-figure bien formée
2. « le dénombrement est valide/invalide/non déterminé », Rel. 2 Tracé (dénombrement manuel ou comparaison avec figure connue)-compteur OU Rel.3 mesure-compteur
3. « (le tracé et le dénombrement) sont valides/invalides/non déterminés », 2 rel. : 1 et 2 ; ou 1 et 3
4. « (le tracé ou le dénombrement) sont valides/invalides/non déterminés », Rel.1
5. « la mesure est valide/invalide/indéterminée » (eut égard au tracé ou au compteur) Rel.4 mesure-tracé Rel.3 mesure-compteur

36. Le lien *mesure* - nombre d'hexagones d'un côté des triangles de base est erroné.
37. Ou de modifier localement afin de tester une modification à généraliser à l'ensemble du script, ce que l'on trouvera notamment lors de la phase de généralisation.

6. « le résultat est valide/invalide/non déterminé », Rel6 mesure-compteur-tracé

Comme souligné plus haut, ces faits de validation deviennent dépendants du but des élèves lorsqu'ils sont mis en relation avec les actions visant ce but. Ainsi, si l'on veut analyser l'activité de l'élève, on doit rechercher les faits construits mettant en relation les actions, le but poursuivi et la validité interprétée de la rétroaction. Ces faits mettent alors en relation des faits premiers ayant pour origine l'élève (les actions), des faits premiers ayant pour origine le milieu ou le dispositif (les rétroactions), des faits premiers ayant pour origine la situation (la validité de la figure, la validité du lien figure-dénombrement). Ces faits sont ici des *faits assertoriques* : la relation est établie mais n'est pas expliquée, il n'y a pas de « médiateur » entre les faits mis en relation.

Faits tiers

Toujours en suivant la théorie des catégories de Peirce, on peut définir les faits tiers comme étant la médiation expliquant la relation entre d'autres faits.

Dans notre situation, où l'élève doit résoudre un problème par une activité de programmation et est soumis à des rétroactions, nous identifions quatre types de faits tiers pour l'élève : les faits « Action-Rétroaction+TEA », « Action-Rétroaction+Ajustement », « Action-Rétroaction+Localisation », et les « meta-faits », construits à partir des faits précédents.

ART Les faits « Action-Rétroaction+TEA » (ART), ou « fait TEA » sont les faits construits sur la médiation de la relation Action-Rétroaction par le théorème-en-acte (TEA) qui organise l'activité de l'élève. Une suite d'actions de modification d'un script, suivie d'une exécution produisant une rétroaction, est le signe d'une activité organisée : c'est la manifestation d'un théorème-en-acte. La rétroaction quant à elle représente la relation Action-Effet de l'action. Si la rétroaction est vue comme une validation de l'action, le TEA se trouve « renforcé » ou confirmé. En revanche, si la rétroaction est invalidante, le TEA se retrouve mis en tension avec les nécessités du problème : ce TEA produisant un résultat erroné n'est pas compatible avec les nécessités qui soutiennent le problème. Ces faits, que nous pourrions qualifier de « faits problématiques » puisqu'ils permettent d'avancer dans la construction du problème, sont donc essentiels pour comprendre comment les élèves cheminent dans le problème, et notamment pour illustrer ce cheminement par un espace des contraintes(Orange, 2005). Nous dirons qu'un ART est *valide* si la rétroaction est le signe d'un script valide. Dans le cas contraire, nous qualifierons l'ART d'*invalide*. Cette validité (ou non) est liée au résultat de l'action, elle ne concerne pas (directement) le TEA. Ainsi, un TEA non valide — c'est-à-dire non compatible avec une au moins des nécessités du problème — peut produire des ART valides.

ARL Les faits « Action-Rétroaction+Localisation » (ARL, ou « fait localisation ») concernent la médiation de la relation Action-[Rétroaction invalidante] par la localisation et l'interprétation de l'erreur. La localisation et l'interprétation d'une erreur dans le script est un fait tiers, médiation entre l'action sur le script et la rétroaction. $\mathbf{b}(E^p(S \odot \mathcal{A}))$ est ce qui dénote la localisation des erreurs, c'est-à-dire dans notre situation les boucles erronées ou considérées comme telles. L'identification de ces boucles par les élèves indique une prise en compte de la structure du programme et de la rétroaction. Cela signifie notamment que les élèves mobilisent les concepts de séquence et de boucle : sans ces concepts, il n'est pas possible de faire le lien entre ce que le programme donne à voir et l'algorithme dont il est l'écriture.

Faut-il distinguer la localisation de l'interprétation ? La localisation consiste ici à faire le lien entre une manifestation de l'erreur et une (ou plusieurs) boucles. L'interprétation consistant, en fonction de la boucle, à identifier les raisons de l'erreur, soit le lien entre la boucle identifiée et la rétroaction. Dans notre situation, il nous semble que cette distinction est possible mais peu pertinente. En effet, les indices fournis par l'artefact, les données*, ne permettent la localisation de l'erreur que s'il y a une interprétation de l'erreur en fonction de l'action. Dans la figure 2.21 (p. 195), on trouve différentes rétroactions issues de scripts $TS(5)$ erronés. Il s'agit ici d'erreurs simples, c'est-à-dire pouvant être dues à une seule erreur sur une seule boucle. Nous indiquons en légende l'erreur qui peut expliquer la rétroaction : ainsi, « $b_1 + 1$ » signifie que la boucle b_1 trace un hexagone de trop, soit $b_1 = 5$ dans le cas du TS5. Examinons ce cas. Les données* permettant d'invalider le tracé sont les suivantes :

— il y a au moins un chevauchement (milieu du grand côté de gauche, ou sommet du premier triangle de base)
— le nombre d'hexagones dénombré est de 34, alors qu'un TS5 est fait de 33 hexagones
— le tracé sort en partie de l'espace d'exécution.

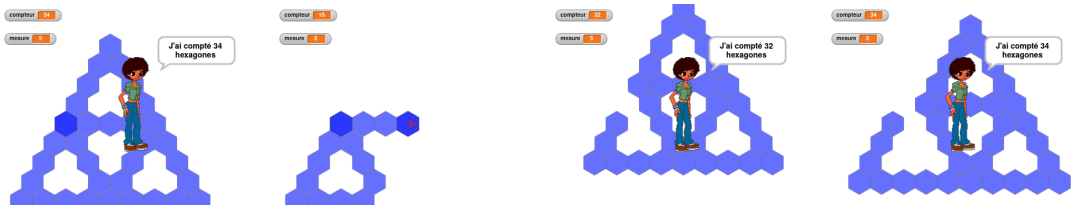
Ce dernier point n'invalide pas de façon stricte le tracé, mais il est néanmoins un signe de résultat inattendu : même si cela n'a pas été dit explicitement, les scripts doivent permettre de voir l'intégralité du TS tracé. Concernant le chevauchement, d'où peut venir l'erreur ? Il y a trois possibilités : soit b_1, soit b_3, soit b_8 [38]. Ce sont les seules boucles pouvant tracer l'hexagone marquant le chevauchement. Si les élèves identifient l'une de ces boucles comme source possible de l'erreur, cela signifie qu'ils interprètent le script, en le mettant en relation avec la rétroaction : même si cette interprétation se limite au constat de la localisation d'une boucle, sans se définir sur les raisons de l'erreur sur cette boucle, localisation et interprétation sont liées. De plus, nous verrons plus loin que, lorsqu'on se base sur les actions de programmation des élèves, la seule indication nous permettant d'identifier les boucles identifiées par les élèves comme étant erronées est l'action suivante de modification de ces boucles. Ces faits suivant les faits de localisation sont des faits d'ajustement.

ARA Les faits « Action-Rétroaction+Ajustement » (ARA, ou « fait ajustement ») sont les faits construits sur la médiation d'un ART invalide par un « ajustement » du TEA : si l'application du TEA par une suite d'actions produit un résultat erroné, une nouvelle action peut être envisagée afin de corriger ce résultat. Cette correction n'est pas un résultat du TEA, elle est un ajustement de l'action en fonction de la rétroaction, afin de supprimer la manifestation de l'erreur. Par suite, elle est le signe de la nécessité de modifier, d'ajuster, d'amender le TEA en jeu. Ces faits d'ajustement sont directement liés aux faits de localisation vus précédemment, et ils en sont les révélateurs : c'est lorsqu'un élève modifie une certaine boucle b_i que l'on peut penser qu'il a localisé une source d'erreur possible en b_i.

Articulation des faits Ces trois types de faits que l'on envisage de reconstruire grâce aux traces de programmation des élèves sont fortement imbriqués. Ainsi par exemple, si les élèves mobilisent le TEA « pour passer d'une instance à la suivante, je rajoute 1 à chaque boucle » [39], la rétroaction sera la manifestation d'une erreur sur b_7, qui implique un tracé du côté correspondant

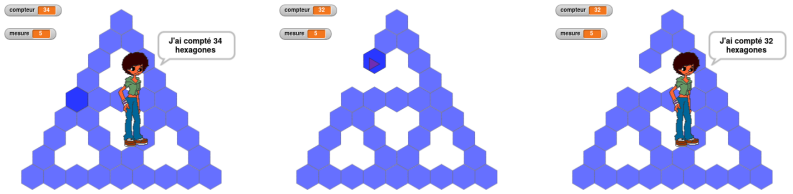
38. Notons que la rétroaction serait très similaire en cas d'erreur à la fois sur b_1 et b_8 : dans la version des scripts utilisés, le chevauchement rend la couleur de l'hexagone plus foncée, mais si un premier chevauchement est très visible, un deuxième l'est beaucoup moins : Normal Un chevauchement Deux chevauchements

39. TEA majoritairement présent en début de généralisation, que nous notons « +1 /+1 /+1 » en suivant les conventions de simplification par famille de boucles.



(a) b1+1, b1+1 en cours, b1-1, b2+1

(b) b6+1, b6-1, b7-1



(c) b8+1, b8+1 en cours, b8-1

Figure 2.21 – Quelques rétroactions sur erreurs

trop court de un hexagone (voir figure 2.21b, p. 195). Ce fait TEA entre alors en tension avec une nécessité locale du problème : « le tracé doit être bien formé, sans trou ni chevauchement ». Trois faits de localisation cohérents avec le script sont alors possibles : [b_7 est trop court], ou [b_8 est trop long], ou [b_7 est trop court et b_8 est trop long]. La seule indication nous permettant d'identifier le fait de localisation construit, est le fait d'ajustement qui le suit. Si les élèves *ajustent* b_7 en lui ajoutant encore un, c'est qu'ils auront construit le fait de localisation [b_7 est trop court de un]. Dans cette hypothèse, le résultat produit sera valide [40] Cet ajustement n'est pas la manifestation d'un TEA concernant l'action « passer d'une instance à la suivante », mais la manifestation d'un TEA plus local, permettant de « corriger le script lorsqu'un côté est trop court de 1 hexagone ». Dans l'idéal, cela devrait amener les élèves à ajuster leur TEA initial, qui deviendrait alors « pour passer d'une instance à la suivante, je rajoute 1 à chaque boucle sauf pour b_7 où je rajoute 2 » [41]. Il pourrait aussi devenir « pour passer d'une instance à la suivante, je rajoute 1 à chaque boucle et je rajoute encore 1 à b_7 » [42].

Le fait d'ajustement est donc le résultat d'une tentative visant à régler une tension TEA-nécessité en considérant le fait de localisation. Suivant le résultat de cet ajustement (la validité ou non vue dans la rétroaction), il fera ou non bouger le TEA en œuvre. Notons néanmoins que cette adaptation du TEA n'est pas garantie : il est sans doute possible que les élèves, à certains moments, considèrent l'ARA comme l'action définissant leur but. Dans ce cas, l'objectif de l'élève en agissant ne serait plus de résoudre la tension TEA-nécessité, mais juste d'obtenir un résultat valide : il s'intéresse au résultat, à la résolution du problème, et non à la procédure, à la construction du problème.

40. On suppose ici que l'instance initiale est un script valide.
41. soit « +1 /+1 /+2 »
42. « +1 /+1 /+1+1 »

Faits et REX Ainsi, l'identification des faits tiers Action-Rétroaction+TEA construits par les élèves est une condition nécessaire pour établir comment les élèves posent et construisent le problème, dans quel REX, et avec quel TEA sous-tendant leurs actions. Les faits ARA et ARL sont quant à eux des faits susceptibles de produire une adaptation du TEA. Si les élèves problématisent, ils mettent en relation ces faits tiers, et les mettent en tension avec les nécessités du problème. Cela a alors une influence sur le cadre des explications possibles et donc sur les invariants opératoires des schèmes mobilisés.

Une raison des erreurs, ainsi qu'une raison de la validité, sont des médiations expliquant la relation entre faits : ce sont des *faits apodictiques* ou *faits raisonnés* (ce qui n'implique pas forcément la validité de ces raisons). Ces raisons dépendent à la fois des faits construits (les données du problème), des nécessités ou des modèles construits, et du « cadre permettant de penser [le] problème » (Doussot et al., 2022, p. 10). Ce cadre, le registre explicatif (REX), « indique une manière habituelle ou établie de confronter les faits ou les idées explicatives pour identifier les conditions de possibilité d'une explication » (ibid., p. 14). C'est ce registre explicatif (REX) qui encadre les possibilités d'inférences et d'invariants opératoires (Vergnaud, 2011). L'objectif pour nous étant de permettre aux élèves de [construire, mobiliser, se situer dans] un REX algébrique.

2.5 Construction et analyse des données

2.5.1 Formalisation des faits à partir de l'histoire du programme et des actions de programmation

Dans cette partie, nous détaillerons la méthodologie nous permettant de reconstituer et transcrire les faits premiers, seconds ou tiers construits par les élèves, en nous basant sur les données recueillies par le système de captation et l'histoire du programme ainsi reconstituée.

Faits Premiers disponibles ou données captées

La captation des événements de l'EPGB nous permet d'établir les faits premiers disponibles, c'est-à-dire les faits premiers pour l'EPGB. Ce sont ici des données pour le chercheur, qui ensuite permettront de reconstituer les faits premiers pour les élèves. Ces événements sont issus d'actions sur l'EPGB. Nous retenons ici les actions suivantes :

- les chargements, rechargements ou sauvegardes des scripts ;
- les navigations parmi les différentes catégories de blocs ;
- les lancements (exécutions) des scripts, leur arrêt ou pause (manuels), leur terminaison (fin « normale ») ;
- les valeurs éventuelles entrées par l'utilisateur ;
- les rétroactions produites ;
- les modifications de la structure du script ;
- les modifications de la représentation du nombre de répétitions des boucles ;
- les créations de variables ou mobilisation d'instructions dédiées à la manipulation de variables ;

Chargements, sauvegarde, navigation Parmi les événements concernant les modifications de l'environnement de l'EPGB (voir 2.2.4, p. 157), nous retenons ainsi les sauvegardes, chargements et rechargements, ainsi que la navigation dans les familles de blocs. Les sauvegardes nous permettent d'identifier des moments où les élèves « fixent » un résultat. Cela est potentiellement

la marque d'un intérêt spécifique du script en cours de modification. Cela peut aussi être une mise en suspens de l'organisation de l'activité en cours : l'organisation en cours, qui dirige les modifications du programme, est en quelque sorte mise de côté : les changements sont sauvegardés, les élèves peuvent alors explorer d'autres voies. Les chargements sont potentiellement le signe d'une « remise à zéro » (RAZ) des modifications : soit pour repartir sur une base stable — ce qui sera notamment le cas lors du rechargement du programme de la séance —, soit pour reprendre les modifications mises en suspens. La navigation parmi les différentes catégories de blocs peut être le signe d'une exploration du langage : les élèves sont potentiellement en recherche d'un élément du langage θ permettant de résoudre un problème spécifique. Nous nous intéresserons bien sûr plus spécifiquement aux moments où les élèves explorent la catégorie « variables ».

Concernant les évènements modifiant l'état du programme (le changement de l'état d'exécution d'au moins un des scripts du programme), nous retiendrons les exécutions, arrêts manuels ou non, les éventuelles valeurs entrées par l'utilisateur ainsi que les rétroactions produites.

Exécutions et terminaison L'exécution d'un script est la manifestation de la recherche d'une rétroaction, il s'agit pour les élèves, en général, de chercher des indications pertinentes sur la validité ou non du tracé, mais aussi des informations complémentaires sur le type d'erreurs et leur localisation (2.4.2, p. 179). On peut légitimement penser que dans la majorité des cas, les élèves testent le résultat de leurs actions afin de les valider ou non. Cependant, le script exécuté n'est pas nécessairement celui en cours de modification : dans ce cas — hors erreur de manipulation — il est probable qu'il soit lancé comme « référence », notamment s'il a déjà été validé. Ainsi, l'exécution d'un script reconnu comme produisant un résultat valide peut permettre par exemple de déterminer le nombre d'hexagones tracés par l'une ou l'autre des boucles, et de faire le lien entre ce nombre, ces boucles, l'instance du script valide et l'instance en cours de modification.

Les arrêts manuels sont les actions des élèves visant à interrompre l'exécution en cours avant sa terminaison normale. Ces arrêts sont susceptibles de correspondre aux moments où les élèves estiment avoir suffisamment d'informations produites par la rétroaction. Il n'est pas toujours utile de laisser le programme se dérouler jusqu'au bout si l'on souhaite vérifier la validité des modifications sur les premières boucles, par exemple. Parfois, ces arrêts manuels peuvent être le signe d'une exploration de la structure du programme, ils prennent dans ce cas le même rôle que la pause. La pause et la reprise, lorsqu'elles sont utilisées, indiquent une exploration décomposée de la structure ou la recherche d'éléments de validation correspondant à plusieurs moments du programme — ici les différentes boucles. Lorsque les élèves laissent le programme s'exécuter jusqu'à son arrêt normal, cela peut indiquer la recherche d'une validation globale, et non seulement de la validation ou de l'étude d'une partie spécifique du script. Le tracé final, la forme, ainsi que l'affichage du nombre d'hexagones dénombrés, sont des indicateurs de la validité du script.

Tests et rétroactions La valeur entrée par l'utilisateur[43] est un élément essentiel de la compréhension de l'activité de l'élève lors de la phase de généralisation : le choix de la valeur, sa régularité ou ses variations, sont des indices importants de ce que cherche à vérifier l'élève lors d'un test. Par exemple, l'utilisation d'une mesure dont un script valide est déjà construit peut permettre de comparer les tracés — au niveau local des boucles, ou au niveau global — ou le dénombrement. L'utilisation d'une même mesure sur plusieurs tests est une possible indication que l'élève fixe une mesure spécifique, c'est-à-dire qu'une instance spécifique est testée, afin de faciliter la validation des différentes actions. La variation de cette entrée entre différents tests peut quant

43. Nous parlons ici d'utilisateur car l'élève passe d'une posture de programmeur à celle d'utilisateur d'un programme.

à elle être le signe d'une validation pour d'autres instances ou de l'exploration des modifications du tracé. Les exécutions simples d'un script ne sont pas précisées dans les transcriptions des actions de programmation : elles sont, sauf exception signalée, présentes implicitement à la suite d'une étape de modification des boucles (voir Étapes p. 198). Enfin, la rétroaction est un élément essentiel à connaître, puisque cette rétroaction permet le contrôle par l'élève de son activité.

Boucles et structure Du point de vue de la structure du programme, nous nous intéresserons plus spécifiquement aux modifications apportées aux boucles (ou à la représentation du nombre de répétitions de chacune de ses boucles), puisque, dans notre situation, elles suffisent à définir le script (si aucune autre modification de structure n'a eu lieu). Les changements autres dans la structure (ajout ou suppression d'instructions, modification d'autres instructions que les boucles) seront néanmoins à identifier afin de déterminer la position du problème par l'élève, et de comprendre les difficultés ou erreurs d'interprétation des rétroactions.

Création d'éléments nouveaux du langage Les événements liés à la création ou la manipulation de variables seront aussi relevés, puisque directement liés à notre recherche. Nous ne traitons pas, en partie du fait des limitations du système de captations, les créations d'autres « lutins », ni de nouveaux blocs. Le dispositif n'empêche pas la création de nouveaux blocs, — donc de nouveaux éléments du langage — mais bloque leur modification, à la fois pour des raisons techniques, mais aussi parce qu'il ne s'agit pas ici de changer de niveau d'abstraction. C'est une limite du dispositif qui devra être dépassée : il n'y a une trace de la création d'un nouvel élément du langage formulé en termes de bloc que si ce bloc est utilisé, c'est-à-dire déplacé sur la zone de programmation. Or, *a posteriori*, il semble que ces créations de blocs sont des actions d'ordre sémiotique et les aspects sémiotiques paraissent essentiels dans le cadre d'une réflexion sur la construction du concept de paramètre. Avec le dispositif actuel, nous pouvons donc passer à côté de certains de ces évènements sémiotiques si les deux conditions suivantes sont réunies :

1. le bloc créé n'est pas utilisé ;
2. on ne dispose pas de captation audio et vidéo du groupe créant ce bloc.

Rappelons que algorithmique et algèbre partagent cette particularité concernant les langages permettant leur mise en œuvre : il est possible de créer de nouveaux mots dans ces langages, dont la portée est limitée au traitement du problème, voire plus localement.

Étapes

L'ensemble de l'activité de programmation des élèves est retranscrit par *étape*. Nous considérons une *étape* de programmation comme un ensemble d'actions similaires et consécutives de programmation sur un même objet. On retrouve ainsi dans la transcription des actions les trois familles suivantes :

— les modifications de boucles
— les tests
— les chargements ou sauvegardes

Les modifications de boucles sont considérées comme une étape aux conditions suivantes :

— elles concernent le même script,
— elles concernent des boucles différentes,
— elles sont consécutives, c'est-à-dire non séparées par d'autres actions.

Une étape, basiquement, est donc soit [un ensemble de modifications des boucles différentes d'un script suivi (d'une exécution ou d'une opération de chargement ou sauvegarde)], soit [un ensemble de tests indépendants de toute modification], soit [une opération de sauvegarde ou de chargement]. Pour chaque étape, nous préciserons un ensemble de propriétés la définissant. Ces différentes propriétés sont explicitées ci-après. Toutes ces propriétés s'appuient sur les traces de programmation captées.

Faits Premiers reconstitués

À partir des événements captés et de l'histoire du programme, il est possible de reconstituer les faits premiers construits par les élèves. Sans disposer de ce que les élèves disent de ce qu'ils font, ces faits sont hypothétiques, mais probables. Dans ce qui suit, nous nous attacherons à expliciter comment nous reconstruisons les faits « tâche prescrite », « but (ou sous but) poursuivi » et « actions de l'élève ».

La tâche prescrite et script attendu La tâche prescrite, ou plus précisément la tâche *intermédiaire* prescrite, précise le script attendu : soit une instance précise, soit une certaine instance parmi plusieurs (17, 19 ou 21), soit le script générique. Cet attendu est précisé par le commentaire qui accompagne la tête de script (voir figure 2.22, p. 199). Il est aussi implicitement visible dans le bloc d'initialisation : 

sous-entend que la *mesure* est initialisée à X. Pour rappel (voir 2.1, p. 131), ces blocs d'initialisation initialisent la variable *mesure* ainsi que la dimension des hexagones tracés, afin que la forme soit ajustée à l'espace de traçage (figure 2.23, p. 200). En outre, le script d'initialisation du $TS(n)$ rend transparente la demande d'entrée de la part de l'utilisateur et le stockage de la valeur entrée dans la variable *mesure* (figure 2.23c, p. 200).

(a) scripts « 5 », « 6 » et « 7 »

(b) scripts « 11 », « 22 » et « n »

Figure 2.22 – Têtes de scripts proposés aux élèves et tâches prescrites

La tâche prescrite est censée correspondre au but que se donne les élèves, lorsque ce but est la création du script traçant un certain TS. Mais on verra que certains groupes utilisent ces têtes de script seulement comme « zone de dépôt » d'une copie de script, sans prendre en compte la tâche prescrite. Ainsi, après avoir complété le script $TS(7)$, ils vont compléter le script suivant, c'est-à-dire le script immédiatement disponible à droite du script $TS(7)$ pour un TS8, alors que le script attendu est $TS(11)$.

(a) Script $TS(5)$ **(b)** Script $max22$

(c) Script $TS(n)$

Figure 2.23 – Blocs d'initialisation des scripts

But et instance en cours de construction Le but des élèves, c'est-à-dire dans notre situation le script qu'ils cherchent à construire, peut se déduire — parfois avec beaucoup d'incertitude — des éléments suivants :

— la tâche prescrite
— le script précédemment traité
— les valeurs choisies pour les différentes boucles
— les valeurs entrées par l'utilisateur.

Ainsi, si les élèves modifient le script S dont le commentaire en entête précise de « compléter pour un TS6 », il est probable que leur but soit de construire le $TS(6)$. Si le script précédemment traité était le $TS(5)$, il est encore plus probable qu'ils passent à l'instance suivante. Si en outre ils modifient $b_1 \leftarrow 5$, cela renforce la probabilité que leur but coïncide à la tâche prescrite, puisque $b_1 = 5$ est valide pour $TS(6)$. Cette probabilité augmentera encore avec le nombre de valeurs de

boucles dans $V^6(S)$: l'instance en cours de construction est cohérente avec la tâche prescrite. De même, si les élèves modifient le script leur demandant de tracer le script générique, mais qu'ils utilisent des valeurs compatibles avec le $TS(6)$ et qu'ils choisissent d'entrer comme valeur « 6 », il est très probable qu'ils se soient définis comme but de construire le $TS(6)$: la valeur entrée est cohérente avec l'instance en construction. Cependant, comment identifier le but des élèves si, pour le même exemple que ci-dessus, ils effectuent $b_1 \leftarrow 6$? Sont-ils en train de faire un TS6, en considérant que la première boucle doit tracer autant d'hexagones que la valeur de la mesure ? Sont-ils en cours de construction d'un ($TS(7)$) car ils ont déjà identifié que pour un certain $TS(i)$, $b_1 = i - 1$? Tentent-ils simplement d'ajuster leur script pour obtenir une figure bien formée, sans prendre en compte l'instance du script voulue ? Ainsi, dans certains cas de figure, il est difficile de se prononcer quant à l'instance voulue par les élèves ou le but poursuivi. Dans la mise en forme des données recueillies, nous préciserons les éléments suivants :

— la tâche prescrite, qui est identifiable en général par l'entête du script en cours de modification

— la valeur entrée par l'utilisateur (ou les valeurs)

— l'instance effective, qui sera déduite soit de l'ensemble des valeurs des boucles, soit, par défaut, de la valeur de la boucle b_1 : si $b_1 = i$ on considérera alors que l'instance effective est $TS(i + 1)$

— le but des élèves, soit l'instance qu'ils cherchent à construire, si ce but nous parait identifiable ; dans le cas contraire nous indiquerons ce doute par un « ? ».

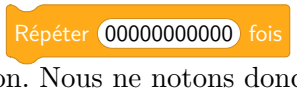
Actions (boucles et expressions) Si l'on omet les modifications de structure des scripts, les actions des élèves dans le cadre de notre situation sont décrites par l'ensemble des modifications effectuées sur les représentations du nombre de répétitions des boucles, ce que nous représentons par l'ensemble $\mathcal{A}$. Comment construisons-nous cet ensemble ? Comme nous nous intéressons au fait défini comme relation action-effet (fait second), éventuellement complété par un médiateur (fait tiers), les actions des élèves à un moment donné seront définies comme étant *l'ensemble des modifications consécutives des boucles d'un même script, pour un même but.* Par « modifications consécutives des boucles », nous entendons des actions de programmation qui ne concernent que les boucles. Ainsi, les actions sur les boucles se terminent lorsque les élèves lancent un test, changent de but, ou effectuent une sauvegarde ou un chargement. Pour un même but, l'ensemble des actions est donc constitué de l'ensemble des modifications des boucles entre deux tests, ou entre la première modification liée au but et un test. Dans la transcription de l'activité des élèves, nous préciserons d'une part les intitulés des boucles de $\mathcal{A}$, d'autre part les expressions définies par les élèves pour ces boucles. Ainsi, si $\mathcal{A} = \{b_1 \leftarrow$ (15) - (1)$, b_2 \leftarrow$ (15)$\}$, nous indiquerons :

— « b1b2 » comme représentation de $\mathbf{b}(\mathcal{A})$

— « 15-1 /15 » comme représentation des expressions entrées.

Précisons que la représentation des intitulés des boucles suit l'ordre d'intervention des élèves sur les boucles. « b1b2 » signifie que les élèves ont modifié b_1 *puis* b_2. Pour des raisons de concision, la modification par les élèves de l'ensemble des boucles du script sera représentée par « Ba », tandis que la modification de l'ensemble des boucles excepté la boucle b_7 sera notée « Be » [44]. Dans ces deux cas, nous utiliserons la convention de simplification par famille de boucles (précisée p. 164 et dans le tableau 2.1, p. 170)

44. Mélanges franglais de « B » pour *boucles* et « a » pour *all* ou « e » pour *except.*

Les expressions des boucles sont les transcriptions textuelles des instructions utilisées, c'est-à-dire la suite des caractères entrés par les élèves dans l'EPGB pour décrire le nombre de répétitions d'une boucle. Par exemple, si les élèves ont modifié b_1 ainsi : 

alors nous noterons « 00000000000 » comme représentation de l'expression. Nous ne notons donc pas « 0 », qui est le dénoté ou la valeur évaluée à l'exécution, de cette expression. Un double crochet « [] » notera un paramètre d'une instruction laissé vide : $b_1 =$ 

sera noté $b_1 = [] - 1$.

Faits Seconds

Comme signalé p.192, les faits seconds qui nous intéressent dans notre situation sont essentiellement des faits de validation ou d'invalidation. Ces faits sont des mises en relation de faits premiers.

Validité absolue des actions et boucles erronées Nous préciserons si les modifications $\mathcal{A}$ faites par les élèves sur un script S aboutissent à un script valide considérant le but p identifié, c'est-à-dire si $S \odot \mathcal{A} = TS(p)$. Si ce n'est pas le cas, nous indiquerons les intitulés des boucles erronées ($\mathbf{b}(E^p(S \odot \mathcal{A}))$). Il s'agit ici de mettre en relation les actions des élèves et la validité du script en fonction du but identifié, « indépendamment de toute autre chose » : nous ne présumons pas ici que les élèves ont construit ou non le fait « nos actions sont valides considérant notre but p ».

Prise en compte de la rétroaction Si les actions des élèves s'appliquent sur un script ayant des erreurs, nous préciserons si toutes les erreurs sont prises en compte ou non, nonobstant la validité des modifications opérées. Il s'agit ici de considérer la prise en compte des erreurs, c'est-à-dire notamment de la rétroaction. Nous dirons que la prise en compte de la rétroaction est *totale* si toutes les boucles erronées sont traitées ($\mathbf{b}(\mathcal{A}) = \mathbf{b}(E^p(S))$), indépendamment de leur validité après modification. Nous dirons que cette prise en compte de la rétroaction est *partielle* si au moins une boucle erronée n'est pas traitée ($\exists b_i \in \mathbf{b}(E^p(S)) | b_i \notin \mathbf{b}(\mathcal{A})$). La localisation des boucles erronées n'étant pas toujours aisée, nous préciserons, le cas échéant, si les boucles erronées identifiées par les élèves sont *avant* ou *après* les boucles erronées identifiées par le chercheur. Une modification de boucle b_i dont la prise en compte de la rétroaction est définie comme *avant* signifie que b_i n'est pas une boucle erronée de S, mais que b_{i+1} l'est. De même, nous noterons *après* pour b_i si b_i n'est pas erronée mais que b_{i-1} l'est. Cela nous semble être une indication au moins de la prise en compte de la rétroaction par les élèves, avec une localisation de l'erreur inexacte mais proche. Si rien n'est indiqué pour cette propriété, cela signifie qu'aucune boucle erronée n'a été traitée, ce qui pourrait être une indication soit de la localisation erronée des erreurs, soit d'une non prise en compte de la rétroaction, et donc d'une démarche qui s'apparenterait plutôt à la démarche essai-erreur.

Validité relative des actions La validité des actions opérées sur des boucles sera, elle aussi, précisée. Ces actions, que l'on peut qualifier de corrections, pourront être totalement ou partiellement valides. Cette validité relative sera notée *totale* si toutes les boucles corrigées sont valides ($\forall expr \in S \odot \mathcal{A}, expr \in TS(p)$). Elle sera notée *partielle* si elle n'est pas totale et qu'au moins une boucle est correctement corrigée ($\exists expr \in S \odot \mathcal{A}, expr \in TS(p)$).

Faits Tiers : Tea et adaptation/ajustement

Tous les faits évoqués plus haut sont soit des faits pour l'EPGB, soit des faits pour le chercheur. Dans cette partie, nous détaillerons comment nous mobilisons ces faits pour tenter d'établir les faits tiers (voir Faits tiers p. 193) construits par les élèves, ces faits tiers étant essentiels afin de pouvoir construire les espaces des contraintes (Orange, 2005).

ART Pour reconstituer les faits TEA, et donc en particulier le TEA organisant l'activité de l'élève pour une tâche donnée, nous nous baserons sur les éléments suivants :

- le but identifié des élèves
- les liens identifiables entre les actions de modifications et le but
- la comparaison des actions de modification pour ce but avec celles des buts précédents et suivants
- la comparaison des ajustements opérés pour ce but, les précédents et les suivants

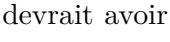
En effet, si on considère que le TEA s'applique à une famille de situations, des situations similaires devraient mobiliser un TEA semblable. Ainsi, si un TEA détermine les actions des élèves pour passer de l'instance d'un TS4 à un TS5, il est probable que ce même TEA soit à l'œuvre pour passer de l'instance d'un TS5 à un TS6 : il s'agit d'une même famille de situations « passer d'une instance à la suivante ». Dans le cas d'une situation de généralisation de motifs les TEA sont les relations permettant de construire les différentes instances en fonction d'éléments déjà connus.

Dans notre situation, de façon générale, pour la construction d'une certaine instance $TS(p)$, il faudra chercher à mettre en relation les valeurs des boucles du script S construit, soit avec l'instance p visée, soit avec les autres valeurs des boucles de S, soit avec les autres valeurs des boucles des scripts précédemment traités, ou éventuellement en combinant ces relations.

Lorsque le but est « construire le script $TS(p+1)$ à partir de $TS(p)$ », nous considérerons en priorité les liens entre les boucles de S et celles de $TS(p)$. La construction du script se basant sur une construction précédente, il est probable que le TEA mette en relation b_i^{p+1} et b_i^p.

Lorsque le but est « construire le script $TS(p+k)$ (ou de façon plus générale $TS(f(p))$) à partir de $TS(p)$ », nous considérerons en priorité les liens entre les boucles de S et celles de $TS(p)$ combinées avec k ou p (ou f). La construction du script se basant sur une construction précédente avec une augmentation de la mesure de k, il est probable que le TEA mette en relation b_i^{p+k}, b_i^p et k (ou $f(p)$).

Lorsque le but est de construire une instance $TS(p)$ précise, indépendamment des précédentes, nous considérerons en priorité les liens que l'on peut établir entre b_i^p et p. Nous serons aussi attentifs à la possibilité d'un lien entre boucles du même script (b_i^p et b_j^p) : par exemple pour tenter de déterminer si $b_7^p = 2 \times p - 2$ ou si $b_7^p = 2 \times b_6^p$. Dans le doute, nous favoriserons l'hypothèse d'un lien avec p, considérant que l'activité implique plutôt un traitement trans-objectal plutôt que intra-objectal (Piaget et García, 1983) de part sa généralité.

Lorsque le but est de construire une instance $TS(p)$ précise avec p comme nombre à caractère générique, nous chercherons à identifier le lien entre b_i^p et p, et surtout à décrire le type de généralisation qui semble être en vigueur dans l'expression de b_i^p. Nous détaillerons ces types de généralisation plus loin. Enfin, lorsque le but est de construire le script générique $TS(n)$ où n désigne la valeur entrée par l'utilisateur, nous chercherons à identifier les liens que font les élèves entre b_i, le signe s utilisé par les élèves pour représenter n dans Snap !, et son dénoté $\delta(s)$. Pour être valide, on devrait avoir non seulement $\delta(s) = n$, mais aussi $s =$ 

.

ARL et ARA Un fait TEA (étape i), lorsque le résultat est erroné, est normalement suivi (pas forcément immédiatement, étape $i+k$) par la construction d'un fait de localisation et d'un fait d'ajustement [45]. En effet, à la suite d'actions visant un but directement lié au problème, si la rétroaction invalide le script modifié, les élèves doivent localiser et identifier l'erreur, puis ajuster leur script en conséquence. Seules les actions de l'étape i et les ré-actions d'ajustement à l'étape $i+k$ nous sont accessibles par les traces de programmation. Les actions de l'étape $i+k$, mises en relation avec les erreurs de l'étape i, nous permettent d'identifier le fait de localisation construit par les élèves : les boucles modifiées à l'étape $i+k$ sont les boucles (ou une partie des boucles) identifiées comme source de l'erreur par les élèves. Leur modification est une indication de l'ajustement, donc de l'interprétation de l'erreur. Ainsi par exemple, si à la suite des actions de l'étape i, $b_1 = 5$ et si à l'étape $i+k$, $\mathcal{A} = \{b_1 \leftarrow 6\}$, on fera l'hypothèse que les élèves ont identifié la boucle b_1 comme étant source de l'erreur et qu'elle est trop courte de 1 (fait de localisation). Le fait d'ajustement sera ici « on augmente de 1 le nombre de répétitions de la boucle b_1 ». Il sera noté « ajust.+1 » dans les transcriptions. S'il parait raisonnable de penser que l'ajustement est rendu nécessaire non par le caractère erroné du TEA mais par une erreur de manipulation, nous préciserons qu'il s'agit d'une « correction ». Par raisonnable, nous entendons étayé par les actions précédentes et/ou suivantes. Prenons un exemple classique : les élèves mettent en œuvre le TEA $\{+1/+1/+2\}$ sur plusieurs instances consécutives (TS5, TS6, TS7), lors de la construction de l'instance suivante (TS8) ils rajoutent 1 à chaque boucle y compris b_7, puis augmentent $b7$ de 1. On peut raisonnablement penser que le TEA qui organise l'activité pour TS8 est toujours $\{+1/+1/+2\}$ mais qu'en modifiant à la volée l'ensemble des boucles, ils aient entré une valeur non désirée pour b_7. L'ajustement suivant est donc plutôt une correction, et non un ajustement du script compensant l'invalidité du TEA. On ne peut cependant pas écarter définitivement la possibilité que le TEA $\{+1/+1/+1\}$ soit à l'œuvre en TS8 : les actions suivantes permettront de renforcer la probabilité que l'un ou l'autre de ces TEA soit mobilisé.

Type de généralisation

Un aspect essentiel pour l'analyse de notre situation consiste à tenter d'établir le type de généralisation mis en œuvre dans l'activité, et à représenter les TEA en fonction de ces types de généralisation.

Radford (2008) identifie plusieurs *niveaux* de généralisation :

— la généralisation arithmétique : comment passer d'une instance à une autre ;

— la généralisation « naïve » (ou « induction naïve ») : trouver une certaine instance, mais sans formaliser la méthode ; donc *faire*, mais sans expliciter comment *faire faire* ;

— la généralisation algébrique : en formalisant la méthode pour n'importe quelle instance, avec une symbolisation ; dénotation et analycité sont alors présents.

Zazkis et Liljedahl (2002b) y rajoutent un intermédiaire, la « généralisation disjonctive », soit un traitement particulier pour chacun des cas, et non une procédure s'appliquant à tous les cas. Cette généralisation disjonctive implique donc que plusieurs TEA sont à l'œuvre pour une même famille de situations.

45. Dans le cas où les élèves modifient un script et ne l'exécutent pas, ou lorsqu'un script erroné est exécuté mais que les élèves ne modifient pas ce script, on peut imaginer qu'un ARL est construit, mais il est impossible d'y avoir accès sans les interactions langagières.

De notre côté, nous nous intéressons au passage de la généralisation arithmétique à la généralisation algébrique, puisque c'est ce passage qui va nécessiter la construction du concept de variable dans un rôle de paramètre. Nous avons donc besoin d'identifier les *manifestations d'éléments tendant vers l'analycité et la dénotation*. Nous nommons ces manifestations « type de généralisation », en utilisant « type » dans le sens « Ensemble des caractères distinctifs (choisis d'après des critères divers) de certains groupes d'objets, d'individus, permettant leur classification » [46].

Nous gardons comme premiers types les niveaux de généralisation de Radford (2008) (voir 2.1.1, p. 2.1.1) que nous codons ainsi :

— *GA* pour la généralisation arithmétique. Exemple : les élèves connaissant $b_1^4 = 3$ effectuent l'action $b_1^5 \leftarrow 4$.
— *GAN* pour la généralisation arithmétique naïve. Exemple : les élèves construisent « directement » l'instance du TS(22) sans prendre en considération les boucles d'autres instances créées.

Afin de déterminer les autres types, nous cherchons à observer comment les élèves passent d'un raisonnement arithmétique à une formulation algébrique (ou algorithmique) : comment le $b_1 = 11$ de l'instance 12 devient $b_1 = mesure - 1$ pour la *mesure* entrée par l'utilisateur. Ces types correspondent à des éléments se situant à la frontière de la généralisation algébrique : pour construire un script générique, les élèves doivent formaliser les généralisations arithmétique ou naïve qu'ils mettent en œuvre. Si les élèves appliquent la GAN « pour b_1 on fait moins un », comment vont-ils exprimer de façon algorithmique cette procédure plus ou moins implicite ? L'analyse des traces de programmation fait apparaître les « caractères distinctifs » suivants :

— *NG* pour le déjà bien connu *nombre à caractère générique* : $b_1 = 12 - 1$.
— *PNG*, moins générique, qui désigne la *perte de généricité du nombre* : de $b_1 = 12 - 1$ pour l'instance 12 — le 12 est ici sans doute un NG —, on passe à $b_2 = 11 - 1$. Du point de vue du résultat, ces boucles sont valides pour l'instance 12. Cependant, le « 11 » n'a plus le caractère d'exemple porté par le « 12 » sur cette instance ; c'est un obstacle vers la généralisation.
— *NPG*, pour un « nombre potentiellement générique », désignant l'utilisation d'un nombre portant une possible généricité, un nombre qui *pourrait* représenter n'importe quel nombre (et pas spécifiquement une instance) : par exemple « 18754555444799999963 » (46k, étape 21) qui pourrait s'interpréter comme « un nombre quelconque, même aussi improbable que celui-là ». Nous considérons aussi que 10, 100 ou 1000 sont possiblement génériques (46a, 46k, 46b) : ce sont des nombres particuliers qui ont déjà une généricité implicite (dizaine, centaine).
— *NSG*, un *nombre-signe générique*, le signe d'un nombre particulier : nous retenons notamment les éléments neutres et absorbants « 1 » et « 0 ». Ainsi le groupe 46e évoque « il faut un multiple de tout ça [...] un truc qui finit par un zéro [...] parce que zéro c'est un multiple de tout » (d'où l'idée aussi de traiter séparément les puissances de dix). « 0 » reste un signe-nombre particulier, mais doté d'une propriété générique.
— *SG*, le signe générique, donc un symbole autre qu'un nombre et représentant un nombre quelconque, ou/et un marque-place.
— *param*, le paramètre tant attendu, cette fois désignant une variable dans le langage du programme, sur lequel ou peut opérer comme si le nombre était connu.

Il s'agit en fait du système de signes mobilisé par les élèves pour formaliser leurs procédures de façon algorithmique, et plus spécifiquement de façon algébrique pour les paramètres des boucles.

46. Cnrtl https://www.cnrtl.fr/definition/TYPE

Relations et formalisation des TEA Nous appelons *relation* entre deux valeurs, la procédure permettant de passer d'une de ces valeurs à la seconde. De façon plus spécifique, lors de la phase de généralisation, les élèves doivent établir et formaliser une relation entre le nombre d'hexagones du triangle de base (ou la valeur entrée dénotant ce nombre) et le nombre de répétitions d'une boucle particulière. Nous formaliserons cette relation en utilisant une combinaison d'opérateurs, de nombres, et de caractères distinctifs de la généralisation. Cette procédure formalisée est une représentation simplifiée du TEA, et nous identifierons cette relation au TEA qu'elle représente. Par exemple si les élèves font $b_1 \leftarrow 15 - 1$ pour l'instance 15, le 15 a un rôle de nombre générique (NG). La relation sera ainsi notée « NG - 1 ». Le TEA qu'elle représente pourrait s'exprimer ainsi : « pour un certain nombre 15, le nombre de répétitions de b_1 est ce nombre 15 auquel on enlève 1 ».

Types de généralisation Le *type de généralisation* est constitué de l'ensemble des caractères distinctifs de la généralisation auquel nous rajoutons si nécessaire le caractère distinctif « R » indiquant que les élèves mettent en relation des éléments de la situation. À la suite des indications sur le système de signe, nous préciserons entre parenthèse les signes utilisés. De même, si une relation est établie, sa représentation R sera suivie d'une indication permettant de préciser si cette relation a changé par rapport aux précédents théorèmes (1) , ou si seuls les opérandes de cette relation ont changé(0). La distinction entre types de généralisation ne prend bien sûr pas en compte les signes spécifiques ou l'indication d'un changement de relation : « NG(15) R(1) » est du même type que « NG(12) R(0) ». Ce codage de la généralisation nous permet de rassembler tous les *NG R* sans perte d'information sur les valeurs utilisées.

Prenons comme illustration le groupe 46i, étapes 35 à 39 (tableau 2.2,p. 206).

Étape	Script	Ins.Util.	Méthode	Entrée	Action	Théorème	Type généralisation
35	n	15	b_1	15-1		NG-1	NG(15) R(1)
36	n	15	b_2	15-1		NG-1	NG(15) R(0)
37	n	15	$b_3b_4b_5b_6$	15-1		NG-1	NG(15) R(0)
38	n	15	b_7b_8	15-1		NG-1	NG(15) R(0)
39	n	15	b_7	30-1	Ajust. *2	[2*NG]-1	PNG(30=15*2) R(0V)

Tableau 2.2 – Exemple du groupe 46i

À l'étape 35, $b_1 = 15 - 1$ pour une entrée utilisateur de 15 indique l'utilisation de 15 comme nombre générique *NG*. Le théorème est donc « pour un certain nombre 15, le nombre de répétitions de b_1 est ce nombre 15 auquel on enlève 1 », codé *NG-1*. C'est une nouvelle relation ($\square - \square$), le codage définissant le type généralisation à l'œuvre sera donc *NG(15) R(1)*. Pour les étapes 36, 37 et 38, on garde la même relation, donc *R(0)*. Pour l'étape 39, l'entrée $b_7 = 30 - 1$ indique une perte de généricité (le 15 devient 30), pour une relation dont seul change un opérande. Le théorème sera noté *[2*NG]-1* (les crochets indiquant que le contenu n'est pas littéral mais calculé), et on notera la perte de généricité de 15 au profit de 30 *PNG(30=15*2)* ainsi que le seul changement d'un opérande de la relation *R(0V)*.

Exemple illustratif

Nous allons ici illustrer les éléments précédents à partir d'un exemple, le début de la séance 3 du groupe 46a. Dans la figure 2.24 (p. 207), nous avons un exemple partiel des premières transcriptions obtenues à partir des actions de programmation. Il s'agit ici d'une sélection d'actions considérées comme pertinentes dans la situation, et mises en forme (et en ordre) automatiquement. La deuxième colonne indique le « type d'action » (chargement, exécution de scripts, modification de valeurs, modification d'une instruction contenant une variable...). La

46_a, le 11/01/2019 de 14:38:06 à 14:52:05

session: beiqxw45qpai9xf9sydqvc080fio8iof

session	type	scriptName	ScriptId	time	start	event	image	compteur	mesure
b..f	*LOAD/SAVE*			14:38:06	0:00.000				
b..f	*LOAD/SAVE*			14:38:18	0:12.445	LOBA-sierpinski-programme2-v2019			
b..f	*STRUCTURE*			14:38:33	0:26.864	DUPLIC_391-452(391)			
b..f	*VARIABLE*	Quand b est pressé(481)	481	14:38:36	0:30.107	(I3) compteur prend la valeur 0 [drop loc:bottom duplic 26864]			
b..f	*EXEC*			14:38:51	0:45.279	KEY_b			
b..f	*EXECUTION*	receiveKey(481)	481	14:38:51	0:45.302	START receiveKey(481)			
b..f	*EXECUTION*	receiveKey(481)	481	14:39:04	0:58.177	FIN receiveKey(481)			
b..f	*SNP*		481	14:39:04	0:58.188	SNP FIN481		24	5
b..f	*EXECUTION*	receiveKey(481)	481	14:39:36	1:29.831	START receiveKey(481)			
b..f	*AFFICHAGE*			14:39:47	1:40.991	AFFBL_variables			
b..f	*EXECUTION*	receiveKey(481)	481	14:39:49	1:42.948	FIN receiveKey(481)			
b..f	*SNP*		481	14:39:49	1:42.961	SNP FIN481		24	5
b..f	*VALEURS*	Quand b est pressé(481)	481	14:42:34	4:28.508	(BOUCLE)b1i0(VAL): répéter *4* fois (commandes)<<3>>			
b..f	*VALEURS*	Quand b est pressé(481)	481	14:42:37	4:30.634	(BOUCLE)b2i0(VAL): répéter *3* fois (commandes)<<2>>			
b..f	*VALEURS*	Quand b est pressé(481)	481	14:42:37	4:31.523	(BOUCLE)b3i0(VAL): répéter *3* fois (commandes)<<2>>			
b..f	*VALEURS*	Quand b est pressé(481)	481	14:42:39	4:33.130	(BOUCLE)b4i0(VAL): répéter *3* fois (commandes)<<2>>			
b..f	*VALEURS*	Quand b est pressé(481)	481	14:42:41	4:35.130	(BOUCLE)b5i0(VAL): répéter *3* fois (commandes)<<2>>			
b..f	*VALEURS*	Quand b est pressé(481)	481	14:42:43	4:37.418	(BOUCLE)b6i0(VAL): répéter *4* fois (commandes)<<3>>			
b..f	*VALEURS*	Quand b est pressé(481)	481	14:42:47	4:40.889	(BOUCLE)b7i0(VAL): répéter *7* fois (commandes)<<6>>			
b..f	*VALEURS*	Quand b est pressé(481)	481	14:42:56	4:50.466	(BOUCLE)b8i0(VAL): répéter *3* fois (commandes)<<2>>			
b..f	*EXEC*			14:42:57	4:50.627	GREEN_button			
b..f	*EXEC*			14:42:57	4:51.487	GREEN_button			
b..f	*EXEC*			14:42:59	4:52.759	STOP_button(all)			
b..f	*SNP*			14:42:59	4:52.776	SNP STOPbutton		24	5
b..f	*EXEC*			14:43:00	4:54.278	KEY_b			
b..f	*EXECUTION*	receiveKey(481)	481	14:43:00	4:54.287	START receiveKey(481)			
b..f	*EXEC*			14:43:00	4:54.308	KEY_h			
b..f	*EXEC*			14:43:00	4:54.373	KEY_space			
b..f	*EXECUTION*	receiveKey(481)	481	14:43:17	5:11.433	FIN receiveKey(481)			
b..f	*SNP*		481	14:43:17	5:11.444	SNP FIN481		32	5
b..f	*VALEURS*	Quand b est pressé(481)	481	14:43:31	5:25.160	(BOUCLE)b8i0(VAL): répéter *4* fois (commandes)<<3>>			
b..f	*EXEC*			14:43:31	5:25.304	STOP_button(all)			
b..f	*SNP*			14:43:31	5:25.324	SNP STOPbutton		32	5
b..f	*EXEC*			14:43:33	5:26.599	KEY_b			
b..f	*EXECUTION*	receiveKey(481)	481	14:43:33	5:26.607	START receiveKey(481)			
b..f	*EXEC*			14:43:33	5:27.046	KEY_b			
b..f	*EXECUTION*	receiveKey(481)	481	14:43:33	5:27.048	STOP receiveKey(481)			
b..f	*SNP*			14:43:33	5:27.060	SNP STOP			5
b..f	*EXECUTION*	receiveKey(481)	481	14:43:33	5:27.073	START receiveKey(481)			
b..f	*EXECUTION*	receiveKey(481)	481	14:43:33	5:27.127	STOP receiveKey(481)			
b..f	*EXEC*			14:43:33	5:27.127	KEY_b			
b..f	*SNP*			14:43:33	5:27.136	SNP STOP			5
b..f	*EXECUTION*	receiveKey(481)	481	14:43:33	5:27.141	START receiveKey(481)			
b..f	*EXECUTION*	receiveKey(481)	481	14:43:33	5:27.154	STOP receiveKey(481)			
b..f	*EXEC*			14:43:33	5:27.154	KEY_b			

Figure 2.24 – Exemple de transcription des actions de programmation (46a, S3)

troisième précise le bloc de tête du script concerné par l'action : le nom du bloc, qui généralement correspond à l'instruction, ainsi qu'un numéro identifiant unique, numéro noté également dans la colonne 4 (« scriptId ») Les cinquième et sixième colonnes (« time » et « start ») précisent le moment du déclenchement de l'action, en horaire UTC puis en millisecondes depuis le lancement de l'EPGB [47] Pour référencer une action de programmation, nous utiliserons le temps de la colonne « start » (en bleu). La septième colonne (« event ») précise l'événement. Les événements concernant les modifications de valeurs dans une instruction sont rendues d'avantage visible, car elles sont les éléments essentiels de l'activité de généralisation demandée aux élèves. Lorsqu'une instruction voit un de ses paramètres modifié, la nouvelle valeur est précisée entre deux astérisques, l'ancienne étant encadrée par les caractères « « » et « » ». Les modifications de valeurs précisent aussi le numéro de l'instruction, ou, s'il s'agit d'une modification du corps d'une boucle, le numéro de la boucle ainsi que le numéro d'ordre de cette instruction dans ce corps de boucle. Par exemple en 4'28, *(BOUCLE)b1i0(VAL) : répéter *4* fois (commandes)«3»* signifie :

— qu'il s'agit d'une modification touchant une boucle,

— qu'il s'agit de la boucle b_1,

— qu'il s'agit du nombre de répétitions de la boucle ($i0$),

— que la valeur « 3 » a été remplacée par « 4 ».

Ainsi, cela décrit l'action $b_1 \leftarrow 4$. Les trois dernières colonnes rendent compte de la rétroaction en fin d'exécution : le résultat affiché, ainsi que la valeur des variables compteur (si le résultat est énoncé par le lutin) et mesure.
Ainsi en 0'12, (ou la deuxième action sélectionnée), les élèves chargent le programme de base de la séance 2 (*LOBA*). Ils dupliquent ensuite une partie du script fourni (*duplic*), à partir de la troisième instruction (*i3*) qui est une instruction concernant une variable(

). Ce script est déplacé sous l'instruction de tête de script 

, cette instruction étant identifiée par le numéro 481. Par construction, on sait que ce script est celui dont le commentaire fixe la tâche « Modifier le script pour un TS5 ». Durant la deuxième exécution, les élèves affichent la famille de blocs correspondant aux variables (*AFFBL_variables*, 1'40).
Les huit événements suivants correspondent à deux exécutions successives du script ainsi modifié [48] : *START receiveKey(481)* désigne le lancement du script dont le bloc de tête est précisé (481). Ce script a été exécuté suite à l'appui sur la touche « b » (*KEY_b*, 0'45). *FIN* indique une terminaison normale du script, c'es-à-dire que le script s'est exécuté jusqu'au bout sans soucis ni intervention. Le *STOP* que l'on trouve par exemple en 5'25, indique un arrêt manuel du programme par les élèves. Le numéro d'identifiant unique permet de différencier les scripts. Nous rappelons que chaque lancement ou arrêt de script donne lieu à une capture de l'écran d'exécution, rétroaction indiquée par le type *SNP* (voir figure 2.25, p. 209).
Cette première étape permet ensuite d'assembler les différentes actions. Pour les événements de 4'28 à 6'12, les élèves modifient le script $TS(4)$ pour tracer et dénombrer un TS5 ; ils cherchent donc à passer d'une instance à la suivante. On peut déterminer les étapes suivantes :

1. Suite de modification de l'ensemble des boucles b_1 à b_8, en ajoutant 1 à chaque valeur par rapport au script précédent. (4'28-4'50)

47. Plus précisément, le début de l'action pour l'ensemble des actions excepté les arrêts de scripts, pour lesquels ce temps correspond au moment de l'action auquel on ôte 500ms. Ces 500ms correspondent à une durée par défaut de l'action, qu'il est nécessaire de définir pour l'intégration aux transcriptions des interactions langagières.
48. Notons que pour les exécutions, c'est le nom anglais du bloc qui est utilisé.

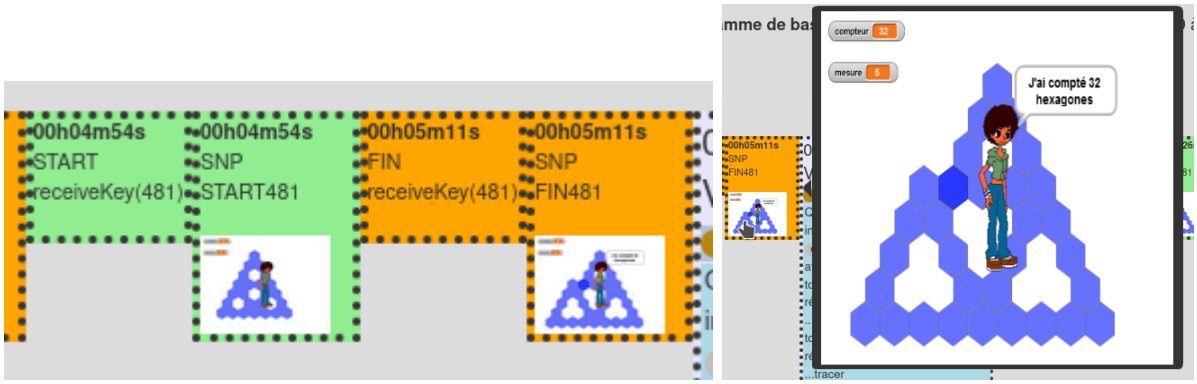


Figure 2.25 – Captures d'écran du rendu de l'histoire du programme, 46a S3 lignes 16 et 17 (sans et avec le « popup » permettant de mieux visualiser la capture)

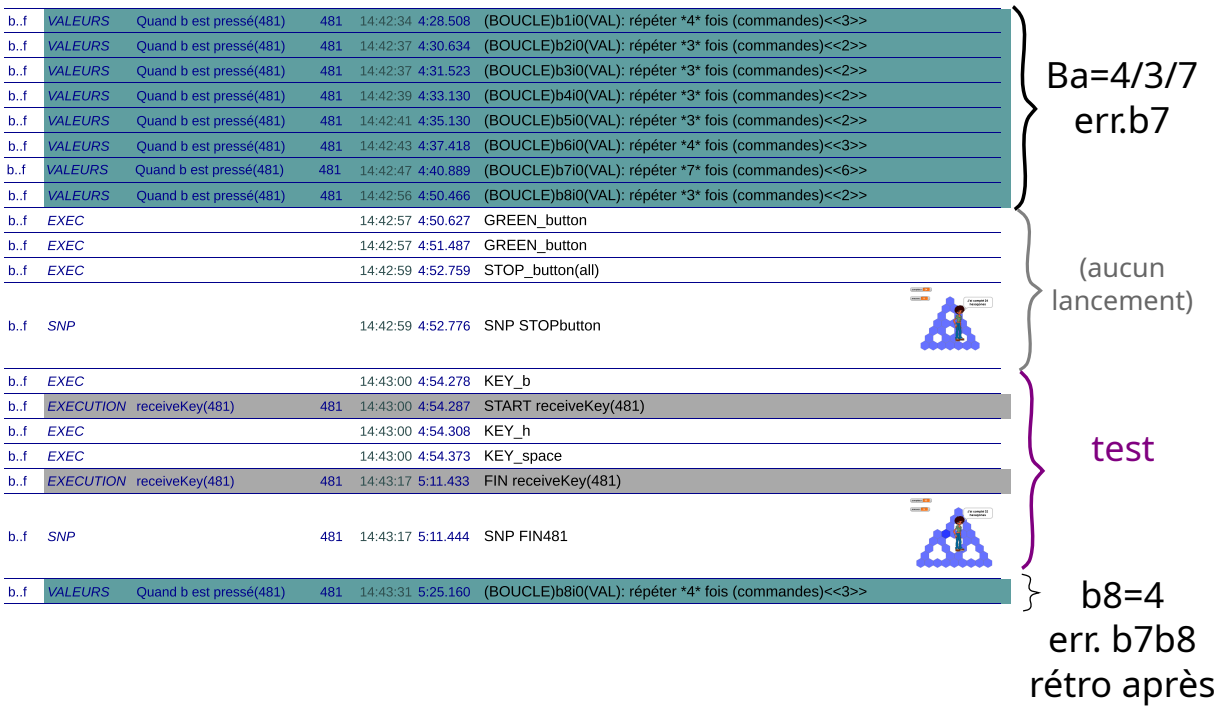


Figure 2.26 – Première étape de la transcription : sélection et organisation des événements liés à la modification des boucles

2. Test du script obtenu (voir la rétroaction figure 2.25, p. 209) : produit la donnée* « script erroné en b_7 » (4'52)
3. Ajustement de la boucle b_8 : $b_8 \leftarrow b_8 + 1$: les élèves modifient ici la boucle située « après » la boucle erronée (5'25)
4. Nouveau test du script, avec plusieurs arrêts manuels : produit la donnée* « script erroné en b_7 et b_8 » (5'44)
5. Modifications $b_8 \leftarrow b_8 - 1$ et $b_7 \leftarrow b_7 + 1$ (5'51-5'52)
6. Nouveau test, avec rétroaction validante (6'12).

Nous avons donc ici trois étapes Action-Rétroaction. L'étape 1 sera notée ainsi :

méthode : Ba (toutes les boucles sont modifiées)

entrée : 4 /3 /7 ($B_1 \leftarrow 4$, $B_2 \leftarrow 3$, $B_7 \leftarrow 7$)

théorème : b(i+1)=+1 /+1 /+1 (« pour passer d'une instance à la suivante, on ajoute un à chaque boucle »)

type : GA (généralisation arithmétique)

erreurs : b7

Pour l'étape 2 :

méthode : b8 (seule la boucle b_8 est modifiée)

entrée : 4 ($b_8 \leftarrow 4$)

Action : Ajust.+1 (C'est un ajustement par incrémentation de la valeur du nombre de répétitions de la boucle b_8)

erreurs : b7b8

R.interprétée : après (et non valide)

Cela nous donne une deuxième transcription des actions de programmation, dont celle correspondant à l'exemple traité ici est représentée figure 2.27 (p. 210). Cette deuxième transcription nous

groupe	étape		script_attendu	instance	instance_effective	instance_utilisateur	méthode	entrée	action	théorème	type_généralisation	action_valide	erreurs	erreur compteur	rétroaction_interprétée	correction_adap
46_a	1	5	5	5			Ba	4/3/7		b(i+1)=+1/+1/+1	GA		b7			
46_a	2	5	5	5			b8	4	Ajust.+1				b7b8		après	
46_a	3	5	5	5			b8b7	3/8	Ajust.-1/+1			V			totale	totale
46_a	4	6	6	6			Ba	5/4/9		b(i+1)=+1/+1/+1	GA		b7			
46_a	5	6	6	6			b7	8	Ajust.-1				b7		totale	
46_a	6	6	6	6			b7	10	Ajust.+2			V			totale	totale
46_a	7	7	7	7			Ba	6/5/12		b(i+1)=+1/+1/+2	GA	V				
46_a	8	11	8	8			Ba	7/6/14		b(i+1)=+1/+1/+2	GA	V				
46_a	9								LOAD							

Figure 2.27 – Deuxième étape de la transcription : épisodes action-réaction

permet d'établir les faits premiers, seconds et tiers qui seront utilisés ensuite pour construire différentes représentations des enchainements problématiques (exploration de la généralisation algébrique, factualisation, et espaces des contraintes).

2.5.2 Construction des représentations de la factualisation

Les transcriptions faites vont servir de support pour construire deux représentations de l'enchainement des faits construits par les élèves : l'une focalisée sur l'enchainement des TEA (ce qui pour nous est une description de l'activité des élèves), l'autre sur le détail des faits construits et comment ils contribuent à faire bouger les TEA.

Enchainement des TEA

La première représentation est partiellement automatisée en se basant sur la transcription par épisodes actions-test (figure 2.28, p. 214). Seul le placement des différents nœuds est réalisé manuellement, et ce placement, tout comme la distance entre les nœuds, n'a pas de signification particulière. Les nœuds principaux (en bleu clair) sont constitués par les TEA (avec leur dénomination). Chacun de ces nœud-TEA est accompagné du codage du type de généralisation impliqué. En dessous sont représentés par des cercles l'ensemble des étapes (ou épisodes) impliquant ce TEA. Chacun de ces cercles précise les informations suivantes :

— Au centre, l'instance visée et éventuellement l'instance effective si elle est différente (en bleu). La lettre « n » représente le script générique.
— En haut à droite, la valeur entrée par l'utilisateur (pour le script générique).
— En bas à gauche, le numéro de l'étape.
— Le contour code la validité de l'action ainsi que la prise en compte de la rétroaction :
  — Vert pour une action valide, corrigeant s'il y a lieu l'ensemble des erreurs présentes lors de la précédente exécution.
  — Vert pointillé pour une prise en compte partielle des boucles erronées, mais dont toutes les corrections sont valides (par exemple si $b_1b_2b_3$ sont erronées, que seules b_1 et b_2 sont corrigées, mais avec des corrections adéquates).
  — Orangé pour une prise en compte partielle des boucles erronées, et dont une partie seulement des corrections est adéquate (par exemple seule b_1 est correctement corrigée dans l'exemple précédent).
  — Rouge dans les autres cas.

Les étapes consécutives sont reliées par un arc fléché, sur lequel sont disponibles d'autres informations (notamment accessibles sur la représentation numérique) :

— Si, d'une étape à l'autre, la relation établie entre l'instance et le nombre de répétitions d'une boucle est stable, la couleur de l'arc est fixée au vert ; si cette relation change, la couleur est fixée au rouge. S'il n'y pas de relation (par exemple lors de la généralisation arithmétique), l'arc reste en gris.

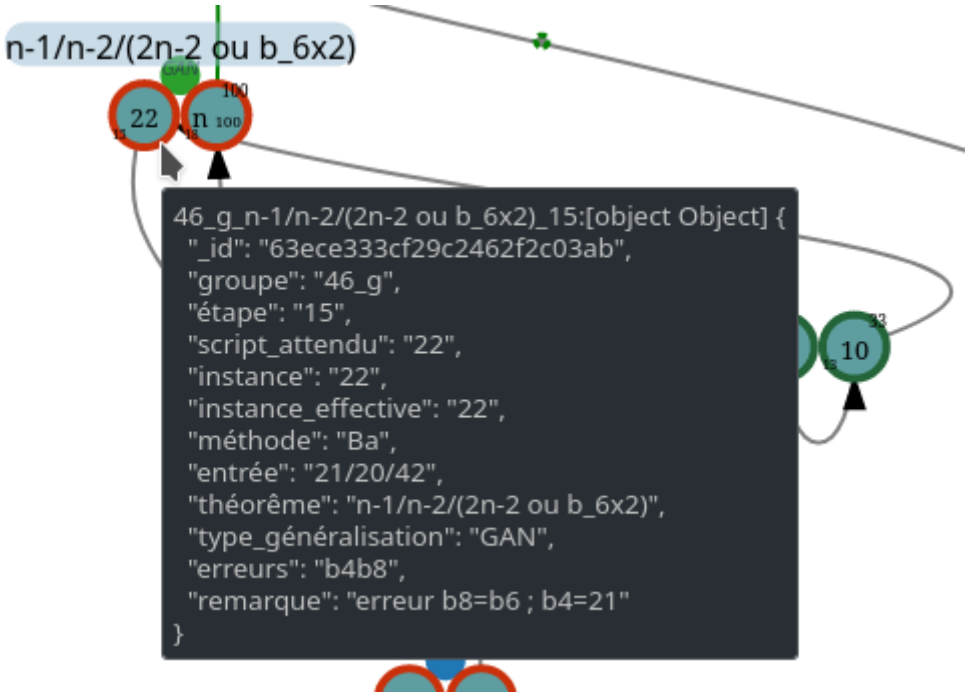


Figure 2.29 – Fenêtre contextuelle surgissant lors du survol d'une étape

Sur la représentation sous forme numérique, des informations complémentaires sont rendues accessibles sous la forme d'une fenêtre contextuelle. Dans la majorité des cas, l'ensemble des propriétés d'une étape est affiché (figure 2.29, p. 211).

Faits et TEA

Les différentes traces à disposition sont utilisées afin de construire une représentation de l'enchainement des faits construits par les élèves, ainsi que les effets sur les TEA (figure 2.30, p. 215). On indique dans cette représentation les faits seconds qui semblent être construits par les élèves. Dans les cartouches sont précisés : le problème des élèves, les actions pour résoudre ce problème, et la validation ou non de ces actions en prenant en comte la rétroaction et le problème posé. Le problème peut consister à chercher à :

— Passer d'un script instancié à un autre, par exemple pour construire le $TS(5)$ à partir du $TS(4)$. Ceci sera codé « TS4 à TS5 ».

— Construire directement une instance, par exemple construire $TS(1789)$ en utilisant seulement la valeur de la mesure. Ceci sera codé « TS1789 »

— Construire le script générique. Cela sera codé par « TSn(p) », où p est la valeur entrée par l'utilisateur. « TSn(12) » indique ainsi que le script générique est testé en entrant une valeur de 12 pour la *mesure*.

Si l'action est validée, le cartouche a un fond vert, sinon le fond reste blanc. Il semble parfois nécessaire d'indiquer les actions des élèves même si elles n'ont pas été suivies d'un test. Dans ce cas, il s'agit d'un fait premier, qui est sur fond gris. Ainsi, dans l'exemple donné figure 2.30 (p. 215) le cartouche TSn: 0-1/0-2/0+9 représente une hypothèse de recherche faite par les élèves, qui éclaire leur action suivante, action qui va aboutir à la construction du fait TSn(5): mesure-1/mesure-2/mesure+9 faux. Les faits sont placés chronologiquement suivant l'axe horizontal puis vertical. Ils sont rassemblés lorsqu'ils ont potentiellement un effet sur un même TEA. Ces effets sur les TEA sont représentés par des flèches. Une flèche noire indique un fait qui confirme en quelque sorte un TEA, c'est-à-dire qu'il étend la validité du TEA à une situation supplémentaire. Une flèche rouge indique une contradiction avec le TEA [49]. La flèche est en pointillés si son « effet » sur un TEA est partiel. Par exemple Tsn(6): b7=mesure+4 vrai ne fait que « contribuer » à la validation du TEA énonçant que $b_7^n = n + n - 2$, puisque dans ce cas particulier, $n = 6$ et donc $b_7^n = n + 4 = n + n - 2 = 10$.

2.5.3 Représentation de l'exploration de la généralisation algébrique

Type de généralisation et schémas

En utilisant le tableau récapitulatif des différentes action-réaction, on construit (automatiquement) la représentation des explorations de la généralisation algébrique (figure 2.32, p. 215). On s'intéresse ici au type de généralisation mis en œuvre et aux signes utilisés. Chaque nœud correspond à un type de généralisation associé à un ou plusieurs signes et éventuellement une relation entre boucle et *mesure* (voir 2.5.1, p. 204). Les arcs reliant les nœuds permettent d'ordonner chronologiquement cette exploration. Cette représentation permet d'illustrer et de comparer l'exploration des possibles généralisations pour les différents groupes, et notamment d'identifier les schémas reconnus par l'élève afin de généraliser le script.

Construction des espaces faits-contraintes

Dans un dernier temps, l'ensemble des résultats produits par ces données nous permettra de d'établir un espace « faits-contraintes » (figure 2.33, p. 215), permettant de voir l'articulation entre les faits et les nécessités construits, qui font avancer la problématisation des élèves dans la construction du concept de variable dans un rôle de paramètre. Cet espace faits-contraintes se différencie des espaces de contraintes traditionnellement représentés dans le CAP par l'absence de mention du REX d'une part, et par une lecture organisée d'autre part.
L'identification du REX dans lequel se trouvent les élèves est une tâche complexe, elle l'est d'autant plus lorsqu'on s'appuie essentiellement sur l'activité des élèves et non sur ce que les élèves disent de leur activité. Dans le texte de l'analyse des données, nous proposerons des hypothèses réalistes, lorsque cela nous semble possible, concernant le REX sur lequel s'appuient les élèves dans un épisode donné, mais le degré d'incertitude nous parait trop important pour mentionner ces REX dans l'espace des contraintes.
D'autre part, nous proposons ici une représentation de l'enchainement des faits et nécessités construits, et non seulement les relations entre données et nécessités. En effet, notre l'analyse s'appuyant sur l'activité de l'élève, nous pouvons établir comment les faits s'enchainent et sur quoi ils s'appuient, mais il est plus incertain de ne localiser que les éléments faisant avancer le problème et leurs relations : cela n'est possible que lorsqu'on dispose des interactions langagières.
L'espace faits-contraintes est ainsi construit en croisant l'ensemble des résultats préalables avec l'espace des contraintes issu de l'analyse *a priori*. Les faits construits sont représentés en gris. Les nécessités du problème, permettant de construire le concept de paramètre, sont en bleu. Les nécessités locales, spécifiques à *ce* problème de généralisation, sont en bleu clair. Certains éléments sont sur un fond blanc : il ne s'agit pas de nécessité concernant directement le paramètre, mais ce ne sont pas non plus des nécessités locales — cela concerne en général des aspects sémiotiques—. Les tensions qui nous semblent se manifester sont représentées en bleu lorsqu'elles concernent les nécessités. S'il y a tension ou incohérence entre des faits, nous le représentons avec un trait plein rouge lorsque cela est manifeste, et avec des tirets rouges lorsque cette tension nous semble susceptible d'avoir été rencontrée par les élèves.

49. Nous parlons de « contradiction » et non d'« invalidation » car, comme l'analyse le montrera, un fait ne validant pas un TEA ne l'invalide pas pour autant : le TEA reste tant qu'il n'est pas remplacé par un autre.

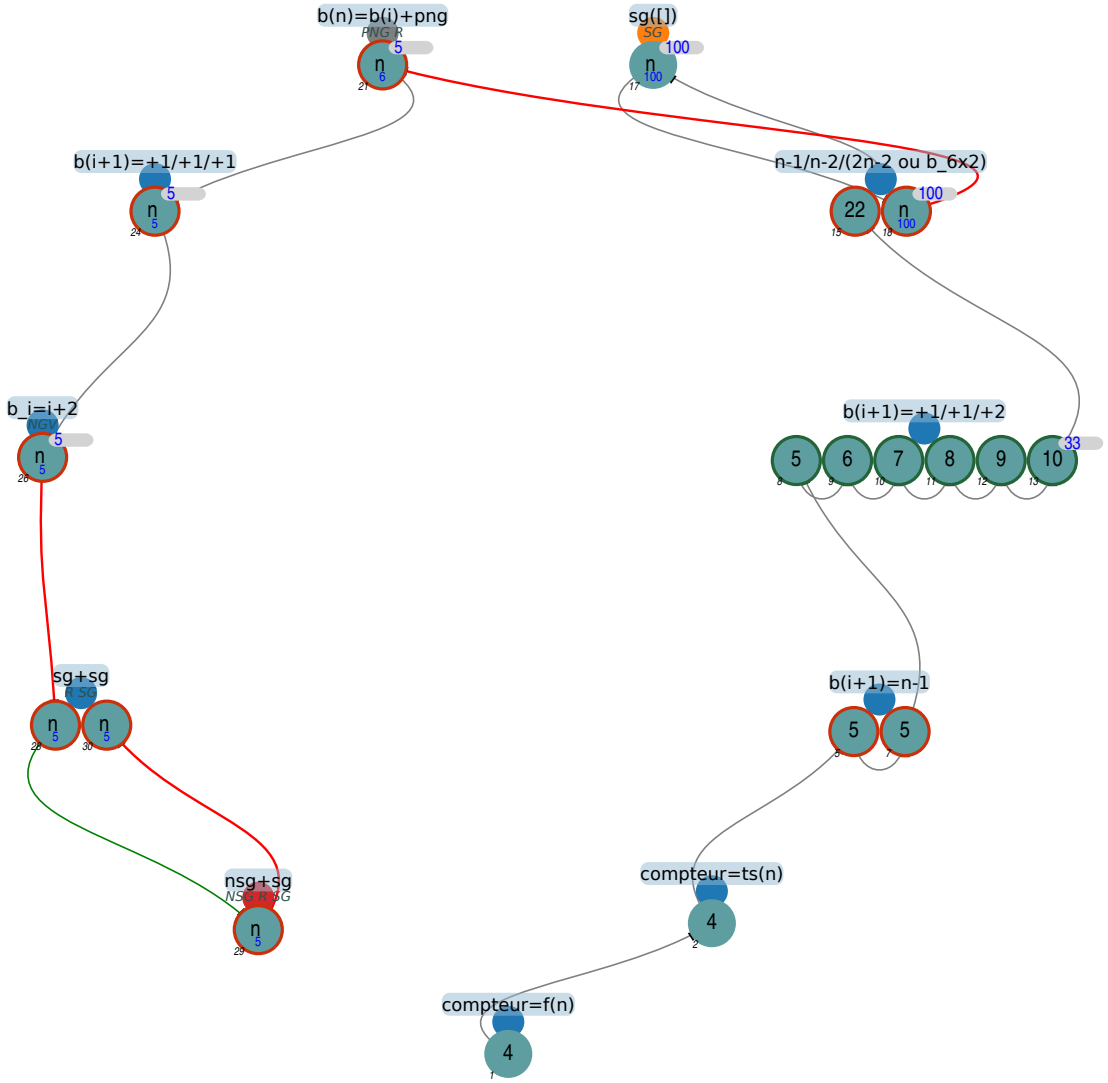


Figure 2.28 – Enchainements des TEA - 46g

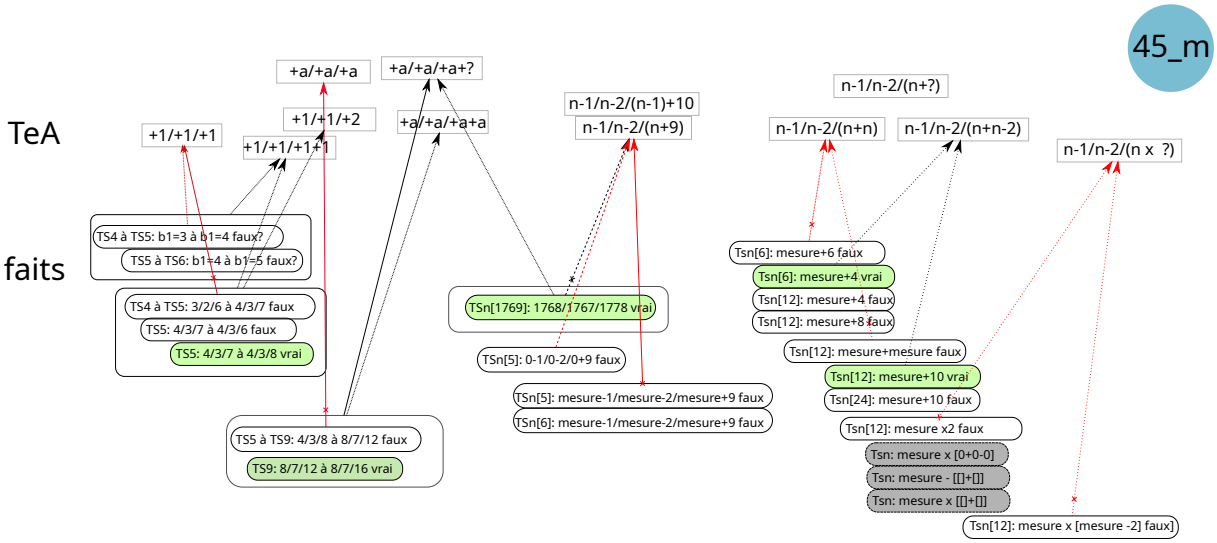


Figure 2.30 – faits et TEA pour le groupe 45m

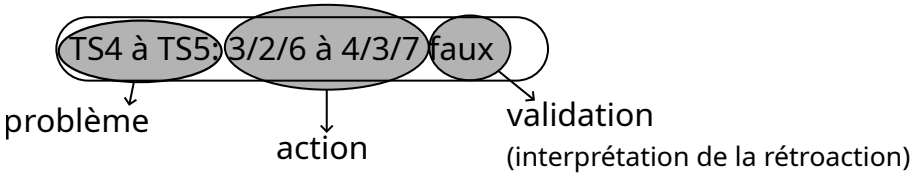


Figure 2.31 – Codage d'un cartouche « fait »

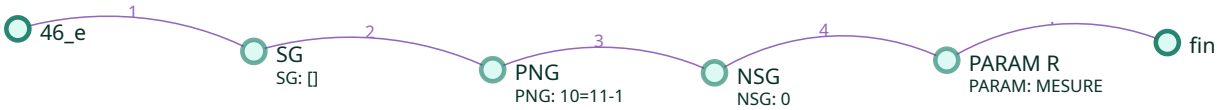


Figure 2.32 – Type de généralisation - 46e

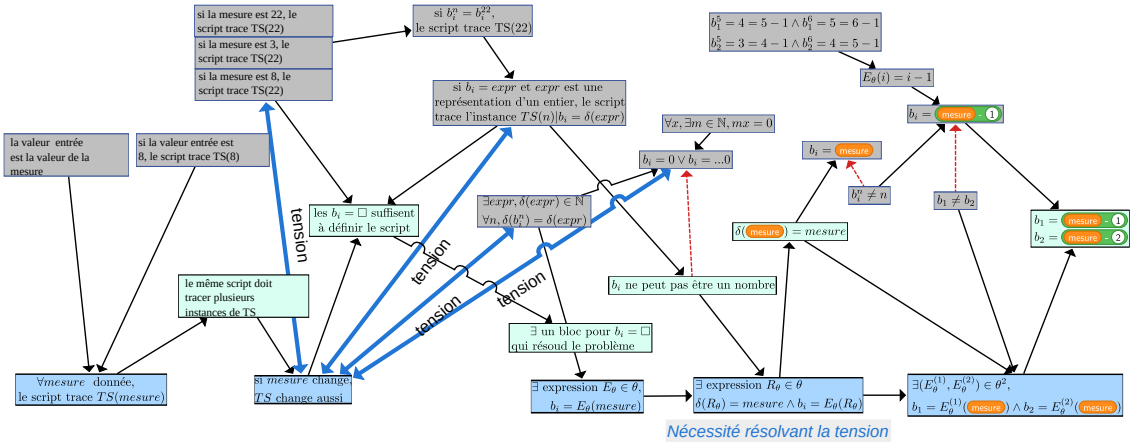


Figure 2.33 – Espace faits-contraintes - 46e

2.6 Résumé de la construction des représentations et de l'analyse des données

Nous avons cherché dans ce chapitre à établir une *méthodologie permettant d'identifier les faits et nécessités construits par les élèves, ainsi que leur mise en tension, à partir des traces d'actions de programmation.* Concernant l'organisation de la représentation des données, pour chaque groupe :

1. Nous commençons par reconstituer l'histoire du programme. Cette reconstitution se base sur les événements captés, et est automatique. Il s'agit d'une représentation diachronique peu compatible avec un document imprimé. Ces histoires sont dans les annexes numériques. Une version synchronique est aussi disponible, c'est cette version qui nous permet d'extraire l'écriture d'un algorithme à un moment donné.
2. Une représentation de l'histoire montrant la succession des événements de programmation liées au problème (donc avec un tri, en ne prenant en compte que les événements liés à la modification des boucles, ainsi que les tests) est extraite de ces événements captés. Cette représentation est notamment utilisée pour les transcriptions des interactions langagières : une fois ces événements synchronisés avec les enregistrements audio et vidéo, ils peuvent être directement intégrés dans les transcriptions.
3. En nous appuyant sur ces trois représentations, nous rassemblons les différentes actions des élèves lorsqu'ils modifient les scripts en *étapes*, qui regroupent un ensemble de modifications organisées avec un même objectif, généralement ponctué d'un test.
4. Ces différentes étapes sont répertoriées dans un tableur. Une première analyse a lieu, tentant de définir ces étapes, notamment en cherchant à établir le TEA mobilisé ou le type de généralisation en cours.
5. À partir de ce tableur, une représentation de l'activité des élèves, montrant l'enchainement étapes, est créée (automatiquement).
6. À partir de ce même tableur, une représentation de l'enchainement des types de programmation est créée (automatiquement).
7. D'autres représentations peuvent être extraites de ce tableur, par exemple pour répertorier quel TEA est mis en œuvre par quels groupes.

En mobilisant l'ensemble de ces éléments :

— nous décrivons et faisons une première analyse de l'activité des élèves ;

— nous établissons à partir de cette analyse les faits et TEA construits par les élèves (cela concerne notamment la généralisation arithmétique) ;

— ces éléments sont complétés par une recherche des régularités : quels sont les schémas identifiés par les élèves lorsqu'ils tentent de généraliser, et comment les mobilisent-ils ensuite ?

La construction de l'espace faits-contraintes, en croisant l'espace construit *a priori* et les résultats établis précédemment, permet de visualiser les articulations et tensions entre faits et nécessités, lors de la phase de généralisation, lorsqu'ils construisent (ou non) le concept de paramètre.

Chapitre 3

Analyse

Sommaire du chapitre

Liste des figures

Liste des tables

3.1 Introduction

Dans cette partie nous analyserons l'activité des élèves — plus précisément des binômes d'élèves — à partir de leurs traces de programmation et des représentations des données construites, complétées par les interactions langagières entre membres des binômes, lorsque celles-ci sont disponibles. Il s'agit ici d'établir comment les élèves avancent dans leur problématisation de la variable dans un rôle de paramètre, c'est-à-dire de déterminer comment les élèves posent et construisent le problème de la généralisation, et quelles sont les nécessités construites relatives à la variable dans un rôle de paramètre. On s'intéresse donc à la phase de généralisation de la situation (séances 3, 4 et 5), lors de laquelle les élèves devaient :

1. construire les scripts traçant les TS5, TS6 et TS7 à partir du script $TS(4)$ construit en séance 2 ;
2. construire des instances plus éloignées (TS11, TS19 à TS22) ;
3. construire le script générique permettant de tracer et dénombrer n'importe quelle instance suivant la valeur de la mesure entrée par l'utilisateur.

3.1.1 Analyse des groupes

Les groupes seront analysés en trois ou quatre étapes [1], chaque étape permettant d'avancer dans la compréhension de la position et de la construction du problème chez les élèves [2].

Description de l'activité

La première étape consiste à décrire l'activité des élèves, en s'appuyant sur les traces de programmation, et à produire une première analyse de cette activité. Il s'agit ici de recenser les problèmes qu'ils se sont posés, les différents épisodes par lesquels ils sont passés. Dans cette partie seront aussi présentés les différents obstacles rencontrés par les élèves, y compris ceux d'origine instrumentale, et ceux ayant une origine sémiotique, s'ils ne sont pas directement liés à la généralisation.
L'objectif est de rendre visible les différents épisodes, de donner un premier niveau d'explicitation de certaines actions (hors généralisation) ou interprétation des rétroactions. Il s'agit de donner à voir les éléments de l'activité qui vont nous permettre d'établir les faits possiblement construits — ou non — par les élèves.

Recherche de régularités

Ce deuxième niveau d'analyse, s'appuyant sur le précédent, consiste à rechercher les faits et TEA construits par les élèves, et comment ils se construisent.
Le regard est principalement porté sur les moments de généralisation arithmétique ou de généralisation algébrique naïve (Radford, 2008, p. 3). L'objectif est notamment de tenter une reconstitution des faits (validants ou non) mobilisables par les élèves lors de la phase de généralisation du script.

1. Suivant leur avancée dans le problème de généralisation.
2. Les diverses représentations de l'activité des élèves sont précisés dans la section 2.5.2 (p. 2.5.2).

Généralisation

Pour ce troisième niveau, nous nous intéressons plus spécifiquement au problème de la généralisation du script. Il s'agit d'établir ce qui permet aux groupes de généraliser ou non le script. Rappelons que dans notre situation nous considérons, en suivant Radford (2006a, p. 5) que généraliser un motif de façon algébrique consiste notamment à identifier des points communs dans une séquence, être conscient que ces points communs s'appliquent à l'ensemble des instances du problème étudié, et être capable d'utiliser ces points communs pour produire une expression permettant de déterminer n'importe quelle instance du problème étudié :

> Generalizing a pattern algebraically rests on the capability of grasping a commonality noticed on some elements of a sequence S, being aware that this commonality applies to all the terms of S and being able to use it to provide a direct expression of whatever term of S. (ibid., p. 5)

Il s'agit donc d'étudier comment les élèves passent des faits construits à partir des instances particulières $p_1, p_2, ..., p_k$ à une expression de θ permettant d'établir directement p_n pour tout n (figure 3.1, p. 224). Dans notre situation, cette généralisation doit produire des expressions différentes pour les familles de boucles $B_1 = \{b_1, b_6\}$, $B_2 = \{b_2, b_3, b_4, b_5, b_8\}$ et $B_7 = \{b_7\}$.

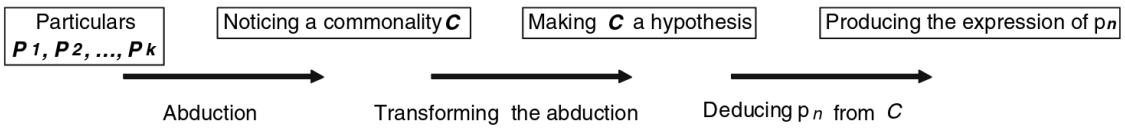


Figure 3.1 – Architecture d'une généralisation algébrique de motifs, (Radford, 2008, p. 3)

L'objectif est ici d'identifier les schémas reconnus par les élèves — et notamment ceux concernant la boucle b_7 —, de mettre en avant les aspects sémiotiques liés à la construction d'une expression dans θ de ces schémas par ces élèves, et de questionner les éventuels obstacles à cette généralisation algébrique.

Espace des faits-contraintes

Nous nous appuyons sur l'ensemble des éléments qui émergent des précédentes analyses pour reconstruire l'espace des faits-contraintes du problème posé par les élèves. Cet espace donnant à voir l'articulation entre les faits construits ou mobilisés, les nécessités ou contraintes locales au problème et les nécessités qui font avancer la problématisation de la variable dans un rôle de paramètre par les élèves. Dans les espaces de contraintes tels qu'on les conçoit en général dans le Cadre de l'Apprentissage par Problématisation (CAP), on montre « comment les nécessités se propagent en s'appuyant sur le registre empirique et le registre explicatif » (Orange, 2012, p. 42). Dans nos représentations, nous ne faisons pas apparaitre le registre explicatif, « qui organise les explications [...] du problème travaillé » : en effet, lorsqu'on s'appuie sur les traces de programmation et non sur les discours qui les accompagnent, la détermination de REX dans lequel se situe les élèves nous parait très incertaine. Nous pouvons juste faire l'hypothèse que les élèves qui problématisent dans notre situation voient leur(s) REX évoluer vers un REX algorithmique et algébrique.

3.1.2 Choix des groupes

Nous disposons des traces de programmation de vingt-et-un groupes sur les deux classes. Parmi ces groupes, six (trois dans chaque classe), devaient aussi être filmés et leurs interactions langagières enregistrées sur les trois séances de la phase de généralisation du script. Un certain nombre d'éléments se retrouvent chez les différents groupes, nous allons donc analyser seulement une sélection de ces groupes. Cette sélection est faite en fonction :

1. des caractéristiques de la phase de généralisation et du type de généralisation qui semble mis en œuvre (voir 2.5.3, p. 212),

2. de la présence ou non de captation vidéo et audio,

3. de l'intérêt spécifique de chaque groupe (quels éléments d'analyse supplémentaires amène-t-il ?).

Les groupes sont rassemblés en groupements, caractérisés par la représentation des types de généralisation. Pour chaque groupement, une synthèse sera faite en incluant des éléments provenant des traces de programmation des groupes non analysés en détail.

Groupement 1

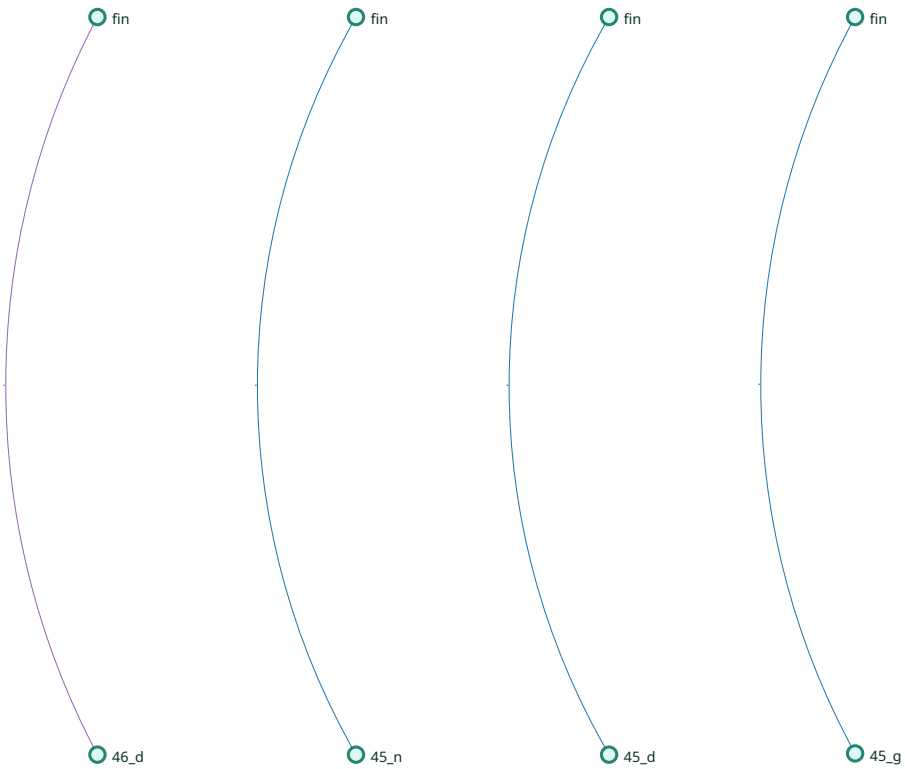


Figure 3.2 – Groupement 1 : Pas de généralisation

La première catégorie de groupes, le Groupement 1, concerne les groupes n'ayant pas commencé à généraliser (figure 3.2, p. 225). Nous analyserons les groupes 46d et 45d [3]. Le groupe 46d a produit de nombreuses instances successives valides, alors que le 45d a eu des difficultés à construire les premières instances, tous comme le 45g. Le groupe 45n n'a quant à lui construit aucune instance valide.

Groupement 2

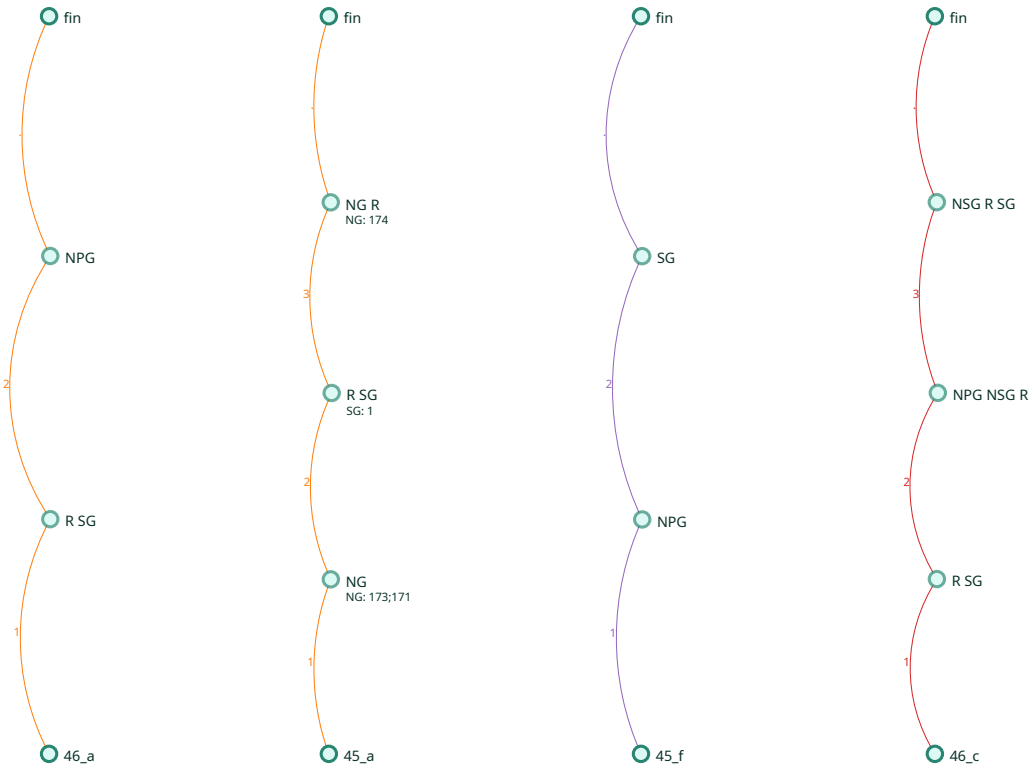


Figure 3.3 – Groupement 2 : Pas de paramètre, linéaire

Le Groupement 2 rassemble les groupes qui n'ont pas mobilisé le paramètre, mais qui semblent avoir commencé à poser le problème de la généralisation du script. Ces groupes montrent une exploration du problème assez linéaire, enchainant divers types de généralisation (figure 3.3, p. 226). Les groupes 45a et 45f seront analysés en détail, ces deux groupes permettant d'observer l'émergence de types de généralisation que l'on retrouve dans les deux autres groupes non détaillés (46a et 46c). Le groupe 45f est aussi l'objet d'une captation audio et vidéo qui viendra compléter, illustrer ou expliquer les traces de programmation.

Groupement 3

Le Groupement 3, est constitué des groupes n'ayant pas non plus mobilisé de paramètre, mais qui montrent une démarche moins séquentielle que le groupement 2, représentée par une ou plusieurs boucles sur les graphes représentant l'évolution des éléments concernant la généralisation (figure 3.4, p. 228). Nous disposons d'une captation audio et vidéo du groupe 46g, et le groupe 46i, qui sera, lui aussi, analysé en détail, montre une caractéristique présente chez les autres groupes (46l, 46b, 45k, 46f) : des aller-retours entre des représentations mobilisant des nombres génériques (NG) et celles mobilisant des nombres qui ont perdu leur généricité (PNG).

3. Pour rappel, « 45d » désigne le groupe *d* de la classe de quatrième 5. Dans les schémas représentant les types de généralisation identifiés, le même groupe sera désigné par « 45_d », pour des questions de lisibilité.

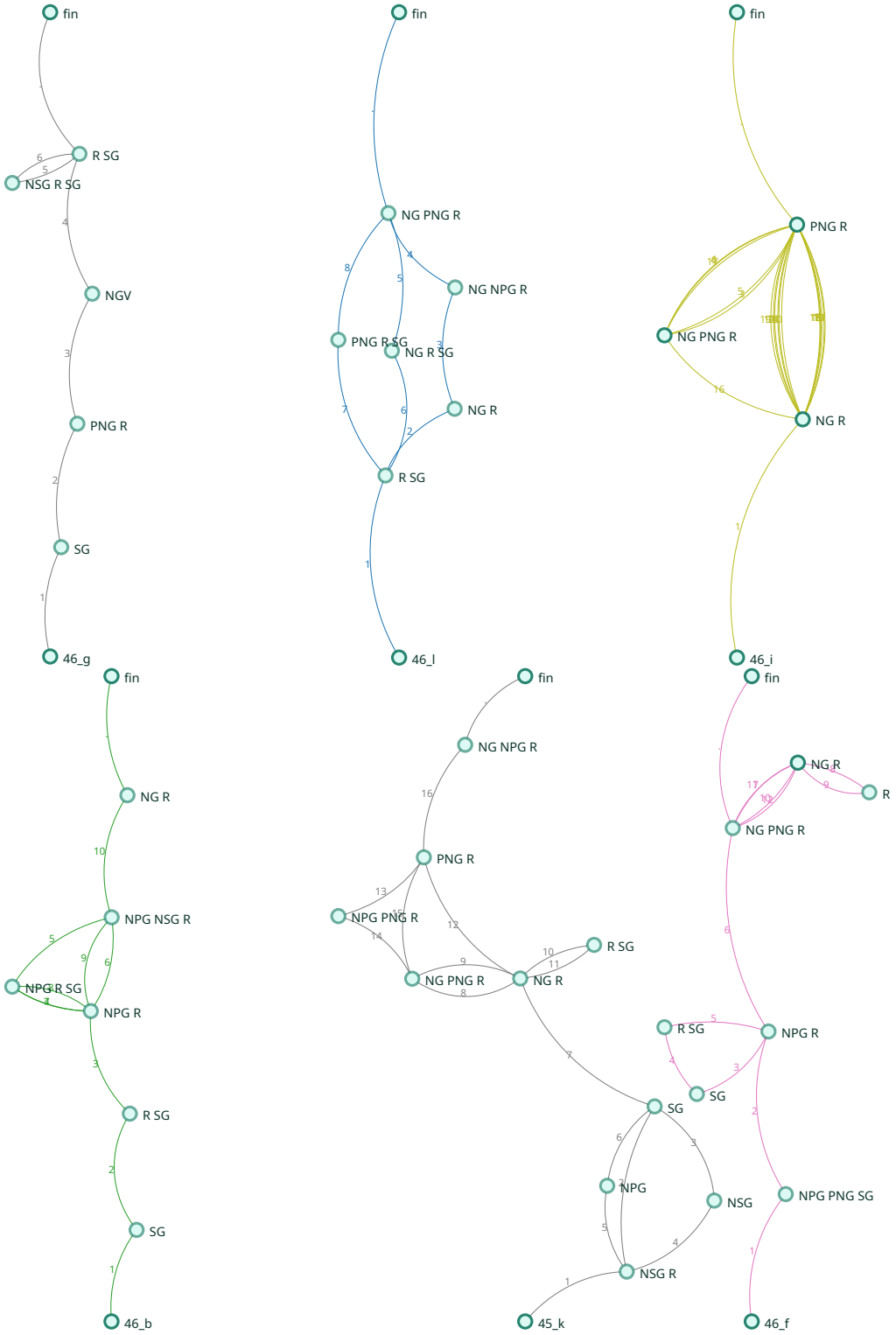


Figure 3.4 – Groupement 3 : Pas de paramètres, boucles

Groupement 4

Le Groupement 4 est formé des groupes ayant mobilisé le paramètre, ou plus précisément la représentation mesure, avec, comme le groupement 3, une démarche d'exploration non linéaire (figure 3.5, p. 230). Les groupes 46m et 46e seront détaillés, illustrant l'apparition de la nécessité d'une représentation dans θ de la *mesure*, par des voies différentes. Le groupe 45e est en outre le seul groupe ayant manifesté une écriture explicite de la relation entre la *mesure* et le nombre d'itérations de la boucle b_7.

Figure 3.5 – Groupement 4 : Paramètre, boucles

Groupement 5

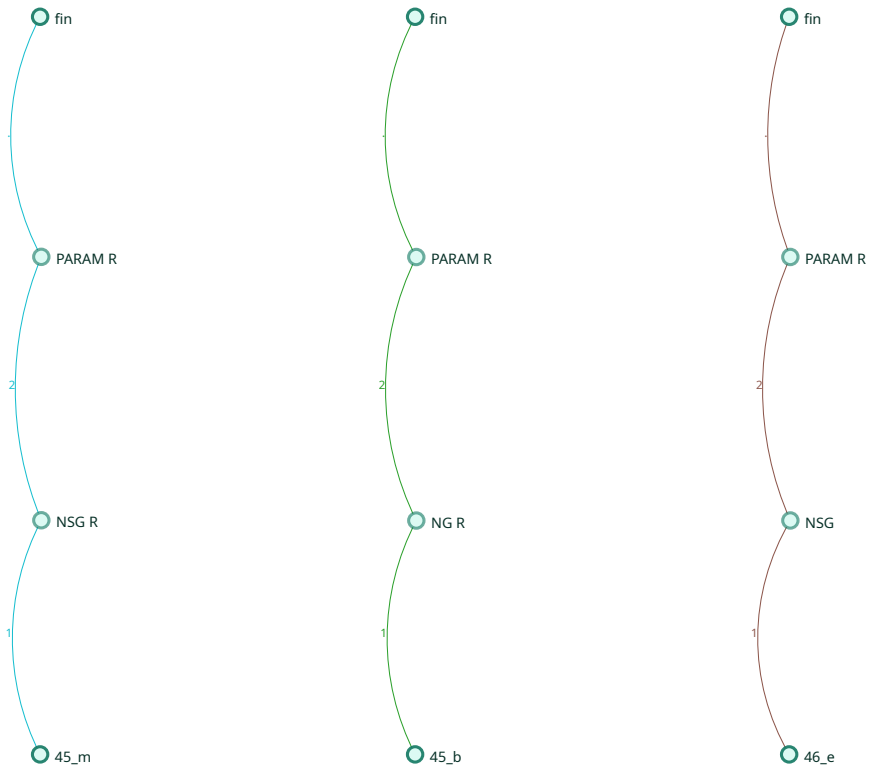


Figure 3.6 – Groupement 5 : Paramètre, linéaire

Le dernier groupement (Groupement 5) rassemble les groupes qui ont mobilisé le paramètre tout en ayant une démarche assez séquentielle.(figure 3.6, p. 231). Nous analyserons les groupes 45m et 46e. Le groupe 45b sera évoqué en synthèse pour sa spécificité concernant le problème posé par les élèves. Le groupe 46e permettra d'éclairer la construction des nécessités du problème par leurs interactions langagières notamment.
Nous rajouterons quelques éléments concernant le groupe 45h. Lors des séances 4 et 5, deux groupes ont pris cet identifiant (le groupe censé être le 45i, et le 45h prévu). Comme les deux groupes ont à chaque fois commencé par charger le programme de base de la séance, il est impossible de les différencier, et nous avions décidé d'exclure les groupes 45i et 45h de l'analyse. Cependant, après une nouvelle observation de l'activité de ces groupes indissociables, il est apparu que l'un d'eux avait abouti à une formulation valide du script générique. La démarche lors de cette séance 5 semblant similaire à celle des autres groupes de ce groupement, nous inclurons quelques remarques les concernant en synthèse du groupement.

Récapitulatif

La table 3.1 précise le groupement associé à chacun des groupes suivis. Certains identifiants de groupe ne sont pas associés à un groupement pour les raisons suivantes :

a. Poste non occupé [4].

4. En début d'activité, chaque binôme se voyait attribuer un identifiant de groupe lié à la position du poste informatique dans la salle.

b. Double utilisation de l'identifiant 45h, groupes indissociables.

c. Reconstitution de l'histoire du programme impossible (raisons non encore identifiées)

d. Double utilisation volontaire de l'identifiant 45j par le groupe 45i. Les deux groupes sont dissociables par analyse de leur activité, seul le groupe 45j « original » est conservé.

e. Confusion avec le groupe 45j, écarté par erreur.

groupe	groupement
45a	2
45b	5
45c	(a)
45d	1
45e	4
45f	2
45g	1
45h	(b)
45i	(d)
45j	4(d)
45k	3
45l	4
45m	5
45n	1

groupe	groupement
46a	2
46b	3
46c	2
46d	1
46e	5
46f	3
46g	3
46h	(c)
46i	3
46j	(e)
46k	4
46l	3
46m	4
46n	(a)

Tableau 3.1 – Association groupe-groupement

3.2 Groupement 1

3.2.1 Groupe 46d

Le groupe 46d, absent lors de la séance 5, a été capable de construire une instance en fonction de la précédente, mais n'a pas commencé de généralisation algébrique naïve (en construisant une certaine instance), ni un début de généralisation algébrique.

Description de l'activité

Le groupe 46d, après avoir construit le script $TS(5)$, va construire, avec succès, les instances successives de 6 à 17. Les élèves de ce groupe ne vont en revanche pas poser le problème de la construction du script générique.

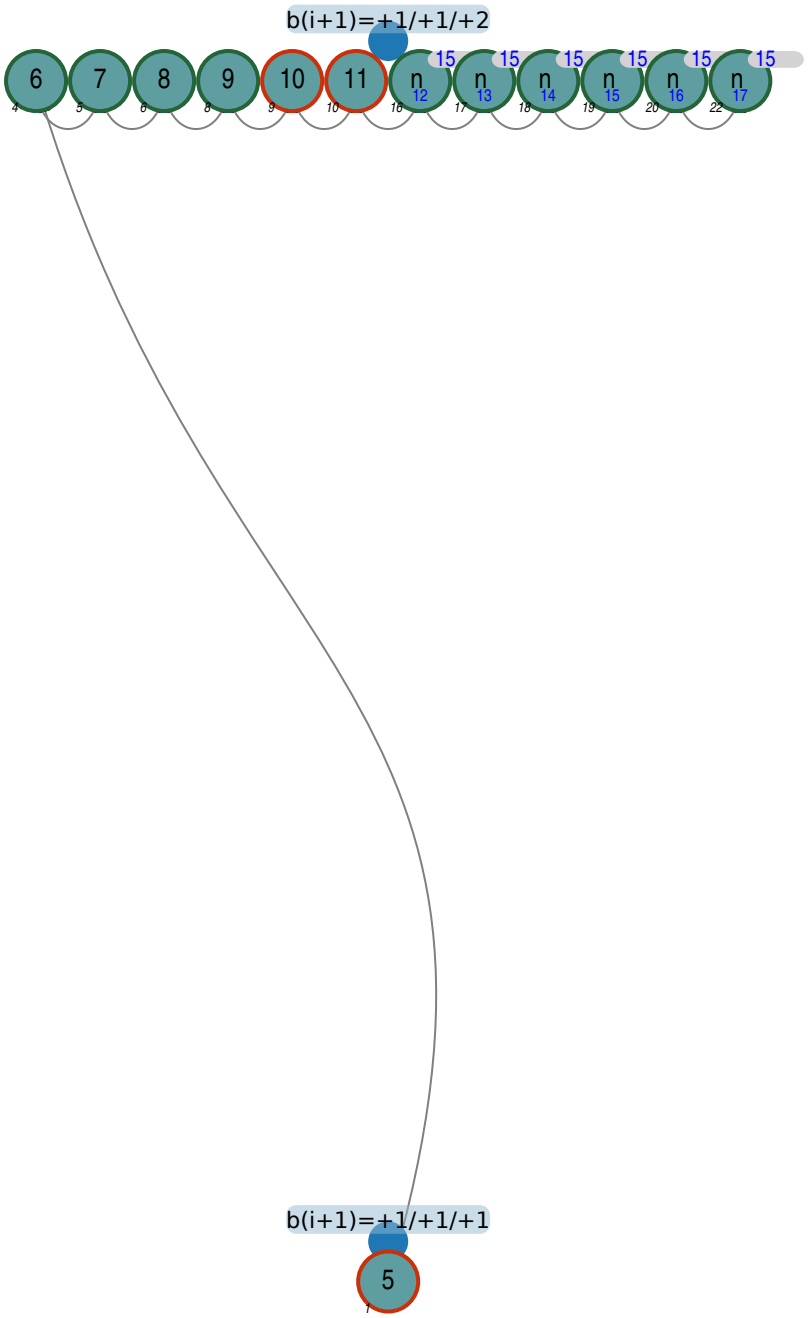


Figure 3.7 – Organisation de l'activité - 46d

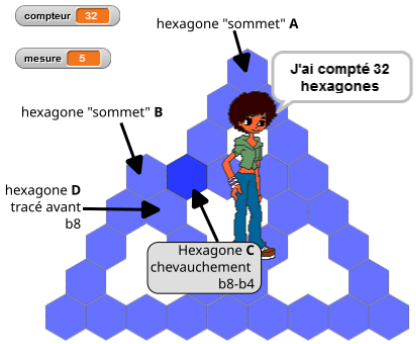
Une construction classique du $TS(5)$: interprétation de la rétroaction Le groupe 46d commence par exécuter le script $TS(4)$, duplique ce script comme demandé par l'enseignante, et le modifie en ajoutant un au nombre de répétitions de chacune des boucles, TEA que nous notons $+1/+1/+1$: ils passent de $\{3/2/6\}$ à $\{4/3/7\}$ (étape 1). Ils obtiennent ainsi :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 32, 5 \right)$$

Ces élèves identifient la boucle b_8 comme étant celle qui est erronée (alors que c'est le côté tracé par b_7 qui est trop court de un hexagone). Ils considèrent que cette boucle doit tracer un hexagone de plus, et ajoutent donc un au nombre de répétitions de $b8$ (étape 2), ce qui produit la rétroaction :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 33, 5 \right)$$

Si le nombre d'hexagones est exact, le tracé n'est pas valide, et est bien reconnu comme tel par les élèves. Ils remettent alors b_8 à son ancienne valeur, et augmentent b_7 de un, aboutissant ainsi à un tracé valide (étape 3). Cette erreur d'interprétation suivant d'un ajustement valide est présente



(a) Rétroaction pour $TS(5)$ avec $b_7 = 6$

(b) Rétroaction pour $TS(5)$ avec $b_7 = 6$ et $b_8 = 4$

Figure 3.8 – Un enchainement classique lors de la construction de $TS(5)$

chez quatre autres groupes[5]. La première rétroaction (figure 3.8a, p. 234) montre le côté erroné du tracé de deux façons : la forme ne correspond pas à un TS, et l'hexagone de couleur foncée **C** représente un chevauchement. Ici, le binôme semble considérer que le nombre d'hexagones du côté en partie tracé par b_8 est insuffisant, et effectivement ce côté semble n'être constitué que de quatre hexagones : l'hexagone « sommet » **A**, tracé par b_7, suivi des trois répétitions du tracé. Rappelons que cette boucle doit bien tracer seulement trois hexagones, puisque l'hexagone « sommet » **B** de fin de tracé est déjà présent (commun avec le tracé de b_1). L'hexagone **D** n'est dans ce cas pas compté comme faisant partie du tracé, puisqu'il a été tracé avant le début de b_8, et il aurait été en bleu foncé si b_8 l'avait « de nouveau » tracé. L'hexagone **C**, qui pointe une erreur, est tracé en dernier, ce qui peut expliquer que les élèves considèrent la dernière boucle comme étant erronée. Après modification, ce groupe obtient le tracé figure 3.8b (p. 234), tracé qui va être correctement interprété. Le tracé de b_8 est bien de cinq hexagones, mais la forme n'est toujours pas valide. Le décalage, cette fois bien rendu visible, est à imputer à la boucle précédente, b_7, qui aurait dû tracer un hexagone de plus. Les élèves ont possiblement effectué mentalement une translation du côté tracé, ce qui les a conduit à anticiper un chevauchement de l'hexagone **B**. Une autre possibilité étant que ce binôme, voyant que l'erreur vient de la boucle précédente, annule sa modification pour la reporter sur cette nouvelle source d'erreur possible identifiée.

5. 46a (étape 3), 46c (7), 45d (7), 45b (8).

Construction de multiples instances consécutives Suite à cela, le groupe 46d duplique le script $TS(5)$ obtenu, et le modifie en ajoutant un à chaque boucle, exceptée b_7 qui voit son nombre de répétitions augmenté de deux. Le tracé étant validé, ils vont réitérer cette méthode (duplication du dernier script obtenu et application du TEA $+1/+1/+2$) pour construire les scripts traçant le TS6, le TS7, puis le TS8 et en début de séance 4 le TS9 (étapes 4-8).
Ces deux dernières instances n'étaient pas demandées et sont construites sur les scripts pour lesquels le script d'initialisation fixe la mesure à 11 et à 22 respectivement. Ils obtiennent ainsi les deux rétroactions suivantes :

$$TS(8)[] \rightsquigarrow \left(\text{  } \text{} , 60, 11 \right)$$

$$TS(9)[] \rightsquigarrow \left(\text{  } \text{} , 69, 22 \right)$$

Ces deux rétroactions sont considérées comme valides par les élèves, puisqu'ils passent à la construction de l'instance suivante. Ainsi, la non cohérence de la mesure avec le tracé, ou le fait que la figure n'occupe pas tout l'espace n'est pas pris en considération : seule la forme avec une absence de chevauchement et de trou compte.

Construction inter-objectale Les élèves de ce groupe, continuant leur construction d'instances par incrément de un, passent alors à la construction du script $TS(10)$ à partir du $TS(9)$, toujours en utilisant la même méthode (étape 10). Cependant, avant d'avoir terminé la construction du $TS(10)$ (n'ayant pas encore modifié b_7 et b_8), ils dupliquent de nouveau ce script pour créer $TS(11)$. Ce binôme passe ainsi de $TS(9) = \{8/7/7/7/7/8/16/7\}$ à $TS(10) = \{9/8/8/8/8/9/16/7\}$ puis $TS(11) = \{10/9/9/9/9/10/18/8\}$. On voit ainsi qu'ils appliquent le TEA $+1/+1/+2$ à $TS(10)$ pour construire $TS(11)$, en propageant l'erreur (étapes 11-12). Cela nous laisse à penser que ce groupe construit une instance à partir de la précédente, et seulement à partir de la précédente. La propagation de l'erreur nous indique sur quoi porte les relations que ces élèves établissent lorsqu'ils créent une nouvelle instance. Ils ne se basent pas sur la mesure attendue (ils n'en sont pas encore à la généralisation algébrique, même naïve) : il n'y a pas de relation trans-objectale établie (Piaget et García, 1983). De même, ils ne construisent pas non plus de relation entre les valeurs du nombre de répétitions des boucles d'une même instance. En effet, une fois établie la première valeur (10), il est possible de construire le script en constatant :

— que la deuxième valeur est la première moins un ($b_2^p = b_1^p - 1$),
— que la valeur suivante est identique à la seconde ($b_3^p = b_2^p$),
— tout comme les troisième, quatrième et cinquième valeurs ($b_5 = b_4 = b_3 = b_2$),

— que la sixième valeur est la précédente plus un, ou bien la même valeur que la première ($b_6^p = b_5^p + 1 = b_1$),

— que la septième valeur est le double de la sixième ($b_7^p = 2b_6^p$),

— et enfin que la dernière valeur est identique à la deuxième (par exemple).

Mais, ces élèves n'établissent pas non plus ces relations intra-objectales, sinon l'erreur ne se serait pas propagée : ils auraient établi que $b_7^{11} = 2b_6^{11} = 2 \times 10 = 20$. Ils font donc des relations inter-objectales, comparant les propriétés d'un objet à celles d'un autre, ou une instance à une autre : $b_{i\neq 7}^{p+1} = b_i^p + 1$ et $b_7^{p+1} = b_7^p + 2$, ce qui est une autre façon d'exprimer le TEA $+1/+1/+2$. Nous verrons que certains groupes, à certains moments au moins, passent par des relations intra-objectales, ce qui complique la généralisation (voir 2.1.5, p. 146).

Raisons de la construction inter-objectale Cette double erreur ne pourra être ni validée ni invalidée immédiatement par un test : les élèves ont copié le $TS(9)$ sous l'en-tête du script qui est censé tracer n'importe quelle instance de TS, que l'on peut exécuter en appuyant sur la touche « n » (

), mais ils dupliquent l'intégralité du script obtenu pour préparer le $TS(11)$. Ils disposent ainsi de deux scripts différents qui vont s'exécuter simultanément [6] lors d'un même événement : l'appui sur la touche « n » (46d, S4, 4'52).

(a) Lancement du script générique

(b) Lancement simultané de deux instances du script générique

Figure 3.9 – Rétroaction lors du lancement normal ou double du script générique

Un premier événement ne lancera que le script censé tracé le TS11, puisque les élèves déclenchent l'exécution en cliquant sur le script. Cette première exécution produi la rétroaction de la figure 3.9a (p. 236), déjà vue en groupe classe en début de séance. Les élèves ne vont pas entrer de valeur et vous stopper ce script. Ils vont ensuite enchainer deux double-lancements qu'ils vont annuler avant d'entrer une valeur, obtenant la rétroaction de la figure 3.9b (p. 236). Ces deux double-lancements annulés laissent penser que les élèves ont identifié un problème. Il est possible que ce soit en obtenant une rétroaction au lancement qui n'est pas celle attendue (le message invitant à entrer une valeur n'est pas visible). Il est aussi possible qu'une rétroaction interne à l'EPGB renforce ce doute : en effet, les scripts en cours d'exécution sont entourés dans l'EPGB

6. Il n'y a en fait pas de réel parallélisme dans Snap !, c'est un parallélisme simulé.

d'un halo bleuté (voir figure 3.10, p. 240 [7]), et la signification de ce halo a été évoquée en début de séance 4 par l'enseignante, lorsqu'elle a précisé et montré l'utilisation du mode « pas-à-pas » dans Snap ! Les deux scripts sont ainsi identifiés comme étant lancés, et restent bleutés car en attente d'une entrée de la part de l'utilisateur. Cette mise en valeur semble permettre aux élèves d'identifier non pas la double exécution sur un même événement, mais leurs erreurs concernant les boucles b_7 et b_8. Ils vont effectuer les actions suivantes, dans l'ordre (étapes 11-12) :

1. $b_7^{10} \leftarrow 18$
2. $b_7^{11} \leftarrow 20$
3. $b_8^{10} \leftarrow 8$
4. $b_8^{11} \leftarrow 9$

Les scripts étant disposés côte à côte dans l'espace de programmation, tout se passe comme si les élèves avaient constaté les relations par comparaison sur une même ligne horizontale (les cadres de la figure 3.10, p. 240). Ainsi, les valeurs qui doivent changer quand on passe d'un script à l'autre, sont celles qui changent quand on passe d'un script au script qui lui est adjacent à droite. Ils changent ainsi ce qui devrait changer ($b_7^{10} = b_7^9 + 2$) puis propagent le changement ($b_7^{11} = b_7^{10} + 2$), puis ils procèdent de façon similaire pour les boucles b_8. Cet épisode semble confirmer, à ce stade, pour ces élèves, une construction inter-objectale [8]. Cette construction inter-objectale est probablement favorisée, si ce n'est induite, par les alignements horizontaux des différentes boucles. On peut ainsi faire l'hypothèse raisonnable que les élèves vont probablement davantage chercher à créer des relations inter-objectales plutôt que intra-objectales.
Une fois ces ajustements faits, le groupe 46d va d'une part résoudre le problème de l'exécution simultanée, et d'autre part expérimenter sur le sens de variable mesure.

Construction d'un sens de la variable *mesure* Les élèves vont de nouveau lancer une exécution en cliquant sur le script $TS(11)$, ce qui ne conduit pas, cette fois-ci, à une exécution simultanée. La demande d'entrée de la part de l'utilisateur semble leur poser problème [9]. En effet, suite à cette demande, ils arrêtent une première fois le script, et modifient l'instruction d'affichage afficher la variable mesure en afficher la variable compteur : cherchent-ils à ne plus faire afficher la demande d'entrée pour la mesure, cette demande n'ayant pas de sens pour eux puisqu'ils construisent des instances spécifiques ? Après un test qui s'avérera infructueux (si on considère cette dernière hypothèse), ils rétablissent l'instruction d'affichage originelle (46g, S4, 5'42).
La signification de la variable mesure, dont la valeur est décidée par l'utilisateur pour ce script, semble être en jeu à ce moment. Le binôme va tout d'abord lancer le script $TS(9)$, construit sur l'entête pour un TS22 ; puis ils vont procéder à deux tests, en entrant une valeur de 80 puis 4, tests qui semblent être considérés comme invalides (notamment le deuxième qui est

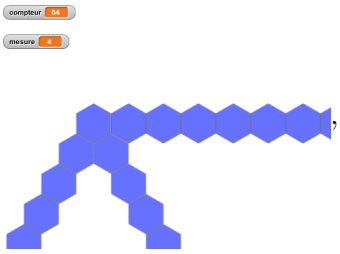
7. Cette figure est une reconstitution de l'état de l'EPGB du groupe 46d au moment qui nous intéresse, à partir de l'histoire du programme de ce groupe et de leur programme en fin de séance, automatiquement sauvegardé. Nous faisons l'hypothèse qu'il n'y a pas eu de déplacement des scripts dans l'espace de programmation, les en-têtes des scripts du programme lancé en début de séance étant à la même place en fin de séance.

8. On aurait pu aussi faire l'hypothèse que les élèves ont identifié la relation entre le nombre de répétitions d'une boucle et la valeur de la mesure du TS voulu, mais dans ce cas on s'attendrait à la modification complète d'un objet avant de passer au suivant : on aurait ainsi eu les modifications de b_7^{10} et b_8^{10} avant de passer à b_7^{11} et b_8^{11}.

9. C'est un point récurrent que nous traiterons pour d'autres groupes (45d, 46g, 46i).

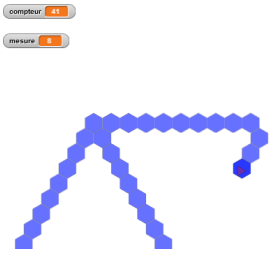
stoppé lorsque le tracé sort de l'espace de rétroaction) :

$$TS(11)[80] \rightsquigarrow \left(\text{  } \text{} , 69, 80 \right)$$

$$TS(11)[4] \rightsquigarrow \left(\text{  } , [], 4 \right)$$

Un troisième essai va être effectué avec une entrée de 10, mais cette fois l'exécution est déclenchée par un appui sur la touche « n », ce qui conduit à une exécution parallèle visiblement problématique, pour lequel le message de demande d'entrée de la part de l'utilisateur se déplace avec les hexagones créés (voir figure 3.11a, p. 241 [10].) et se termine en gardant l'invite affichée (figure 3.11b, p. 241).
Après un peu moins dune minute sans action sur l'EPGB, les élèves corrigent le script $TS(11)$ en en changeant l'événement déclencheur : 

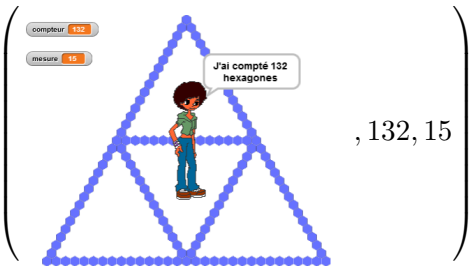
(46d, S4, 12'23).
Une fois ce problème réglé, les élèves poursuivent leur expérimentation avec des entrées de 8 (vue comme invalide, sortant du cadre), puis de 15, sans doute considérée comme valide, puisqu'ils passent alors à la construction d'une nouvelle instance.

$$TS(11)[8] \rightsquigarrow \left(\text{  } , [], 8 \right)$$

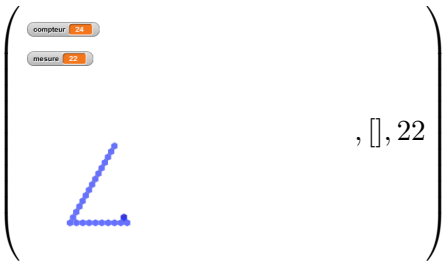
$$TS(11)[15] \rightsquigarrow \left(\text{  } \text{} , 87, 15 \right)$$

10. Reconstitution de ce que voient les élèves, nous n'avons pas accès à la rétroaction de façon dynamique

Construction de multiples instances consécutives (bis) Ce groupe va alors construire cinq autres instances successives (étapes 16-20), en associant l'événement déclencheur à l'appui d'une nouvelle touche pour chaque script (les touches « p », « q », « r », « s » et « t » pour tracer respectivement les TS12, TS13, TS14, TS15 et TS16). Les quatre premiers scripts sont exécutés en entrant une valeur de 15 pour la mesure, le script $TS(15)$ étant stoppé avant la fin de son exécution (figure 3.12, p. 241). Dans ce cas, on peut considérer que les élèves valident le tracé sans attendre la fin de l'exécution puisqu'ils enchainent directement sur l'instance suivante $TS(16)$.
Une nouvelle perturbation issue de l'EPGB va apparaitre alors : les élèves vont cliquer sur l'icône permettant le reprise d'un script en pause (46d, S4, 23'08), pensant sans doute reprendre l'exécution du tracé du TS15, or le script $TS(4)$ est en pause depuis plusieurs minutes [11], et c'est ce script qui est donc remis en route, conduisant à une rétroaction ininterprétable — sauf à avoir identifié le script relancé, ce qui est peu probable. Une nouvelle occurrence de ce fonctionnement de l'EPGB va avoir une influence sur les faits construits : la rétroaction issue du $TS(16)[15]$ est possiblement valide — le tracé restant dans l'espace d'exécution —, mais un test suivant mêlant lancement et reprise d'un ancien script donnera une rétroaction visiblement erronée :

$TS(16)[15] \rightsquigarrow$ 

$, 132, 15$

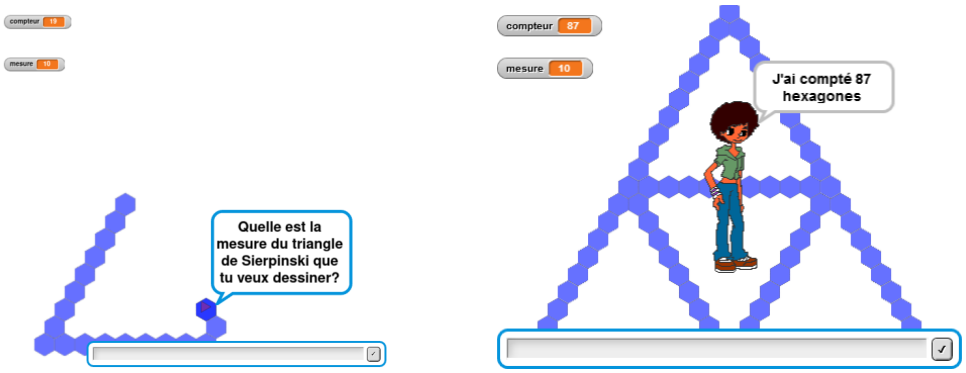
$TS(16)[22] \rightsquigarrow$ 

$, [], 22$

Les élèves de ce groupe vont alors stopper l'exécution de tous les scripts en appuyant sur le bouton idoine, évoqué en groupe classe en début de séance, puis relancer $TS(16)[15]$ avant de construire le tracé du TS17 (étape 22) en omettant, comme précédemment, les boucles b_7 et b_8 (figure 3.13a, p. 242). Celles-ci seront cette fois corrigées en deux étapes : correction de b_7 et test (figure 3.13b, p. 242), puis correction de b_8 avec un test avorté suite à la fin de la séance.
Les élèves de ce groupe étant tous deux absents en séance 5, ils n'auront pas l'occasion de poser le problème de la généralisation.

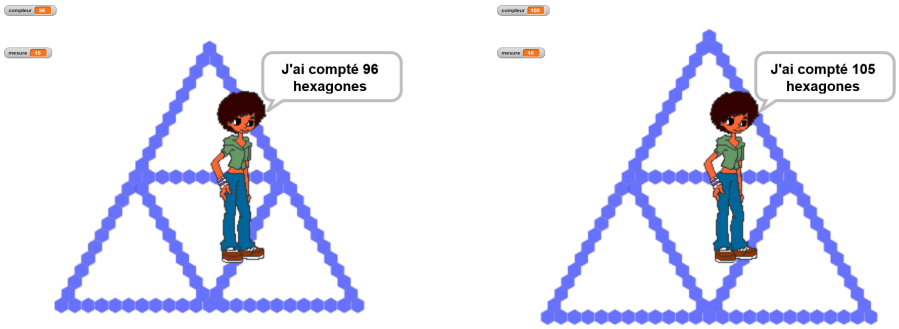
11. Il a été mis en pause à 15'22, soit presque huit minutes plus tôt.

Figure 3.10 – Disposition probable des scripts du groupe 46d lors d'une exécution parallèle

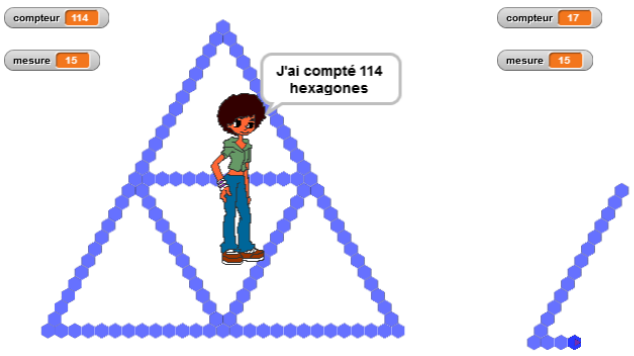


(a) En cours d'exécution **(b)** Rétroaction finale

Figure 3.11 – Rétroactions suite à l'exécution simultanée de deux scripts identiques (46d)

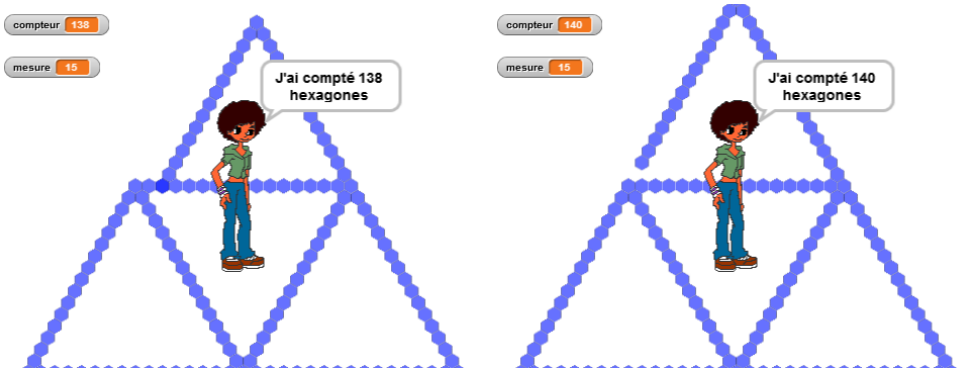


(a) $TS(12)[15]$ **(b)** $TS(13)[15]$



(c) $TS(14)[15]$ **(d)** $TS(15)[15]$

Figure 3.12 – Différentes instances, une seule mesure (46d)



(a) $TS(17)[15]$, b_7 et b_8 non modifiées **(b)** $TS(17)[15]$, b_7 corrigée

Figure 3.13 – Rétroactions pour le tracé d'un TS17 (46d)

Recherche de régularités

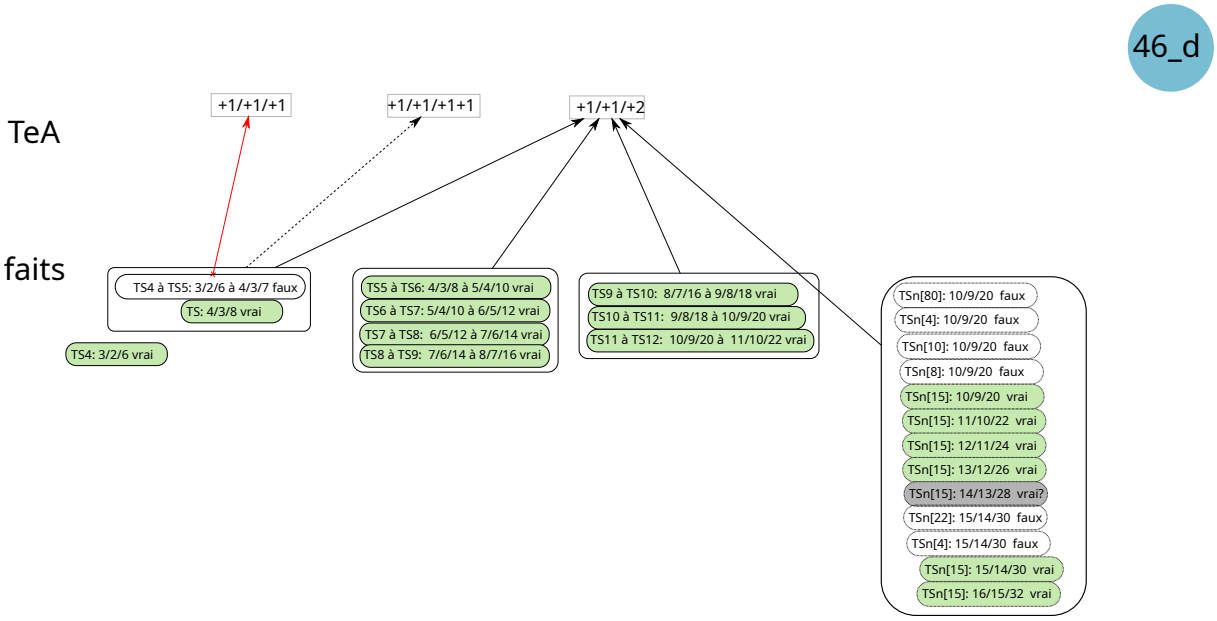


Figure 3.14 – Faits et TEA, groupe 46d

Afin de pouvoir généraliser, les élèves doivent identifier un schéma, une régularité entre les différents motifs, ici constitués des scripts permettant de tracer les TS. Dans cette partie, nous précisons les faits et TEA possiblement construits par les élèves lors de cette phase de recherche de régularités, illustrés dans la figure 3.14 (p. 243).
Les faits et TEA construits par ce groupe montrent qu'un TEA valide a rapidement été construit, et que ce TEA a été mise en œuvre de façon répétée pour construire les différentes instances successives. La valeur entrée par l'utilisateur pour les scripts qui auraient dû être génériques, ne correspond à la mesure du TS tracé : la valeur entrée, qui sera assignée à la variable *mesure*, est associée à l'affichage (au niveau de zoom), et non à la mesure du TS.

Construction rapide d'un TEA valide Lors de la recherche de construction d'une instance connaissant la ou les précédentes, le groupe 46d, après avoir classiquement expérimenté le TEA $+1/+1/+1$, l'a immédiatement invalidé au profit du TEA valide $+1/+1/+2$. Ce TEA sera le seul mobilisé sur les deux séances, les deux passages incluant une erreur sur les boucles b_7 et b_8 étant très probablement un oubli de traitement des modifications plutôt que l'application d'un curieux TEA qui n'entrainerait pas de modification sur certaines boucles.
La première construction valide, constituée en ajoutant d'abord un au nombre de répétitions de chacune des boucles, puis en ajustant b_7 en ajoutant de nouveau un, semble invalider le TEA $+1/+1/+1$, qui ne sera plus mobilisé. D'autre part, ces actions peuvent renforcer la probabilité que les TEA $+1/+1/+1+1$ — « je rajoute un à chaque boucle puis de nouveau un à b_7 » —, ou $+1/+1/+2$ — « je rajoute un à chacune des boucles sauf pour b_7 pour laquelle je rajoute deux » —, soient valides. Même s'ils sont équivalents en termes de résultats, lorsqu'on cherche à étendre ces TEA à d'autres situations, on obtiendra des propositions différentes. Par exemple, si on passe non plus de l'instance p à l'instance $p+1$, mais de l'instance p à l'instance $p+a$, le premier TEA pourrait évoluer en $+a/+a/+a+1$ — « je rajoute a à chaque boucle puis je rajoute un à b_7 », alors que le deuxième pourrait évoluer en $+a/+a/+2a$. Selon le TEA considéré

comme valide par les élèves, la généralisation prendra des voies aboutissant sur des TEA invalides (première formulation) ou valides (deuxième formulation). Ici, il semble que les élèves mobilisent et confirment la validité du TEA +1/+1/+2 : chacune de leurs actions en est une mise en oeuvre sans hésitation. Lorsqu'il nous semble qu'un TEA est *plus probablement* validé qu'un autre, nous les différencierons par des tirets pour le moins probable, des traits pleins pour le plus probable. Nous verrons que, pour certains groupes, plusieurs TEA semblent être validés pour une même action.

Constructions de multiples instances consécutives valides Les élèves vont accumuler des faits validant ce TEA, attendu, $+1/+1/+2$, sur un total de seize instances consécutives, c'est-à-dire en construisant l'instance de mesure $p+1$ à partir de l'instance de mesure p. La consigne générale donnée aux élèves était de construire les instances permettant de tracer les TS5, TS6 et TS7, puis un TS11, suivi d'un TS17 ou TS19 ou TS21 avant de construire le script générique. Les élèves de ce groupe n'ont pas ici respecté cette injonction. D'autres groupes ont prolongé la création d'instances successives avant de passer au script générique, mais sans en créer autant :

— le groupe 46a a construit les instances jusqu'à 10, avant de passer à 15 puis 27 ;

— le groupe 46b a construit les instances jusqu'à 11, avant de passer à 22 ;

— le groupe 46c a, lui aussi, construit les instances jusqu'à 11, mais a poursuivi par l'instance traçant et dénombrant un TS15 ;

— le groupe 46d a aussi construit les instances jusqu'à 11, avant de passer au script générique ;

— le groupe 46g a construit les instances jusqu'à 10, avant de passer à 22 ;

— le groupe 46i a reconstruit les instances de 5 jusqu'à 11, mais seulement après avoir dans un premier temps construit les instances 5, 7, 11, 13 et 15 (ces dernières sans succès) ;

— le groupe 46k a construit les instances jusqu'à 8, avant de passer à 11 puis 21 ;

— le groupe 46m a construit les instances jusqu'à 9, avant de passer à 22 ;

— le groupe 45k a construit les instances jusqu'à 8, avant de passer à 19 ;

— le groupe 45i a, lui aussi, construit les instances jusqu'à 8, mais a poursuivi sur l'instance 11 ;

— le groupe 45e a construit les instances jusqu'à 10 ;

— le groupe 45d a construit les instances jusqu'à 8 ;

Notons que ces instances non attendues sont moins présentes dans la classe de 45 : la séance 4 ayant eu lieu après celle de la classe de 46, avec l'enseignante, nous avons décidé de mettre en question ce procédé de création systématique d'instances consécutives, en utilisant les scripts créés par le groupe 46d. *A posteriori*, cette épisode s'avère proche d'une utilisation des « caricatures » dans le CAP : on reprend les productions de certains groupes de sortes que ces productions montrent de façon homogène et impersonnelle certaines caractéristiques à discuter (Orange, 2012, p. 102). Ici, de par le dispositif, les productions sont homogènes et non identifiables. Cependant, nous avons improvisé cette discussion pour éviter que les élèves ne perdent du temps dans la création de multiples instances, et nous n'avons pas véritablement à chercher à les faire problématiser sur cette question : il s'agissait plutôt d'illustrer la lourdeur du travail dans ce cas. C'est pourtant en mettant au travail ce problème (faut-il créer chacune des instances, les unes après les autres ?) que nous aurions pu faciliter la construction de la nécessité d'une part de construire n'importe quelle instance, mais aussi de construire une méthode générique permettant de construire n'importe quelle instance.

Quoiqu'il en soit, le groupe 46d est capable d'enchainer les constructions d'instances, et on peut se demander jusqu'où seraient allés ces élèves s'ils avaient été présents la séance suivante. Auraient-ils continué à tester les instances les unes après les autres jusqu'à trouver la réponse au problème initial (trouver le plus grand TS constructible avec 1400 hexagones) ? Se seraient-ils rendus compte que le nombre d'hexagones croit faiblement à chaque instance, et donc que l'on peut faire des « sauts » pour accélérer la recherche ? [12] Auraient-ils cherché alors à résoudre le problème en construisant certaines instances spécifiques, comme les groupes 45a ou 45b ? La création d'une instance à partir de la précédente, pour ce groupe, finit en tout cas par ne pas être coûteuse, ni en terme d'effort cognitif (le TEA fonctionne), ni en termes de temps. Les instances 14 et 15 sont ainsi construites en une minute et trente secondes, la nécessité d'aller plus vite n'apparait pas.

Généralisation algébrique

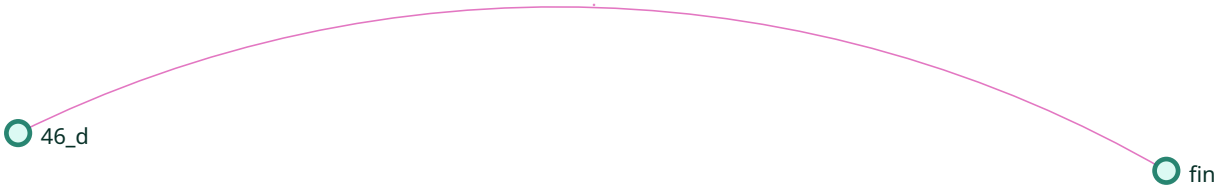


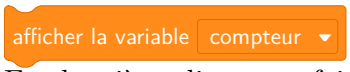
Figure 3.15 – Type de généralisation - 46d

Ce groupe n'a pas commencé à raisonner de façon algébrique, et n'a pas établi la nécessité d'une généralisation algébrique naïve. Cependant, une relation semble avoir été établie concernant la variable mesure.

La mesure n'est pas la mesure Le groupe 46d semble avoir éprouvé des difficultés lors de l'utilisation de l'en-tête du script générique. Celui-ci, contrairement aux autres en-têtes, ne fixe pas la mesure de façon arbitraire (choisie pour la situation), mais propose une invite :« quelle est la mesure du triangle de Sierpinski que tu veux dessiner ». La mesure est alors initialisée à la valeur entrée par l'utilisateur. Pour rappel [13], nous avons rendu invisible l'usage de l'instruction demander … et attendre en l'intégrant dans le bloc d'initialisation. Les élèves, lors de leurs premières rencontres avec l'invite, semblent ne pas savoir qu'en faire, puisqu'ils ne proposent aucune valeur et arrêtent l'exécution du script. Ainsi, lorsque les élèves dupliquent le script $TS(9)$ et le placent sous l'entête du script générique, ils se confrontent à l'irruption d'une demande qui n'est pas nécessaire, ni anticipée, dans le cadre de leur exploration. En testant différentes valeurs pour un script dont on peut supposer, à ce stade, qu'ils sont convaincus de sa validité [14], ils vont aboutir au fait « Pour que le tracé soit valide il faut entrer une valeur de 15 », cette valeur étant ensuite appliquée sur toutes les instances suivantes, et les deux tests (avec une valeur de 22 puis de 4) qui ont été rendus nécessaires pour cause de rétroaction ininterprétable de l'EPGB vont confirmer cette validité, appuyée ensuite par deux ultimes tests sur les dernières instances. Ces faits n'ont ici pas de lien avec le TEA, les élèves cherchent le sens de cette valeur qu'il faut entrer. Ils ne l'associent pas à la valeur de la mesure, c'est-à-dire au nombre d'hexagones d'un côté

12. Rappelons qu'au bout de seize instances, le plus grand nombre d'hexagones utilisés est de 141...
13. Voir la partie Analyse *a priori*, p. 140.
14. Il s'agit de la sixième itération de la mise en œuvre de leur TEA, qui n'a jusqu'ici pas été mis en défaut. De plus, si leur script avait été identifié comme invalide, les élèves auraient modifié des valeurs de nombres de répétitions de boucles, ou éventuellement touché à la structure du script, ce qui n'est pas le cas ici.

d'un triangle de base, car sinon la valeur entrée devrait bouger quand ce nombre d'hexagones d'un côté varie. Malgré le texte de l'invite, pour ces élèves la valeur entrée n'est pas la mesure. Cette valeur de 15 est seulement la valeur qui permet aux scripts de tracer les instances de façon « correcte », c'est-à-dire sans sortir de l'espace d'exécution, et sans que le tracé soit de trop petite dimension. D'autres groupes ont été confrontés à cette interprétation, et montrent une utilisation de la valeur entrée comme celle de la valeur du « zoom », ce que l'on constate aussi pour le groupe 45d. En fin de séance 4, les élèves atteignent la limite de leur valeur (le tracé sort légèrement de l'espace d'exécution). Ici, il est difficile de savoir si les élèves considèrent la valeur entrée comme devant être adaptée à la valeur de la mesure du TS voulu, ou s'il s'agit d'une valeur « magique » en quelque sorte, *la* valeur qui fait que ça marche.
La valeur entrée à l'invite n'est pas associée à la valeur de la mesure du TS voulu, mais elle est peut-être associée à la variable mesure — ou tout du moins à son affichage. Trois points nous semblent être des indices de cette conception.
En premier lieu, lors de leur première rencontre avec l'invite d'entrer une « valeur pour la mesure », les élèves semblent vouloir faire « disparaitre » cette invite en tentant d'annuler, en quelque sorte, l'affiche de la variable mesure : ils remplacent 

par 

(46d, S4, 5'36), donc *on n'affiche pas la mesure.*
En deuxième lieu, une fois la possibilité de faire disparaitre l'invite écartée, ces élèves sont passés par le test de leur $TS(9)$ sur l'entête du TS22 :

$$TS(9)[] \rightsquigarrow \left(\text{  } \text{, } 69, 22 \right)$$

Cette rétroaction montre une valeur de la mesure affichée (22) sans rapport avec la mesure du TS (9), et pourtant le tracé est valide : il ne parait ainsi pas y avoir de relation entre mesure et la valeur de la mesure du TS. En outre, la figure est « un peu petite », fait qui, combiné avec le fait qu'une valeur de 80 donne une figure bien trop petite, et une valeur de 4 bien trop grande, contribue à lier cette valeur entrée par l'utilisateur et le niveau de zoom, ou la dimension des hexagones. C'est effectivement le cas, puisque le bloc d'initialisation détermine la dimension des hexagones tracés en fonction de la valeur entrée.
En dernier lieu, en début de séance, l'enseignante a cherché à partager un fait avec la classe : lorsqu'on entre une valeur, cette valeur se retrouve stockée dans la variable *mesure* dont le contenu est visible. Ce fait est possiblement mobilisés par les élèves dans cet épisode.
Ainsi, même si cette relation entre valeur entrée, variable mesure et facteur d'agrandissement, est sinon erronée, du moins incomplète, on peut se demander si ce n'est pas un début de conceptualisation du paramètre : un objet du langage θ représente, réfère, dénote, la valeur d'une propriété liée au problème travaillé. Il aurait été ainsi intéressant de voir comment ce groupe aurait changé (ou pas) sa conception de ce que représente la valeur entrée en se confrontant a d'autres faits.
En tout état de cause, cette dissociation entre la mesure du TS tracé et la valeur de la variable mesure affichée est possiblement induite par une maladresse dans les tâches que nous avons prescrites aux élèves. En effet, pour le script correspondant au TS22, le commentaire lié, indiquant la tâche prescrite est :« Compléter pour un TS17 ; TS19 ou TS21 ». L'idée était d'inciter les élèves

à faire un « saut » vers une instance plus grande, sans que cette instance soit un multiple de la dernière instance tracée (11). Dans les faits, aucun élève n'a tracé le TS17 ni le TS19. Aucun n'a non plus tracé le TS21, même si on peut suspecter que certains élèves ont pensé le faire, suivant une règle implicite — et erronée — identifiant la mesure avec le nombre de répétitions de la première boucle (puisque pour un TS22, $b_1 = 21$). Cette tâche s'est donc avérée inutile. De surcroit, affichant « 22 » quel que soit le tracé effectué par le script, donc quelle que soit la valeur de la mesure du TS choisi, ce script peut amener à la construction de faux-faits dissociant valeur entrée et mesure du TS. Les tâches fixant la valeur de la mesure doivent prescrire la construction du script traçant le TS correspondant à la mesure.

Bilan

Le groupe 46d a accumulé des faits validant le TEA $+1/+1/+2$. Certains faits valident aussi la relation [valeur entrée par l'utilisateur] - [facteur de réduction de la taille des hexagones], tandis que d'autres, en partie issus de l'EPGB et en partie de la situation, valident l'absence de relation [valeur entrée par l'utilisateur] - [mesure du TS tracé]. À ce stade, les élèves de ce groupe ne peuvent poser le problème de la généralisation du script.

3.2.2 Groupe 45d

Le groupe 45d fait partie des rares groupes n'ayant pas stabilisé la construction d'une instance à partir de la précédente. Leur activité permet d'illustrer la construction progressive d'un nouveau TEA, ainsi qu'une évolution dans l'interprétation des rétroactions.

Description de l'activité

En trois séances, les élèves de ce groupe parviendront, après de nombreux essais, à construire quatre instances successives des scripts.

Construction progressive du $TS(5)$ (étapes 1-7) Les élèves du groupe 45d, contrairement à la plupart des autres groupes, commencent par modifier la première, deuxième et dernière boucle, en augmentant le nombre de répétitions de un. Le résultat est erroné, mais le premier triangle de base tracé est proche de l'attendu (figure 3.17a, p. 249). Ils modifient alors de la même façon les boucles restantes, toujours en ajoutant un, et en changeant de nouveau b_2, ce que nous soupçonnons être une erreur, car rien dans la rétroaction précédente ne permet d'expliquer cet ajustement. Le « trou » entre les côtés tracé par b_1 et b_2 lors de ce deuxième essai (figure 3.17b, p. 249), semble être considéré comme une manifestation de l'erreur sur b_2, qui sera corrigée lors de l'action suivante. Lors de cette même action, un ajustement de b_6 est effectué, lui donnant le même nombre de répétitions que les boucles B_2. Il est possible que la boucle b_6 ait été modifiée afin de prendre la même valeur que les boucles précédentes, mais le lien avec la rétroaction est difficilement visible : aucun indice d'erreur n'est à proximité du tracé de la boucle b_6 (voir figure 3.18, p. 249). La rétroaction produite montre un tracé dont la forme est valide, mais qui doit être considéré comme erroné puisque manifestant des hexagones se chevauchant (figure 3.17c, p. 249).
Le binôme va effectivement interpréter cette rétroaction comme la manifestation d'une erreur puisqu'ils vont agir pour modifier le tracé. En modifiant b_7 (moins un), et b_6 et b_8 (plus un), ils manifestent une localisation des erreurs qui s'affine — puisque localisée sur les boucles effectivement erronées —, mais qui reste imprécise. Le résultat produit (figure 3.17d, p. 249) est encore mieux interprété, puisque la longueur insuffisante du côté tracé par b_7 semble identifiée et correctement modifiée. Le côté b_8, pourtant lui aussi erroné, car trop long d'un hexagone,

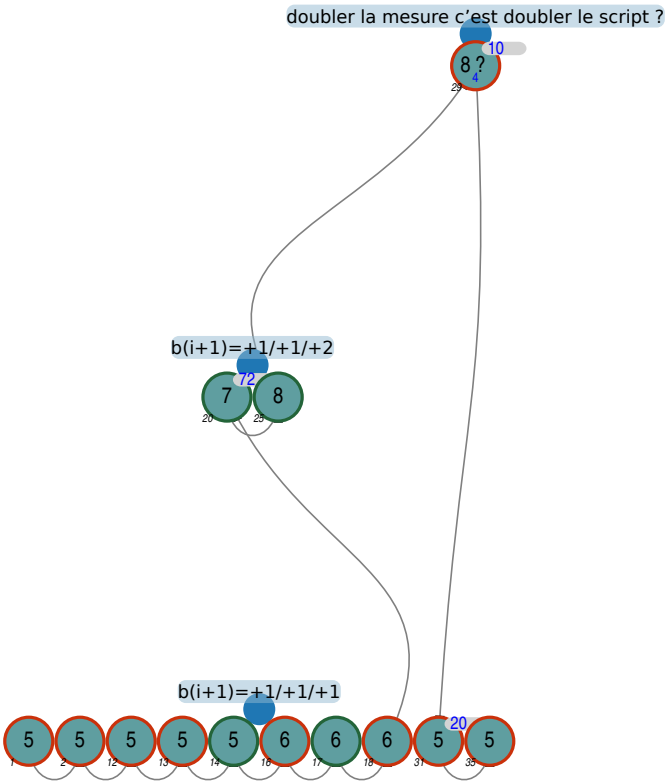
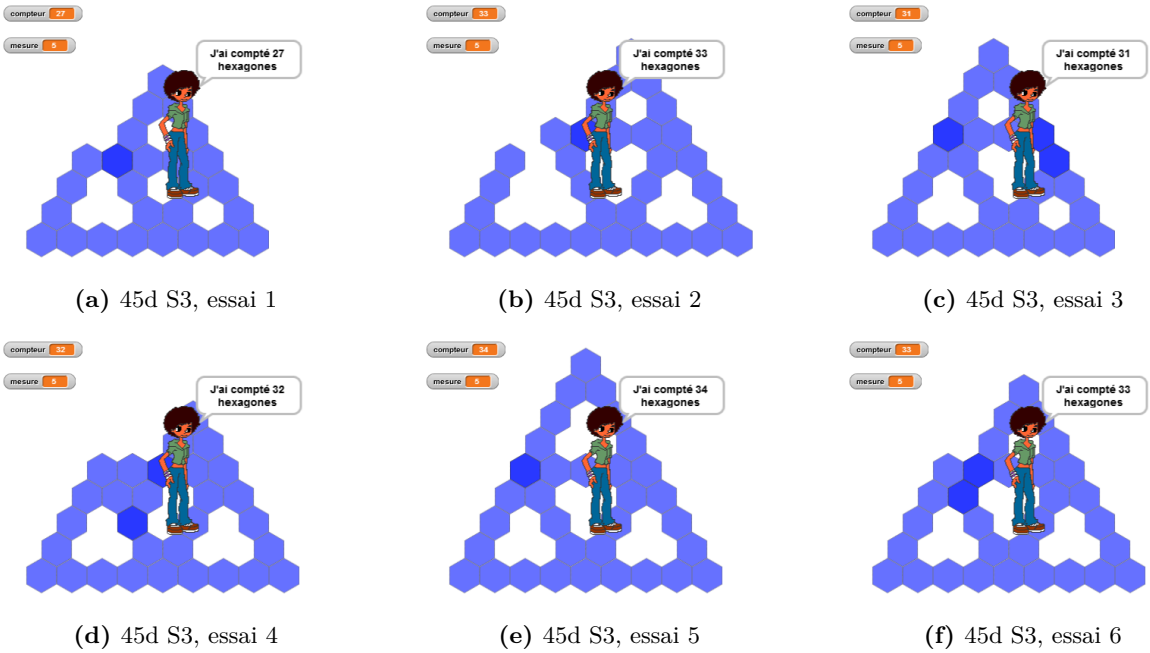


Figure 3.16 – Organisation de l'activité - 45d

n'est quant à lui pas modifié lors de cette phase. Il est possible que l'erreur soit identifiée et que son traitement soit repoussé, mais il est plus probable qu'elle ne soit pas identifiée : la boucle b_8 aboutit au tracé d'un côté de cinq hexagones, ce qui correspond à la mesure attendue. Cependant, c'est un hexagone de trop comme on le voit figure 3.17e (p. 249). Le groupe 45d va alors modifier la boucle b_7 avant de comprendre son erreur au vu de la rétroaction (figure 3.17f, p. 249) et d'obtenir un tracé finalement valide, en toute fin de séance. Les élèves, peu à peu, semblent préciser la bonne localisation et interprétation des erreurs.
Lors de la séance suivante, les élèves vont de nouveau chercher à construire un $TS(5)$ à partir du script $TS(4)$ (étapes 10-15), en commençant par ajouter un, et seulement à la boucle b_1. Suite à la rétroaction qui semble valider le tracé de cette boucle, ils vont propager cette modification de « plus un » sur l'ensemble des boucles excepté b_8. Une fois la correction faite, ils interpréteront correctement la rétroaction (figure 3.19a, p. 250) pour ajuster b_7 et ainsi obtenir un tracé de TS5 valide.

Construction progressive du $TS(6)$ (étapes 16-19) Pour passer au TS6, ce groupe va encore une fois ajouter un à chaque boucle, en plusieurs étapes : d'abord les boucles b_1 à b_4, puis b_5 et b_6, pour finir par b_7 et b_8. Une rétroaction similaire à celle de la figure 3.19a est de nouveau produite (figure 3.19b, p. 250) et correctement interprétée.



(a) 45d S3, essai 1 **(b)** 45d S3, essai 2 **(c)** 45d S3, essai 3

(d) 45d S3, essai 4 **(e)** 45d S3, essai 5 **(f)** 45d S3, essai 6

Figure 3.17 – Rendu des premiers tests - 45d, S3

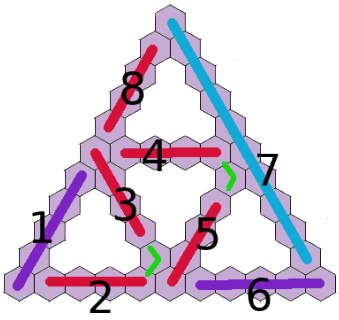
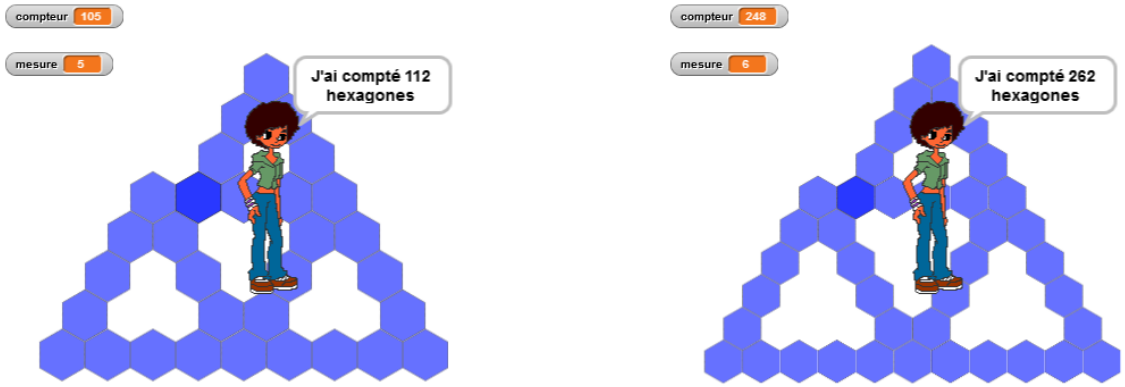


Figure 3.18 – Numéros de boucles et hexagones tracés (TS6)

Il est notable que le groupe 45d, sur toutes ces rétroactions, ne considère pas le nombre d'hexagones dénombrés comme le signe d'une erreur. Lorsqu'ils ont dupliqué les scripts, le bloc d'initialisation du compteur n'a pas été sélectionné, il n'est donc jamais remis à zéro. Les élèves considèrent malgré tout le TS5 qu'ils ont tracé comme étant valide, de même que leur TS6, puisqu'ils passent à l'instance suivante. Le nombre d'hexagones des TS5 et TS6 a pourtant été établi lors des séances précédentes [15], mais il n'est ici pas mobilisé.

Construction du $TS(7)$ et $TS(8)$ (étapes 20-25) Pour passer du TS6 au TS7, cette fois le binôme 45d modifie l'ensemble des boucles d'un coup avant de tester leur script, et ils le font de façon adéquate : plus un pour toutes les boucles, sauf b_7 qui est incrémentée de deux (étape 20). Cependant, ils ont construit ce script sur celui dont on attendait qu'il soit générique : le fait que le programme leur demande une valeur pour la mesure semble les perturber — tout comme

15. 33 pour le TS5 (145 affiché), 42 pour le TS6 (304 affiché).



(a) Erreur b_7 sur TS5 (45d, S4)

(b) Erreur b_7 sur TS6 (45d, S4)

Figure 3.19 – Deux rétroactions courantes - b_7 et compteur erronés (45d, S4)

le groupe 46d —, puisqu'ils lancent le script $TS(n)$ sept fois de suite, en l'interrompant lors de la demande d'entrée (figure 3.20a, 45d, S4, 9'51-10'26). Ils testeront ensuite avec différentes entrées. D'abord 72, qui fournit une rétroaction dont manifestement la dimension, la valeur du « zoom », est erronée. Suivront, avec les mêmes effets, les valeurs 200 puis 50. Ils termineront par tester 10, et cela semble être considéré comme valide (figure 3.20c, p. 250) puisqu'ils passent alors à la construction de l'instance suivante, le TS8. Ce script, construit sur celui dont l'en-tête invitait à tracer un TS11, est valide du point de vue du tracé dès la première modification globale de l'ensemble des boucles. Là encore, le nombre d'hexagones dénombrés par la variable *compteur* — qui continue d'augmenter — ne semble pas induire de signification d'erreur, pas plus que la dimension du tracé — qui occupe une bonne partie de la zone d'exécution —, ni l'affichage de la valeur de la mesure — 10 pour un TS7, 11 pour un TS8. La valeur entrée pour le script générique semble ici liée au zoom et non à la mesure du TS.

(a) Première exécution avortée du $TS(n)$

(b) Un zoom invalidant

(c) Un zoom validant

Figure 3.20 – Un script $TS(n)$ perturbant (45d, S4)

Exploration pour le script générique En début de séance 5, le groupe 45d va commencer par dupliquer le script $TS(4)$ sur le script générique, et le lancer avec une valeur entrée identique à celle de la mesure du TS attendu (4), qui produira une rétroaction validante, et interprétée comme telle. À la suite de quoi, ces élèves vont de nouveau dupliquer le $TS(4)$ et le rajouter à la suite du script $TS(n)$, avant de tester une valeur entrée de 10, soit proche du double de la

mesure du TS original. Le résultat étant visiblement invalide, mais difficilement interprétable (figure 3.21a, p. 251), ils vont tester avec une autre valeur en entrée (1), ce qui ne rend pas la rétroaction plus facilement interprétable (figure 3.21b, p. 251). Ils vont alors modifier la première partie du script en ajoutant un aux huit premières boucles, qu'ils testeront cette fois avec 20 comme entrée. La difficulté d'interprétation du résultat (figure 3.21c, p. 251) débouche sur une « remise à zéro » (RAZ), en rechargeant le programme de la séance.
Ils vont cette fois modifier directement le script $TS(4)$ en ajoutant un à chaque boucle, puis en corrigeant b_7. Après avoir dupliqué ce script sur l'en-tête du script générique, le binôme va alors tester ce script en entrant différentes valeurs : 10, 155, 5000, 40, 5000, pour finir par 12 en fin de séance.

(a) **(b)** **(c)**

Figure 3.21 – Rétroaction pour un script « doublé » (45d, S5)

Recherche de régularités

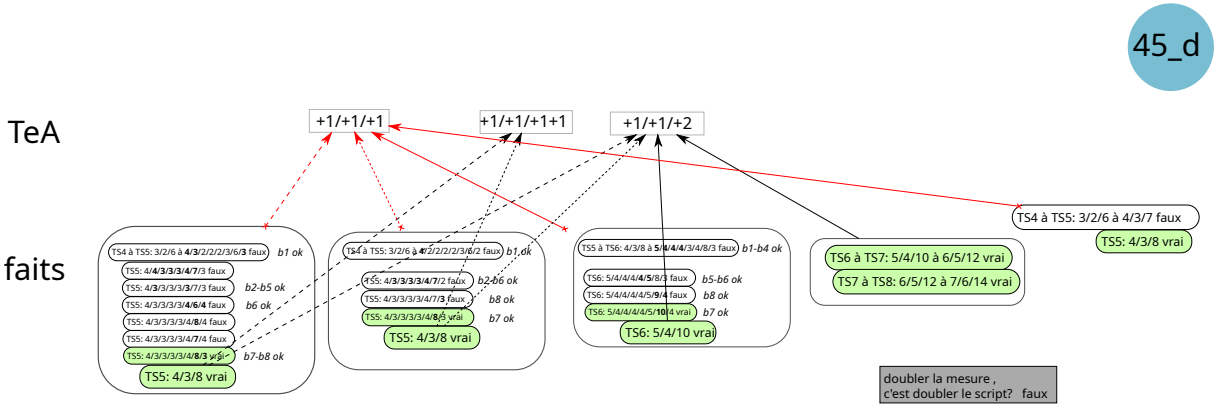


Figure 3.22 – Faits et TEA, groupe 45d

Si l'on s'intéresse aux faits et TEA construits par le groupe 45d (figure 3.22, p. 251), plusieurs remarques sont à faire :

— peu de faits concernant les instances des scripts sont construits ;

— l'organisation de l'activité des élèves permettant d'aboutir à une version d'un script connaissant la précédente semble très peu stabilisée.

En outre, la description de l'activité des élèves semble montrer que de nombreuses actions sont des ajustements progressifs du script, des réactions à la rétroaction, et non la mise en œuvre d'un TEA permettant de modifier globalement un script.

Comprendre la rétroaction et diviser le script Dans la représentation des faits et TEA construits par les élèves (figure 3.22, p. 251), nous avons rajouté pour ce groupe des faits premiers d'une granularité plus fine que pour d'autres analyses, en précisant les boucles modifiées (en gras), leurs valeurs et la validité de ces valeurs.
Pour la plupart des groupes, les élèves modifient l'ensemble des boucles d'un script (en traitant éventuellement d'une façon différente et décalée la boucle b_7), puis ajustent rapidement les éventuelles erreurs présentes. Dans le cas du groupe 45d, les modifications du script ne sont pas globales, mais par parties. Pour les trois premiers scripts construits, l'enchainement des boucles traitées est le suivant, erreurs mises à part (étapes 1 à 19) :

1. $b_1b_2b_8 \rightarrow b_2b_3b_4b_5b_6b_7$
2. $b_1 \rightarrow b_2b_3b_4b_5b_6b_7 \rightarrow b_8$
3. $b_1b_2b_3b_4 \rightarrow b_5b_6 \rightarrow b_7b_8$

Ainsi, la partition du script n'est pas stable, et ne semble pas correspondre à une partition organisée du tracé. On aurait pu diviser le traitement du script afin d'observer et de vérifier la construction des différents triangles de base, par exemple. Ici, si les élèves cherchent sans doute à vérifier leurs premières actions avant de passer aux suivantes, le moment choisi pour effectuer les tests n'apparait pas clairement. Cela témoigne probablement des doutes des élèves — qui veulent vérifier à intervalles variables la validité de leurs actions —, mais aussi de leur difficulté initiale à interpréter les rétroactions ainsi que de l'évolution de cette capacité. Ainsi, il aura fallu :

— sept modifications du premier script pour deux parties, avec des erreurs sur les boucles b_2, b_6, b_7 et b_8 ;
— quatre modifications du deuxième et troisième script pour trois parties, avec des erreurs seulement sur b_7.

En outre, pour le premier script, les six rétroactions (étapes 2 à 7) ont été suivies de trois modifications erronées, une correction partielle et partiellement erronée, une correction partielle totalement valide et pour finir une correction totale et valide. Pour les deux autres scripts en revanche, il n'y a pas eu de correction invalide : toutes les réactions des élèves à la rétroaction ont été valides. La construction du premier script semble ainsi avoir permis aux élèves de construire des faits premiers de validation adéquats : l'observation des erreurs suite aux modifications faites, combinées aux réussites, permet de finir par identifier les parties tracées par chacune des boucles. Cependant, la capacité des élèves à valider un script semble se limiter aux faits premiers « le tracé est valide » et « le tracé n'a ni trou ni chevauchement ». Le nombre d'hexagones dénombrés, tout comme le lien entre la mesure et le tracé, ne semblent pas être pris en compte.

Invalider et révolutionner des TEA Cette division de l'activité rend difficile l'ajustement des TEA mis en œuvre par les élèves. En effet, le TEA qui semble se manifester est celui, répandu, consistant à ajouter un à chacune des boucles : *si on augmente la mesure de 1, on augmente chaque boucle de 1.* Ce TEA, qui chez d'autres groupes évolue après une ou deux instances construites, reste en vigueur pour trois instances différentes successives avant d'aboutir au TEA $+1/+1/+2$ en fin de séance 4, et réapparait en début de séance 5. Tout se passe comme si ce TEA restait « plus ou moins » valide. En fait, on retrouve, à l'échelle de la communauté scientifique formée par les deux élèves, le problème de la réfutation d'un paradigme scientifique. En effet, comme le rappelle Juignet, pour Kuhn il est impossible de simplement invalider de façon définitive un paradigme sans qu'un autre paradigme puisse le remplacer, car « rejeter un paradigme sans lui en substituer simultanément un autre, c'est rejeter la science elle-même »(Kuhn, Meyer et Luminet, 2018, p. 117). Ici, le TEA $+1/+1/+1$ est invalidé par l'action, mais comme les élèves

ne disposent pas encore d'un TEA qui lui serait substituable, ils en maintiennent la mobilisation. Chaque nouveau fait de construction valide (« TS5 : 4/3/8 vrai », « TS6 : 5/4/10 vrai ») semble renforcer la probabilité que le TEA $+1/+1/+2$ soit vrai. Lorsque cette accumulation est suffisante, le nouveau TEA devient plus acceptable que l'ancien, mais ici sans que l'ancien ne disparaisse complètement : la « révolution scientifique » qu'aurait constitué l'abandon du TEA $+1/+1/+1$ au profit de $+1/+1/+2$ n'a pas totalement eu lieu. Cette absence de théorie de remplacement, en quelque sorte, peut aussi expliquer la partition du script qu'opèrent les élèves : peut-être, à un moment donné, s'agit-il non plus de « tester si ça marche », mais plutôt d'« identifier ce qui ne marche pas ». Ainsi, à chaque fait de construction valide, « rajouter un à b_7 puis encore un » (que nous notons $+1/+1/+1+1$) semble aboutir sur une construction valide : de façon simultanée, le TEA $+1/+1/+1$ est en partie invalidé, et les TEA $+1/+1/+1+1$ et $+1/+1/+2$ sont, eux, en partie validés. Nous verrons que chez certains groupes, c'est le TEA $+1/+1/+1+1$ qui sera mis en œuvre dans un premier temps. Ce groupe 45d a bien observé une régularité : à chaque fois que l'on fait plus un et encore plus un pour la boucle b_7, le tracé devient valide. Cette régularité n'est pas forcément formalisée ou explicitée, mais ici elle aboutit au raccourci — efficace — « ajouter 1 puis encore 1, c'est rajouter 2 ». On peut se demander, étant donné que le fait numérique « ajouter un et encore ajouter un, c'est comme ajouter deux » est sans doute connu des élèves, pourquoi ceux-ci ne passent-ils pas plus vite à ajouter directement deux à b_7. Comme Radford (2006a, p. 5), nous pensons que pour généraliser de façon algébrique, il faut observer une régularité et être conscient que ce point commun s'applique à tous les termes de la séquence d'instances étudiées (avant de la formaliser). Ici, non seulement la régularité est peut-être difficilement observable à partir de deux constructions d'un même script (TS5), mais aussi il n'est pas évident que, pour ces élèves, cette régularité soit valable pour tout passage d'une instance à la suivante. Ces élèves semblent tout juste initier un processus de généralisation.

Généralisation algébrique

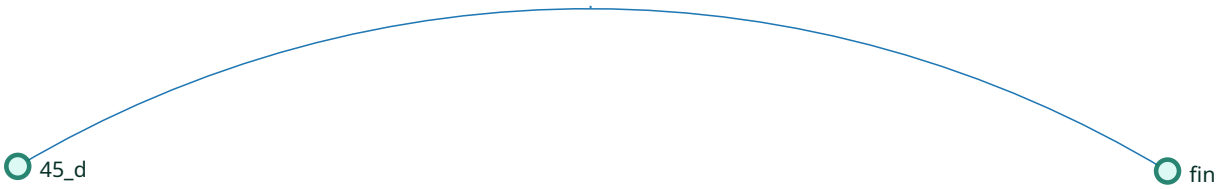


Figure 3.23 – Type de généralisation - 45d

La généralisation que nous observons chez ces élèves est limitée, lors de ces séances, au passage d'une instance à la suivante. Ils semblent en mesure de construire une instance à partir de la précédente (généralisation arithmétique), mais ils ne sont pas encore capables de construire directement une certaine instance (ce qui serait une manifestation d'une généralisation algébrique naïve). *A fortiori* ils ne peuvent pas formaliser de méthode de construction pour une quelconque instance. On peut se demander si les élèves ont bien posé le problème : ont-ils pris note de la contrainte qu'un même script doit tracer n'importe quelle instance ? Les tests effectués en fin de séance 5 en exécutant le script $TS(n)$ en entrant des valeurs variables (4, puis 10, 155, 5000...) pourraient le laisser penser. Cependant, on a vu que cette valeur entrée par l'utilisateur semblait plutôt associée à la valeur de l'agrandissement ou de la réduction de la dimension des hexagones. Le lien [mesure entrée] - [mesure du TS tracé] est visible une seule fois en début de séance 5.

Ensuite, après avoir « doublé » (littéralement) le script traçant le TS4 [16], ces élèves le testent avec une valeur entrée de 10, ce qui est certes proche du double de la mesure initiale, mais, n'étant le double exact, cela laisse à penser que c'est une valeur arrondie permettant au tracé d'être entièrement représenté dans l'espace d'exécution.

Bilan

Pour ce groupe, qui réussit tout juste une généralisation arithmétique encore hésitante et non stabilisée en séance 5, il est difficile de prétendre qu'une des nécessités du problème a bien été construite. Cela pose la question de la limite d'une situation se basant exclusivement sur les rétroactions, accompagnée de moments de bilan en groupe classe. Ici, les élèves n'avancent pas suffisamment dans les différentes explorations, et les faits « partagés » en début de séance avec l'ensemble de la classe, ne sont pas mobilisés : ce groupe ne construit pas suffisamment de faits sur lesquels construire le problème, ou ne pose pas correctement le problème.

3.2.3 Synthèse groupement 1

Les analyses des groupes sélectionnés du groupement 1 — pas de généralisation — font apparaitre quelques éléments notables.
En premier lieu, les interprétations des rétroactions par les élèves semblent se baser en priorité sur la forme tracée : est-ce un TS bien formé ou non ? Les groupes étudiés ici prennent en compte les trous ou chevauchements comme des signes d'erreur, qui invalident leur tracé, et leur script. Cependant, la dimension globale du tracé, son inscription ou non dans l'espace d'exécution et la place occupée dans celui-ci, ne semble pas toujours prise en compte : à part lorsque cette dimension est visiblement problématique, avec un tracé peu visible ou un tracé qui est presque totalement en dehors de l'espace d'exécution, elle n'est pas prise en considération pour la validation. De même, la validation ne semble pas dépendre, chez ces élèves, de l'adéquation entre la mesure affichée (ou prescrite) et le nombre d'hexagones d'un triangle de base de la figure tracée. Le nombre d'hexagones dénombrés, au moins pour le groupe 45d, n'est pas non plus mis en lien avec le TS tracé, et n'est pas un critère d'invalidation.
Cette non construction de certains faits premiers de validation est possiblement due au problème que se posent les élèves : ces deux groupes, tout comme le groupe 45n qui complète ce groupement, paraissent avoir pour préoccupation première de construire une suite d'instances de TS bien formés. Le nombre d'instances construites est très variable : seulement deux tentatives non concluantes (TS5 et TS6) pour le groupe 45n en trois séances, quatre pour le groupe 45d en trois séances, et treize instances successives pour le groupe 46d, sur seulement deux séances. Le nombre d'instances construites n'est pas en lien avec l'avancée dans le problème : aucun de ces groupes ne pose le problème de la généralisation du script, et aucun n'a cherché à construire une certaine instance particulière. Il s'agit ici pour les élèves de construire une instance à partir d'une autre proche, ce que Radford (2008, p. 3) nomme une « généralisation arithmétique ».
Afin de faire cette généralisation arithmétique, les élèves de ces trois groupes — comme la plupart des autres groupes — ont commencé par appliquer le TEA $+1/+1/+1$: « Pour passer d'une instance à la suivante, j'augmente la valeur du nombre de répétitions de chacune des boucles de un ». La non validité de ce TEA va amener les élèves des groupes 46d et 45d à remplacer par le TEA valide $+1/+1/+2$: « Pour passer d'une instance à la suivante, j'augmente la valeur du nombre de répétitions de chacune des boucles de un sauf b_7 que j'augmente de deux ». Cependant, malgré l'invalidité constatée du TEA $+1/+1/+1$, les élèves du groupe 45d ne vont pas le remplacer

16. C'est-à-dire après avoir dupliqué le script et avoir copié ces blocs à la suite du script initial.

immédiatement : il leur faudra une nouvelle application invalidée de ce TEA pour le remplacer. Pour le groupe 45n, ce TEA restera en vigueur sur les trois séances. Ainsi, l'invalidation d'un TEA n'implique pas systématiquement son éviction : il faut une alternative pour le remplacer, et, suivant les élèves, le TEA erroné peut être plus ou moins résistant, c'est-à-dire qu'il faudra plus ou moins de faits construits avant de l'invalider totalement au profit d'un autre TEA.
On peut noter, en dernier lieu, que le groupe 45d a utilisé une méthode consistant non à modifier l'ensemble du script, mais à le modifier par partie. S'il a prolongé l'utilisation du TEA malgré une première invalidation, cette façon de procéder, en divisant le problème, semble leur avoir permis de concevoir un TEA valide et stable : les constructions suivantes se feront sans erreurs.

3.3 Groupement 2

3.3.1 Groupe 45a

Le groupe 45a est un binôme de deux garçons réputés peu à l'aise en mathématiques selon l'enseignante de la classe. Leur activité témoigne de la difficulté à construire des nécessités ou des savoirs en ne se basant que sur des faits invalides, c'est-à-dire sur des faits invalidant une hypothèse ou un TEA : sur les trois séances, seule l'exécution du script donné $TS(4)$ donnera une rétroaction valide.

Description de l'activité

Le groupe 45a, sans avoir obtenu une version valide du $TS(5)$, va néanmoins tenter de construire des instances beaucoup plus grandes, notamment pour tracer et dénombrer un TS174.

Un semblant d'exploration Les élèves commencent par exécuter de façon répétée le script fourni. Cette action pourrait être considérée comme une exploration du script, en essayant d'arrêter son exécution à certains moments pour comprendre son exécution. Cependant, ces exécutions multiples sont issues de frappes répétitives de la touche « a », qui lance l'exécution du script $TS(4)$ et simultanément stoppe une exécution préalable de ce même script. Les élèves procèdent ainsi à 13 frappes en moins de dix secondes, ce qui ne leur permet de produire que les rétroactions suivantes :

$$TS(4)[] \rightsquigarrow \left(\text{  } \quad , [], 4 \right)$$

et

$$TS(4)[] \rightsquigarrow \left(\text{  } \quad , [], 4 \right)$$

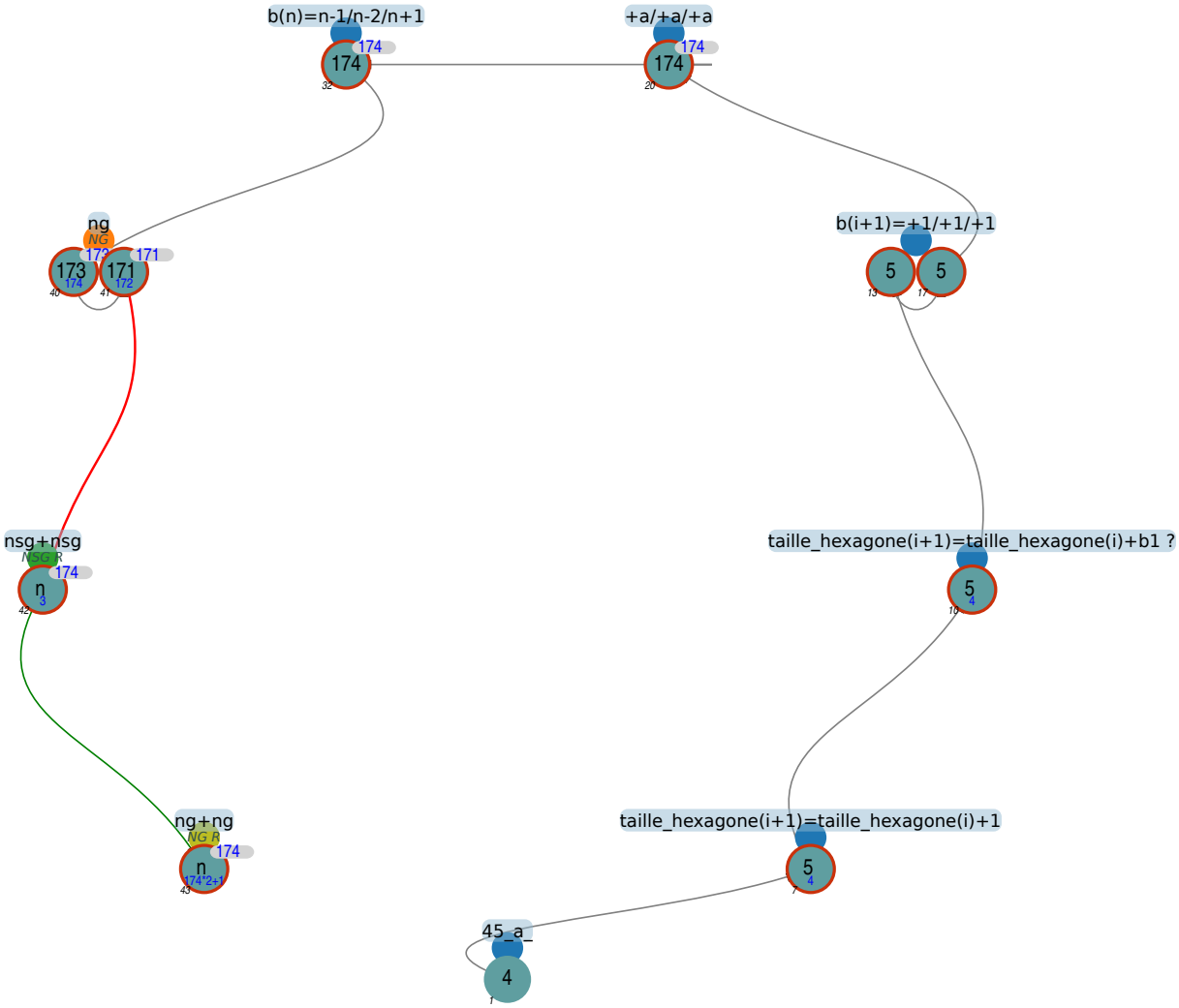
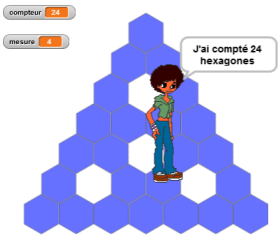


Figure 3.24 – Organisation de l'activité - 45a

Au bout de ces treize exécutions, ils laissent le script se dérouler jusqu'au bout, ce qui aboutira à la seule rétroaction valide des trois séances de généralisation :

$$TS(4)[] \rightsquigarrow \left(\text{  } \text{} , 24, 4 \right)$$

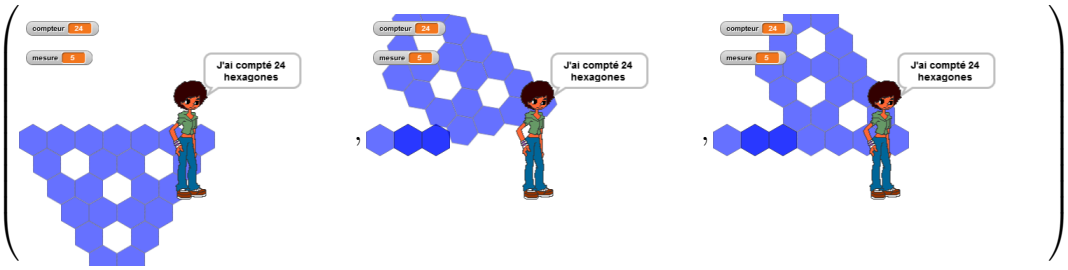
Une réelle exploration du déroulé du script aurait été illustrée par des arrêts à différents moments de l'exécution, ce qui est rendu possible par la relative lenteur de ce premier script, incluant des pauses à chaque tracé d'hexagone. Selon nous, tout au plus s'agit-il d'une exploration ludique du comportement de l'EPGB lors de l'appui répété d'une touche déclenchant une exécution de script.

Les élèves dupliquent ensuite le script, comme la consigne leur en a été donnée en début de séance. Ils réitèrent alors les multiples lancements, cette fois en pressant 9 fois la touche « b » (qui produit l'exécution du script $TS(5)$ qu'ils doivent construire) en l'espace de 6 secondes. Là encore, difficile d'y voir une exploration organisée du script. Ils laissent finalement le script se dérouler. Comme nous considérons qu'à ce moment de la séance, ils répondent à la consigne initiale de dupliquer le $TS(4)$ pour construire un $TS(5)$, ils produisent le fait premier suivant :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 24, 5 \right)$$

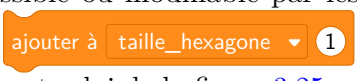
Cette rétroaction semble bien être considérée comme invalide, puisque les élèves vont à partir de ce moment s'attacher à modifier le script, sans doute pour répondre à leur but d'obtenir un script permettant de tracer et dénombrer un TS5. Pour cette première phase de leur activité de modification du script, on peut observer trois étapes distinctes.

Construction du TS5 : changement de la structure Dans cette première étape, le binôme va modifier la structure du programme, en modifiant les instructions « tourner de ... ». Ils obtiendront ainsi les trois affichages suivants :



Après avoir remis les valeurs initiales, ils vont relancer l'EPGB.

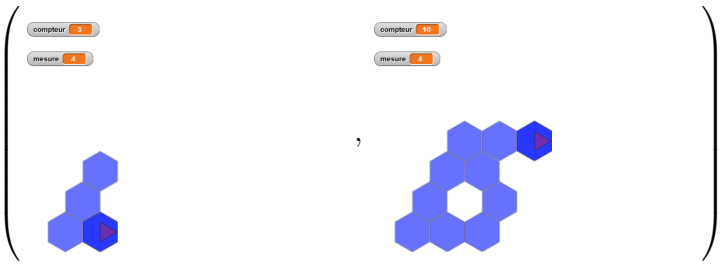
Construction du TS5 : changement de la taille des hexagones Lors de cette deuxième tentative, les élèves du groupe 45a vont agir sur une variable[17] : « taille_hexagone ». Cette variable est définie dans le bloc d'initialisation et permet d'ajuster la dimension des hexagones tracés en fonction de la mesure, afin que l'ensemble du tracé tienne sur l'écran d'affichage. Ce bloc d'initialisation n'est pas accessible ou modifiable par les élèves.

Ils vont ainsi insérer l'instruction 

en début de chaque boucle du script $TS(4)$. Le programme obtenu est celui de la figure 3.25a. Cette modification de l'ensemble des boucles est suivie d'une exécution dont la rétroaction montre clairement un caractère erroné (figure 3.25b, p. 258). Les élèves vont alors remettre le programme dans sa version initiale (le script $TS(4)$), et vont — contrairement au début de la séance — explorer le script solution

17. Rappelons que la variable informatique a été découverte et définie lors de la séance précédente, avec la variable « compteur » (compteur).

en l'arrêtant à des moments choisis, en l'occurrence la fin du tracé de la boucle b_1 puis la fin de b_3, obtenant les deux rétroactions suivantes :



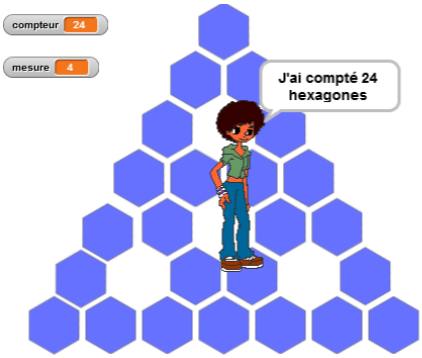
Ils rajoutent alors, avant la boucle 1 et avant la boucle 2, l'instruction 

,

Block_388
Quand a est pressé
initialisation4
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter 3 fois (commandes)
.....ajouter à taille_hexagone 1
.....tracer
tourner de 120 degrés à gauche
répéter 2 fois (commandes)
.....ajouter à taille_hexagone 1
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter 2 fois (commandes)
.....ajouter à taille_hexagone 1
.....tracer
tourner de 120 degrés à droite
répéter 2 fois (commandes)
.....ajouter à taille_hexagone 1
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter 2 fois (commandes)
.....ajouter à taille_hexagone 1
.....tracer
tourner de 120 degrés à gauche
répéter 3 fois (commandes)
.....ajouter à taille_hexagone 1
.....tracer
tourner de 120 degrés à gauche
répéter 6 fois (commandes)
.....ajouter à taille_hexagone 1
.....tracer
tourner de 120 degrés à gauche
répéter 2 fois (commandes)
.....ajouter à taille_hexagone 1
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

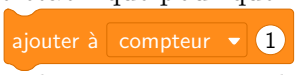
(a) Version avec « taille_hexagone+1 »

(b) Exécution avec « taille_hexagone+1 »



(c) Exécution avec « taille_hexagone+3 »

Figure 3.25 – 45a, S3 : utilisation de « taille_hexagone »

obtenant à l'exécution le résultat lui aussi visiblement erroné de la figure 3.25c (p. 258). À ce stade, il nous semble que les élèves n'ont pas encore posé le problème : ils semblent reconvoquer les faits construits dans la séance précédente. Ces faits ont été partagés en début de séance 3, où il a été établi que pour que le programme « apprenne à dénombrer les hexagones », une instruction 

devait être ajoutée pour chaque tracé d'un hexagone, ce qui de fait inclut une insertion dans chaque boucle. Leur script donné figure 3.25a (p. 258) est donc très semblable, au changement de la variable près. La modification effectuée ensuite, en ajoutant le nombre d'hexagones supposés être tracés par une boucle, est lui-aussi caractéristique de ce qu'il s'est déroulé en séance 2.
Ce n'est que suite à cette deuxième rétroaction erronée que les élèves vont revenir sur le problème initial : dupliquer le script traçant un TS4 et le modifier pour qu'il trace un TS5.

Construction du TS5 : changement du nombre de répétitions des boucles Après duplication, le groupe 45a va modifier chacune des boucles en augmentant son nombre de répétitions de un, ce qui produira la rétroaction :

$$TS(5)[] \rightsquigarrow \left(\text{  }, 32, 5 \right)$$

Cette rétroaction est bien vue comme invalidante, mais les élèves identifient la boucle b_8 comme étant erronée (et de fait, c'est son tracé qui produit la visualisation de l'erreur). Ils ajusteront b_8 afin de supprimer le chevauchement et obtiendront ainsi en fin de séance :

$$TS(5)[] \rightsquigarrow \left(\text{  }, 31, 5 \right)$$

En début de séance 4, ce binôme va réitérer la duplication suivie de l'augmentation des nombres de répétitions de un. Cependant, ils ne vont pas réagir à la rétroaction en modifiant le script, mais plutôt en changeant de but : ils vont alors modifier le script pour lequel on attend une généralisation, en le testant presque systématiquement pour une *mesure* de 174. La consigne rappelée en début de séance était pourtant de construire différentes instances avant de passer au script générique.

Construction de l'instance 174 : compenser par une boucle En dupliquant leur script $TS(4)$ sur $TS(n)$, les élèves ont omis de copier l'instruction d'initialisation du compteur, ce dont ils vont se rendre compte lors du premier test (effectué donc avant toute modification). En entrant « 174 » comme valeur de la mesure du TS voulu, ils obtiennent :

$$TS(n)[174] \rightsquigarrow \left(\text{  } \text{} , 88, 174 \right)$$

Ils ajoutent alors l'instruction d'initialisation du compteur pour obtenir une rétroaction identique à leur précédent test sur $TS(5)$, à une homothétie près.

$$TS(n)[174] \rightsquigarrow \left(\text{  } \text{} , 32, 174 \right)$$

On peut ainsi constater que ces élèves semblent prendre en compte tout à la fois le tracé, les chevauchements et le nombre d'hexagones dénombrés pour établir la validité ou non de leur script. Ils vont alors modifier l'ensemble des boucles en leur affectant comme nombre de répétitions $\{173/172/176\}$ [18], ce qui produira :

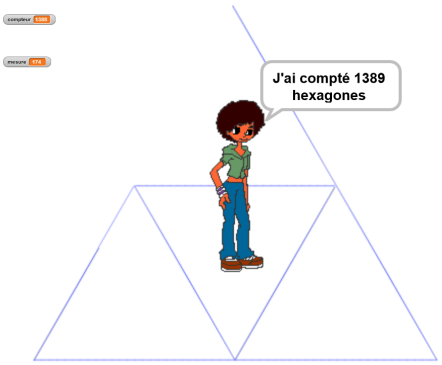
$$TS(n)[174] \rightsquigarrow \left(\text{  } \text{} , 1384, 174 \right)$$

L'identification de b_7 comme étant erronée est certes faite, mais la modification ($b_7 \leftarrow 178$, soit une compensation en ajoutant deux) ne produit pas un résultat valide. Après avoir appliqué $\{b_6 \leftarrow 174, b_8 \leftarrow 174\}$ sans succès, le binôme va de nouveau avoir recours à un changement dans la structure du programme, tout d'abord en cherchant à « aligner » les côtés tracés par b_7 et b_8, en effectuant une rotation de 180° puis 0° entre ces deux boucles. Ils obtiennent ainsi le résultat rendu figure 3.26b (p. 261). Après quelques tâtonnements, ils ajouteront une neuvième boucle ($b_9 \leftarrow 174$) pour tracer le côté manquant, ce qui aboutit à un résultat proche de celui attendu (programme 3.26a, p. 261, figure 3.26c, p. 261). Les élèves ont donc fait tracer trois côtés du triangle de base par trois boucles dont le nombre de répétitions est la valeur de la *mesure*. Le résultat valide aurait dû être dans ce cas pour ces trois boucles $\{b_7 = 173, b_8 = 173, b_9 = 172\}$.

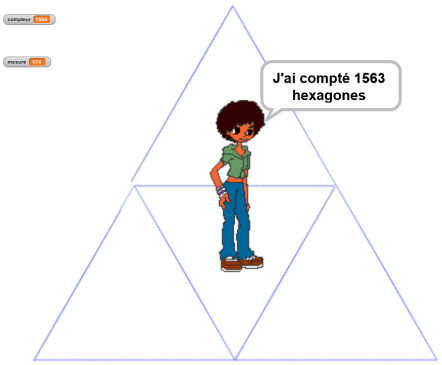
18. Donc $b_1 = b_6 = 173$, $b_2 = b_3 = b_4 = b5 = b_8 = 172$ et $b_7 = 176$.

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter 173 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 172 fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter 172 fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter 172 fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter 172 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 174 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 178 fois (commandes)
.....tracer
tourner de 0 degrés à gauche
répéter 174 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 174 fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(a) Version TSn avec alignement b_7b_8 et ajout d'une boucle b_9, S4



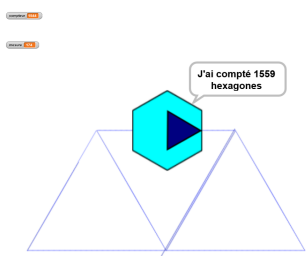
(b) Alignement b_7b_8



(c) Ajout d'une boucle b_9

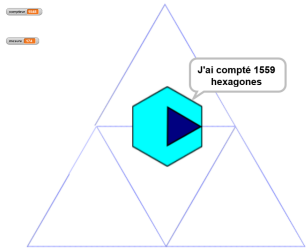
Figure 3.26 – 45a, S4 : une boucle pour un côté

En séance 5, ce groupe va de nouveau repartir sur la construction de l'instance 174 à partir du script générique. Ils fixent ainsi le script à $\{173/172/175\}$, se différenciant ainsi de la séance précédente avec $b_7 \leftarrow 175$. Là encore, ils vont modifier la structure, tout d'abord en tentant d'ajouter une instruction avancer de 1000 pas en alignement avec b_8, puis en rajoutant une boucle b_9 alignée avec b_8, dont la rétroaction est :

$$TS(n)[174] \rightsquigarrow \left(\text{  } \text{} , 1559, 174 \right)$$

Le tracé sortant de l'écran, ils corrigent alors pour revenir à une version proche de celle obtenue en fin de séance 4 (programme 3.26a, p. 261), avec $b_7 = 175$ et $b_6 = b_8 = b_9 = 174$. Ils obtiennent ainsi :

$$TS(n)[174] \rightsquigarrow \left(\quad , 1559, 174 \right)$$



Le groupe va alors corriger b_6 puis ajuster ensemble les boucles b_7 et b_8 — en leur donnant à chaque fois la même valeur, jusqu'à obtenir le script 3.1 (p. 263), produisant la rétroaction figure 3.28a (p. 264). La nouvelle organisation du script est alors celle de la figure 3.27 (p. 262).

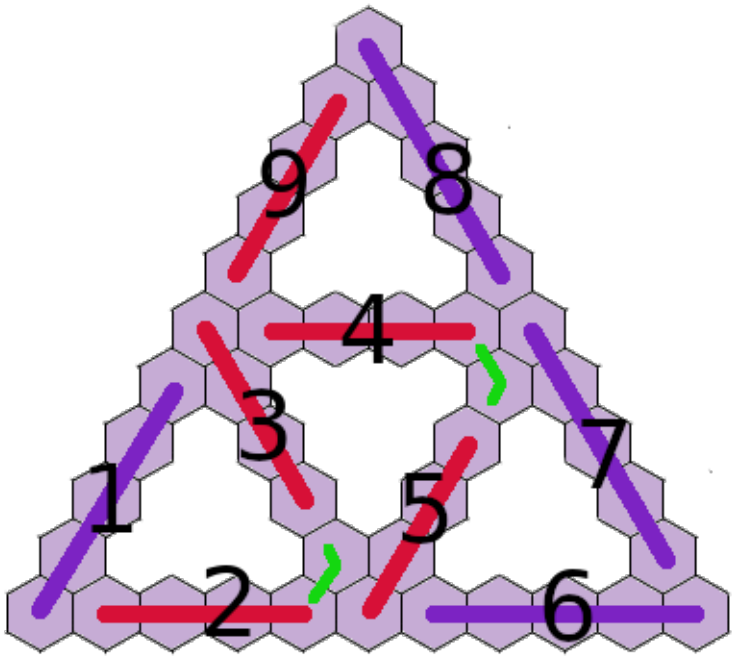


Figure 3.27 – Organisation du script pour une boucle - un côté

Cette version trace correctement le « grand côté » devant être tracé originellement par $b7$, puisque $b_7 + b_8 = (174 - 1) \times 2$. Cependant, b_9 devrait être à 172, il est ici de 174. Mais le chevauchement produit par la fin du tracé de b_9 sur b_1 est peu visible (voir figure 3.28b, p. 264). Ainsi, ce tracé est sans doute ici considéré comme valide, puisque les élèves vont changer de but, cherchant à construire l'instance 173 en effectuant $\{B_1 \leftarrow 172, B_2 \leftarrow 171, b_7 \leftarrow 175, b_8 \leftarrow 171\}$ (la boucle b_9 reste inchangée). Les élèves procèdent alors à un test, mais au lien de rentrer la valeur « 173 », une erreur de frappe leur fait entrer « 17" ». Nous considérons que l'entrée voulue était bien 173, et donc qu'ils cherchaient bien à construire l'instance 173, car sur un clavier Azerty, les guillemets doubles sont obtenus par la même touche que le chiffre 3. Cependant, cette erreur

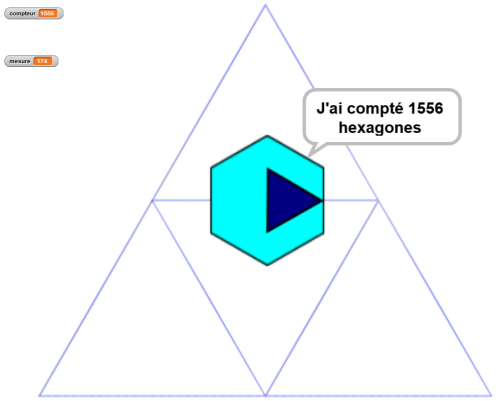
```
Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter 173 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 172 fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter 172 fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter 172 fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter 172 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 173 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 173 fois (commandes)
.....tracer
tourner de 0 degrés à gauche
répéter 173 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 174 fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || compteur |]] ||
hexagones |]]
tracer
```

Script 3.1 – Version TSn avec alignement b_7b_8 et ajout d'une boucle b_9, S5

va ainsi produire une rétroaction difficilement interprétable :

$$TS(n)[17"] \rightsquigarrow \left(\begin{array}{c} \text{compteur } 202 \\ \text{mesure } 17" \\ \\ \\ \end{array} \quad , 202, 17" \right)$$

Les élèves semblent alors identifier leur oubli de modification de b_9 en effectuant $b_9 \leftarrow 173$, ce qui bien qu'incorrect est cohérent avec leur méthode. Cette modification ne pouvant être issue d'une interprétation de la rétroaction, cela indique que les élèves ont sans doute cherché à relire et comprendre leur script, ou à ré-appliquer leur méthode. Suite à cela le binôme va, de nouveau, recourir à de multiples frappes de touche lançant le script, avant de sauvegarder leur programme puis de le recharger.

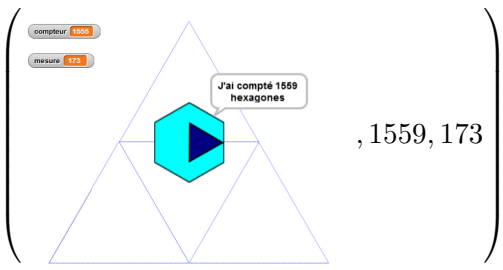


(a) Rétroaction du script ci-contre. Valide ou invalide ?

(b) Zoom sur le chevauchement b_9 — b_1

Figure 3.28 – Rétroaction pour le TS174 (45a, S5)

Construction d'une instance générique À ce stade, les élèves ont construit — ou tout du moins pensent avoir construit — une instance valide pour TS174, et ils commencent donc à chercher comment passer à une autre instance. Après la tentative infructueuse décrite ci-dessus pour passer au TS173, ils vont radicalement changer de méthode en indiquant pour chaque boucle une valeur de répétitions identique : 173. Ils obtiennent la rétroaction suivante :

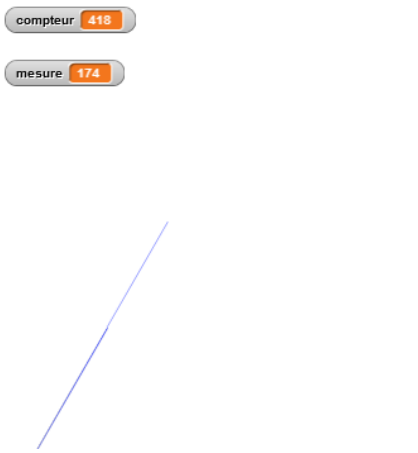
$TS(n)[173] \rightsquigarrow$  $, 1559, 173$

Leur script est erroné, mais le binôme semble l'identifier comme valide, puisqu'ils passent aussitôt à une autre instance, 171, en appliquant la même méthode — toutes les boucles à 171—, obtenant ainsi la rétroaction figure 3.29b (p. 265). L'erreur est légèrement plus visible, mais sans doute encore insuffisamment (la zone d'exécution n'est pas en plein écran), et de plus les élèves ne peuvent faire le lien entre le nombre d'hexagones dénombrés et le nombre d'hexagones attendus, cette valeur leur étant inconnue. Ils vont ensuite affecter à chaque boucle l'expression « 1 + 1 » (1 + 1), procédant ainsi avec la même méthode (le même expression pour toutes les boucles), et là encore pour un résultat difficilement interprétable en termes de tracé (figure 3.29c, p. 265).

(a) Rétroaction pour $b_a \leftarrow 173$.

(b) Rétroaction pour $b_a \leftarrow 171$.

(c) Rétroaction pour $b_a \leftarrow 1 + 1$.



(d) Rétroaction pour $b_a \leftarrow 174 + 174$.

Figure 3.29 – Rétroactions obtenues lors de l'affectation à chaque boucle de la même expression

Les élèves semblent considérer que le résultat est erroné pour une valeur de 174 entrée, et affectent à chaque boucle « 174 + 174 » (174 + 174). Le caractère non valide du tracé est visible, et les élèves ne laissent pas le script s'exécuter jusqu'au bout (figure 3.29d, p. 265)
Dans la minute qui leur reste avant la fin de la séance, le groupe 45a clique sur les opérateurs de quotient, produit, différence et somme, directement dans la zone de palette des commandes, sans entrer de valeur.

Recherche de régularités

La construction des faits et TEA du groupe 45a (figure 3.30, p. 266) montre que trois explorations des possibles sont expérimentées par les élèves :

1. modifier la taille des hexagones ;
2. modifier la structure du tracé ;
3. modifier un script en se basant sur un autre script ;

En outre, on peut constater un début de généralisation algébrique naïve.

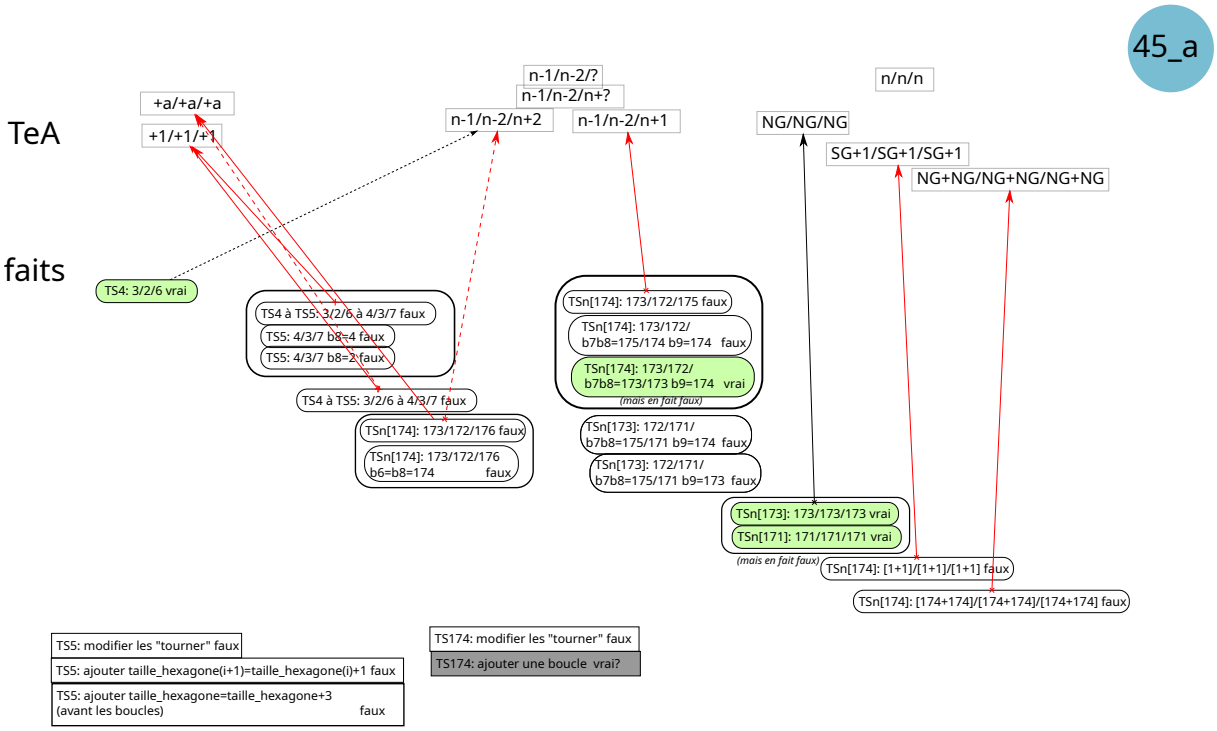
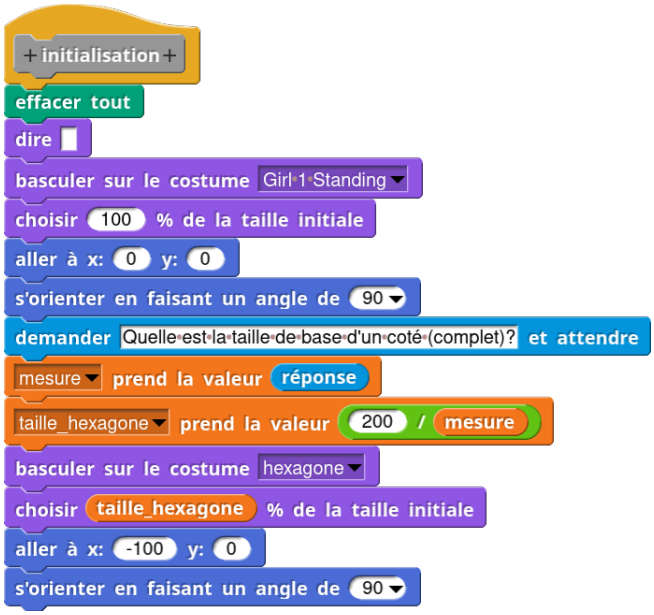


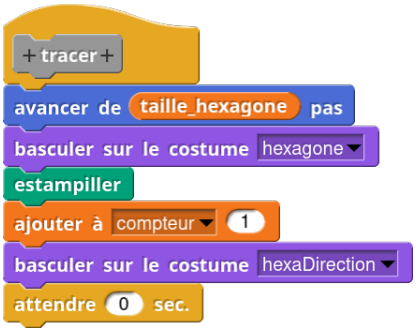
Figure 3.30 – Faits et TEA, groupe 45a

Une question de taille La modification de la taille des hexagones, semblant reproduire l'activité faite en séance 2, est vite invalidée : non seulement la dimension des hexagones tracés ne change pas, mais en plus ils ne sont plus jointifs (figures 3.25b et 3.25c, p. 258). Il est probable que l'hypothèse sous-jacente faite par les élèves est que si on augmente la mesure d'un TS de un, sa « taille » change. Ce terme de « taille » est en effet polysémique et a été problématique lors d'une première expérimentation faite en Master, lors de laquelle le nombre d'hexagones d'un côté du triangle de base avait été dénommé « taille » : ainsi, « taille » pouvait tout à la fois dénoter le nombre d'hexagones d'un triangle de base, la variable permettant de dimensionner un hexagone pour que le tracé reste inclus dans l'espace d'exécution, mais aussi la dimension du « lutin » lui-même, défini par une instruction choisir taille_hexagone % de la taille initiale. Lors de l'expérimentation analysée ici, une partie de cette polysémie a été ôtée en renommant, de façon plus juste, le nombre d'hexagones d'un côté en « mesure ». Nous avons aussi rendu la variable « taille_hexagone » partiellement cachée : on n'affiche pas sa valeur par défaut sur la zone d'exécution, mais elle reste accessible dans la catégorie d'instructions « variables ». En outre, les blocs (ici procédures) déterminant la valeur de cette variable ainsi que son utilisation, ont été rendus inaccessibles aux élèves (voir le détail figure 3.31, p. 267). Nous pensions ainsi supprimer l'effet polysémique, mais ce groupe invalide cette hypothèse [19]. Cependant, comme les élèves ont rapidement abandonné cette piste, l'effet sur leur construction du problème reste superficiel.

19. Il ne sera pas le seul, voir par exemple les groupes 46i ou 46e.



(a) Procédure (bloc) d'initialisation générique



(b) Procédure (bloc) de tracé d'un hexagone et déplacement

Figure 3.31 – Les deux blocs cachés aux élèves

Modifier l'algorithme et modifier le tracé Dans un premier temps, puis, après avoir tenté de construire $TS(5)$, les élèves de ce groupe expérimentent la modification de la structure du tracé, c'est-à-dire la façon dont se déroule le tracé. Comme si *pour changer de TS il fallait changer de structure*, ou que pour modifier un tracé, il faut modifier les instructions ou encore modifier l'algorithme. Lors de la séance 2, les élèves ont dû modifier un script pour ajouter des instructions permettant de dénombrer des hexagones, mais sans modifier la structure du tracé, c'est-à-dire la suite de déplacements, de rotations et de « tampons » permettant de tracer le TS4. Ici, ils commencent par modifier les rotations, avant d'abandonner cette piste, construisant sans doute le fait « Pour tracer un TS5, modifier les instructions "tourner" ne fonctionne pas ». Peu d'autres groupes ont ainsi touché à la structure même du tracé, la plupart semblent avoir intégré le fait que pour changer de TS tracé, il faut et il suffit de modifier le nombre de répétitions de chacune des boucles. Pour ce groupe, cette modification du tracé a été fondamentale, leur faisant construire des relations entre la mesure et les boucles, très différentes de celles des autres groupes. Ainsi, ce groupe passe du problème « comment modifier les b_i pour passer d'un $TS(4)$ à un $TS(5)$ » au problème « Comment modifier la structure du tracé pour qu'il soit correct ». En construisant cette structure, tout se passe comme si les élèves considéraient qu'une boucle ne devait concerner qu'un seul côté d'un triangle de base — ce qui est une possibilité. La valeur (176, puis 175) qui sera appliquée à la boucle b_7 lors de la construction du TS174, associée au niveau de zoom ne permettant plus de distinguer individuellement les hexagones, laisse à penser que cette boucle ne trace que la moitié du grand côté, ou plutôt, pour ces élèves, environ un côté des triangles de base[20]. Il faut donc prolonger le tracé sur un côté, ce que fait ce binôme en transformant une rotation en un alignement (tourner de 180 degrés puis tourner de 0 degrés).

20. Notons que pour des mesures assez petites comme celles qui sont demandées, la rétroaction suite à l'application du TEA $+1/+1/+1$ laisse penser *qu'il manque des hexagones*, alors que pour de grandes valeurs comme 174, on voit davantage *qu'il manque (presque) un côté d'un triangle de base.*

L'ajout d'une boucle supplémentaire est donc nécessaire, ce que font ces élèves. Leur activité est en fait bien organisée et semble montrer une maitrise du concept de boucle (ou d'itération), ainsi qu'une compréhension globale de la méthodologie du tracé (une suite de côtés tracés par des boucles). Cependant, la rétroaction va amener les élèves à construire de « faux faits », des énoncés que tout laisse à penser qu'ils sont vrais, mais qui, dans la réalité mathématique ou algorithmique, sont faux : leur script $S_{174} = \{b_1 = 173, b_2 = b_3 = b_4 = b_5 = 172, b_6 = 173, b_7 = b_8 = 173, b_9 = 174\}$ leur apparait comme valide, le chevauchement de $b9$ et b_1 n'étant que très difficilement visible[21]. Si ce groupe avait travaillé sur une instance plus petite, l'erreur (ou les erreurs ensuite) aurait été rendue visible. La figure 3.32 (p. 268) en témoigne, pour un TS14. Quoiqu'il en soit, les élèves considèrent cette nouvelle structure comme étant valide, et ils vont se baser dessus pour tenter de généraliser leur script.

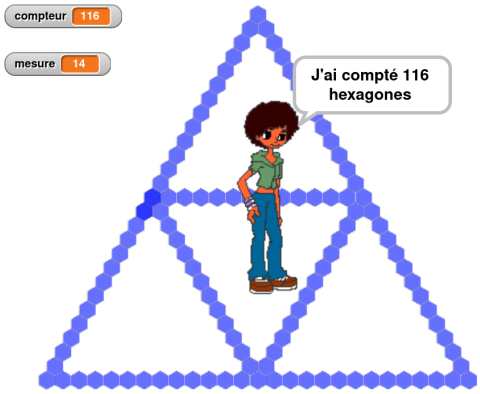


Figure 3.32 – La méthode de construction du groupe 45a appliquée à un TS14

Un TEA répandu, et son extension Si on écarte momentanément cette question de la structure, une question demeure : sur quoi se base ce groupe pour construire ces quelques instances, du point de vue des expressions à utiliser dans les boucles ?
Une première transformation d'un $TS(4)$ en $TS(5)$ a consisté à passer de $TS(4) = \{3/2/6\}$ à $TS(5) = \{4/3/7\}$: *si on augmente la mesure de 1, on augmente chaque boucle de 1.* Ce TEA que nous notons $+1/+1/+1$, est présent en première tentative dans quasiment tous les groupes. D'un point de vue plus formel ce théorème s'écrit : $b_i^{p+1} = b_i^p + 1$. Plusieurs cadres explicatifs peuvent le soutenir :

- un lien avec les propriétés géométriques du triangle équilatéral : « si on augmente la taille d'un côté, la taille des deux autres côtés augmente de la même façon » ;
- un appui sur la proportionnalité, avec une conception classiquement erronée : « si on agrandit une figure d'une certaine façon, on agrandit de la même façon tous les éléments de la figure ». On sait que, très souvent, au lieu de considérer l'agrandissement des dimensions comme un produit, les élèves traduisent cela par l'ajout d'une même valeur à toutes les dimensions des éléments de la figure.
- un lien entre les boucles et les côtés des triangles de base : « chaque boucle trace un côté, si j'augmente un côté d'un hexagone j'augmente le nombre de répétitions de la boucle de un »

21. Avec la classe suivante, enseignante et chercheur ont convenu qu'il était nécessaire d'écarter si possible ces mauvaises interprétations, et une démonstration du programme du groupe 45a a été analysée en début de séance, servant ainsi en quelque sorte de caricature (Orange, 2012, p. 102).

— une relation arithmétique : « pour passer d'une instance à la suivante on fait plus un ».

Ces quatre cadres sont aussi des explications au théorème $+a/+a/+a$ ($b_i^{p+a} = b_i^p + a$) qui semble à l'œuvre lorsque ce groupe tente de construire une première fois un TS174. Ils se basent sur le script donné $TS(4)$, et passent ainsi b_1 de 3 à 173, b_2 de 2 à 172, et b_7 de 6 à 176. On a ainsi $b_i^{4+170} = b_i^4 + 170$. Ce TEA peut être vu comme une extension du TEA $+/+1/+1$ à un ensemble de situations plus étendu : il n'agit plus seulement de passer d'une instance à la suivante, mais d'une instance à une autre. Le TEA $+1/+1/+1$ aurait dû être invalidé suite aux rétroactions, mais comme les élèves ne sont pas parvenus à construire une instance de TS5 valide, il n'y a pas de TEA alternatif susceptible de le remplacer. Cette persistance du TEA $+1/+1/+1$ peut sans doute expliquer aussi les tentatives de généralisation. On peut ainsi notamment le reconnaitre dans les actions $b_i \leftarrow 1+1$. Il est aussi possible que la forme de ce TEA, consistant à *faire la même chose pour toutes les boucles*, aboutisse à $b_i^{173} = 173$, ou encore $b_i^{174} = 174 + 174$. Pour construire ces deux TEA, les élèves font une relation intra-objectale (Piaget et García, 1983), croisant deux objets différents et situés graphiquement côte-à-côte.

Un début de généralisation algébrique naïve Ces deux applications de TEA erroné sont invalidées par les élèves grâce à la rétroaction. Lors du début de la séance suivante, ils construisent leur script de l'instance 174 en organisant leur activité différemment, fixant $\{B_1 = 173, B_2 = 172, B_7 = 175\}$. L'hypothèse que nous faisons est que les élèves cherchent ici non plus à passer d'une instance à une autre, mais à construire directement une instance (ce qui relèverait alors du trans-objectal) : ils sont dans une phase de recherche de généralisation algébrique naïve, cherchant une relation entre la mesure et les valeurs des boucles, sans forcément pouvoir la formuler de façon rigoureuse. Nous nous appuyons sur différents éléments pour appuyer cette proposition. D'un part, il est difficile de considérer que les élèves ajustent le script obtenu à la séance précédente ($\{173/172/176\}$) car il parait peu probable que l'erreur rendue visible par la rétroaction aboutisse à une diminution de b_7 au lieu d'une augmentation, puisque seulement un peu plus de la moitié du « grand côté » a été tracé, ce qui est clairement visible dans la rétroaction. D'autre part, le début de la séance 5 a été l'occasion de partager les faits suivants en classe : « pour une mesure de 3, $b_1 = 2$ », « pour une mesure de 4, $b_1 = 3$ », « pour une mesure de 5, $b_1 = 4$ », un élève précisant « ben par exemple mesure six c'est cinq mesure cinq c'est quatre bah c'est tout le temps un de moins » (45 Prof, 6'06). Ainsi, le fait « pour une mesure donnée, pour trouver b_1 on fait moins un » est partagé, et il est probable qu'il ait été mobilisé par le groupe 45a — notamment car en accord avec les premiers faits construits par ce groupe. En revanche, et volontairement, aucune relation permettant de trouver b_7 n'a été évoquée lors de ce partage de faits avec le groupe classe. Les élèves doivent donc mobiliser des faits permettant de construire la relation de la valeur du nombre de répétitions de b_7 avec la mesure, et ce groupe semble tenter ici une valeur plus grande que la mesure, ce qui correspondrait a une instance du TEA $\{p-1/p-2/p+?\}$ pour une mesure p. Les tentatives ultérieures ($b_i^{173} = 173$, $b_1^n = 1+1$, $b_i^{174} = 174 + 174$) semblent elles-aussi montrer une recherche de généricité.

Contourner la difficulté de b_7 On peut supposer que la succession de faits invalidant leurs premières tentatives encourage les élèves à chercher une solution alternative : non pas chercher une relation entre b_7 et la mesure, mais compenser l'erreur de tracé en prolongeant b_7 et en rajoutant une boucle b_9. Ils procèdent alors par ajustement progressif jusqu'au faux fait « S_{174} est valide ». Ils vont ensuite essayer de construire directement l'instance 173 en construisant $\{b_1 = 172, b_2 = b_3 = b_4 = b_5 = 171, b_6 = 172, b_7 = 175, b_8 = 171, b_9 = 174\}$. Il est dans ce cas difficile d'établir ce que les élèves ont tenu pour vrai lors de l'organisation de cette action. Les

relations pour b_7 et b_8 ne sont en effet pas stabilisées. Il semble même que les relations pour B_1 et B_2 ne le soient pas beaucoup plus. En effet, si, lors de cette exploration, le binôme semble mobiliser les faits partagés concernant ces relations, ce n'est plus le cas après seulement deux essais, à la suite de quoi ils passent au TEA $b_i^p = p$ en affectant 173 à toutes les boucles, puis en le réitérant avec la valeur 171. Ces deux mises en œuvre considérées comme étant valides par les élèves semblent ainsi confirmer le TEA $b_i^p = p$, puisqu'il est de nouveau appliqué avec une variation sémiotique avec les opérateurs « 1+1 » et « 174+174 » qui entament la phase de généralisation algébrique proprement dite, et que nous analyserons dans la partie suivante.

Généralisation algébrique

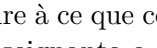
Figure 3.33 – Type de généralisation - 45a

Durant cette phase de généralisation algébrique, les élèves vont en premier lieu utiliser ce qui semble être un NG (nombre générique), qui sera ensuite remplacé par un signe générique (SG) dans une relation. Cette tentative rapidement invalidée, la relation sera conservée (quoique erronée) en remobilisant un nombre générique.

Un nombre générique Concernant la phase de généralisation algébrique, dans un premier temps le groupe 45a pense être capable de construire une instance quelconque d'un script construisant un certain TS de mesure donnée : lorsqu'ils construisent le script $S_{173} = \{173/173/173\}$, qu'ils testent bien avec une mesure de 173, le lien entre la mesure choisie et la valeur du nombre de répétitions des boucles est ici évident. Ils testent d'ailleurs sur le même script une application de leur TEA $b_i^p = p$ avec une valeur de 171 pour p et pour la mesure entrée : $S_{171} = \{171/171/171\}$. Il est possible que les élèves fassent ici une sorte de simulation de ce qui devrait se passer si on entrait la valeur 171 : l'ordinateur devrait mettre 171 à chacune des boucles. En effet, en début de séance, l'enseignante a terminé son bilan en groupe classe par la consigne : « donc il va falloir améliorer ce programme pour et lui donner des instructions pour qu'il puisse changer bah la valeur de ces nombres correctement »(45 Prof, S5, 12'29). Plus tôt, elle avait précisé le résultat attendu du script générique : lorsqu'on utilise une copie du $TS(4)$ sur $TS(n)$, $TS(n)[4]$ est valide, mais pas $TS(n)[5]$: en parlant du nombre d'hexagones sur les côtés du triangle de base, un élève remarque « mais yen a quatre »(45 Prof, S5, 4'26), ce à quoi l'enseignante ajoute « bah j'en ai pas cinq, alors effectivement il nous fait un dessin il nous fait un triangle de Sierpinski mais il ne fait pas celui que l'utilisateur a demandé, il ne fait pas celui de mesure cinq »(45 Prof, S5, 4'35). Les élèves mobilisent sans doute ici ce fait partagé. Ainsi, assigner aux boucles le nombre 173 pour une mesure de 173, tout comme assigner le nombre 171 pour une mesure de 171, semble caractériser un début de recherche de généricité. En effet, la généricité vient ici d'un début de recueil de faits issus de d'une famille de problèmes (construire un script traçant et dénombrant une certaine instance de TS) :

> C'est donc le recueil de problèmes qui induira le questionnement sur la relation "particulier" versus "général". Celle-ci intervient d'ailleurs à plusieurs niveaux : au phénomène précédent s'ajoute le fait qu'en variant les situations, on fera apparaître la portée d'une procédure, au-delà du contexte dans lequel un premier problème a permis son exposition. (Vitrac, 2005, p. 10)

Si le script S_{173} est une réponse à un premier problème, S_{171} étend la portée de la procédure. On retrouve donc ici les nombres à caractère générique de Balacheff (1987), les nombres donnés prenant un statut de nombre générique (« un objet présent non pour lui-même, mais en tant que représentant caractéristique d'une classe d'individus. »), ce que nous codons « NG »[22]. Ainsi, les élèves ont une procédure générique qui leur semble adéquate, mais qui pour l'instant n'est pas formalisée. Comme remarqué plus haut, cette affectation d'une même expression à toutes les boucles est peut-être une adaptation de leur TEA non remplacé $+1/+1/+1$: *on fait la même chose pour toutes les boucles.*

Vers le paramètre ? Dans un deuxième temps, les élèves vont chercher à résoudre un nouveau problème : comment faire pour donner « suffisamment d'instructions à l'ordinateur pour qu'il puisse le faire [changer la valeur des nombres correctement] »(45 Prof, S5, 13'07), c'est-à-dire comment faire en sorte que la machine puisse déterminer le nombre de répétitions de chacune des boucles suivant la mesure entrée. En effet, on a vu par exemple avec les Babyloniens qu'une procédure donnant un exemple de résolution pour une certaine instance d'un problème impliquait que la prise en charge de la généricité soit faite par le scribe étudiant, le « dispositif d'exécution » de Samurçay et Rouchier (1985). Or dans notre cas, le dispositif d'exécution (la machine, l'ordinateur) ne peut prendre la construction d'une procédure générique à sa charge. On aurait pu s'attendre à ce que ce groupe se dirige rapidement vers la variable  , puisqu'en début de séance l'enseignante a précisé et montré que la valeur entrée par l'utilisateur était stockée dans cette variable, avec un rappel de ce qu'est une variable informatique (45 Prof, S5, 10'23-12'25), et que la valeur des boucles, pour ces élèves est identique à la valeur de la mesure entrée. Mais on retrouve ici une question sémiotique : on doit trouver une expression dont le dénoté est le même que le dénoté de la mesure, ou que le dénoté de la variable  . Or, non seulement on peut avoir deux expressions différentes pour un même dénoté[23], mais en plus ce dénoté varie : pour une même expression E, on pourrait avoir $\delta(E) = 171$ ou $\delta(E) = 173$. Les élèves semblent traduire cette tension fixe-variable par la nécessaire existence d'un signe portant la généricité, allié à un moyen de permettre à l'ordinateur de savoir quoi faire de ce signe, c'est-à-dire à opérer dessus. Or, un autre fait partagé est sans doute mobilisé ici par effet de contrat : dans le début de séance, il a été précisé que « dans Snap on est on a la possibilité de faire des opérations | des calculs »(45 Prof, S5, 8'02) en montrant et détaillant la catégorie « opérateurs ». Un effet commun du contrat didactique étant « si on nous le montre, c'est qu'on doit l'utiliser », les élèves sont incités à utiliser un opérateur, qui a l'avantage de faire « bouger » les choses, puisque d'un ou plusieurs nombres on obtient un autre nombre. C'est peut-être ce qui explique l'utilisation par les élèves de l'expression  : le premier « 1 » exprime sans doute la généricité, c'est un signe générique, ou plutôt ici ce que nous considérons être un nombre-signe générique (NSG)[24]. Ce NSG est un signe marquant ce que l'ordinateur devrait remplacer, par

22. Voir les conventions pour le type de généralisation 2.5.1, p. 204

23. Voir l'exemple connu de Frege, avec « l'étoile du jour » et « l'étoile du matin » qui dénotent toutes deux la planète Vénus.

24. Notons d'ailleurs qu'avec les EPGB, on peut définir une variable nommée  et ainsi écrire  , et ainsi avoir  comme étant vrai !

171 ou 173 ou toute autre valeur entrée. Mais c'est aussi un nombre, puisque c'est bien par un nombre que l'on devrait le remplacer [25]. L'opérateur, quant à lui, indique une relation permettant d'utiliser ce signe, ce que l'on pourrait traduire en « faire une certaine opération avec un certain nombre ». Or le seul TEA mis en œuvre par les élèves est d'ajouter un à chaque boucle, d'où peut-être l'utilisation de l'opérateur ( ) + (1). Cependant, le deuxième signe (1) pourrait tout aussi bien être un signe générique : l'expression (1) + (1) aurait alors comme sens « ajouter à un certain nombre une certaine valeur ». D'ailleurs, lorsque l'exécution de cette solution produit une rétroaction qui invalide le tracé, les élèves reviennent alors au nombre générique (cette fois 174, qui correspond bien à leur entrée de test), en conservant l'opérateur, mais en remplaçant les deux signes (1) par ce nombre générique 174. Cette solution est elle-aussi invalidée et marque la fin de la séance, et par la même occasion de la recherche.

Construction des nécessités

Les éléments présentés ci-dessus permettent de construire l'espace des faits-contraintes pour le groupe 45a (figure 3.34, p. 288).
La nécessité que le script trace TSp pour un p donné semble être correctement construite : lors des tests du script générique, les élèves entrent une valeur qui correspond à la mesure du TS souhaité. Une fois la structure du script « correctement » modifiée, c'est-à-dire débouchant selon les élèves sur une construction valide, cela implique qu'il faut aussi changer les valeurs indiquant le nombre de répétitions des boucles. Ainsi, pour un TS173, toujours selon les élèves, il faut mettre ces valeurs à 173, et pour un TS171 à 171. Comme remarqué précédemment, ce fait (le tracé est valide selon les élèves), est en contradiction avec les faits précédemment construits, concernant par exemple b_1 et b_2, mais le binôme ne semble pas faire cette contradiction : le fait « S_{174} est valide » est ainsi défactualisé, notamment sans doute puisque construit dans le cadre d'un autre problème pour les élèves.
La nécessaire existence d'une expression dans Snap ! mettant en relation les boucles et la valeur du TS voulue est issue de la tension entre le caractère immuable de l'expression entrée pour représenter le nombre de répétitions d'une certaine boucle et la nécessité que le script trace diverses instances. La résolution de cette tension implique que b_i ne peut pas être nombre, puisque ce nombre définit l'instance tracée, et qu'il existe donc une expression qui convient pour plusieurs mesures. De plus, comme l'expression doit concerner plusieurs valeurs possibles, il doit exister un signe exprimant cette généricité. Ainsi le groupe 45a imagine la possibilité que l'expression soit un opérateur. Ici, il s'agit d'une addition, mais qui n'a pas forcément le sens d'une somme dans ce cas, à moins que ce ne soit une mise en œuvre partielle du TEA $+1/+1/+1$. Pour exprimer la généricité de la relation mesure-boucle, soit ce qui va indiquer à l'ordinateur « comment il va changer les valeurs correctement », le signe générique (1) est ainsi combiné à l'opérateur.
Si les élèves avaient pu continuer à expérimenter à la suite de l'échec de cet essai et du suivant (qui remobilise un NG), peut-être auraient-ils pu construire la nécessité d'une relation entre b_i et la mesure (relation non nécessaire pour les élèves à ce stade puisque selon eux la valeur du nombre de répétitions d'une boucle *est* la valeur de la mesure) et par suite la nécessaire existence d'une expression de Snap ! dénotant la valeur de la mesure. Mais en l'occurrence, ces deux nécessités ne semblent pas être construites [26], on peut ainsi dire que pour ces élèves, le concept de paramètre n'est pas encore construit.

25. D'autres nombres-signe génériques seront visibles dans d'autres groupes, notamment le « 0 ».
26. Ce qui est représenté dans l'espace des faits-contraintes par des pointillés bleus.

Bilan

L'enchainement de la construction des faits chez ce groupe nous amène à faire les remarques suivantes.
D'une part, les élèves ont construit très peu de faits seconds validant leur activité, et certains faits invalidants n'ont pas été suivi d'ajustements aboutissant à une solution valide. Il semble ainsi difficile de construire des faits tiers médiateurs à partir uniquement de non-validation. En revanche, dès que ce binôme a cru valider sur deux itérations le TEA $b_i^p = p$, ils l'ont ensuite étendu à la phase de généralisation. L'obtention de faits validant parait ainsi nécessaire à la construction de faits tiers, et par suite de nécessités.
D'autre part, ces faits construits ne sont pas tous liés au même problème pour les élèves : il s'agit soit de trouver les valeurs pour les b_i, soit de trouver comment modifier la structure du script pour le rendre valide. Les faits construits dans un problème ne paraissent ainsi pas mis en relation — ou en tout cas plus difficilement — avec les faits construits dans un autre problème. Ici par exemple, le fait second « S_{174} est vrai » est lié au problème de structure. Mais les élèves ne semblent pas mettre en tension les faits premiers mobilisés dans ce problème avec ceux qu'ils ont construit lors de l'instance 173. Tout se passe comme si les élèves ne mobilisaient pas le fait premier $S_{174} = \{b_1 = 173, b_2 = b_3 = b_4 = b_5 = 172, b_6 = 173, b_7 = b_8 = 173, b_9 = 174\}$, qui produit un résultat considéré comme valide, lorsqu'ils construisent le fait « $S_{173} = \{173/173/173\}$ est valide », alors que l'instanciation des boucles apparait contradictoire [27].
Un début de recherche de généralisation est visible, avec une recherche d'expression générique dans le registre de représentation sémiotique de Snap ! Ne pouvant se baser sur une relation mesure-boucle établie et reconnue comme vraie par les élèves — relation non établie par absence ou insuffisance de construction de scripts valides —, il s'agit ici pour les élèves de trouver une combinaison de signes permettant à l'ordinateur de « changer les valeurs correctement », mais ils ne peuvent mettre de sens derrière ces signes.

3.3.2 Groupe 45f

Le groupe 45f est le premier des groupes pour lesquels il nous semblait voir un début de généralisation (figure 3.47, p. 295). C'est aussi un groupe qui semble s'être appuyé, du moins à certains moments, sur les liens entre ce qui est tracé et les valeurs du nombre de répétitions des boucles, et non plus seulement sur les relations entre ces nombres. Une des difficultés rencontrées par ces élèves est sans doute une absence de compréhension de la structure du script, qui notamment empêche la construction de faits tiers valides lors de l'interprétation des rétroactions. D'autre part, une captation vidéo et audio de ce binôme est disponible pour les trois séances étudiées. Les informations portées par ces données entrent parfois en contradiction les hypothèses que l'on peut faire à partir des seules traces de programmation des élèves.

27. Contradiction non vue représentée par des pointillés rouges dans les espaces faits-contraintes.

Description de l'activité

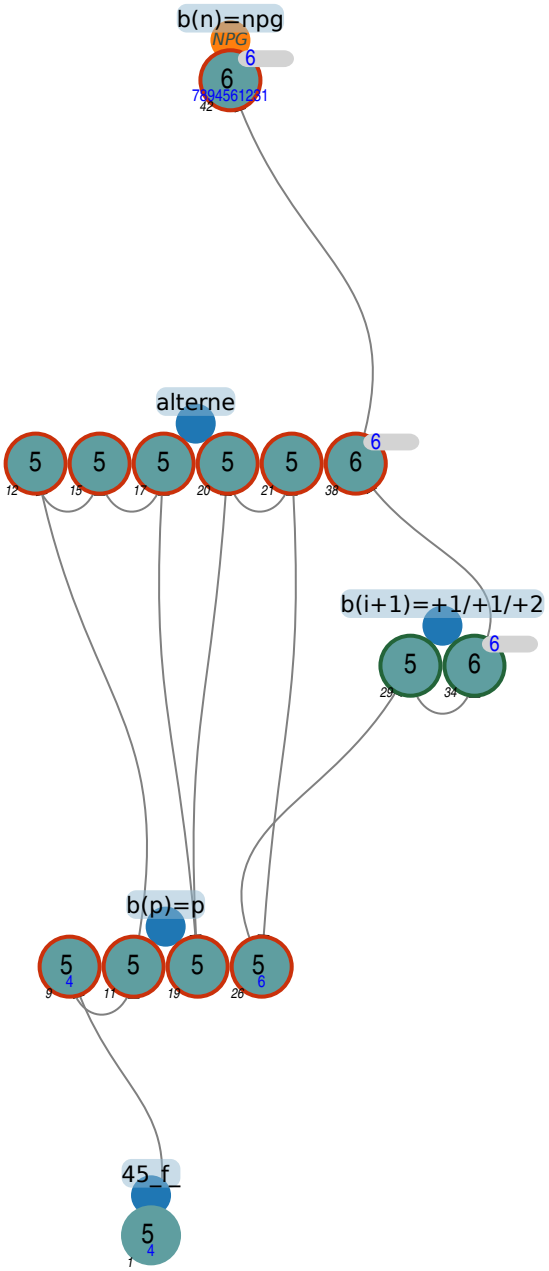
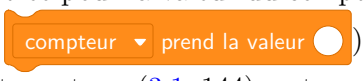


Figure 3.35 – Organisation de l'activité - 45f

Le groupe 45f n'aboutira à la construction d'un TS5 valide qu'au bout de 29 étapes (en séance 4). Il ne construira qu'une seule autre instance (TS6) avant de poser le problème de la généralisation du script. Un TEA peu courant est présent en séance 3, et fait son retour en séance 5.

Un problème de lancement (étapes 1-3) Le groupe 45f commence par dupliquer le $TS(4)$ sous l'entête du script devant tracer le TS5, modifie l'initialisation du compteur (*compteur* ← 5) et ajoute un bloc d'entête de script « drapeau vert » : 

. Les élèves tentent alors plusieurs lancements en cliquant sur le drapeau vert ( ), sans succès. S'ils sont les seuls dans cette classe à avoir rajouté cet entête « drapeau vert », d'autres groupes ont eux-aussi commencé la séance 3 en tentant de lancer une exécution en cliquant sur le drapeau vert [28], ce qui ne produit aucune exécution, le script étant vide. Au bout de quelques tentatives infructueuses, le binôme va modifier l'entête de lancement, en remplaçant le « quand "b" pressé » par « quand "5" pressé », puis tester de nouveau plusieurs appuis sur le drapeau vert, toujours sans succès. Cette difficulté initiale est sans doute induite par la séance précédente, lors de laquelle le script sur lequel travaillaient les élèves était unique et se déclenchait en utilisant cet événement. Il est possible que les élèves font l'hypothèse que le drapeau vert lance l'exécution, et que cela déclenche le script traçant le TS5 lorsque c'est cette mesure qu'ils souhaitent tracer, une des élèves précisant (45f, transcription S3, 3'45) « bon ah du coup faut faire cinq | attends un |triangle de cinq | de mesure cinq ». Le « 5 » est bien identifié à la mesure choisie, et est donc reconnu comme déterminant pour le tracé. Huit tentatives plus tard, l'enseignante va rappeler à ce groupe les raisons de l'absence de drapeau vert (plusieurs scripts)

Entre l'intention et l'erreur (étapes 5-8) Si on se base sur les traces de l'activité, les élèves vont modifier un peu plus tard les instructions « quand "5" est pressé » en « quand "b" est pressé » (45f, S3, 4'11) puis « compteur prend la valeur "5" » en « compteur prend la valeur "b" » (45f, S3, 4'33). Ce double changement consécutif du « 5 » en « b » pourrait nous laisser penser qu'un enjeu sémiotique est ici à l'œuvre, puisqu'on associe au compteur le même signe que celui associé au déclenchement du script. Il s'agit en fait d'une erreur de manipulation : l'enseignante corrige le changement d'entête (3.1, 134) [29]. Elle appuie sur la touche « b » (3.1, 142) afin de lancer l'exécution, mais l'EPGB est dans un état d'attente d'une entrée pour la valeur du compteur (le curseur est actif pour entrer le paramètre de la commande 

) : c'est pour cette raison que l'enseignante précise « ah non faut que tu sortes » (3.1, 144), autrement dit, il faut sortir de cet espace permettant d'entrer un paramètre. Ici, l'intention que nous pourrions prêter aux élèves à partir des événements de programmation est contredite par les transcriptions issues des captations vidéo. Cela nous permet de souligner que l'analyse des groupes à partir des seules transcriptions d'événements ne peut pas être vue comme étant une analyse de « ce qu'il s'est passé », mais plutôt de « ce qu'il semble s'être passé, nonobstant les informations non disponibles » — ce qui est le cas de toute analyse se basant sur une récolte de données.

28. Dans cette classe, les groupes 45d, 45g, 45i, 45m et 45b.
29. Les références des transcriptions sont codées :

— Extrait dans le corps du texte : (table, indice), où « table » est la référence du tableau d'un extrait de transcription donné dans le corps du texte, et « indice » l'indice de la transcription dans ce tableau

— Transcription (*ELAN* 2020) en annexe : (groupe, transcription séance, temps) où « groupe » désigne le groupe concerné, « séance » la séance — le mot « transcription » précisant qu'il s'agit de la transcription issue de la capture vidéo et audio, pour différencier de la transcription des événements de programmation —, et « temps » le moment de l'événement transcrit.

121	05 :33.025	EXEC	GREEN_button
122	05 :33.031	EXECUTION	START receiveGo(619)
123	05 :33.054	EXECUTION	FIN receiveGo(619)
124	05 :33.871	T1	madame est-ce que vous pouvez venir s'il vous plait parce qu'en fait on appuie sur le drapeau ça marche pas
125	05 :37.712	TProf	alors
126	05 :38.277	TProf	je vous ai dit tout à l'heure le drapeau on va pas l'utiliser
127	05 :42.019	TProf	parce que drapeau vert c'est souvent utilisé quand on a un seul programme
128	05 :46.811	TProf	là yen a plusieurs
129	05 :48.762	TProf	donc quand si vous voulez lancer ce programme là
130	05 :51.603	TProf	vous faites "quand"
131	05 :52.871	TProf	ah vous c'est quand cinq c'était écrit cinq au départ ?
132	05 :55.683	T1	non c'était zéro je crois
133	05 :57.396	TProf	c'était pas "b" ?
134	05 :58.736	VALEURS	i1 : Quand *b* est pressé<<5>>
135	06 :01.257	TProf	maintenant quand je vais taper sur "b" parce que
136	06 :03.495	TProf	là "a" ça lance celui de quatre là "b" ça lance celui de cinq
137	06 :06.742	TProf	là "c" celui de six
138	06 :08.762	TProf	là "d" etc
139	06 :10.039	T1	mais on a mis valeur euh de cinq
140	06 :13.178	TProf	alors
141	06 :14.534	TProf	ça veut dire
142	06 :16.287	TProf	que lorsque je vais appuyer sur "b" qu'est-ce qu'il se passe
143	06 :19.000	T1	euh
144	06 :19.386	TProf	ah non faut que tu sortes
145	06 :20.615	VALEURS	i3 : compteur prend la valeur *b*<<5>>
146	06 :21.811	TProf	scuse moi j'avais pas vue
147	06 :22.821	TProf	remets cinq
148	06 :27.227	TProf	voilà
149	06 :27.834	VALEURS	i3 : compteur prend la valeur *5*<<b>>
150	06 :28.099	TProf	entrée XXX
151	06 :29.144	EXEC	KEY_enter
152	06 :29.643	TProf	maintenant quand j'appuie sur "b"
153	06 :31.346	EXEC	KEY_b
154	06 :31.364	EXECUTION	START receiveKey(481)
155	06 :31.831	TProf	qu'est-ce qu'il se passe dans le programme
156	06 :31.831	GProf	montre rétroaction
157	06 :33.802	TProf	est-ce qu'il fait bien
158	06 :34.950	TProf	vous avez votre cahier de cours
159	06 :36.207	T2	euh il est là bas je vais aller le chercher
160	06 :44.251	EXECUTION	FIN receiveKey(481)
161	06 :48.811	TProf	le premier programme
162	06 :48.811	GProf	montre TS4 sur cahier puis script
163	06 :50.485	TProf	quand je lance "a"
164	06 :50.485	GProf	suit le script puis montre TS4 sur le cahier
165	06 :52.039	TProf	y fait ça
166	06 :52.039	GProf	entoure le TS4 cahier avec le doigt
167	06 :53.128	TProf	quand on va lancer "b" il va faire ça
168	06 :53.128	GProf	suit le script TS5 puis montre TS5 sur le cahier
169	06 :55.811	T1	hmm

Transcription 3.1 – 45f, S3 : enjeu sémiotique ou erreur de manipulation

Raccourcis sémiotiques Un premier élément sémiotique est aussi à l'œuvre dans ce passage : Le script dont la tâche prescrite associée est de construire un TS5 est désigné dans un premier temps par l'action à produire pour lancer son exécution : « donc quand si vous voulez lancer ce programme là, vous faites "quand" » (3.1, 129-130). Notons que cette expression (« vous faites "quand" ») peut-être interprétée de deux manières :

— *vous créez l'événement correspondant au "quand"* (c'est-à-dire presser la touche correspondante)

— *vous cliquez sur* 

Ces deux interprétations sont valides dans l'EPGB, et lancent bien l'exécution du script correspondant. Dans un deuxième temps l'enseignante associe l'exécution de ce script non plus à l'action en liaison avec l'entête, mais à l'action seule : « quand je vais taper sur "b" »(3.1, 135). Une nouvelle réduction a ensuite lieu : le fait d'appuyer sur une certaine touche pour lancer un certain script est réduit à la lettre, qui se retrouve associée au script lui-même ou à la tâche prescrite pour le script (le problème qu'il résout ou celui qu'il doit résoudre) : « le "a" lance celui de quatre »(3.1, 136) débouche sur « quand je lance "a" » (3.1, 163). Cette désignation du script (ou de ce qu'il produit, ou est censé produire) par une lettre se retrouve ensuite mobilisée par les élèves. Ainsi, en séance 5, E1 remarque « mais t'es pas sur le "n" là [...] faut que tu te mettes sur "n" » (45f, transcription S5,19'52-19'59). La lettre finit par désigner plusieurs objets (figure 3.36, p. 289) :

— l'action à effectuer pour lancer le script ;

— le script lancé lorsqu'on appuie sur la touche correspondant à cette lettre ;

— la tâche prescrite ou le but fixé par les élèves pour ce script ;

— la valeur de la mesure du TS tracé (ou devant être tracé) par ce script.

Ce raccourci sémiotique *aurait pu* être à l'œuvre lorsque le groupe 45f commence par modifier 

. Il a été constaté, et a engendré des confusions, lors d'une première expérimentation en Master, pour laquelle les scripts étaient lancés en cliquant sur le chiffre correspondant à la mesure de l'instance voulue.
De plus, ce raccourci est aussi associé à une idée d'itération, voire de nombre : le fait présenté par l'enseignante « là "a" ça lance celui de quatre là "b" ça lance celui de cinq, là "c" celui de six, là "d" etc »(3.1, 136-138) peut amener la reconnaissance d'un motif :

$$\begin{array}{ccc} \text{a} & \rightarrow & 4 \\ \text{b} & \rightarrow & 5 \\ \text{c} & \rightarrow & 6 \\ \text{d} & \rightarrow & 7 \\ \text{e} & \rightarrow & 8 \\ & \ldots & \end{array}$$

Ainsi, lorsqu'une lettre en suit une autre, elle semble désigner un script qui doit tracer l'instance qui suit celle désignée par la lettre précédente. Ceci peut inciter les élèves à construire une suite d'instances successives, ce qui obère la possibilité d'aboutir à une généralisation naïve, et de surcroit à une généralisation algébrique. C'est par exemple ce que l'on peut retrouver pour les groupes 45e, 45j,46a, 46b, 46c, 46d, 46g, ou encore 46m qui vont produire une suite d'instances successives, alors que la tâche prescrite du script que l'on peut désigner par "e" est « Compléter pour un TS11 » et celui par "f" est « Compléter pour un TS17 ; TS19 ou TS22 ».

On retrouve ce raccourci sémiotique dans les transcriptions des autres groupes filmés. De notre part, nous désignerons de préférence les scripts par leur tâche prescrite, où en précisant l'entête du script, notamment pour le script générique. Néanmoins, mais ce raccourci pourra être utilisé, notamment pour les groupes dont une captation est disponible.

Construction du TS5 : appui sur la figure et ses propriétés (étape 9-10) Ce binôme va ensuite fixer la valeur du nombre de répétitions de toutes les boucles à trois (qui est aussi la valeur du nombre de répétitions de la boucle b_1) : E1 précise « j'ai envie de le changer ici | euh bah trois | comme yen a trois » (45f, transcription S3, 5'45), en montrant le TS4 exemple sur son cahier. On voit ici que ces élèves ne s'appuient pas sur le nombre de répétitions des boucles, mais sur la figure. En effet, passer de $\{b_1 = 3, b_2 = 2, b_3 = 2...\}$ à $\{b_1 = 3, b_2 = 3, b_3 = 3...\}$ s'explique difficilement, à moins de considérer le tracé possible : on *aurait pu* commencer le tracé du premier triangle de base avec trois segments de trois hexagones (figure 3.37, p. 289). L'hypothèse d'un appui sur la figure exemple du cahier est par ailleurs renforcée par l'injonction précédente de l'enseignante d'utiliser le cahier de cours. À ce stade, les élèves semblent chercher des explications dans un cadre géométrique ; le REX en vigueur est ici géométrique.

Block_481

```
Quand b est pressé
initialisation5
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 3 fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter 3 fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 3 fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || compteur |]] || hexagones |]]
```

Script 3.2 – $b_i = 3$ (45f, S3)

Figure 3.38 – « comme yen a trois » (45f, S3)

L'exécution produit :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 26, 5 \right)$$

Ce qui conduit E1 à s'interroger sur l'origine du trou à la jonction des côtés tracés par b_1 et b_3 : « pourquoi il a pas fait lui lui ? » (45f, transcription S3, 6'39). La réaction à cette rétroaction est cohérente : comme il manque un hexagone, il faut en tracer un de plus, et ce ne peut être que b_1 qui trace cet hexagone. Les élèves ici associent « trou » à « il en manque » (un hexagone n'a pas été « fait »). C'est une association logique, mais erronée : un trou dans un tracé (avec l'algorithme de tracé choisi) est un décalage, plutôt qu'une absence. Ainsi, l'hexagone H3 qui est censé être au-dessus du point de départ du tracé (figure 3.39, p. 289) est bien tracé, mais pas au bon endroit. C'est effectivement en faisant tracer un hexagone de plus par b_1 que l'erreur va être corrigée, mais parce que cet ajout va entrainer un décalage du reste de la figure, qui permet de repositionner l'hexagone H3. Cependant, comme les différents tracés ne sont pas conservés, les élèves ne peuvent s'y référer.

Construction du TS5 : appui sur les valeurs et alternance (étapes 9-29)
Le tracé est toujours erroné, et E2 identifie deux problèmes sur la figure :

- le triangle de base du haut n'est pas tracé : « oui mais en plus on a pas la pointe du triangle » (45f, transcription S3, 7'13) — REX géométrique,
- le nombre d'hexagones des côtés du triangle de base droite n'est pas le même que celui du triangle de base de gauche : « là ils en mettent que trois alors que là quatre » (45f, transcription S3, 7'19) — REX géométrique

De ces deux problèmes, E1 semble ne traiter que le deuxième en fixant les valeurs des boucles b_1 à b_5 à 4 :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 32, 5 \right)$$

Le problème du décalage réapparait ainsi, sans qu'aucun des deux problèmes identifiés ne soit résolu. E2 semble encore s'intéresser aux différentes étapes du tracé — donc à l'algorithme : « en fait au début là ça a été bien commencé | après ici c'était bon | mais en fait c'était XXX là et là parce que là yavait rien » en montrant le cahier puis la rétroaction (45f, transcription S3, 8'03). E2 cherche ici des explications dans un REX géométrique et algorithmique. En revanche, E1 semble s'intéresser davantage aux nombres : « bah attends c'est une fois sur deux » (45f, transcription S3, 8'10). Elle modifie alors le script en alternant les valeurs de 3 et 4, soit en

faisant $b_{2i+1} \leftarrow 4$ et $b_{2i} \leftarrow 3$. Le tracé produit est plus difficilement interprétable :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 30, 5 \right)$$

À partir de ce moment et jusqu'à la fin de la séance, les élèves de ce groupe vont tenter diverses alternances : après l'alternance 4-3, elles vont tenter 4-8, puis 5-4, 4-2 avant de revenir au 4-3. Elles termineront la séance en explorant le tracé grâce au mode pas-à-pas, avec l'enseignante qui leur a suggéré et expliqué ce mode. Ce groupe semble avoir basculé d'un REX géométrique et algorithmique (le tracé de la figure et son organisation) à un REX numérique et algorithmique (une alternance de nombres).
En séance 4, ce groupe reprend la première tentative de la séance 3, en donnant la même valeur à toutes les boucles. La différence étant que cette fois ces élèves affectent la valeur 5, qui est la mesure du triangle recherché : « faut le mettre en cinq » (45f, transcription S4, 1'22). D'autre part, ils modifient directement le script donné, sans le dupliquer au préalable.

$$TS(4)[] \rightsquigarrow \left(\text{  } \text{} , 42, 4 \right)$$

Après quelques minutes, les élèves vont rétablir le script dans sa version originelle $\{3/2/6\}$ (45f, S4, 5'11-5'43). Elles vont ensuite dupliquer ce script et le modifier, deux minutes plus tard, pour qu'il trace un TS5, en choisissant les valeurs exactes : $TS(5) = \{4/3/8\}$ (45f, S4, 7'08). Cette soudaine réussite est due à leur voisine, qui leur a expliqué le TEA $+1/+1/+2$: « eh les filles faut changer, vous mettez un de plus à chaque fois, sauf le au six vous mettez huit » (45f, transcription S4, 7').

Vers la généralisation (étapes 30-34)
Après avoir copié ce dernier script sous l'entête visant la construction du script générique, l'enseignante leur précise ce qui est attendu lorsqu'on lance le script générique (3.2). Elle tente ainsi de faire partager les faits suivants :

- le programme doit tracer « pour n'importe quelle mesure » (3.2, 434) ;
- si je veux un TS de mesure 12, je rentre douze lorsque « la personne me demande combien je veux » (3.2, 439) ;
- si les valeurs sont celles d'un TS5, le script ne va pas tracer un TS12 (3.2, 442) ;
- il faut changer « les nombres qui sont dans les répéter » (3.2, 445) ;
- le nombre de répétitions de la première boucle est la mesure moins un, traduit par les élèves en « ah bah faut mettre un truc en dessous en fait » (3.2, 454-456) ;

— il ne faut pas changer le programme quand on change la mesure entrée donc il faut trouver « une astuce [...] pour que ce soit l'ordinateur qui se débrouille tout seul » (3.2, 466-468) ;
— il y a des opérateurs pour faire des calculs (3.2, 470-474).

431	10 :52.505	TProf	alors maintenant
432	10 :54.000	TProf	vous avez fait celui de mesure cinq
433	10 :56.198	TProf	donc vous avez | modifié | correctement vos boucles | pour que la figure soit un de cinq
434	11 :01.910	TProf	et ce que je vous demande de faire c'est un programme qui tracerait pour n'importe quelle mesure
435	11 :07.277	TProf	ça veut dire que quand je lance le "n"
436	11 :09.459	EXEC	KEY_n
437	11 :09.474	EXECUTION	START receiveKey(478)
438	11 :09.533	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
439	11 :11.564	TProf	là la personne me demande combien je veux là par exemple douze
440	11 :15.089	TProf	de mesure douze
441	11 :16.875	ENTRÉE	ANSW <<12>>
442	11 :19.485	TProf	sauf que là il va pas me faire un mesure douze parce que j'ai recopié celui de mesure cinq
443	11 :23.980	T1	du coup faut tout changer
444	11 :25.619	EXECUTION	FIN receiveKey(478)
445	11 :26.821	TProf	il va falloir il faut changer les nombres qui sont dans les répéter
446	11 :31.000	TProf	sauf
447	11 :31.217	T2	douze
448	11 :33.732	TProf	alors si je regarde celui de mesure cinq
449	11 :37.013	EXECUTION	START receiveKey(388)
450	11 :37.079	TProf	que vous avez fait
451	11 :38.742	TProf	XXX
452	11 :38.930	EXECUTION	STOP receiveKey(388)
453	11 :40.477	EXEC	STOP_button(all)
454	11 :40.960	TProf	euh quand c'est mesure cinq la première boucle est à quatre
455	11 :45.316	TProf	quand c'est mesure quatre la première boucle est à trois
456	11 :48.356	T2	d'accord | ah bah faut mettre un truc en dessous en fait
457	11 :50.604	TProf	c'est ça
458	11 :51.326	T2	pour douze faudra mettre onze
459	11 :52.940	TProf	c'est ça
460	11 :53.851	T1	ah
461	11 :54.019	TProf	sauf que
462	11 :56.009	TProf	si vous me mettez pour douze XXX mettre onze
463	11 :58.841	TProf	mais si l'utilisateur veut autre chose que douze et qu'il veut seize
464	12 :04.505	TProf	ça veut dire que vous allez devoir encore changer le programme ?
465	12 :09.198	TProf	c'est ça qu'est dommage
466	12 :10.831	TProf	c'et pour ça justement que nous justement on aimerait que vous trouviez une astuce | pour que
467	12 :14.871	TProf	ce soit l'ordinateur
468	12 :17.653	TProf	qui se débrouille tout seul à changer XXX
469	12 :21.386	T2	ah
470	12 :21.910	TProf	c'est pour ça tout à l'heure que je vous ai montré qu'il y avait des opérations
471	12 :23.861	T2	oui ça j'avais vu
472	12 :24.172	AFFICHAGE	AFFBL_operators
473	12 :25.366	TProf	donc | vous m'avez dit tout à l'heure c'est un de moins
474	12 :28.495	TProf	ici

Transcription 3.2 – 45f, S4, extrait 1

Ces faits correspondent en partie aux nécessités locales que l'on souhaite faire construire aux élèves. On voit cependant qu'avec l'enseignante, nous induisons possiblement un REX plus numérique que géométrique pour la construction du script, puisque nous incitons les élèves à établir des relations entre des nombres. En fait, nous considérions que suite à la séance 2 et au bilan de début de séance 3, les élèves avaient construit les faits liés à l'algorithme et à la structure du tracé, en identifiant notamment quelle boucle traçait quelle portion. On se rend compte *a posteriori* que ces faits partagés en groupe classe n'étaient pas mobilisés par tous les groupes, et que certains ont donc dû les reconstruire.

Les élèves du groupe 45f modifient alors le script en mettant toutes expressions du nombre de répétitions des boucles à une chaine de caractères vide (notée []) : répéter ◯ fois. On interrogera plus tard le sens que l'on pourrait accorder à ce signe. Les élèves enchainent en mettant les valeurs adaptées pour un TS6 ({5/4/10}), testé avec la valeur de l'instance tracée, 6 — mais cela est fait en copiant le script de leur voisine.

$$TS(n)[6] \rightsquigarrow \left(\text{  } \text{} , 42, 6 \right)$$

En séance 5, le groupe 45f — encore une fois accompagné par l'enseignante, que ces élèves sollicitent beaucoup — commence par dupliquer le script $TS(4)$ sous l'entête du script ayant vocation à être générique, l'exécutent avec une entrée de 4 qui correspond à la mesure du TS tracé, et activent le pas à pas. L'enseignante veut leur préciser [30] aussi, de nouveau, que ce script doit s'adapter à l'entrée : « donc quand je vais mettre quatre il fait bien quatre [...] donc par contre si je le lance et que je mets autre chose que quatre, ben y doit, vous devez lui dire de modifier les boucles » (45f, transcription S5, 1'02-1'16). Le problème est posé aux élèves, mais les élèves posent-ils le problème ?

Retour de l'alternance (étapes 37-40) Ces élèves vont alors modifier le script pour tracer un TS6 (« donc euh on fait sur combien ? sur six comme hier » 45f, transcription S5, 2'53). Après une hésitation sur la valeur de b_1 fixée dans un premier temps à 6, elles initialisent le compteur à 6 (comme en début de première séance) puis affectent, de nouveau, une alternance, ici 5-4-4 : $TS(6) = \{5/4/4/5/4/4/5/4\}$ ce qui produit la rétroaction :

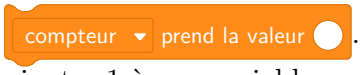
$$TS(n)[6] \rightsquigarrow \left(\text{  } \text{} , 43, 6 \right)$$

Cette alternance, déjà présente en séance précédente, est expliquée par les élèves en faisant référence à un fait partagé qui est écrit au tableau : « si mesure c'est six cinq quatre quatre » (3.3, 155). Autrement dit, $mesure = 6 \implies b_1 = 5 \wedge b_2 = 4 \wedge b_3 = 4$. Les autres valeurs des boucles ne sont pas indiquées au tableau, et l'élève E1 a fait « comme la prof » : « cinq quatre quatre cinq quatre quatre » (3.3, 158). Ici encore, il semble que les élèves se situent dans un REX numérique, et ne s'appuient pas sur l'organisation du tracé. Elles relancent ensuite le script plusieurs fois en changeant la valeur entrée : 5, puis 7, puis de nouveau 6 : le tracé devrait changer, mais

30. De nouveau, puisque cela avait été fait auprès de ce groupe en séance 4, et abordé en groupe classe en début de séance 5.

148	08 :12.388	T1	mais qu'est-ce t'as fait toi ?
149	08 :13.698	T2	c'est ça le six ça marche pas parce que là on a faux sur le truc
150	08 :13.698	G2	balaye le script avec la main
151	08 :16.785	T1	bah pourquoi déjà t'as mis compteur valeur six là
152	08 :18.419	T1	faut rien mettre | compteur
153	08 :21.276	T2	répéter cinq fois parce que en fait là-bas c'est écrit
154	08 :21.276	G2	regarde le tableau
155	08 :24.326	T2	si mesure c'est six cinq quatre quatre
156	08 :26.823	T2	mais après
157	08 :27.531	T1	et ben on fait comme la prof
158	08 :29.363	T2	cinq quatre quatre cinq quatre quatre

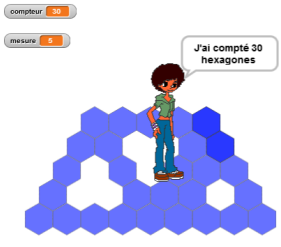
Transcription 3.3 – 45f, S5, extrait 1

ce n'est pas le cas, à un facteur d'agrandissement près. L'initialisation du compteur est alors considérée comme source de l'erreur (3.3, 151-152) puis une tentative de correction est faite : 

. Cette instruction, en théorie, n'est pas valide, mais sous Snap !, ajouter 1 à une variable « vide » équivaut à ajouter 1 à une variable ayant une valeur nulle.

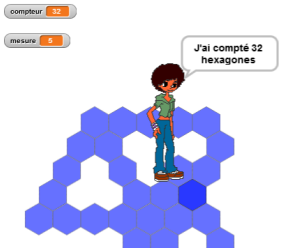
Un nombre particulier (étape 41) E1 va alors tester une action très particulière : $b_1 \leftarrow$ 7894561230. On peut se demander s'il ne s'agit pas ici d'un « Nombre Potentiellement Générique »(NPG) : ce pourrait être n'importe quel nombre, y compris celui-ci [31]. Cependant, rien dans les échanges entre élèves ne permet de confirmer cette hypothèse (45f, transcription S5, 9'13-9'51). En revanche, le lien entre la valeur de b_1, le compteur, et les hexagones tracés est bien établi par E2 : lorsque E1, constatant que le tracé est hors de l'espace d'exécution, et que la valeur affichée du compteur continue de changer, pense que « ça fait que bugger » (45f, transcription S5, 9'34), E2 répond « non c'est que ça trace sept mille machin bidule fois » (45f, transcription S5, 9'37). E2, comme en début de séance 3, est ici dans un REX géométrique et algorithmique, et non seulement numérique.

Identification des boucles : nécessité (étapes 44-62) Le binôme va alors s'atteler à reconstruire le script permettant de tracer un TS5 : « alors | si on veut faire cinq il faut » (45f, transcription S5, 10'36) : ils entrent les valeurs par alternance {4/3/3/4/3/3/4/3} et lancent un test avec une valeur entrée correspondant à celle de la mesure voulue (5) — ce qui confirme que les élèves ont bien identifié le lien entre la mesure et la valeur du nombre de répétitions de la première boucle, et le fait que la valeur entrée par l'utilisateur correspond à la mesure voulue. La rétroaction obtenue est interprétée comme étant le résultat d'une erreur sur la boucle b_5.

$$TS(n)[5] \rightsquigarrow \left(\text{  }, 30, 5 \right)$$

31. Obtenu en pressant les touches du clavier numérique ligne par ligne et de gauche à droite.

Différents tests sont effectués pour les valeurs de b_5, en commençant par $b_5 \leftarrow 6$, c'est-à-dire en doublant la valeur initiale : les élèves semblent identifier le côté de « la pointe du triangle » non tracé à la boucle b_5.

$$TS(n)[5] \rightsquigarrow \left(\text{ 

} , 32, 5 \right)$$

Suivront une suite de tests :

— b_5 sera passée de 6 à 5 puis à 4 puis à trois,

— puis b_4 sera fixée à 4 puis à 3 puis de nouveau à 4,

— b_3 prendra alors la valeur 3 puis la valeur 4,

— ensuite b_7 et b_8 seront modifiées, passant de 4 à 3,

— b_4 et b_5 suivront, passant de 3 à 4 puis revenant à la valeur 3 et en poursuivant l'alternance 4-3 jusqu'à b_7

Cette suite d'essais laisse à penser que les élèves tâtonnent, procèdent par essai-erreur sans anticiper leurs actions ni analyser les résultats. E1 ponctuera alors cette exploration d'un « ah ça me saoule ça » (45f, transcription S5, 14'43) qui marque à la fois son agacement et le début d'une prise en compte de la nécessité de chercher d'où vient l'erreur. Le binôme utilisera alors le mode pas à pas pour tenter de comprendre le fonctionnement du script et d'identifier quelle boucle trace quelle partie : « déjà faut savoir où est-ce que c'est ici » (3.4, 385)

Identification des boucles : difficultés d'interprétation Les erreurs concernant plusieurs boucles, et sans qu'il y ait d'unité dans les différentes familles de boucles, les rétroactions sont rendues difficilement interprétables (figure 3.40, p. 290).
En outre, le début du tracé induit les élèves en erreur : l'enchainement des hexagones tracés par les boucles b_3 et b_4 leur semble erroné, puisqu'il donne l'impression qu'il manque un hexagone (figure 3.41, p. 290, et 45f, transcription S5, 16'19-16'31). Pour les élèves, l'erreur viendrait de b_3, boucle qu'ils modifient en lui ajoutant une répétition. Cette correction, erronée est questionnante, puisqu'elle semble impliquer que, pour ce groupe, seule la valeur du nombre de répétitions importe : si on demande de tracer cinq hexagones, ils seront tracés à partir d'un des sommets des triangles de base — ce qui est faux. Ici, les élèves prennent en compte la figure et le nombre de répétitions, mais pas l'algorithme. Cette difficulté qu'ont ces élèves à interpréter cet hexagone qui n'est pas encore tracé se retrouve dans d'autres groupes. Il est probable que cela provienne en partie de l'absence de nécessité visible de « tourner » avant le sommet. Ainsi, comme l'illustre la figure 3.42a (p. 290), à plusieurs reprises, le tracé change de direction sans aller jusqu'à un sommet, alors que le sommet en question n'est pas encore tracé. Un autre algorithme de tracé aurait été possible, conservant la contrainte « ni trou ni saut », mais en ajoutant celle de ne pivoter « qu'avec raison » : dans ce cas, le tracé d'un côté se poursuit jusqu'au sommet s'il n'a pas encore été tracé, ou avant s'il l'a déjà été (voir le script correspondant figure 3.3, p. 286). Cette alternative n'avait pas été envisagée.

368	15 :24.146	E2	fin b1
369	15 :24.146	T2	quatre fois
370	15 :25.860	T2	XXX
371	15 :26.903	E2	b2 en cours
372	15 :26.903	T2	répéter
373	15 :27.711	T1	trois fois
374	15 :31.854	E1	début b3
375	15 :31.854	T1	quatre
376	15 :33.121	T1	un
377	15 :34.164	T1	deux
378	15 :35.624	T1	quatre
379	15 :36.891	E1	b3 trop long
380	15 :36.891	T1	oh
381	15 :37.792	T2	non déjà là c'est faux
382	15 :38.792	T1	ouai
383	15 :39.313	E1	montre fin b2 puis fin b3
384	15 :39.313	T1	donc là faut que t'en mette quatre et là faut que t'en mette trois
385	15 :41.568	T2	déjà faut savoir où est-ce que c'est ici
386	15 :44.102	T1	bah oui c'est ici
387	15 :45.500	T2	mais où ici
388	15 :45.500	G2	balaye le script avec la souris
389	15 :47.524	T1	là faut que tu mettes
390	15 :47.524	G1	montre entête
391	15 :48.947	T2	mais non
392	15 :51.711	T1	faut que tu mettes cinq
393	15 :51.711	G1	pointe les hexagones de b2 un à un
394	15 :52.084	EXEC	STOP_button(all)
395	15 :52.094	SNP	SNP STOPbutton
396	15 :52.096	EXECUTION	STOP receiveKey(478)
397	15 :52.106	SNP	SNP STOP
398	15 :53.514	EXEC	KEY_n
399	15 :53.527	EXECUTION	START receiveKey(478)
400	15 :53.981	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
401	15 :56.388	T2	là regarde
402	15 :59.682	PASaPAS	SBS_slider(44)
403	15 :59.818	EXEC	KEY_enter
404	16 :02.575	ENTRÉE	ANSW <<5>>
405	16 :05.307	T2	là on est là
406	16 :06.475	E2	b1 en cours
407	16 :06.475	T2	répéter quatre fois
408	16 :08.847	T2	un | deux | trois
409	16 :13.344	E2	début b2
410	16 :13.344	T2	répéter trois fois
411	16 :15.400	T2	un deux trois
412	16 :16.885	T2	encore
413	16 :19.201	E2	sur b3-b4
414	16 :19.201	T2	là déjà c'est faux
415	16 :20.549	T1	ici c'est pas bon
416	16 :28.785	T2	après
417	16 :31.792	VALEURS	b3 : répéter *5* fois <<4>>

Transcription 3.4 – 45f, S5, extrait 2

Modification de la structure (étapes 64-72) Comme les différents essais sont infructueux, les élèves modifient alors la structure, en ajoutant un bloc 

à la fin du corps de la boucle b_6, bloc qu'ils modifieront ensuite (60 degrés, figure 3.4, p. 287). En effet, au moment où le tracé est produit par la boucle b_7, E2 constate que le tracé repart vers le bas (3.5, 556). C'est en fait la boucle b_8 qui débute, alors qu'il devrait continuer vers le haut (pour prolonger b_7) : il faut donc compenser, en quelque sorte, cette rotation, de 120° ou 60° selon le sens (figure 3.43, p. 291). Cette erreur va se répercuter sur la suite de la séance, y compris lorsqu'une autre élève vient modifier les scripts à la place de ce binôme. Celle-ci détermine la valeur du nombre de répétitions de b_7 en la mettant en relation avec les valeurs de b_1 et b_2 : $b_7^5 \leftarrow 12(b_1^5 \times b_2^5)$; puis

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [2 × [*mesure* − 1]] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [2 × [*mesure* − 1]] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
répéter [*mesure* − 2] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [*mesure* − 3] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [*mesure* − 3] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

Script 3.3 – Script solution « Pivoter avec raison »

555	21 :37.394	E2	b7
556	21 :37.394	T2	en fait ça part par là alors qu'en fait ça devrait pas partir par là
557	21 :38.996	T1	mais oui
558	21 :40.704	EXECUTION	FIN receiveKey(478)
559	21 :40.719	SNP	SNP FIN478
560	21 :41.332	T1	on dirait euh qu'c'est euh ce qu'on met à chaque fois au de degrés
561	21 :44.024	EXECUTION	START receiveKey(478)
562	21 :44.251	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
563	21 :44.743	EXECUTION	STOP receiveKey(478)
564	21 :44.751	SNP	SNP STOP
565	21 :47.742	T2	faut mettre un autre
566	21 :49.046	T2	faut mettre celui-là
567	21 :50.052	T2	faut mettre cent vingt degrés
568	21 :52.003	T1	non soixante
569	21 :53.878	T1	tu veux dire
570	21 :54.835	T1	faut qu'on mette soixante en gros

Transcription 3.5 – 45f, S5, extrait 3

$b_7 \leftarrow 20$ après avoir modifié b_1 à 5 et b_2 à 4. Le binôme terminera en tentant d'ajuster la valeur de b_7 en effectuant $b_7 \leftarrow 11$ puis $b_7 \leftarrow 10$. La dernière rétroaction obtenue étant la suivante :

$$TS(n)[5] \rightsquigarrow \left(\text{  } \text{} , 43, 5 \right)$$

Block_478
Quand n est pressé
initialisation
compteur prend la valeur
afficher la variable mesure
tourner de 120 degrés à droite
répéter 4 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 3 fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter 3 fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 4 fois (commandes)
.....tracer
.....tourner de 60 degrés à droite
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 4 fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

Script 3.4 – Modification de la structure (45f,S5,22'23)

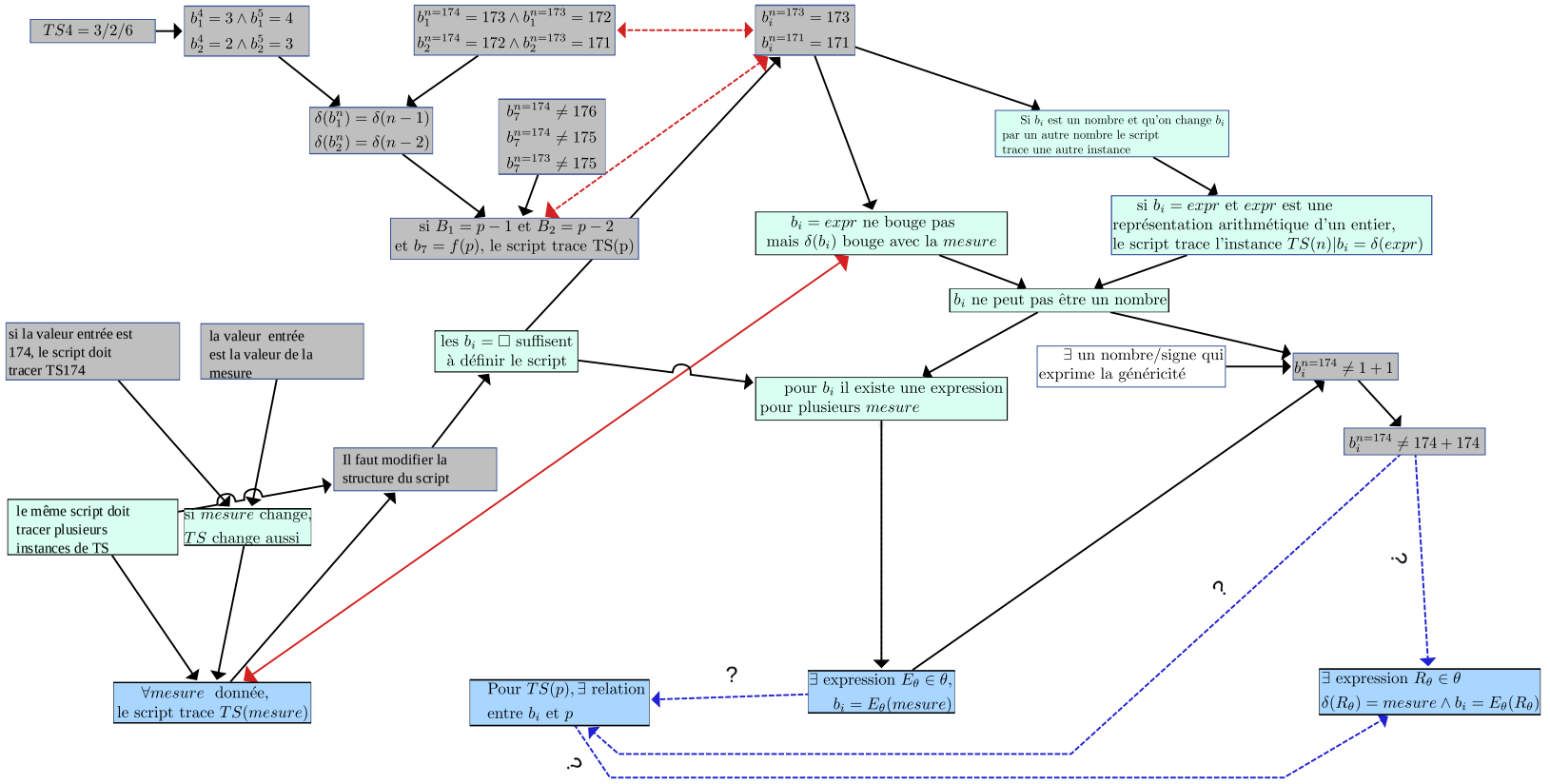


Figure 3.34 – Espace des faits-contraintes - 45a

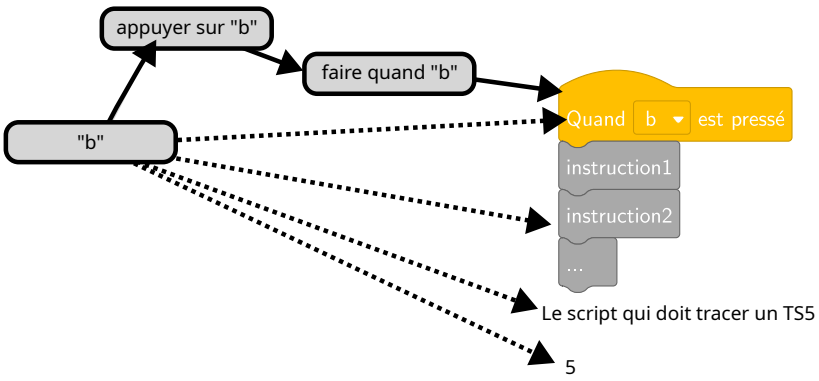


Figure 3.36 – Un raccourci sémiotique (45f, S3)

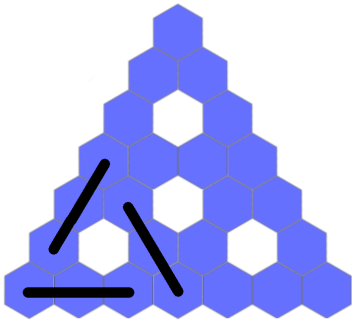
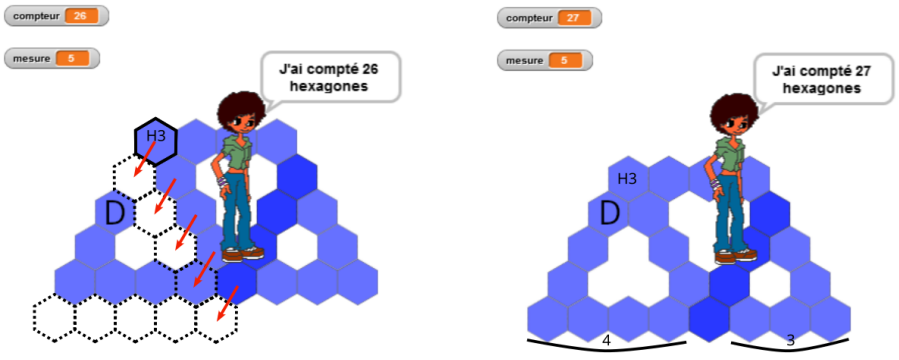


Figure 3.37 – « trois comme y'en a trois », (45f,S3)



(a) $b_i = 3$ **(b)** $b_1 = 4, b_{i\neq 1} = 3$

Figure 3.39 – La gestion d'un manque dans le tracé d'un TS5. Le "D" indique le premier hexagone tracé (45f, S3)

(a) 14'50 **(b)** 15'03 **(c)** 15'52

Figure 3.40 – Des rétroactions difficilement interprétables (45f, S5)

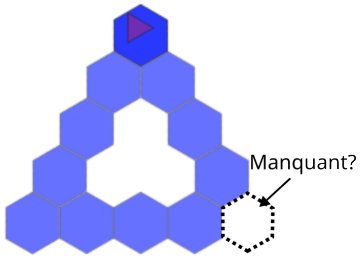
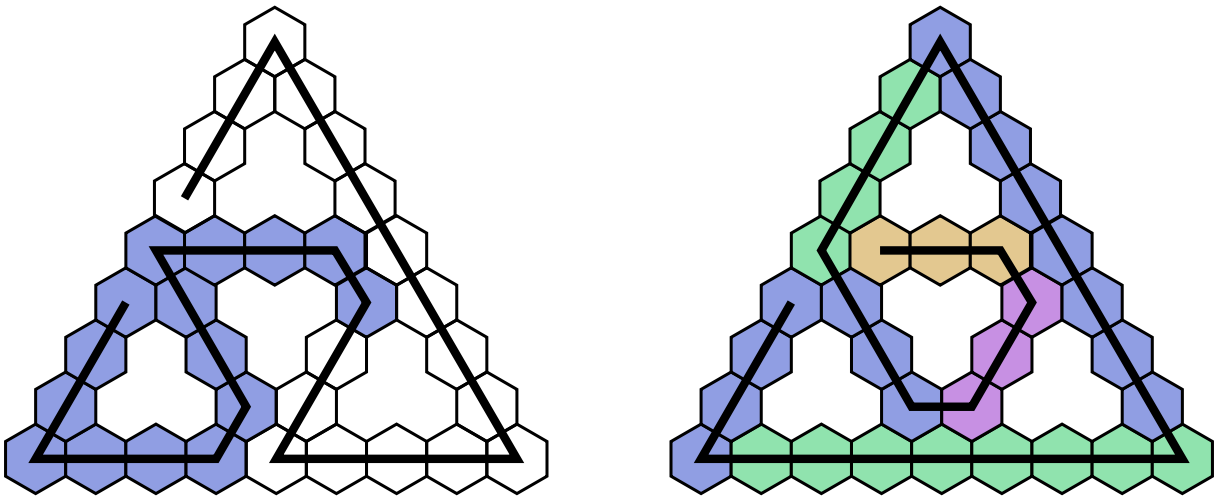


Figure 3.41 – Jonction b_3-b_4 : un hexagone manquant ?



(a) Sans raison **(b)** Avec raison

Figure 3.42 – Pivoter avec ou sans raison, deux algorithmes de tracé possibles

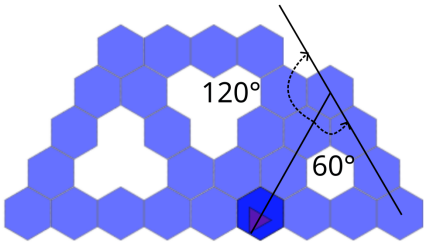


Figure 3.43 – Compenser un manque d'hexagones par une rotation (45f, S5, 22'23)

Recherche de régularités

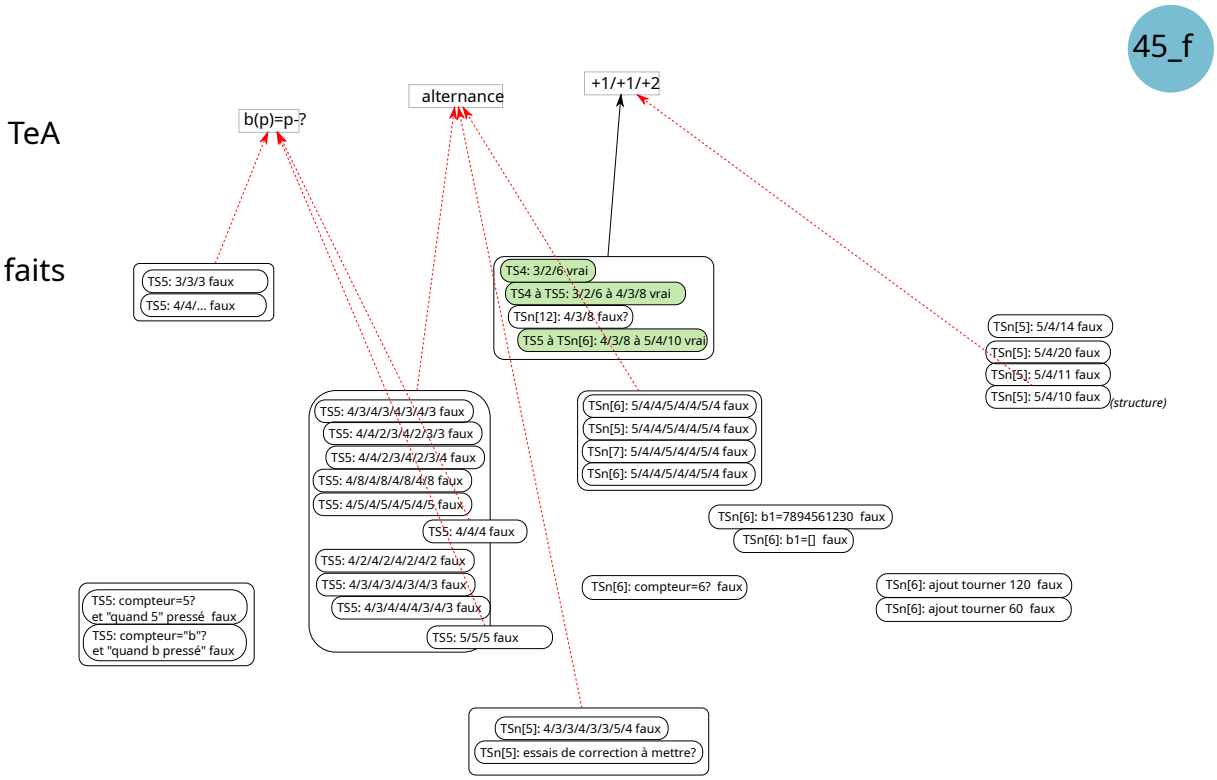


Figure 3.44 – Faits et TEA, groupe 45f

Le groupe 45f construit peu de faits validant leurs actions, donc peu de faits tiers. Un TEA original, alternant des valeurs, semble à l'œuvre, et de façon persistante. Il induit des difficultés d'interprétation que les élèves n'ont pas pu dépasser.

Un TEA inamovible Comme le groupe 45d, le groupe 45f construit peu de faits tiers de validation. On voit (figure 3.44, p. 292) que la plupart des faits construits sont issus du TEA d'une définition des valeurs des boucles « par alternance ». Ce TEA, comme nous l'avons dit plus haut, se base sans doute en premier lieu sur la figure (voir figure 3.37, p. 289) avant de ne concerner que les valeurs. Le nombre d'occurrences des faits construits à partir de ce TEA devraient l'invalider, mais il reste mobilisé tout au long des trois séances. Cela laisse à penser que ces élèves, en dépit de leurs essais de compréhension de l'algorithme en associant les boucles à leur tracé grâce au mode pas à pas, n'ont pas identifié l'organisation globale du tracé : ils modifient en général une seule boucle sans réussir à faire de lien avec le tracé issu des autres boucles. Cette compréhension de la structure de l'algorithme aurait dû être initiée lors de la séance 2, mais lors de cette séance, le binôme a fait des essais qui sont restés localisés aux premières et deuxièmes boucles. La méconnaissance de la structure du programme, de la logique du tracé, est ici un obstacle à la construction de faits tiers de validation. Les seuls faits validants sont issus soit de l'enseignante, soit de leur voisine : c'est sans doute insuffisant pour permettre de construire une alternative possible à leur TEA.

Cumul d'erreurs et interprétation En outre, le cumul d'erreurs sur plusieurs boucles non forcément consécutives rend l'interprétation de la rétroaction bien plus difficile, puisqu'elle nécessite une recherche globale et organisée des erreurs. Par exemple, la rétroaction du TS5 donnée figure 3.40a (p. 290) nécessite de constater :

1. que b_1 semble correcte (on a bien les 5 hexagones du côté),
2. que b_2 l'est sans doute aussi,
3. que la boucle b_3 est trop longue d'un hexagone,
4. que la boucle b_4 est trop courte de un hexagone (si on a compris qu'elle ne traçait aucun des sommets correspondant au côté tracé),
5. que la boucle b_5, dont le tracé est décalé à cause de b_3, est trop longue de un hexagone (alors même que la forme tracée est plus proche de la forme idéale que s'il n'y avait pas eu d'erreur en b_3),
6. que la boucle b_6 est trop courte d'un hexagone (et non de deux comme on pourrait l'imaginer, ce qui est dû au double décalage issu des erreurs sur b_3 et b_4),
7. que la boucle b_7 est trop courte d'un nombre d'hexagones difficilement identifiable (ici 5, puisqu'on identifie que le tracé de b_8 commence lors du premier chevauchement)
8. et donc que b_8 est probablement correcte (ce qui est très peu visible ici et impose d'imaginer ce qu'aurait été le tracé de cette boucle si toutes les autres avaient été correctes).

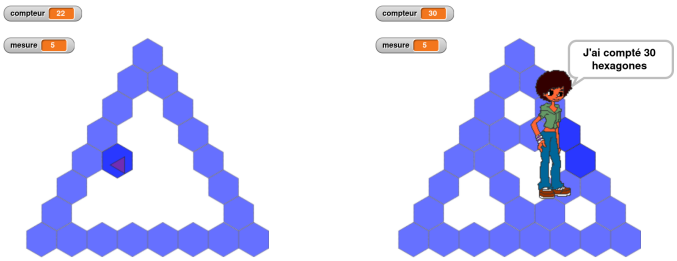
Comme les boucles tracent successivement les quatre triangles de base, une méthode de recherche et correction d'erreur efficace, et généralement constatée, est de traiter les erreurs dans l'ordre de leur apparition, ce que ne font pas ces élèves malgré leur connaissance du mode pas à pas, utilisé par peu d'autres élèves.
Nous avons évoqué plus haut un algorithme alternatif (« pivoter avec raison », script 3.3, illustré figure 3.42b, p. 290). Cette alternative aurait-elle rendue la correction des erreurs, et notamment des erreurs multiples, plus aisée ? Il semble que non, pour au moins deux raisons :

— le tracé commençant par le « grand triangle », soit les côtés externes du TS, celui-ci peut paraitre valide jusqu'à ce qu'un des côtés intérieurs soit tracé et indique une erreur par un chevauchement par exemple (voir figures 3.45a et 3.45b, p. 294)

— le tracé des côtés externes implique quatre boucles, et si la première est identique à celle de notre situation, les deux suivantes, pour être intéressantes du point de vue algébrique, mettent les élèves en difficulté lorsqu'il s'agit d'établir la relation entre mesure et nombre de répétitions, puisque ici cette relation est identique à la boucle b_7 : $2 \times (mesure - 1)$. Cette difficulté d'établir une relation pour b_7 sera abordée dans l'analyse des prochains groupes.

Ainsi, si les élèves procèdent dans un premier temps avec un TEA similaire à celui qui a été constaté majoritairement dans notre situation, en ajoutant un à chaque boucle, la rétroaction produite est complexe à interpréter (figure 3.46a, p. 294). En effet, tant que la quatrième boucle n'est pas totalement tracée, aucune erreur ne semble visible (figure 3.46b, p. 294), alors que les deuxième et troisième boucles sont erronées. La quatrième boucle provoquant un chevauchement (figure 3.46c, p. 294), une réaction normale serait de considérer que cette boucle trace un hexagone de trop, et donc qu'il faut diminuer son nombre de répétitions de un. Ceci semble être validé lorsque la cinquième boucle commence (figure 3.46d, p. 294), et de même au début de la sixième. L'exécution complète pouvant laisser penser que l'erreur est due à la sixième boucle (figure 3.46e, p. 294). L'identification des boucles erronées et la nature des erreurs est ainsi relativement complexe.

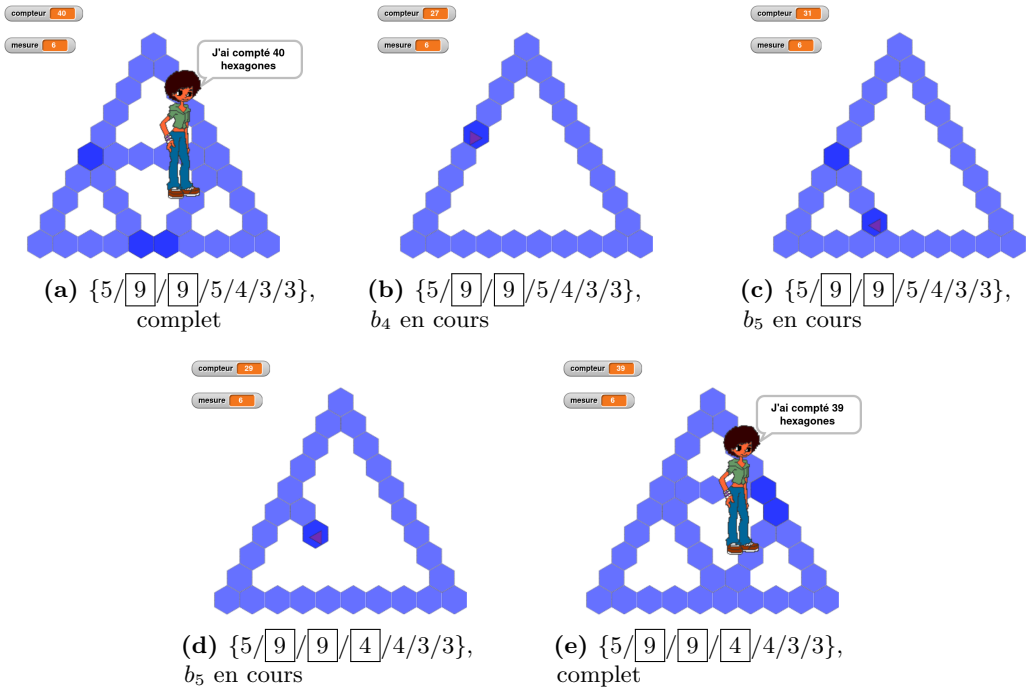
32. Les valeurs encadrées sont celles qui sont erronées.



(a) début b_5 **(b)** complet

Figure 3.45 – Erreurs multiples sur le script « pivoter avec raison » : {4/7/7/3/3/2/2} [32]

Quel que soit l'algorithme de tracé choisi, il semble que le cumul d'erreurs soit dans tous les cas un obstacle à l'interprétation de la rétroaction (et la construction de faits tiers de localisation), à moins d'organiser l'exploration des erreurs en suivant la séquence des boucles. Comprendre qu'un algorithme est, entre autres choses, une suite organisée d'instructions pourrait aider à mettre en œuvre cette démarche de recherche d'erreurs.



(a) {5/9/9/5/4/3/3}, complet **(b)** {5/9/9/5/4/3/3}, b_4 en cours **(c)** {5/9/9/5/4/3/3}, b_5 en cours

(d) {5/9/9/4/4/3/3}, b_5 en cours **(e)** {5/9/9/4/4/3/3}, complet

Figure 3.46 – Application du TEA « on ajoute un à chaque boucle » à l'algorithme « pivoter avec raison », pour passer d'un TS4 ({4/8/8/4/3/3}) à un TS5 ({5/10/10/5/4/4})

Généralisation algébrique

Figure 3.47 – Type de généralisation - 45f

Les élèves du groupe 45f ont tout juste abordé le problème de la généralisation du script. Le problème semble être posé, et les élèves chercheront par quoi remplacer les valeurs du nombre de répétitions des boucles : un nombre qui pourrait être n'importe quel nombre (NPG) ou un possible signe générique (SG).

Le script générique : position du problème L'enseignante intervient à plusieurs reprises auprès de ce groupe afin de préciser ce qu'on attend du programme générique, ce qu'elle avait déjà souligné en plénière en début de S4 et début de S5.
L'élève E2 a bien établi qu'« ils veulent un programme pour n'importe quelle mesure » (45f, transcription S4, 12'45) et que « en fait faut trouver un truc où en fait faut trouver un truc que l'ordinateur il trouve tout seul »(45f, transcription S4, 13'04) : le problème est posé. Plus tard, après que E1 a annoncé « on a bon », E2 expliquera qu'« après faut être sûr et lui il veut pas six et t'as l'impression que par exemple il veut quinze | et ben il faut que l'ordinateur il change tout seul ». En fin de séance 4, après que E2 a précisé que « l'ordinateur il change tout seul », E1 semble envisager l'utilisation des opérateurs, comme suggéré par l'enseignante : en montrant (sans doute) la zone où sont les opérateurs, elle affirme « du coup faut qu'on mette des trucs comme ça pour là j'pense ». Autrement dit, si la valeur doit bouger, il faut faire un calcul, ce qu'avait évoquée l'enseignante en début de séance en plénière : « je refais pas le programme à chaque fois il faut que l'ordinateur se débrouille pour faire la première boucle. Donc c'est l'ordinateur qui doit faire ses propres calculs. » (45 Prof, S4, 26'20). Cette nécessité ne sera pas remobilisée dans la séance suivante. En séance 5, dès que le binôme se met en place, les élèves font appel à l'enseignante, qui va leur réexpliquer le fonctionnement attendu (3.6).

33	00 :52.996	TProf	voilà
34	00 :55.611	TProf	et | là si je le lance
35	00 :58.338	TProf	si j'appuie sur "n"
36	00 :59.050	EXEC	KEY_n
37	00 :59.068	EXECUTION	START receiveKey(478)
38	00 :59.501	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
39	01 :00.133	TProf	il me demande
40	01 :01.021	T1	oui
41	01 :01.021	G1	hoche la tête
42	01 :02.189	TProf	donc quand je vais mettre quatre il fait bien quatre
43	01 :03.113	ENTRÉE	ANSW <<4>>
44	01 :03.953	T1	oui euh ça on le savait on était en train de faire ça hier
45	01 :05.425	TProf	ça vous savez
46	01 :07.347	PASaPAS	SBS_slider(7)
47	01 :07.965	TProf	donc par contre si je le lance et que je mets autre chose que quatre
48	01 :11.381	TProf	ben y doit
49	01 :14.425	TProf	vous devez lui dire
50	01 :16.270	TProf	de modifier les boucles

Transcription 3.6 – Retour sur le script générique (45f, S5)

À la suite de cet épisode, les élèves se sont probablement confrontés à la tension entre un script qui ne change pas et une valeur entrée qui change : après avoir modifié le script générique pour tracer un TS6 (erroné), ils vont lancer l'exécution en répondant « 6 » à l'invite (« c'est là qu'faut mettre six », 45f, transcription S5, 4'05), puis avec une entrée de cinq, puis une entrée de sept avant de retenter une valeur de six. Le résultat, à un léger facteur d'agrandissement près, est toujours le même. On peut ainsi penser que, malgré le fait que les élèves de ce groupe ne soient jamais parvenus à construire une instance à partir d'une instance précédente, le problème de la généralisation a bien été posé, au moins pour E2.
Par ailleurs, en fin de séance 4, après que E2 a précisé que « l'ordinateur il change tout seul », E1 semble envisager l'utilisation des opérateurs, comme suggéré par l'enseignante : en montrant (sans doute) la zone où sont les opérateurs, elle affirme « du coup faut qu'on mette des trucs comme ça pour là j'pense ». Autrement dit, si la valeur doit bouger, il faut faire un calcul, ce qu'avait évoqué l'enseignante en début de séance en plénière : « je refais pas le programme à chaque fois il faut que l'ordinateur se débrouille pour faire la première boucle. Donc c'est l'ordinateur qui doit faire ses propres calculs. » (45 Prof, S4, 26'20). Cette nécessité ne sera pas remobilisée dans la séance suivante.

Des nombres particuliers ou génériques En fin de séance 4, après l'intervention de l'enseignante leur expliquant le comportement attendu du script, E2 précise qu'il ne faut pas mettre douze, « parce que si on met douze après on sera obligées de changer à chaque fois » (3.7, 491). Le nombre douze a été donné en exemple d'absence d'adaptation du programme (transcription 3.2, 442). Ici on peut se demander si le douze est un cas particulier ou un cas générique pour ces élèves : est-ce que ça ne marche pas parce que c'est douze, ou est-ce que ça ne marche pas parce que c'est un nombre (qui donc ne varie pas) ?
Un autre nombre, potentiellement porteur de généricité, a été utilisé par les élèves : $b_1 \leftarrow$ 7894561230. Comme précisé plus haut, dans les interactions des élèves, rien ne nous permet réellement de confirmer l'hypothèse d'un nombre pris comme exemple de tous les nombres possibles. Cependant, ce nombre ayant été entré à la suite de l'exploration de la tension fixe-variable évoquée au paragraphe précédent, il est possible qu'il soit un exemple de cette tension : ce pourrait être ce nombre, comme n'importe quel autre aussi improbable soit-il.

Un signe générique Le signe vide [](dans 

) a été utilisé à deux reprises : pour l'ensemble des boucles dans un premier cas, et pour la boucle b_1 en remplacement du NPG 7894561230. On peut ici se demander si ce signe vide est lié à une préparation du script : on efface puis on met les bonnes valeurs, ou on efface une valeur erronée en attendant de trouver la bonne. On peut aussi émettre l'hypothèse qu'il s'agit d'un début de signe générique (SG) : c'est ce qu'il va falloir remplacer, c'est une marque précisant la place que devront prendre les différentes valeurs si « l'ordinateur se débrouille tout seul ».
Lors de l'épisode où le groupe 45f effectue les actions $b_a \leftarrow []$, suivies de $b_a \leftarrow \{5/4/10\}$ — suite à l'intervention de l'enseignante visant à permettre aux élèves de poser le problème —, E1 demandera trois fois « on met combien » (3.7, 490, 498, 513). La deuxième intervention semble montrer que le « on met combien » concerne la valeur à entrer lors de l'invite, puisqu'elle précise « ah c'est là qu'il faut mettre » juste après l'apparition de l'invite (3.7, 495). Pourtant, elle s'interrogeait, lors du précédent « on met combien », sur la valeur à entrer pour la boucle b_1 : l'action $b_1 \leftarrow []$, qui dans les transcriptions est indiquée au temps 13'12, a en fait eu lieu à 12'54[33]. C'est à ce moment que l'élève supprime la valeur existante pour le nombre de répétitions

33. Geste que l'on observe sur la vidéo.

de b_1, mais l'événement n'est déclenché que lorsqu'elle lance le programme. En effet, lorsqu'une modification d'un paramètre est en cours, le dispositif n'enregistre pas ces actions comme un événement. L'événement sera déclenché lorsque l'entrée du paramètre sera validée, soit avec la touche « entrée », soit par un clic en dehors du paramètre en cours de modification, c'est-à-dire lorsque le bloc perd le focus. Ainsi, l'élève E1 supprime la valeur à 12'54, mais la perte du focus aura lieu lors du lancement du script, à 13'12. Le dispositif ne permet pas, en l'état, de capter les événements atomiques liés à une modification ou une entrée de valeur : si l'élève tape sur une certaine suite de touches et efface certains des caractères tapés, un seul événement sera envoyé, lors de la perte du focus : la chaine de caractères finale.
La première intention de E1 était donc de trouver un nombre à mettre pour remplacer un espace vide, qui marque la place de ce qu'il faut changer. Cependant, E1 semble voir cet espace comme une simple marque qui sera remplacée par un nombre, alors que E2 voit cet espace comme une marque qui doit être remplacée par « un truc que l'ordinateur il trouve tout seul » (3.7, 492), soit une expression de θ qui résoudrait le problème. Ce problème est dû au fait établi par E2 que, si on rentre un nombre (par exemple 12), il faudra changer cette valeur pour toute nouvelle instance : « bah pas douze du coup parce que si on met douze après on sera obligées de changer à chaque fois » (3.7, 491). Elle confirme qu'il y a une tension entre la multitude de TS à tracer (donc de valeurs à entrer) et le fait que si on a des nombres pour les valeurs de répétitions des boucles, cela trace une instance spécifique : à la deuxième occurrence de la question « on met combien », s'interrogeant cette fois sur la valeur à entrer lors de l'invite, E2 voit une impossibilité, puisque « oui mais on peut pas parce que pour l'instant c'est encore le truc de cinq » (3.7, 499), le « truc de cinq » étant le TS5, puisque le script en cours est, mis-à-part b_1, {4/3/8}. E1 cherche un nombre, E2 cherche un signe.
D'un point de vue méthodologique, nous considérons que le signe vide [] — soit le signe ◯ dans une instruction paramétrée de Snap ! — est possiblement un signe générique seulement si le script obtenu est testé. La première occurrence de ce signe n'est donc pas considérée comme un signe générique, tandis que la deuxième le sera — tout en étant bien conscient du doute qui persiste.

Relation boucle-mesure En séance 4, E2 semble identifier d'une part que la valeur entrée par l'utilisateur est la mesure du TS à tracer, et d'autre part que la relation permettant de passer de la mesure au nombre de répétitions de la boucle b_1 est de diminuer la mesure de une unité : lorsque les élèves décident de construire le script qui correspondrait à une entrée de 6 (3.7, 516), E2 en déduit la valeur à entrer : « donc il faut mettre cinq » (3.7, 518). E1 propose le quatre pour la boucle suivante, et E2 et va se référer aux autres scripts pour déterminer les valeurs valides. Cependant, ces faits ne sont pas stabilisés ; en séance 5 les élève reviendront sur l'alternance. Le fait « b_1 c'est la mesure moins un » a été construit avec l'enseignante, mais il n'est pas mobilisé lors de la séance suivante.

Bilan

Ainsi, ce groupe a été très en difficulté tout au long des trois séances et il n'est jamais parvenu à réellement construire (seul) une instance valide. Il n'a donc pas commencé à généraliser, puisqu'aucun motif n'était stable. Cependant, il est probable que les élèves aient néanmoins construit la nécessité de résoudre la tension fixe-variable, entre les valeurs du nombre de répétitions des boucles qui sont fixes et les valeurs variables entrées par l'utilisateur pour tracer un quelconque TS. Cette légère émergence de la généralisation leur a permis de poser le problème : par quoi remplacer les nombres pour que cela fonctionne pour toute mesure entrée ?

485	12 :45.604	T2	ils veulent un programme pour n'importe quelle mesure
486	12 :48.514	T2	et euh après euh ya XXX
487	12 :48.514	G2	montre opérateurs
488	12 :50.019	T2	il est déjà cinquante sept donc je sais pas si on aura le temps hein
489	12 :53.406	T1	eh XXX
490	12 :59.564	T1	on met combien du coup ?
491	13 :01.604	T2	bah pas douze du coup parce que si on met douze après on sera obligées de changer à chaque fois
492	13 :04.940	T2	en fait faut trouver un truc où en fait faut trouver un truc que l'ordinateur il trouve tout seul
493	13 :12.633	VALEURS	b1 : répéter ** fois <<4>>
494	13 :12.776	EXECUTION	START receiveKey(478)
495	13 :12.872	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
496	13 :13.802	T1	merde
497	13 :14.683	T1	ah c'est là que faut mettre
498	13 :17.079	T1	on met combien ?
499	13 :18.069	T2	oui mais on peut pas parce que pour l'instant c'est encore le truc de cinq
500	13 :18.069	G2	balaye le script de la main
501	13 :22.118	T1	du coup faut tout qu'on efface
502	13 :23.732	T1	les trois là
503	13 :28.904	VALEURS	b2 : répéter ** fois <<3>>
504	13 :32.596	VALEURS	b3 : répéter ** fois <<3>>
505	13 :34.227	T2	six
506	13 :35.910	T1	non pas moi non c'est douze moi j'dirais
507	13 :35.920	VALEURS	b4 : répéter ** fois <<3>>
508	13 :38.168	T2	XXX
509	13 :39.584	E2	hors contexte
510	13 :41.167	VALEURS	b5 : répéter ** fois <<3>>
511	13 :45.608	VALEURS	b6 : répéter ** fois <<4>>
512	13 :50.016	VALEURS	b7 : répéter ** fois <<8>>
513	13 :54.802	T1	du coup on met combien ?
514	13 :56.279	VALEURS	b8 : répéter ** fois <<3>>
515	13 :56.435	T2	bah je sais pas faut trouver
516	13 :59.198	T1	mmmh six
517	14 :01.475	T2	essaie
518	14 :05.178	T2	donc là faut mettre cinq

Transcription 3.7 – 45f, S4 : SG ou préparation ?

On peut cependant s'interroger sur le sens que les élèves de ce groupe ont mis dans cette activité : E1 va d'ailleurs affirmer « c'est chiant, mais ça passe notre temps de maths » (45f, transcription S5, 21'14).

3.3.3 Synthèse Groupement 2

Les groupes analysés dans cette partie font partie de ceux qui n'ont pas mobilisé le paramètre, mais qui ont posé le problème de la généralisation, avec une démarche assez linéaire.
Du point de vue de la rétroaction et de son interprétation, trous, chevauchements, et forme invalide du tracé restent les critères principaux retenus pour la validation du script ou des actions de ces groupes. Cependant, la valeur du compteur, c'est-à-dire le nombre d'hexagones dénombrés par le script, est cette fois parfois mobilisée comme critère. Pour le groupe 45f, c'est même une caractéristique essentielle du TS, puisque les élèves de ce groupe non seulement le prennent en compte pour invalider, mais aussi envisagent qu'en modifiant la valeur de la variable compteur en lui affectant une certaine valeur p, le script va tracer le TS nécessitant p hexagones : modifier le compteur devrait modifier le TS tracé.
La situation et l'algorithme de tracé choisi — par son organisation et les rétroactions qu'il produit — compliquent parfois l'interprétation : lorsque des erreurs multiples sont présentes, l'interprétation est rendue complexe. Un autre algorithme est envisageable, mais il ne semble pas résoudre cette difficulté. En fait, il faudrait que les élèves fassent comme le groupe 45d ou

45e, diviser le script pour le modifier partie par partie, problème par problème. Les élèves de ce groupement ont exploré la structure du tracé, la logique de l'algorithme. Le groupe 45a en modifiant les instructions « tourner » et en ajoutant une neuvième boucle — ce qui revient à faire tracer par chaque boucle *environ* un côté du triangle de base —, et le groupe 45f aurait pu faire de même, ayant envisagé une correction de la rotation (entre b_7 et b_8) arrivant trop tôt. Ces questions sur la structure du tracé pourraient amener une meilleure compréhension de l'algorithme et notamment du lien entre le nombre de répétitions et ce qui est tracé par une certaine boucle. C'est peut-être ce qui a eu lieu partiellement pour le groupe 45a (qui a fait ses modifications de structure avant de passer à la généralisation du script), mais cela semble peu visible pour le groupe 45f : on peut supposer que la faible quantité de faits validants construits ne facilite pas la compréhension de l'algorithme. Le groupe 45f, par ailleurs, semblait un temps faire le lien entre les boucles et ce qui était tracé, mais ce lien parait s'être étiolé dans les difficultés d'interprétations d'erreurs multiples. Cependant, le fait que les boucles définissent, finalement, entièrement le script semble être construit.
Les chevauchements semblent pris en compte pour identifier les erreurs et leur localisation, mais encore faut-il qu'ils soient visibles : lorsque le groupe 45a trace l'instance TS174, le chevauchement de deux hexagones n'est pas visible. Leur script, pourtant erroné, est considéré par les élèves comme étant valide : un « faux-fait » est construit. Il ne s'agit pas ici d'une erreur d'interprétation, mais d'une limite de l'EPGB. Nous verrons d'autres faux-faits ayant pour origine l'EPGB, et des conséquences diverses, chez d'autres groupes.
Les élèves vont vers une généralisation algébrique, mais les groupes détaillés dans cette partie n'ont pourtant pas construit de nombreuses instances, et par suite de nombreux faits sur lesquels se baser. Le groupe 45a passe directement du TS5 au TS174, en étendant la portée du TEA $+1/+1/+1$, c'est-à-dire en le rendant applicable à un plus grand nombre de situations. Malgré que leur script TS5 est invalide une fois construit en appliquant ce TEA, les élèves de ce groupe vont le généraliser en $+a/+a/+a$, ici avec $a = 170$, et en basant sur le script fourni $TS(4) = \{4/3/6\}$. Ce TEA ne sera cependant pas stabilisé, puisque après avoir ajusté leur script, les élèves aboutissent au TEA $b(p) = p$, « pour une instance p, je mets toutes les valeurs du nombre de répétitions des boucles p ». De même, le groupe 45f ne stabilise pas son TEA, revenant lors de la généralisation à un TEA « alterne ». Les groupes non détaillés ici (46a et 46c) on en revanche créé de nombreuses instances, et leurs TEA semblent davantage stabilisés. Le groupe 46a étendra lui aussi la portée du TEA qu'ils mobilisent : le TEA valide $+1/+1/+2$ se généralisera en TEA (invalide) $+a/+a/+a+2$. Le nombre de faits validants construits semble ici être corrélé à la stabilité des TEA.
Pour généraliser, les groupes 45a et 46a vont passer par une généralisation algébrique naïve, ou une « induction naïve » pour reprendre les termes utilisés par Radford (2008, p. 3), en construisant des instances « lointaines » des instances précédemment construites : le TS174 pour le groupe 45a et le TS100 (en fait TS101, puis TS102) pour le 46a. Pour ces deux groupes, le script ainsi construit est erroné, mais les tests effectués par les élèves montrent du mouvement concernant le lien entre la valeur de la mesure entrée et la valeur des boucles, notamment la première, et le passage d'une relation inter-objectale à une relation trans-objectale ou intra-objectale, les deux pouvant permettre une certaine forme de généralisation. Ainsi, le groupe 45a entre la valeur 174 pour un script $\{173/172/176\}$: le lien entre les boucles de familles B_1 et B_2 est celui attendu. Au contraire, pour le groupe 46a, le script construit $\{100/100/100\}$ sera testé avec une entrée de 100, identifiant ainsi la valeur de la mesure à la valeur du nombre de répétitions des boucles. C'est ce

que fera aussi le groupe 45a, choisissant une entrée de 173 pour le script {173/173/173}, de 171 pour {171/171/171} et de 174 pour une valeur de b_1 de (174 + 174). Plus tard, le groupe 46a rétablira un lien intra-objectal, entre b_1 et b_2 d'une part ($b_1 = b_2 + 1$, en effectuant $b_1 \leftarrow 101$), et entre b_7 et b_6 d'autre part ($b_7 = 2 \times b_6$, $b_7 \leftarrow 200$), perdant par là-même la relation avec la mesure. Concernant la généralisation du script, on va constater l'exploration de signes spécifiques, l'exploration du langage θ, pour tenter de résoudre le problème. La tension fixe-variable, entre le script dont les expressions ne doivent pas changer et les instances qui varient et impliquent que les dénotés de ces expressions changent, est bien établie. Pour la résoudre, les élèves vont envisager diverses utilisations du langage θ (donc de Snap ! en l'occurrence) :

— pour représenter ce nombre de répétitions qui doit changer sans changer ;

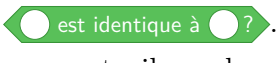
— ou pour représenter une relation avec la mesure, qui change sans changer.

Pour régler le problème de la tension entre la valeur du nombre de répétitions des boucles qui change et l'expression qui ne change pas, soit b_i fixe mais $\delta(b_i)$ variable, une première solution est d'utiliser un nombre qui prendra un caractère d'exemple, un nombre générique (Balacheff, 1987). Le groupe 45a fixe ainsi la valeur de chacune des boucles à 173 pour une entrée de 173, ou de 171 pour une entrée de 171 [34] : ils montrent ainsi ce qui devrait se passer en fonction d'une certaine entrée. Le groupe 46a a, lui aussi, envisagé un tel script, mais avec une entrée de 100 [35]. Le groupe 45f utilisera quant à lui une valeur qui parait aléatoire : 7894561230. Nous considérons que ce caractère de nombre qui pourrait être un exemple générique (NPG, nombre potentiellement générique) est différent du NG vu précédemment : il s'agit ici de montrer que ce pourrait être ce nombre, comme ce pourrait être un autre nombre, ou n'importe quel nombre, aussi incongru soit-il. On peut se demander si le nombre 100 choisi par le groupe 46a n'est pas aussi un NPG : ce n'est pas n'importe quel exemple, c'est un exemple qui porte en lui une certaine généricité (une centaine), et ce groupe testera ensuite $b_1 \leftarrow 10$. On retrouvera d'ailleurs des puissances de 10 chez d'autres groupes.
Cette idée d'un nombre qui pourrait être remplacé par un autre nombre se transforme, pour le groupe 45f, en $b_1 \leftarrow []$ (donc répéter ◯ fois) : le signe vide est ici un signe générique (SG) qui marque la place de ce qu'il faut remplacer, en fonction du tracé.
Ces différents types de signes seront aussi mobilisés pour exprimer non plus le nombre de répétitions, mais ce qui va changer dans l'expression des relations représentant le nombre de répétitions. Ainsi le groupe 45a mobilise le NG 174 pour le script {174+174/174+174/174+174}. Le groupe 46c mobilise des NPG dans les tests $b_1 \leftarrow 2 - 65448974897489784468944$ (lancé avec une valeur entrée de 12), ou $b_1 \leftarrow 2 - 56466$ (avec une valeur entrée de 4) : ces NPG montrent ce qui bouge d'une instance à une autre.
Ce même groupe utilisera, avant ces deux essais, $b_a \leftarrow$ (◯ - ◯), et après ces deux essais, $b_1 \leftarrow$ (0 - ◯). On voit ici l'occurrence du signe générique vide, mais aussi du nombre-signe générique (NSG) « 0 ». Ce NSG est similaire au NSG « 1 » utilisé par le groupe 45a : le script {174+174/174+174/174+174} étant lui-même une forme d'instanciation du script précédent {1+1/1+1/1+1}, tout se passe comme si le nombre-signe « 1 » était voué à être remplacé par un nombre définissant l'instance (174). Le NSG, comme le SG, n'est pas un exemple, c'est un signe marquant dans le

34. Ce qui n'est pas totalement invalide : les élèves ont ajouté une boucle, et chaque boucle doit tracer presque un côté du triangle de base. Les chevauchements n'étant pas suffisamment visibles, les élèves peuvent considérer le script comme valide.
35. Contrairement au groupe 45a, leur script est invalide et identifié comme tel.

texte de l'algorithme une place à prendre, ce que Radford nomme un « index ». Dans l'expression « 0+[] », les deux SG n'ont pas le même rôle : le « [] » désigne la place du nombre dépendant de l'instance (par exemple 56466), tandis que le « 0 » désigne la place du nombre qui dépendra de la boucle, puisque les valeurs de boucles sont différentes (par exemple 2 pour b_1)
Le groupe 46a explorera quant à lui la possibilité d'utiliser une certaine instruction de θ qui permettrait à l'ordinateur « de se débrouiller tout seul ». Les élèves de ce groupe ont envisagé les instructions 

, 

ou encore 

. Les élèves n'associent pas à ces expressions le même sens que le ferait un expert : ils explorent le langage à la recherche de quelque chose qui fonctionnerait. On retrouvera ces explorations dans le groupement suivant.
On peut noter que pour ce groupement, s'il y a bien une recherche de relation et de représentation des nombres variables, aucune expression produite ne correspond, d'une manière ou d'une autre, à la relation entre la mesure et les valeurs du nombre de répétitions des boucles.

3.4 Groupement 3

3.4.1 Groupe 46g

Le groupe 46g, pour lequel une captation audio et vidéo est disponible, semble s'approcher d'une forme de généralisation. De nombreux échanges avec le groupe voisin sont présents, mais, contrairement au groupe 45f, les élèves de ce groupe ne se contentent pas de copier leurs résultats. Le problème ne semble jamais être réellement posé par les élèves, ce qui les empêche de construire des faits tiers, et entraine de nombreux échanges hors contexte en attendant l'enseignante.

Description de l'activité

Le groupe 46g, après une séance 3 très peu productive, va réussir à construire les instances de TS successives de 5 à 10. Ils construiront ensuite partiellement les instances 22 et une version instanciée du script générique (100), semblant montrer que ces élèves sont alors capables de construire n'importe quelle instance. Les tentatives de généralisation seront cependant peu efficaces, laissant augurer une mauvaise position du problème.

Le compteur ou la mesure comme caractéristique (étapes 1-4)
En début de séance — au bout de trente-cinq secondes —, ce groupe fait déjà appel à l'enseignante. Celle-ci leur réexplique la méthode (dupliquer un script puis le modifier) et l'objectif (3.8, p. 303) : modifier le script afin qu'il trace un TS « de la taille au dessus ». Elle attire leur attention sur le nombre d'hexagones des côtés des triangles de base qu'elle montre et dénombre individuellement, en utilisant un des TS exemples du cahier,(3.8, 93-98). La propriété caractéristique des TS (la mesure) est ainsi mise en avant, mais non nommée.
Pour l'élève E1, « c'est pas compliqué » (46g, transcription S3, 4'22) : il suffit de changer l'initialisation du compteur en lui affectant le nombre d'hexagones du TS cible. Ce groupe commence ainsi par l'action *compteur* ← 19, le nombre 19 provenant sans doute du croisement entre l'affichage en cours sur l'espace d'exécution et la consigne « d'en faire un de la taille au dessus ». En effet, l'affichage montre (au chargement du programme de base de la séance [36]) une valeur affichée du compteur de 18 pour une mesure de 4 (figure 3.49, p. 303) ; si on fait « la taille au dessus », on peut obtenir un compteur de 19 (18+1) pour un TS5 (4+1). Pour ce groupe,

36. Cet affichage est ainsi le même pour tous les élèves, mais il n'a pas été anticipé

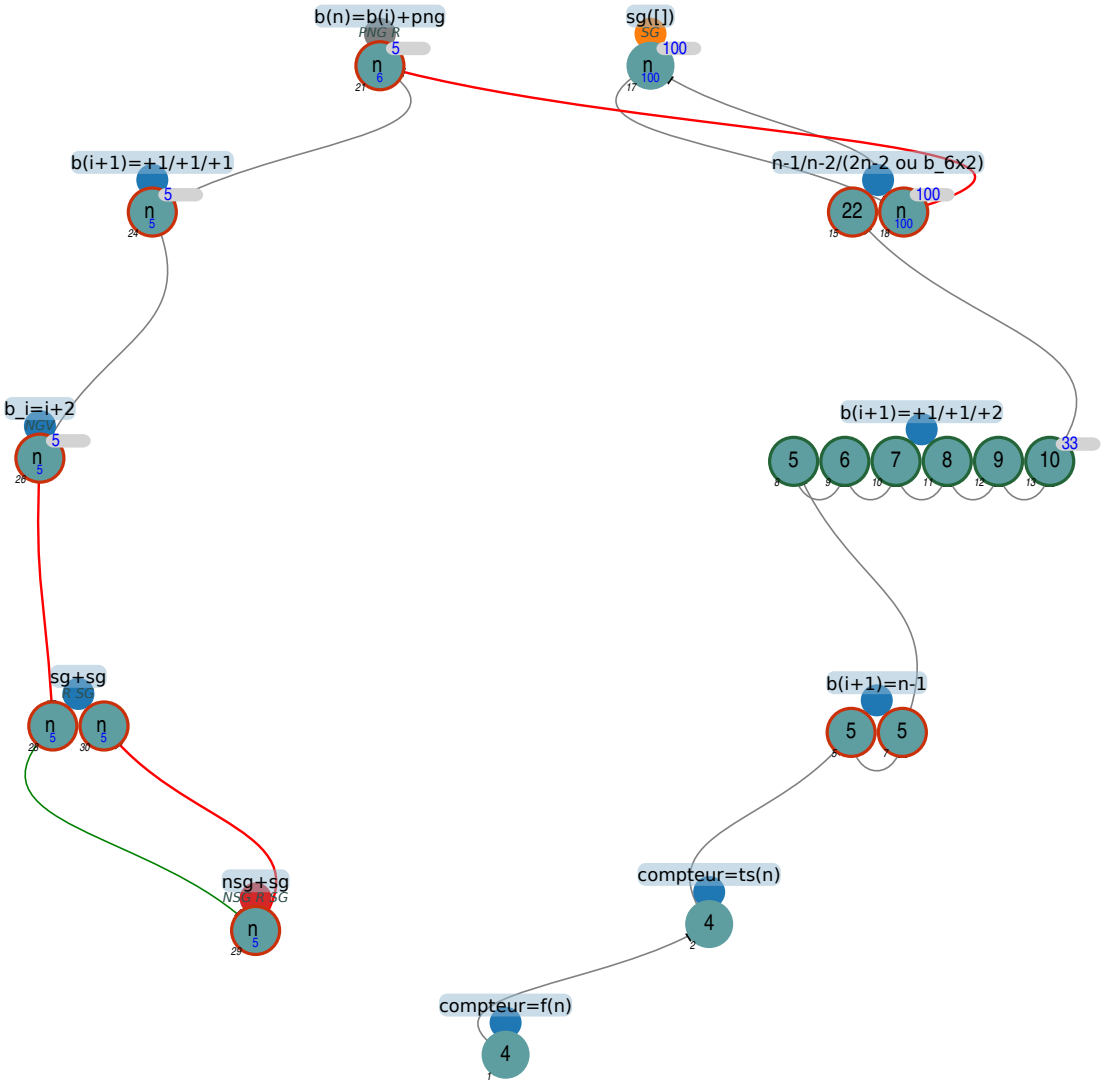


Figure 3.48 – Organisation de l'activité - 46g

à ce moment, le compteur, qui représente le nombre total d'hexagones d'un TS, est l'élément caractéristique de la figure, et il suffit de le changer pour que la figure change. Le groupe 45f a lui aussi momentanément considéré la possibilité d'agir sur le compteur (voir page 275), mais en lui affectant la valeur de la mesure voulue (5), ce que ce groupe a évoqué (« j'mets cinq ? » 46g, transcription S3, 4'37) mais immédiatement écarté. Elles obtiennent ainsi :

$$TS(5)[] \rightsquigarrow \left(\begin{array}{c} \\ \\ \end{array} \right. \begin{array}{c} \\ \\ \end{array} \left. \begin{array}{c} \\ \\ \end{array} \right) , 43, 5$$

Cette rétroaction est considérée comme erronée : E2 précise que « mais non c'est pas ça faut que

88	03 :50.930	TProf	maintenant il va falloir changer des choses à l'intérieur
89	03 :50.930	GProf	balaye TS5 avec la main de haut en bas
90	03 :53.344	TProf	pour que
91	03 :58.269	GProf	ramène un cahier d'élève et l'ouvre à la page des premiers exemples de TS
92	04 :00.707	TProf	pour que maintenent il dessine celui-ci
93	04 :00.707	GProf	montre TS5 sur le cahier
94	04 :02.558	TProf	donc un deux trois quatre cinq
95	04 :02.558	GProf	montre chaque hexagone du côté b1 avec le doigt
96	04 :04.269	T1	ok
97	04 :04.542	TProf	un deux trois quatre cinq
98	04 :04.542	GProf	montre chaque hexagone sur côté b2
99	04 :07.815	TProf	donc ça veut dire | changer les choses à l'intérieur du programme
100	04 :07.815	GProf	montre vaguement plusieurs portions du script TS5
101	04 :11.096	TProf	pour qu'il en fasse un
102	04 :11.096	GProf	dessine b1 et b2 avec la main
103	04 :12.798	TProf	de la taille au dessus

Transcription 3.8 – 46g, S3 : Premier problème

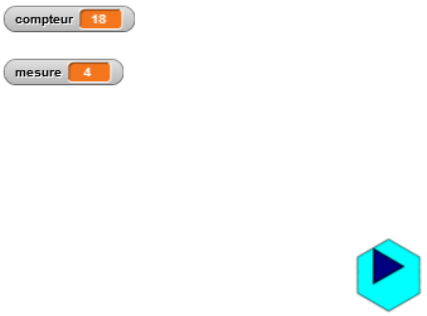


Figure 3.49 – Affichage au chargement du programme de base

tu fasses plus grand[37] » (46g, transcription S3, 5'55), et les élèves vont effectuer une nouvelle modification du compteur : *compteur* ← 33, c'est-à-dire le nombre attendu d'hexagones pour un TS5, fait disponible dans le cahier que les élèves mobilisent à ce moment. E1 souligne d'ailleurs que c'est « parce que là c'est bon parce que c'est de mesure cinq » (46g, transcription S3, 6'13), ce qui met en tension la mesure et le compteur — qui font toutes deux partie des « "aspects" pertinents [du système étudié] par rapport à l'étude que l'on veut faire de ce système, soit l'ensemble des variables par lesquelles on le découpe dans le domaine de réalité où il nous apparaît. » (Chevallard, 1989, p. 53). E1 ne comprend pas « pourquoi c'est faux » (46g, transcription S3, 6'51), et tente de nouveau une modification du compteur à 33, mais cette écriture se rajoute aux signes déjà présents, ce qui produit l'affectation de « 3333 » au compteur. La tension compteur-mesure réapparait alors : E2, en montrant ce qui nous semble être l'affichage de la mesure, croit « que c'est ça qu'il faut changer » (46g, transcription S3, 8'47) tout en s'interrogeant peu après sur ce qui permettrait de modifier la mesure (« le ça |le la mesure elle est où ? »). E1 rétorque que

37. « plus grand » s'applique ici au TS, et non au compteur, puisque celui-ci est de 43 pour une valeur recherchée, connue des élèves, de 33.

« mais non c'est le compteur qu'il faut changer » puisque qu'elle observe « regarde ça monte », en montrant l'affichage du compteur qui est en train d'évoluer au fûr et à mesure du tracé. E2 va remettre le compteur à zéro, mais le conflit n'est pas réglé — ni argumenté. Les élèves vont donc de nouveau avoir recours à l'enseignante (9'31).

Modification des boucles : nécessité (étapes 5-7) E2 semble alors chercher à mettre en relation le tracé, la valeur de la mesure, et le nombre de répétitions de la première boucle (3.9). Après avoir observé le groupe voisin, elle affirme savoir ce qu'il faut changer en montrant les différentes instructions du script de haut en bas (figure 3.50, p. 304) : il ne s'agit plus de changer juste une valeur à un endroit (que ce soit la mesure ou le compteur), mais de modifier l'ensemble de l'algorithme .

(a) de là... **(b)** ... à là.

Figure 3.50 – Ce qu'il faut changer (46g, S3)

Ensuite, elle met en relation la mesure, le tracé, et le nombre de répétitions de la boucle b_1. En croisant ce que dit l'élève et ce qu'elle montre (3.9 et figure 3.51, p. 306), on peut reconstituer le raisonnement suivant :

- Si la mesure est de cinq il faudrait un certain nombre d'hexagones sur le premier côté,
- or le premier côté tracé pour une mesure de cinq est trop court ;
- en séance 2 on a vu que les boucles traçaient les hexagones des côtés un à un,
- donc la première boucle trace le premier côté ;
- pour un TS4 la boucle b_1 est répétée 3 fois, et sur l'écran 3 hexagones sont tracés pour le premier côté,
- pour un TS5 le côté est « la taille au dessus »,
- donc la première boucle trace le premier côté qui est trop court de (au moins) un, et la première boucle est « répéter 3 fois », il faut donc la répéter 4 fois.

La rétroaction produite est ainsi :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 25, 5 \right)$$

318	10 :20.515	T2	je sais s'qui faut changer
319	10 :23.089	G2	balaie de haut en bas le script 2
320	10 :25.962	T1	c'est pas ça qu'faut changer hein
321	10 :27.026	T2	mais si
322	10 :28.146	T1	bah tu veux mettre quoi alors
323	10 :28.983	T2	XXX
324	10 :30.047	T2	demande à la prof
325	10 :33.238	T2	hein ?
326	10 :36.302	T2	attends parce que là tu [trouves]
327	10 :38.870	T2	si ça donc
328	10 :38.870	G2	montre affichage variable mesure puis tracé
329	10 :40.657	T2	là faut mettre
330	10 :40.657	G2	glisse le doigt du tracé au script
331	10 :43.331	T2	euhm
332	10 :47.118	T2	quatre
333	10 :47.118	G2	claque des doigts
334	10 :48.011	T2	mets quatre tu verras
335	10 :49.799	T1	j'te crois pas
336	10 :50.082	T2	le premier il est quatre
337	10 :51.295	T2	non pas là pas là le [jaune]| fin euh ça c'est zéro
338	10 :54.118	T2	mets zéro
339	10 :56.586	T2	mets quatre au deuxième
340	10 :57.870	T1	là ?
341	10 :58.366	T2	hmm hmm

Transcription 3.9 – 46g, S3 : Il faut modifier les boucles

Cette rétroaction n'est pas interprétée par E2 qui déplore « j'comprends pas »(46g, transcription S3, 11'21) et lève la main pour, une fois encore, faire appel à l'enseignante. Les échanges semblent montrer que E2 ne considère pas la représentation d'un chevauchement comme un signe d'erreur. En effet, elle affirme à l'enseignante avoir « essayé plein de trucs », mais que « ça nous fait exactement pareil à chaque fois » (46g, transcription S3, 14'59-15'02). Or, si la forme tracée a bien été « exactement pareille » à chaque tentative, la dernière tentative fait apparaitre un hexagone foncé : cet aspect n'est donc pas pris en compte par l'élève. On retrouvera d'ailleurs plus tard ce souci, que viendra éclairer le groupe voisin en séance 5.
L'enseignante explicite alors le lien entre le nombre de répétitions de la boucle b_1 et le premier côté tracé : « au départ vous avez vu il a tracé ces trois là » suivi de « répéter 3 » en montrant alternativement la première boucle du script et le côté tracé. E1 comprend alors qu'il faut ajouter un, mais comme « nous on a mis répéter quatre fois, là il faut mettre répéter cinq fois » (46g, transcription S3, 15'29), raisonnement qui perturbe, à raison, l'enseignante. On peut penser que ce raisonnement provient aussi de la rétroaction affichée : l'enseignante a mis le programme en pause en fin de tracé de la boucle b_3 (figure 3.52, p. 306), ce qui peut laisser penser que quatre hexagones du premier côté ont été tracés — ce qui est mis en relation avec $b_1 = 4$ —, et donc qu'il en manque un, ce qui entraine qu'il faut faire $b_1 \leftarrow 5$. Cela implique aussi que E1 considère le tracé comme étant issu de leur script, or ce tracé est issu du script « a » (raccourci sémiotique évoqué pour le groupe 45f, et que nous utiliserons aussi), qui correspond au TS4 original.
L'enseignante explicite alors le décalage de la boucle b_3, mais sans revenir sur le sens porté par le codage « hexagone bleu foncé ». Elle précise ensuite que les élèves ont ajouté un hexagone en b_1 et pose la question « et après ? » en montrant alternativement les boucles de chacun des deux scripts visibles, ce qui incite E1 à conclure « qu'il faut en rajouter partout » (3.10).
En séance 4, le groupe 46g va commencer par charger leur travail précédent, puis modifier le script devant tracer un TS5. Ce script est désigné par les élèves non par son objectif, sa tâche prescrite, mais par sa position : « on était au deuxième à celui-là » (46g, transcription S4, 1'35). E1 va alors fixer le nombre de répétitions de toutes les boucles à quatre, mais sans être capable d'en expliquer la raison (« bah chais pas », 46g, transcription S4, 1'45), ce qui aboutira à la rétroaction

(a) si la mesure est 5...

(b) ... et qu'ici il manque un hexagone...

(c) alors là il faut mettre 4

Figure 3.51 – Raisonnement pour changer b_1 (46g, S3)

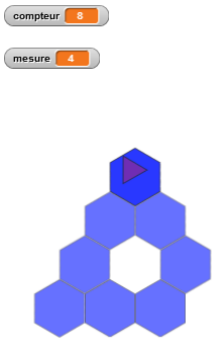


Figure 3.52 – Un TS4 en cours. Combien d'hexagones a le premier côté ? (46g, S3)

suivante, qu'E1 n'arrive pas à interpréter (« c'est quoi cte truc » 46g, transcription S4, 2'10) :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{, } 34, 4 \right)$$

Construction du TS5 : une solution partagée (étape 8) Leur voisine va alors partager

464	15 :46.253	TProf	vous avez bien tracé votre ligne comme ça
465	15 :46.253	GProf	suit tracé b1
466	15 :48.281	TProf	après il a fait ça
467	15 :48.281	GProf	suit tracé b2
468	15 :50.387	TProf	et après il est revenu sur là
469	15 :50.387	GProf	suit b3 jusqu'au départ de b1.
470	15 :52.302	TProf	alors que normalement il aurait du revenir un peu plus haut
471	15 :52.302	GProf	montre trajet virtuel attendu de b3
472	15 :54.976	T1	oui
473	15 :55.622	TProf	donc
474	15 :56.416	TProf	vous en avez rajouté un là
475	15 :56.416	GProf	montre alternativement b1 sur script 1 et 2
476	15 :59.246	TProf	mais après
477	15 :59.246	GProf	montre alternativement b2 script 1 et 2 puis b3 puis b4 en zigzag
478	16 :01.153	T1	faut en rajouter partout
479	16 :02.231	TProf	bah oui
480	16 :02.763	T2	d'accord
481	16 :03.224	TProf	ça vous terminerez ça lundi
482	16 :04.877	T1	ok
483	16 :05.934	TProf	vous sauvegardez

Transcription 3.10 – 46g, S3 : Il faut en rajouter partout

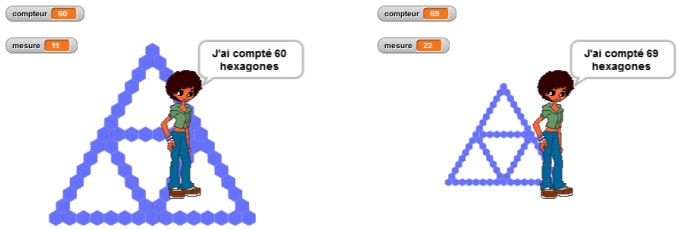
103	02 :40.020	TA	non en gros | à chaque fois | vous changez
104	02 :42.530	TA	là le trois vous mettez le quatre
105	02 :42.530	GA	montre b1 de TS4 puis b1 de TS5
106	02 :43.540	TA	là le deux vous mettez trois
107	02 :43.540	GA	montre b2 de TS4 puis b2 de TS5
108	02 :44.760	TA	là vous mettez trois
109	02 :44.760	GA	idem sur b3 b4
110	02 :45.760	TA	en fait vous en mettez un de plus
111	02 :45.760	GA	idem b5 b6
112	02 :47.100	TA	et baisse
113	02 :47.100	GA	montre script vers le bas
114	02 :47.750	T1	ah ok j'ai compris
115	02 :48.940	TA	attends baisse
116	02 :50.680	TA	et aussi si vous mettez
117	02 :50.680	GA	montre b7
118	02 :51.750	T1	sept
119	02 :51.750	TA	en fait
120	02 :52.570	TA	nan vous doublez de deux
121	02 :52.570	GA	rotation avec la main et deux doigts
122	02 :53.650	TA	ya juste là où vous doublez de deux sinon ça bug
123	02 :53.650	GA	montre b7
124	02 :55.570	T1	ok

Transcription 3.11 – 46g, S4 : Solution partagée

avec le groupe sa solution (3.11) : il faut ajouter un à chaque boucle sauf b_7 pour laquelle « vous doublez de deux sinon ça bug » (3.11, 120). Cette formulation est intrigante : il faut ajouter deux à la boucle b_7 et non la doubler ; mais d'un autre côté on peut aussi trouver b_7 en doublant la valeur de b_6 du script en cours.
Le groupe 46g va alors appliquer les instructions données par leur voisine, et comparer avec l'exemple de TS5 trouvé dans leur cahier afin de valider ce tracé.

Construction de multiples instances consécutives (étapes 9-12) Pour construire le script traçant le TS6, les élèves vont utiliser la relation inter-objectale (Piaget et García, 1983) évoquée par leur voisine. Elles utilisent les valeurs du $TS(5)$ pour construire le $TS(6)$: « et non là c'est trois donc là faut mettre quatre » (46g, transcription S4, 5'15). Pour b_7 il y a une hésitation. E1

commence par suggérer la valeur 12, ce qui peut correspondre soit au doublement de la valeur initiale de b_7 pour le TS4 ($b_7^6 = 2 \times b_7^4$), soit à un doublement de l'écart ($6 \xrightarrow{+2} 8 \xrightarrow{+4(=[+2]\times 2)} 12$). En comparant les valeurs de b_7 sur le $TS(4)$ et $TS(5)$, E1 va finalement se décider pour $b_7^6 = b_7^5 + 2$, en restant sur une relation inter-objectale.
Ce binôme, comme le groupe 46d, va ensuite enchainer sans soucis la construction des scripts pour les TS6, TS7, TS8, et TS9 en appliquant le TEA $+1/+1/+2$. De même, tout comme le groupe 46d, la dimension du TS tracé n'est pas invalidante (figure 3.53, p. 308).



(a) TS8 pour un zoom de TS11 **(b)** TS9 pour un zoom de TS22

Figure 3.53 – Des facteurs de zoom non invalidants (46g, S4)

Une demande perturbante (étapes 13-14) Le groupe 46g va alors construire, toujours avec la même méthode, l'instance traçant le TS10. Mais ce script est construit à partir de l'entête correspondant au script générique : ainsi, au lancement afin de tester leur script, les élèves sont-ils perturbés par la demande faite d'entrer une valeur pour la mesure du TS voulu. E1 relit la demande « quelle est la mesure du triangle » (3.12, 524) et fait un geste semblant représenter un retour un arrière, accompagné par l'arrêt manuel du script. Elles s'intéressent alors au bloc d'initialisation (qu'elles nommeront « initiation ») : c'est effectivement ce bloc qui déclenche, de façon transparente, l'invite. Après réflexion, E1 proposera la valeur 22. Il est possible que cette élève ait établi une relation entre l'écriture chiffrée suivant le mot « initialisation » et les mesures (figure 3.54, p. 308), et l'affichage de la mesure lors de l'invite — qui conserve la dernière valeur effective de la mesure —(figure 3.55, p. 309). D'autres groupes utiliseront ces blocs d'initialisation comme des indices sémiotiques de ce que fait le script. Ne joignant pas le geste à la parole, E1 va entrer la valeur 33, peut-être en faisant référence au nombre d'hexagones d'un TS5. Dans ce cas, on retrouve une nouvelle tension entre les variables caractéristiques d'un TS : sa mesure et son nombre d'hexagones.

Figure 3.54 – Blocs d'initialisation

522	12 :24.674	EXECUTION	START receiveKey(478)
523	12 :24.717	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
524	12 :25.810	T1	quelle est la mesure du triangle
525	12 :29.960	E1	geste de retour en arrière ?
526	12 :32.050	T2	mais attend je monte
527	12 :36.825	EXECUTION	STOP receiveKey(478)
528	12 :37.450	T1	ah euh
529	12 :39.190	E1	montre bloc initialisation
530	12 :39.190	T1	ah oui initiation
531	12 :42.594	EXECUTION	START receiveKey(478)
532	12 :42.646	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
533	12 :43.240	T1	initiation
534	12 :44.480	E1	montre prompt
535	12 :44.480	T1	tu mets euhm
536	12 :46.880	T1	euh
537	12 :47.430	T2	initiation
538	12 :49.290	T1	attends je réflechis
539	12 :50.910	T1	attends tends tends
540	12 :52.010	T1	vingt deux

Transcription 3.12 – 46g, S4 : Que répondre à l'invite ?

Figure 3.55 – Invite au lancement du script générique (46g, S4)

Un but différent Après avoir validé la rétroaction

$TS(10)[33] \rightsquigarrow \Big($  $, 78, 33 \Big)$

— donc toujours sans prendre en considération le niveau de zoom, le lien entre la valeur de la mesure affichée (33) et la mesure du TS tracé (10) —, les élèves estiment que leur but est atteint : « Madame on a réussi nous, on a fini avant toi autreE » puis « on a tout fini nous madame » et encore « on a tout fini » — cette dernière phrase reprise avec fierté à destination du micro (46g, transcription S4, 14'04-14'54). Ainsi, pour les élèves, le but était de compléter l'ensemble des entêtes de script disponibles, et comme c'est le cas, E2 demandera alors « du coup on fait d'autres euh programmes ? ». Rappelons que le problème donné aux élèves sur ces trois séances était :

1. construire quelques instances de scripts consécutives,
2. puis construire des instances en faisant un saut (mesure onze, mesure 22)
3. puis généraliser pour toute mesure.

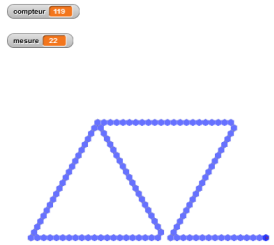
Ici, malgré les indications concernant la tâche prescrite qui accompagnent les entêtes des scripts, les élèves sont restés (comme d'autres groupes), sur la construction, effective et efficace, d'instances successives.

Une relation mesure-tracé E1, malgré tout, ajoute « sauf qu'on a tout faux parce que ça fait pas trente-trois. » La valeur entrée pour la mesure lors de son dernier test ne peut être mise en relation qu'avec un élément de la rétroaction : la mesure affichée de 33. Cependant, le nombre d'hexagones est de soixante-dix-huit, et le tracé correspond à un TS10. E1 montre ainsi que son but, pour ce script, est lié à la valeur entrée. Il est cependant difficile de savoir si ce but était d'obtenir 33 hexagones, ou bien d'obtenir un TS33 — nous penchons plutôt pour la deuxième possibilité. L'enseignante pointera « un problème » (46g, transcription S4, 16'20) : « alors j'ai lancé le même de f, ah oh ya un problème là, pourtant les premiers programmes ils étaient bien ». L'enseignante invalide ainsi le tracé (figure 3.53b, p. 308) mais sans préciser la raison, qui est soit la dimension du triangle, soit — de façon plus rigoureuse — le nombre d'hexagones sur un côté des triangles de base. Après le départ de l'enseignante, E1 mobilisera ce nouveau fait invalidant, en identifiant la raison, en relançant l'exécution du script en cause : « ça en fait que neuf » (46g, transcription S4, 17'05). Elle établit une relation entre la mesure et le tracé.

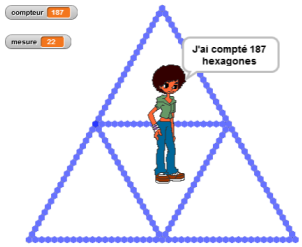
Une collaboration avec le groupe voisin (étapes 15-16) Pendant que E2 discute d'autre chose, E1 cherche, mais se plaint — en s'adressant à elle-même — de ne rien comprendre. La solution viendra une nouvelle fois de leur voisine, qui leur donnera les premières valeurs pour le script qu'elle nomme « f » et qui correspond au TS22. Cependant, pour la boucle b_7, c'est E2 qui donnera la bonne valeur, qui se trouve être aussi la réponse à la grande question sur la vie, l'univers et le reste : 42 [38]. La précédente instance (TS21) n'est pas construite, E2 fait donc soit un lien avec une instance lointaine, soit un lien avec la valeur de la mesure (ou l'instance en cours). Si l'élève se basait sur l'instance du TS11 ($b_7^{11} = 20$), on aurait pu s'attendre a une valeur de 40, puisque comme le rappelle E1 « mais non on doit faire euh fois deux, on l'a pas fait ça » (46g, transcription S4, 20'06). Il est possible que pour E2 $b_7^{2p} = 2b_7^p + 2$, mais rien ne vient confirmer cela : la seule instance dont la mesure est le double d'une autre étant le TS8, construit à partir du TS7, et la réponse rapide de l'élève laisse penser qu'elle ne s'est pas référée à ces scripts. E2 parait donc capable, pour une instance donnée, de déterminer la valeur du nombre de répétitions de b_7. Comme ce groupe n'a jamais été confronté à la nécessité de trouver une relation entre b_7 et la valeur de la mesure, il est probable que la relation — qui a été identifiée de façon contingente — soit intra-objectale : la valeur de b_7^p s'avère être le double de la valeur de b_6^p (pour la même instance). Les doubles des nombres de 1 à 9 sont supposés être des faits numériques connus à ce niveau.

38. Adams, D. (1979). The Hitch Hiker's guide to the Galaxy. Pan Books. Il se trouve que 42 est aussi le caractère ASCII correspondant à l'astérisque « * », qui est souvent utilisé en informatique comme marque-place, auquel peut se substituer n'importe quelle chaine de caractères !

L'exécution produira néanmoins un tracé invalide :

$$TS(22)[] \rightsquigarrow \left(\text{  } \text{} \quad , [], 22 \right)$$

Avec l'aide de leur voisine, en comparant les scripts — et non plus en analysant le tracé — le groupe 46g va corriger la boucle b_4, qui était effectivement erronée (21 au lieu de 20). Le résultat sera validé, malgré la présence d'un hexagone foncé (erreur en b_8).

$$TS(22)[] \rightsquigarrow \left(\text{  } \text{} \quad , 187, 22 \right)$$

Il est notable que la même erreur est présente sur le script de la voisine de ce groupe, et le chercheur est intervenu pour expliciter le sens de cette couleur plus foncée. Le groupe 46g n'a pas semblé prendre en considération ces échanges, mais en séance 5 leur voisine remobilisera cette nouvelle connaissance pour expliquer au binôme 46g en quoi leur tracé est erroné.

Vers la généralisation : des difficultés pour poser le problème (étapes 17-30)
L'enseignante intervient alors, et, s'adressant entre autres au groupe 46g et à leur voisine, précise ce qui est attendu du script générique : « c'est un programme qui [fait] n'importe lequel [TS] » (3.13, 785). Elle prend ensuite un exemple avec une entrée de 100 (proposée par E2), qui devrait « me faire celui de cent » mais « sauf que là il va pas te le faire » (46g, transcription S4, 793-796). Elle précise que, dans ce cas, c'est le TS22 qui a été construit, malgré le fait que « c'est trop petit là on voit rien » (3.13, 804 et figure 3.56, p. 311).

Figure 3.56 – Tracé avec une mesure demandée de 100 (46f, 20'05)

780	22 :44.640	TProf	là ce que vous allez faire maintenant
781	22 :46.540	TProf	puisque là vous avez fini de tracer
782	22 :48.530	TProf	tous les triangles que je demandais
783	22 :50.680	TProf	moi ce que je voudrais
784	22 :52.050	TProf	c'est un programme
785	22 :53.740	TProf	qui fasse | n'importe lequel
786	22 :58.530	TProf	ça veut dire que j'aimerai
787	23 :02.770	TProf	j'aimerai
788	23 :04.160	TProf	voilà | modifier ça
789	23 :06.430	TProf	mais quand je vais lancer
790	23 :08.620	TProf	qu'est-ce que je vais mettre je sais pas moi euh
791	23 :10.400	T2	cent
792	23 :10.870	TProf	cent
793	23 :11.710	TProf	il va me faire celui de cent
794	23 :14.070	TA	XXX
795	23 :16.540	TProf	sauf que là
796	23 :18.620	TProf	il va pas te le faire
797	23 :20.050	TA	ah c'est minuscule on dirait XXX bah si il le fait
798	23 :21.930	TProf	bah non
799	23 :22.800	TProf	il va faire
800	23 :23.590	TProf	j'ai fait un copié | j'ai pris celui là XXX
801	23 :26.400	TA	XXX
802	23 :27.720	TProf	bah oui (rire)
803	23 :29.210	TProf	comment tu vas faire
804	23 :44.430	TA	mais il est trop petit là on voit rien
805	23 :46.060	TProf	comme t'as mis une mesure de cent
806	23 :48.140	Prof	geste de réduction avec les mains
807	23 :48.140	TProf	les hexagones se sont réduits
808	23 :50.000	TProf	par contre comme j'ai juste repris le programme d'avant
809	23 :52.800	TProf	si on devait les compter yen a vingt deux vingt deux
810	23 :55.160	TProf	vingt deux comme ça vingt deux vingt deux
811	23 :57.000	TProf	XXX
812	23 :57.600	T2	madame nous on a fini
813	23 :59.970	T1	mais | on n'a pas du tout fini hein
814	24 :01.630	T2	oui mais fin là euh
815	24 :03.570	T1	là faut qu'on change
816	24 :04.800	TProf	la
817	24 :06.470	T2	pourquoi tu changes ?
818	24 :07.413	VALEURS	(BOUCLE)b1i0(VAL) : répéter ** fois (commandes)<<9>>
819	24 :07.820	T1	passque | si il faut qu'on mette euh
820	24 :09.140	T1	attend
821	24 :10.070	T1	XXX [met] qu'on change
822	24 :10.902	VALEURS	(BOUCLE)b2i0(VAL) : répéter ** fois (commandes)<<8>>
823	24 :11.510	T1	c'est trop compliqué pour toi
824	24 :12.492	EXECUTION	FIN receiveKey(472)
825	24 :13.136	VALEURS	(BOUCLE)b3i0(VAL) : répéter ** fois (commandes)<<8>>
826	24 :14.507	VALEURS	(BOUCLE)b4i0(VAL) : répéter ** fois (commandes)<<8>>
827	24 :16.971	VALEURS	(BOUCLE)b5i0(VAL) : répéter ** fois (commandes)<<8>>
828	24 :18.196	VALEURS	(BOUCLE)b6i0(VAL) : répéter ** fois (commandes)<<9>>

Transcription 3.13 – 46g, S4 : Consignes pour la généralisation

Malgré ces indications, E2 pense de nouveau avoir atteint les objectifs, répétant une nouvelle fois « Madame nous on a fini » (3.13, 812), alors que E1 lui oppose un tranchant « mais on n'a pas du tout fini hein [...] là il faut qu'on change [...] c'est trop compliqué pour toi » ! Elle affecte alors une place vide pour représenter le nombre de répétitions de chacune des boucles : 

, puis teste ce programme avec une valeur entrée de 100.

$$TS(n)[100] \rightsquigarrow \left(\begin{array}{c} \text{compteur } 2 \\ \text{mesure } 100 \\ \text{J'ai compté 2 hexagones} \end{array} , 2, 100 \right)$$

Prenant quelques secondes pour réfléchir, elle va ensuite indiquer à E2 toutes les (bonnes) valeurs à entrer pour construire un TS100 : {99/98/198}. Elle n'a pas d'hésitation pour les boucles b_1 à b_6, on peut donc penser que cette élève a établi les relations $b_1^n = n - 1$ et $b_2^n = n - 2$. L'établissement de la valeur pour b_7^{100} (198) prend trente secondes (3.14). E2 rappelle que « c'est le double » (ce qui semble confirmer la relation intra-objectale évoquée ci-dessus). E1 semble mobiliser les faits suivants :

1. $b_6^{11} = 10$ et $b_7^{11} = 20$: « attends onze, tu mets, c'est cent fois le double » laisse penser que E1 s'intéresse à l'instance 11 pour comprendre ce qu'il faut faire pour trouver « le double » pour l'instance 100 ;
2. $b_6^{22} = 21$ et $b_7^{22} = 42$: « de vingt deux c'est... », laisse penser la mobilisation d'un autre fait comparable au premier, mais sur le TS22
3. $b_6^{100} = 99$

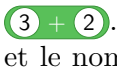
Elle trouve alors la bonne valeur : $2 \times 99 = 198$. Ceci semble confirmer la relation intra-objectale évoquée : $b_7^p = 2b_6^p$. On ne peut cependant pas écarter la possibilité d'un autre calcul : $100 \times 2 - 2$. C'est d'ailleurs sans doute cette technique de calcul rapide qu'a mobilisée l'élève. Mais a-t-elle pour autant établi la relation $b_7^n = 2 \times n - 2$?

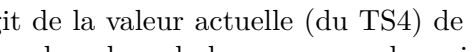
876	25 :29.710	T1	hmm
877	25 :30.660	T1	je réfléchis
878	25 :32.470	T2	il est double
879	25 :33.860	T2	bah
880	25 :37.650	T1	XXX
881	25 :39.720	T1	attends onze
882	25 :40.720	T1	tu mets
883	25 :43.210	T1	c'est cent fois le double
884	25 :45.730	T1	de vingt deux c'est
885	25 :49.680	T2	XXX
886	25 :50.940	T1	uh j'ai déjà une XXX
887	25 :52.630	T2	oui le double de vingt deux c'est quarante quatre hein
888	25 :54.460	T1	oui c'est ça qui rend XXX
889	25 :57.550	T1	ah mais si justement
890	25 :59.990	T1	c'est attends
891	26 :01.190	T1	c'est euh
892	26 :07.500	T1	ok vas-y descends
893	26 :08.444	VALEURS	(BOUCLE)b7i0(VAL) : répéter *198* fois (commandes)<<>>

Transcription 3.14 – 46g, S4 : Comment trouver b_7

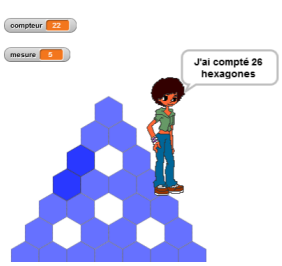
Le test, lancé avec une valeur entrée de 100, montre que cette élève a bien fait le lien entre la valeur entrée, la mesure du TS tracé, et la valeur du nombre de répétitions des boucles. Les élèves terminent la séance en évoquant la nécessité d'un tracé plus rapide (avec cette version du script, cinq minutes et trente secondes environ pour arriver au début du tracé de la boucle b_7). En séance 5, après que l'enseignante a fait un rappel en plénière du problème central de la séance (construire le script générique), le groupe 46g se demande « faut faire quoi du coup ? » (46g, transcription S5, 36"). Leur voisine (autreE) reformule en disant qu'il faut « trouver un moyen pour que ça le fasse tout seul » et complète en rappelant la suggestion de l'enseignante « tu vas sur [calcul] là en vert », ce que E1 identifie bien comme étant la catégories des opérateurs (46g, transcription S5, 46"-56"). Malgré cela, le groupe lève de nouveau la main pour appeler l'enseignante. Après avoir déposé sur l'espace de programmation, puis supprimé, l'opérateur 

et attendu deux minutes, E2 lance le script du TS4, et affirme « oh regarde j'ai réussi » (46g, transcription S5, 3'27), semblant de nouveau confondre tracé valide et problème résolu. E1 confirme à leur voisine que pour elles, « ça le fait tout seul » (46g, transcription S5, 4'03), ce que l'enseignante avait formulé « c'est l'ordinateur qui vous fait tout » (46g-Prof, S5, 16'40) ou encore — faisant référence à ce qui est attendu du script générique — « ça c'est pas à vous de le rechanger, l'ordinateur il est capable de le faire tout seul, de trouver de lui-même | ce qu'il va falloir mettre dans les boucles » (46g Prof, S5, 14'27) [39]. AutreE précisera qu'il faut copier le script du TS4 sous l'entête du script générique, et que lorsqu'on entre un nombre, par exemple 5, « là ça te fait celui de quatre, ben avec ça faut que tu trouves un moyen pour que quand tu tapes cinq et ben ça te fasse un triangle de Sierpinski cinq » (46g, transcription S5, 4'11-4'38). Chez autreE, le problème est bien posé. Le groupe 46g va alors, sans doute par effet de contrat, utiliser l'opérateur 

. Sur l'espace de programmation, ce binôme crée le bloc 

. Le nombre 3 est sans doute utilisé car il s'agit de la valeur actuelle (du TS4) de b_1, et le nombre 2 est choisi tel que, ajouté à trois, il donne la valeur de la mesure voulue, cinq. Les élèves de ce groupe ne semblent pas convaincues : E2 demande à autreE « mais autreE on doit le mettre où le vert ? » (46g, transcription S5, 5'20) et, — après trois minutes de discussion hors contexte en attendant, encore une fois, l'enseignante — E1 effectue l'action $b_1 \leftarrow$ 

et teste le programme en précisant « on va essayer hein » (46g, transcription S5, 9'04).

$$TS(n)[5] \rightsquigarrow \left(\text{ 

}, 26, 5 \right)$$

La rétroaction est validée par E2 : « bah on a réussi autreE hein » (46g, transcription S5, 9'13). Ceci sera contredit, avec raison, par leur voisine, qui leur précisera que non seulement c'est un TS4 et non un TS5, mais aussi que « yen a deux qui sont en foncé, ça veut dire que vous avez surposé deux fois », ce qu'elle reformulera « ces deux là ils sont en foncé [...] ils y sont deux fois » (46g, transcription S5, 9'16-9'30).
L'enseignante arrive alors et reprécise, avec diverses exécutions, ce qui est attendu (3.15) :

39. Les temps sur les transcriptions des élèves et de l'enseignante n'ont pas la même référence : sur les transcriptions des faits partagés par l'enseignante en début de séance, il s'agit de l'écart avec le début de la prise de parole de l'enseignante ; sur les transcriptions élèves il s'agit de l'écart avec le lancement de l'EPGB.

— on demande à l'utilisateur le nombre d'hexagones qu' « il veut sur les petits triangles », c'est-à-dire le nombre d'hexagones sur chaque côté des triangles de base [40],

— si on ne change pas les valeurs des boucles, le programme trace un TS5 même si on veut une mesure de cinq,

— « donc il n'a pas modifié son programme »,

— certes il faut ajouter un pour passer au TS5, mais si on veut un TS6 ou un TS7 « il doit modifier au fur et à mesure »,

— ce n'est pas vraiment qu'« il faut faire tous les programmes », puisqu'on doit « toujours rester sur un »

E1 en déduit qu'il faut changer toutes les valeurs du nombre de répétitions des boucles (3.15, 318), et E2 propose d'affecter 4 à b_1, 5 à b_2, 6 à b_3 etc, soit $b_i = i + 3$ (3.15, 319-324). E1 s'y oppose et construit le script $TS(5)$ à partir du $TS(4)$, en ajoutant un à chaque boucle (étape 24) : le TEA valide permettant de passer une instance à la suivante, que ce groupe avait mis en œuvre en séance 4, n'est plus ici mobilisé. E2 est d'autre part en désaccord avec cette méthode de construction d'une instance, « parce qu'il faut rester sur le même programme ». Ainsi, E1 reste sur le problème de créer l'instance qui correspond à la mesure de cinq entrée, alors que E2 est sur le problème de création du script générique. Après correction de la boucle b_7, E1, en tapant des mains, affirme « on a réussi » (46g, transcription S5, 13'36) : elle est bien sur le problème de construction d'une instance, et pour elle il suffit ensuite de modifier de nouveau toutes les boucles (« et ben après tu changes tout » (46g, transcription S5, 13'41)).
E2 va alors de nouveau proposer d'affecter à chaque boucle un nombre de répétitions incrémenté de un par rapport à la boucle précédente : $b_i = i + 2$ (script 3.5, p. 318). Ceci conduit à une rétroaction visiblement erronée :

$$TS(n)[5] \rightsquigarrow \left(\text{  } \text{} , 54, 5 \right)$$

Cela invalidera la tentative, et E2 précisera « du coup il faut retrouver les mesures hein » (46g, transcription S5, 14'50), laissant entendre que la précédente tentative ne mettait pas en œuvre les liens entre mesure et nombre de répétitions des boucles. Elle commencera alors à réaffecter les valeurs pour un TS5, ce à quoi E1 s'oppose en rappelant « mais la prof elle a dit qu'c'était avec ça » (46g, transcription S5, 15'24). Elle semble montrer la catégorie des opérateurs, et cela est confirmé par les actions suivantes : mettre « à chaque fois » (◯ + ◯). Il s'agit de trouver un bloc qui résout le problème de la généricité, mais sans savoir qu'en faire : « pff là je sais plus quoi foire là, donc on met quoi E2 ? » (46g, transcription S5, 16'05). La solution est évidente : appeler l'enseignante ! Après avoir effectué $b_1 \leftarrow$ (1 + ◯) et l'avoir de suite annulé, ce groupe va passer le reste de la séance (plus de dix minutes), à échanger sur des sujets hors contexte, y compris lorsque l'enseignante fait une intervention auprès de l'ensemble de la classe concernant la variable mesure.

40. E1, à la demande de l'enseignante de savoir « à quoi correspond » une valeur entrée de 4, reliait cette mesure à l'autre propriété caractéristique des TS, le nombre total d'hexagones : un TS4 a 24 hexagones (3.15, 241-245).

234	10 :52.647	TProf	alors
235	10 :53.568	TProf	quand je mets quatre
236	10 :53.568	GProf	montre l'écran
237	10 :55.117	TProf	je relance
238	10 :56.111	EXEC	KEY_n
239	10 :56.138	EXECUTION	START receiveKey(478)
240	10 :56.180	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
241	10 :56.372	TProf	quand je mets quatre
242	10 :57.387	ENTRÉE	ANSW <<4>>
243	10 :57.745	TProf	le quatre correspond à quoi
244	10 :58.234	EXECUTION	FIN receiveKey(478)
245	10 :59.607	T1	bah à vingt quatre hexagones
246	11 :01.196	TProf	le quatre correspond aux hexagones sur les petits triangles
247	11 :01.196	GProf	montre un à un les hexagones d'un côté (b1) triangle de base
248	11 :04.372	T1	hmm
249	11 :04.588	TProf	quatre comme ça
250	11 :05.902	TProf	quatre comme ça
251	11 :05.902	GProf	montre un à un les hexagones d'un côté du triangle de base (b2)
252	11 :06.941	T1	hmm hmm
253	11 :07.333	TProf	quatre comme ça
254	11 :07.333	GProf	montre un à un les hexagones du 3ème côté du triangle de base (b3)
255	11 :09.176	TProf	quand je relance et que je mets cinq
256	11 :09.877	EXEC	KEY_n
257	11 :09.898	EXECUTION	START receiveKey(478)
258	11 :09.947	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
259	11 :11.630	ENTRÉE	ANSW <<5>>
260	11 :12.467	EXECUTION	FIN receiveKey(478)
261	11 :13.078	T1	bah y met quatre
262	11 :13.882	TProf	ah oui yen a que quatre
263	11 :15.411	T2	XXX
264	11 :16.078	TProf	donc il a pas modifi son programme
265	11 :19.549	TProf	ça veut dire que là pour l'instant le programme il reste toujours sur quatre
266	11 :22.876	ECRAN	APP_button
267	11 :24.823	TProf	les boucles là ça correspond toujours aux nombres pour quatre
268	11 :28.215	T1	donc faut faire euh XXX
269	11 :29.960	T2	ben faut ajouter un
270	11 :31.019	T1	oui XXX
271	11 :31.392	TProf	faut ajouter un
272	11 :32.274	T1	donc euh au tout répéter
273	11 :32.274	G1	montre le script de haut en bas
274	11 :33.745	TProf	oui | mais sauf que
275	11 :35.235	TProf	c'est que lorsque je vais le lancer | une première fois
276	11 :37.175	ECRAN	FULL_button
277	11 :40.222	EXEC	KEY_n
278	11 :40.252	EXECUTION	START receiveKey(478)
279	11 :40.289	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
280	11 :41.411	TProf	voilà il le fait
281	11 :41.605	ENTRÉE	ANSW <<4>>
282	11 :42.429	EXECUTION	FIN receiveKey(478)
283	11 :42.980	TProf	mais je le relance aussitôt
284	11 :44.397	EXEC	KEY_n
285	11 :44.423	EXECUTION	START receiveKey(478)
286	11 :44.468	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
287	11 :45.588	TProf	je change pas
288	11 :45.588	GProf	index levé
289	11 :47.078	T1	hmm hmm
290	11 :47.529	TProf	le programme

Transcription 3.15 – 46g, S5, extrait1 : poser le problème(1)

291	11 :49.216	ENTRÉE	ANSW <<5>>
292	11 :49.549	TProf	ça veut dire qu'à chaque fois que je relance
293	11 :50.082	EXECUTION	FIN receiveKey(478)
294	11 :51.597	EXEC	KEY_n
295	11 :51.622	EXECUTION	START receiveKey(478)
296	11 :51.667	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
297	11 :52.352	TProf	il doit modifier au fûr et à mesure
298	11 :52.539	ENTRÉE	ANSW <<6>>
299	11 :53.391	EXECUTION	FIN receiveKey(478)
300	11 :53.599	EXEC	KEY_n
301	11 :53.621	EXECUTION	START receiveKey(478)
302	11 :53.664	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
303	11 :54.460	ENTRÉE	ANSW <<7>>
304	11 :55.215	T1	ah je sais
305	11 :55.311	EXECUTION	FIN receiveKey(478)
306	11 :56.352	T1	ça veut dire que faut faire tous les programmes
307	11 :56.352	G1	geste englobant
308	11 :58.451	TProf	et ben euh
309	11 :59.803	TProf	non on va toujours rester sur un
310	12 :01.567	ECRAN	APP_button
311	12 :01.941	T2	ah
312	12 :02.549	TProf	et
313	12 :03.078	T2	ah d'accord je sais je crois savoir
314	12 :03.551	ECRAN	FULL_button
315	12 :05.019	TProf	t'as une idée
316	12 :05.588	TProf	bon ben essayez et puis après je reviens
317	12 :07.022	ECRAN	APP_button
318	12 :07.352	T1	donc déjà on va changer tout ça
319	12 :09.313	T2	allez vas-y là tu fais quatre
320	12 :09.313	G2	montre b1
321	12 :11.686	T2	cinq six sept huit neuf dix
322	12 :11.686	G2	montre b2 puis b3 puis b4 etc

Transcription 3.16 – 46g, S5, extrait1 : poser le problème(2)

```
Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 4 fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter 5 fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter 6 fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter 7 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 8 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 9 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 10 fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || compteur |]] ||
hexagones |]]
```

Script 3.5 – Une tentative de généralisation (46g, S5)

Recherche de régularités

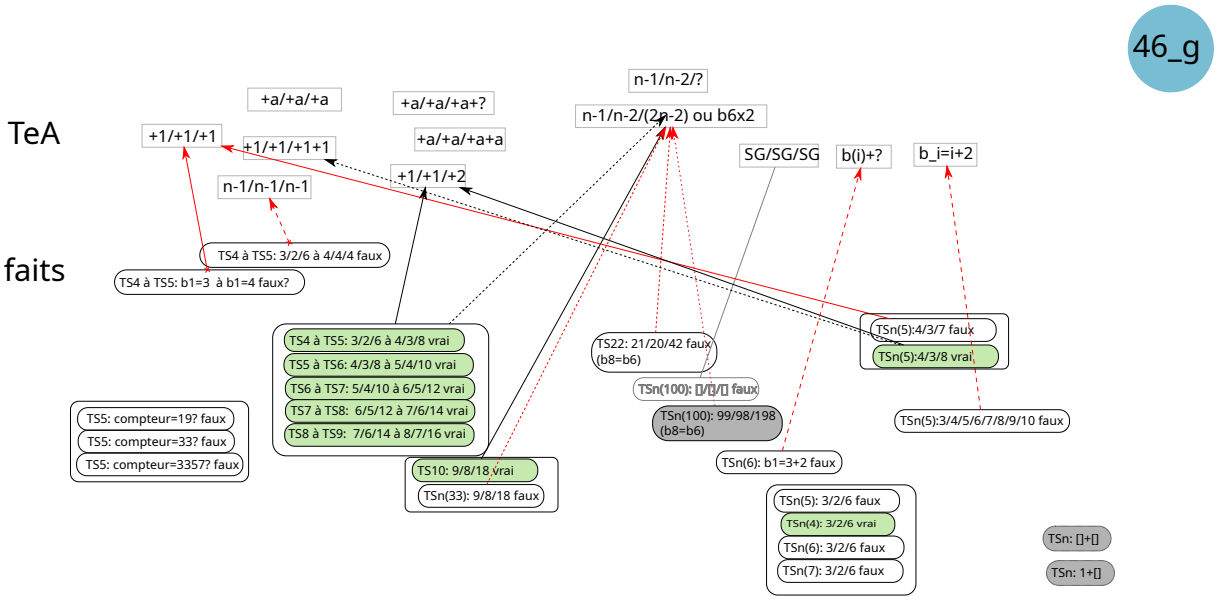


Figure 3.57 – Faits et TEA, groupe 46g

Le groupe 46g semble avoir construit les faits leur permettant de modifier un script directement afin de tracer une certaine instance (généralisation algébrique naïve). Cependant, lorsque le problème de la généralisation va se poser, les élèves de ce groupe ne vont pas mobiliser ce TEA.

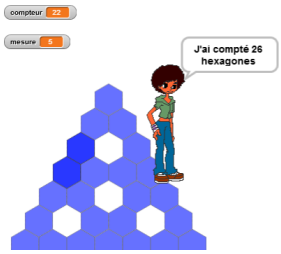
Validation : relation compteur-mesure Le groupe 46g est un des groupes qui cherchent à établir une relation entre le nombre d'hexagones dénombrés — qui est représenté par la valeur finale de la variable *compteur* — et la valeur de la mesure. Ces élèves ont identifié, avec raison, ces deux éléments comme étant des caractéristiques de la situation étudiée. Ainsi, le nombre d'hexagones total du TS est mobilisé :

— comme élément définissant le tracé du TS : en début de séance 3, les élèves tentent de modifier la valeur du compteur pour que cela corresponde à « un de la taille au-dessus », d'abord en augmentant la valeur initiale du compteur pour qu'elle soit *un de plus* que pour un TS4 (passage de 18 à 19), puis en fixant sa valeur à celle attendue pour un TS5 (passage de TS4 à TS5). Pour E1 en début de séance « c'est le compteur qu'il faut changer » (46g, transcription S3, 8'50) puisque c'est cette valeur qui bouge, qui se modifie en cours de tracé, et donc qui est liée à l'état du tracé. En fait, tout se passe comme si E1 voyait le compteur comme représentant l'état du programme, ce qui est en partie exact [41] . Cependant, elle pense que modifier le compteur *modifie* l'état du programme (donc le tracé), alors que le compteur *est modifié* lorsque l'état du programme change (plus précisément lorsqu'un hexagone a été tracé) ;

41. Un algorithme peut être représenté par un « automate à état », définissant notamment un ensemble fini d'états (dont les états initiaux et finaux) et les transitions entre états.

— comme élément de validation du tracé : en début de séance 3, pour E1, le compteur devrait être de 33 pour un TS5 alors que la rétroaction affiche 43 ; en fin de séance 4, lorsque le tracé du TS100 est en cours, cette même élève observe la valeur du compteur en précisant qu'elle doit atteindre 600 à la fin de la boucle en cours (b_6) : « faut qu'ça aille jusqu'à six cents » et comme « ça ne marque pas ça », « c'est pas bon » (3.17).

Cette relation entre la valeur du compteur et la mesure du TS tracé ne sera cependant pas mobilisée autrement. Notamment, lorsque les élèves tentent $b_1 =$ (3 + 2) pour un TS5, la rétroaction suivante est produite :

$$TS(n)[5] \rightsquigarrow \left(\text{  } \text{} , 26, 5 \right)$$

Cette rétroaction n'est pas vue comme invalide, malgré le fait que les élèves ont manifesté à plusieurs reprises leur connaissance du nombre d'hexagones attendus (33) pour un TS5. Il est probable que cette éviction de l'utilisation du compteur comme indice de validation provienne ici de la généralisation qui est en cours : les élèves ne sont pas en train de construire un script permettant de tracer un TS5, mais le programme générique. Le problème posé n'est pas le même : les faits jugés pertinents pour ce problème ne sont donc pas les mêmes. De plus, lors de la construction des instances successives, les élèves ne disposaient pas des faits liant nombre d'hexagones total et valeur de la mesure pour des mesures supérieures à 6, il leur est donc impossible de déterminer la validité du nombre d'hexagones de ces TS.

Validation : position du problème Pour la rétroaction donnée ci-dessus, nous avions déjà évoqué que les élèves considéraient ce tracé comme étant valide. Cela amène à s'interroger sur deux points :

— qu'est-ce qu'un tracé valide pour ces élèves ?

— quel problème se posent ces élèves (ou, formulé autrement, quel est leur but) ?

Ces deux points sont forcément intriqués, la validation dépendant du résultat attendu, et donc du but. À plusieurs reprises, nous avons noté que la validation ne tenait compte ni des chevauchements éventuels, ni de la dimension relative du tracé final. Pour ces élèves, il semble que la validation se limite au fait premier [42] « la forme du tracé est un TS ». Le fait premier « le tracé produit

42. voir Faits premiers,p. 189

988	31 :36.790	E1	fin b6
989	31 :36.790	T1	attends faut regarder un truc
990	31 :39.090	E2	montre affichage compteur
991	31 :39.090	T2	je regarde là
992	31 :39.900	T1	oui
993	31 :40.580	T1	faut qu'ça aille jusqu'à six cents
994	31 :43.480	T1	oh non
995	31 :45.870	T1	c'est pas bon
996	31 :47.160	T2	mais si
997	31 :47.790	T1	bah non ça marque pas ça
998	31 :48.310	T2	ah ouai

Transcription 3.17 – 46g, S4 : Validation par le compteur

des trous » est aussi mobilisé pour invalider, mais si le tracé produit des trous, la forme n'est plus celle d'un TS. En revanche, « le tracé produit des chevauchements » est un fait premier qui, pour ces élèves, n'implique pas « la forme du tracé n'est pas un TS », de même que la dimension relative du TS tracé. De plus, la valeur de la mesure, et notamment lorsque cette mesure est entrée par l'utilisateur, est rarement mobilisée pour la validation. Ce ne sera qu'en séance 5, après intervention de l'enseignante, que E1 commencera à la prendre en compte. Ainsi, lorsque l'enseignante fait une démonstration avec le script générique qui, sur une entrée de 4 trace bien le TS4, mais sur une entrée de 5 trace aussi le TS4, elle demande à E1 « si j'mets cinq il doit m'faire cinq, est-ce qu'il le fait ? » et E1 répond, comme une évidence « ben oui », et insiste, après que E2 a compris que l'enseignante s'attendait plutôt à une réponse négative, « bah si il vient de le faire » (46g, transcription S5, 10'43-10'52).
On voit ainsi que ce groupe semble ne considérer que le problème consistant à *tracer des TS bien formés* : dès qu'un nouveau tracé est considéré comme valide, ces élèves considèrent avoir fini — avoir atteint leur but —, y compris pour le script générique. De plus, pour les différents scripts à compléter, elles se limitent à créer des instances successives. Le problème de la généralisation n'est posé qu'en toute fin de séance 5.
Il est probable que cette absence de position du problème visé — et la position d'un autre problème — soit à la source des nombreuses demandes d'aide auprès de l'enseignante.
On pourrait aussi penser que ne pas poser le problème de la généralisation empêche la construction de faits liés à cette généralisation, puisque nous avons évoqué l'idée que les faits construits étaient dépendants de l'activité et donc du but que se fixaient les élèves. Cependant, grâce à la collaboration fréquente avec leur voisine, ce groupe a pu construire quelques faits et TEA qui relèvent de la généralisation.

De l'intérêt de la collaboration : une généralisation algébrique naïve Comme nous l'avons vu dans la description de l'activité, le groupe 46f et le groupe 46g, qui sont voisins, ont collaboré à plusieurs reprises. C'est ce groupe 46f qui va amener le groupe 46g à prendre en considération les marques de chevauchement. C'est aussi lui qui va aider les élèves à poser le problème de la séance 4 : non pas compléter les scripts avec des instances successives, mais construire aussi des instances plus distantes, le TS22 notamment, et plus tard le TS100.
Ainsi, dans un premier temps, c'est en recopiant sur leur voisine les valeurs pour tracer un TS5, que le groupe 46g va réussir ensuite à construire les instances suivantes, en appliquant systématiquement le TEA valide $+1/+1/+2$. De même, la construction du script permettant de tracer un TS22 se fait au début en reprenant les valeurs entrées par le groupe 46f, mais lorsqu'il s'agit de trouver la valeur pour b_7, E2 la propose spontanément, avant que leur voisine ne l'évoque. E2 a mis en oeuvre le TEA « pour trouver b_7 pour une instance, on double la valeur de b_6 (ou b_1) pour cette même instance ». E1 fera de même lors de la construction de l'instance 100 comme exemple générique. Le groupe 46g, grâce aux faits partagés par le groupe 46f, est ainsi devenu capable de faire une généralisation algébrique naïve : construire n'importe quelle instance d'un script en fonction de la mesure voulue, mais sans formaliser cette construction.

Le retour du TEA $+1/+1/+1$: des faits partagés non construits En séance 4, les élèves ont montré leur capacité à construire une instance à partir de la précédente, mais lorsqu'en séance 5 elles se sont retrouvées à devoir de nouveau créer une instance traçant un TS5 à partir de celle traçant un TS4, elles ont appliqué le TEA $+1/+1/+1$. Comme on l'a vu pour les groupes précédents, et ce sera aussi pour de nombreux autres groupes, ce TEA est généralement mobilisé en séance 3, lors des premières tentatives de construction d'instances successives, pour être

ensuite remplacé, plus ou moins rapidement, par le TEA $+1/+1/+2$. Son arrivée impromptue en séance 5 est questionnante : le TEA $+1/+1/+2$ semblait être effectif, de même que celui de la généralisation naïve « pour construire une certaine instance, j'enlève un pour la première boucle, j'enlève deux pour la deuxième etc... et je double la première boucle (ou la sixième) pour trouver la septième ». Il a fallu une rétroaction montrant le tracé erroné pour que E2 remobilise le TEA $+1/+1/+2$ en se référent aux échanges avec leur voisine : « mais là faut que tu rajoutes deux parce je crois c'est elle l'a dit que c'était faux » (46g, transcription S5, 13'22).
Cette arrivée tardive peut peut-être s'expliquer par les faits partagés en groupe classe en début de séance. En effet, nous avions décidé avec l'enseignante de faire verbaliser les élèves sur ce qu'il fallait faire pour passer d'une instance à la suivante, de façon à ce que ces faits soient partagés et mobilisables pour généraliser. De plus, les relations entre la mesure et les premières boucles devaient elles aussi être construites avec le groupe classe, mais en laissant de côté la relation entre la mesure et la boucle b_7. Mais en procédant ainsi, nous avons sans doute induit des faux faits, ou empêché certains élèves de construire ces faits partagés.
En effet dans un premier temps, pour expliquer le passage du $TS(4)$ au $TS(5)$, l'enseignante a vu compléter sa phrase « lorsque vous avez dupliqué et que vous avez mis la mesure cinq » par un élève qui a affirmé « on a rajouté un ». L'enseignante a voulu attirer l'attention sur la boucle b_7 « et là qu'H disait y'à une boucle que c'est pas rajouter un mais y'à un ptit autre chose » (46 Prof, S5, 7'57-8'15). Cependant, le « plus deux », ce qu'il faut rajouter à la boucle b_7, n'a pas été verbalisé. Non construit préalablement par les élèves, et non verbalisé en groupe classe, ce fait n'a pas pu être mobilisé par ce groupe. De plus, lorsqu'il s'agissait d'établir les relations mesure-boucle, les faits suivants ont été partagés :

- pour $mesure = 8$, $b_1 = 7$ et $b_2 = 6$
- pour trouver la première boucle, il suffit de faire « un chiffre de moins que la mesure » : $b_1^n = n - 1$
- pour trouver la deuxième boucle, « c'est un de moins que la boucle une » (reprise d'une formulation élève) : $b_2^n = b_1^n - 1$
- pour trouver la deuxième boucle, « c'est deux chiffres en moins que la mesure » : $b_2^n = n - 2$ (et $b_2^8 = 8 - 2 = 6$)

Autrement dit, les faits qui ont été partagés ne concernaient de façon précise (c'est-à-dire avec une méthode de calcul) que les boucles b_1 et b_2, ce qui peut expliquer leur extension à b_7 par le groupe 46g : le « rajouter un » s'applique alors à toutes les boucles, ce qui correspond au TEA $+1/+1/+1$.

Généralisation algébrique

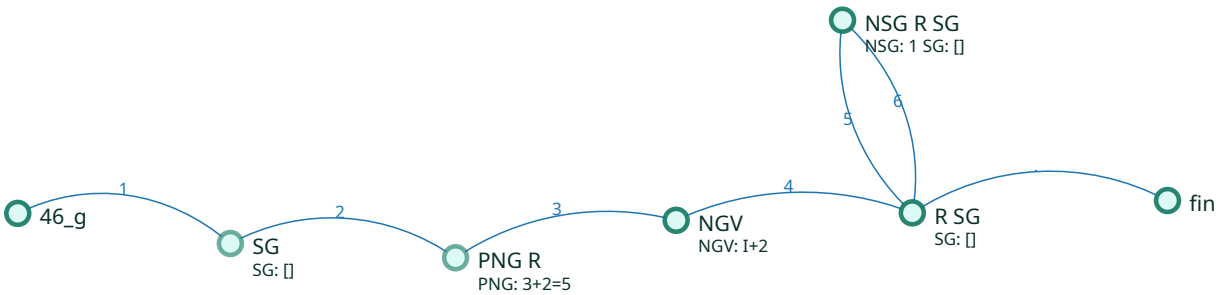


Figure 3.58 – Type de généralisation - 46g

Lors des tentatives de généralisation algébrique, les élèves vont mobiliser divers types de généralisation : un signe générique (vide) comme marque-place, et une relation avec soit un nombre perdant sa généricité (PNG), soit un nombre-signe générique (combiné avec le signe générique vide), soit encore un nombre générique « variable ».

Tension fixe-variable Le groupe 46g a eu des difficultés à poser le problème de la généralisation. Cependant, quatre épisodes sont susceptibles de relever de ce problème, et nous donnent des indications sur les solutions envisagées par les élèves pour régler la tension fixe-variable, entre le script qui doit « faire tous les programmes » et l'impossibilité de modifier ce script entre deux essais. Cette tension est notamment visible en séance 5 dans l'échange rapporté dans la transcription 3.18. Après avoir dupliqué le script traçant le TS4 pour construire le script générique, le binôme a réussi à le modifier pour tracer un TS5, ce qui fait dire à E1 que le but est atteint (3.18, 386). E2 ramène alors au problème de généralisation : le « et après » (3.18, 389) renvoie à la nécessité de faire plusieurs essais successifs, comme montré par l'enseignante quelques minutes plus tôt. Pour E1, il suffit de modifier de nouveau le script (« tu rechanges tout », 3.18, 390). Mais E2 la contredit et rappelle la démonstration faite par l'enseignante : « faut pas rester ». Ce « rester » réfère à l'espace de programmation : nous avions convenu avec l'enseignante, au vu des difficultés des élèves à construire la nécessité d'un seul script, de leur présenter la contrainte d'exécuter le script générique seulement en plein écran (correspondant à l'appui sur le bouton idoine, retranscrit par l'action « ECRAN : FULL_button », 3.15, 276). Dans ce mode, seul l'espace d'exécution de l'EPGB est visible, l'espace de programmation étant inaccessible : on ne peut pas « rester » dans cet espace pour faire les différents tests. Or les élèves sont dans cette configuration : après avoir basculé en plein écran, la première rétroaction de leur test invalidant leur script, les élèves sont obligés de basculer sur l'espace de programmation (« ECRAN : APP_button », 3.15, 310). Comme on ne peut pas rester dans cet espace, on ne peut pas non plus « rechanger » tout.

Des nombres génériques variables Pour résoudre cette tension, une première hypothèse est amenée par E2, sans justification : construire le script en affectant à la suite de boucles $[b_1..b_8]$ respectivement les valeurs de la suite de nombres [3..10] (script 3.5, p.318). Tout se passe comme si E2 « préparait » le script en affectant les différentes valeurs possibles de la mesures aux différentes boucles, comme si la suite des valeurs de boucles était la liste des valeurs possibles de la mesure.

380	13 :31.352	EXECUTION	START receiveKey(478)
381	13 :31.396	ENTRÉE	ASK «Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?»
382	13 :34.428	ENTRÉE	ANSW «5»
383	13 :35.567	EXECUTION	FIN receiveKey(478)
384	13 :36.117	T1	hihi
385	13 :36.117	G1	tape des mains
386	13 :36.843	T1	on a réussi ça fait
387	13 :38.529	T1	un deux XXX
388	13 :38.529	G1	montre un à un les premiers hexagones de b1
389	13 :39.705	T2	et après on fait comment
390	13 :41.333	T1	et ben après tu rechanges tout
391	13 :43.235	T2	mais non
392	13 :44.588	T1	bah je sais pas
393	13 :44.882	T2	parce que faut pas rester

Transcription 3.18 – 46g, S5 : Changer sans changer

En fait, pour E2, ces nombres ne correspondent pas à un tracé effectif, ils sont des exemples de valeurs que pourrait prendre la mesure, mais aussi une façon de montrer la variation de ces valeurs. Une sorte de Nombre Générique Variable (NGV) en somme. Il y a peut-être ici un reste de pensée magique, cette croyance que l'ordinateur comprenne de lui-même ce qu'on attend de lui [43]. La rétroaction invalidant cette hypothèse, E2 en conclut que « on ne va pas faire ça, on va recommencer, du coup il faut retrouver les mesures hein » (46g, transcription S5, 14'50). Ainsi, la nécessité d'une relation entre les valeurs du nombre de répétitions des boucles et la mesure est évoquée.

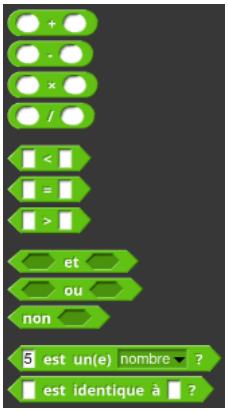


Figure 3.59 – Opérateurs disponibles dans la situation

Représenter une relation Cette relation, pour le groupe 46g, est représentée non pas par une relation arithmétique ou algébrique, mais par le « vert » (46g, transcription S5, 5'20), c'est-à-dire un bloc de la couleur verte (correspondant à la catégorie des opérateurs), dont l'enseignante a pointé la possible utilité. Ainsi, elle précise en plénière de début de séance 5 qu'elle « pense que l'ordinateur est capable de les faire les calculs » (3.19, 264), et que, comme « opérateur » fait penser à « opération », « peut-être qu'il va falloir qu'on utilise maintenant ces instructions là » et ensuite « c'est à l'ordinateur de se débrouiller pour calculer ce qu'il doit mettre la boucle »

43. Nous verrons d'autres exemples à l'œuvre dans certains groupes, notamment la création d'un nouveau bloc nommé faire le programme tout seul (groupe 45b, étape 33)

264	15 :19.400	Prof	je pense que l'ordinateur est capable de les faire les calculs
265	15 :23.320	Prof	alors je ne sais plus si je vous avais montré quels sont les | les outils qui permettent de faire les calculs | sur snap
266	15 :31.240	Prof	et je sais pas si certains ont été si je crois que ya certains groupes qu'ont essayé qu'ont réussi à voir où ça se situait les opérations sur Snap
267	15 :42.090	Prof	alors
268	15 :42.950	Prof	mouvement c'est vraiment les | déplacements bleus on avance on recule
269	15 :47.810	Prof	apparence
270	15 :49.100	Prof	c'est | c'qui
271	15 :52.750	Prof	donc ya pas de son ya pas de stylo ya pas de [controle]
272	15 :55.400	Prof	ya capteur
273	15 :57.010	Prof	ya opérateur
274	16 :00.130	Prof	opérateur ça vous fait pas penser à opération ?
275	16 :04.330	Prof	donc là peut-être qu'il va falloir qu'on utilise maintenant ces instructions là
276	16 :09.470	Prof	et en fait c'est à l'ordinateur de se débrouiller pour calculer ce qu'il doit mettre la boucle

Transcription 3.19 – 46 Prof, S5 : les opérateurs

(3.19, 274-276). Ainsi il semble que pour ces élèves, un signe ( ) ou ( ) permet de résoudre le problème et permet à l'ordinateur de faire les calculs tout seul. C'est effectivement le cas, mais le groupe 46g ne semble voir ici que le côté « opérer », faire, et non la relation permettant de déterminer la valeur du nombre de répétitions d'une boucle en fonction de la mesure, relation que l'on peut formuler avec un opérateur. Si chez certains groupes, l'opérateur (-) est utilisé, et renvoie au fait qu'il faut « faire moins un » par exemple, ce groupe utilise dans un premier temps l'opérateur (5 est un(e) nombre ?), et dans un second temps l'opérateur (+). Le premier, utilisé aussi par le groupe 46m, semble être porteur de la généricité du nombre — puisqu'on doit bien entrer un « nombre » —, mais ne porte pas de relation. Le second en revanche porte une relation, mais cette relation est elle même générique pour les élèves du groupe 46g : à aucun moment elles n'ont dû ajouter quelque chose à la mesure pour trouver une valeur de nombre de répétitions d'un boucle. Il s'agit d'une instruction là encore un peu magique, qui est censée permettre à l'ordinateur de comprendre qu'il lui faut faire une opération.

Perte de généricité et signe générique Cet opérateur sera utilisé à deux occasions et sous trois formes différentes : (3 + 2) en début de séance 5 (9'03) et (+) et (1 +) (cette dernière forme ne débouchant pas sur un test) en fin de séance (15'35 et 17'03). Pour la première forme, une relation avec la mesure semble être implicite ici : le nombre de répétitions de la première boucle du script en jeu était de « 3 », et le but suivi par les élèves était de tracer un TS de mesure « 5 ». On peut imaginer que E1, en examinant la possibilité d'utilisation de ce bloc, a cherché à faire un lien entre la boucle et la mesure, en l'occurrence ici $b_1^5 = b_1^4 + (5 - b_1^4)$. Autrement formulé, E1 cherche une opération mobilisant à la fois la boucle et la mesure, ici en permettant de passer de la valeur de la boucle (3) à la valeur de la mesure (5) en ajoutant 2. Le nombre « 2 » est peut-être ici porteur de généricité, étant ce qui relie le 3 à la mesure, mais cette généricité est non visible. Dans $b_1^{\boxed{5}} = b_1^4 + (\boxed{5} - b_1^4)$, la généricité du 5 peut apparaitre : si la mesure était 6, on écrirait $b_1^{\boxed{6}} = b_1^4 + (\boxed{6} - b_1^4)$. D'une façon plus générique encore, on pourrait écrire $b_1^{\boxed{n}} = b_1^4 + (\boxed{n} - b_1^4)$. En revanche, écrire $3+2=5$ et $3+3=6$ permet difficilement d'identifier le motif commun à ces deux écritures. Nous dirons ici que le nombre deux est une *Perte du Nombre Générique* (PNG). D'autres cas seront évoqués dans les prochains groupes. On verra notamment que les relations intra-objectales, comme $b_7^p = 2b_6^p$ pour le groupe 46g, semblent favoriser cette perte de généricité. Les deux autres formulations impliquent quant à elles ce qui pourrait être des signes génériques (SG) : le fait de ne pas donner de valeur aux paramètres de l'opérateur revient à laisser un espace vide qui marque une place à remplacer. Les élèves, cependant, ne testent pas le script obtenu (script 3.6, p. 326), la fin de séance étant atteinte. Il aurait été intéressant de savoir ce qu'auraient fait ces élèves de ces opérateurs suivant les valeurs entrées. Nous verrons ceci à l'œuvre pour d'autres groupes.

```
Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [ + ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ + ] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [ + ] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [ + ] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [ + ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ + ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ + ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ + ] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || compteur |]] ||
hexagones |]]
```

Script 3.6 – Autre tentative de généricité (46g, S5)

Construction des nécessités

Les élèves du groupe 46g, malgré des difficultés marquées pour poser le problème, ont malgré tout construit quelques faits et nécessités relatifs à la généralisation algébrique.
L'existence d'une relation entre les boucles et la valeur de la mesure semble être (en partie) établie, puisque ce groupe, après s'être inspiré du groupe voisin, est capable de construire des instances indépendantes (ici pour le TS22 et le TS100). Cependant, cette relation reste implicite, et entre en tension avec celle proposée en fin de séance 5 : pour un TS5 :

— Séance 4 : $b_1^5 = 5 - 1 = 4$;

— Séance 5 : b_1 doit être une expression dont le résultat vaut cinq.

La construction, essentielle (et contrainte) pour ce problème, de la nécessité qu'un même script doit être capable de tracer tout TS, quelle que soit la mesure demandée, est tardivement construite mais finalement établie. Celle-ci va entrainer un certain nombre de tensions.
En premier lieu, puisque changer un nombre de répétitions par un autre nombre de répétitions dans un script aboutit à changer l'instance du TS tracé, ce nombre de répétitions doit changer si on veut tracer n'importe quel TS. De plus, les élèves ont constaté que si les valeurs des nombres de répétitions étaient représentées par des nombres (ou plutôt une écriture de nombres), il était nécessaire de « tout rechanger » si on voulait tracer une nouvelle instance avec le même script, ce qui entre en tension avec la nécessité d'un seul et même script que l'on ne modifie pas : le nombre de répétitions ne peut donc pas être une représentation d'un nombre. Ceci, même si on fait varier les nombres à l'intérieur du script (script 3.5), p. 318.

Une expression arithmétique contenant des écritures de nombres ($b_1 =$ 3 + 2) ne fonctionne pas mieux : il doit donc exister une expression, dans le langage θ du problème (Snap !), qui permet de représenter plusieurs mesures, et donc ne contenant pas que des nombres. C'est sans doute ce qui amène les élèves à construire le script 3.6 (p. 326), en affectant à toutes les boucles + : c'est une expression (même si elle est sans rapport avec la relation établie par les élèves entre b_1 et la mesure, par exemple), et elle ne contient pas d'écriture de nombres. Il manque finalement peu de chose pour aboutir à la nécessaire existence d'une expression représentant la valeur de la mesure, et qui soit mobilisable dans une expression.

Bilan

Le groupe 46g, malgré ses difficultés à poser le problème (qui ont entre autres effets entrainé de longs moments d'inactivité), a malgré tout avancé sur la voie de la généralisation : une généralisation algébrique naïve est visible, et une tentative de résolution de la tension fixe-variable a été ébauchée. Ce groupe a profité des faits partagés par le groupe voisin : loin d'être un obstacle, il est probable que c'est ce qui a permis à ces élèves d'établir des relations inter-objectales, première étape vers une généralisation formalisée. Ainsi, si les élèves n'arrivent pas à produire suffisamment de faits permettant d'identifier un motif, un schéma, faut-il les leur fournir de façon plus directe que ce que nous avons fait ? Faut-il remplacer les faits partagés concernant uniquement certaines valeurs des boucles par l'intégralité des scripts ? On peut penser ici qu'il serait intéressant d'employer les caricatures (Orange, 2012, p. 102).

3.4.2 Groupe 46i

Description de l'activité

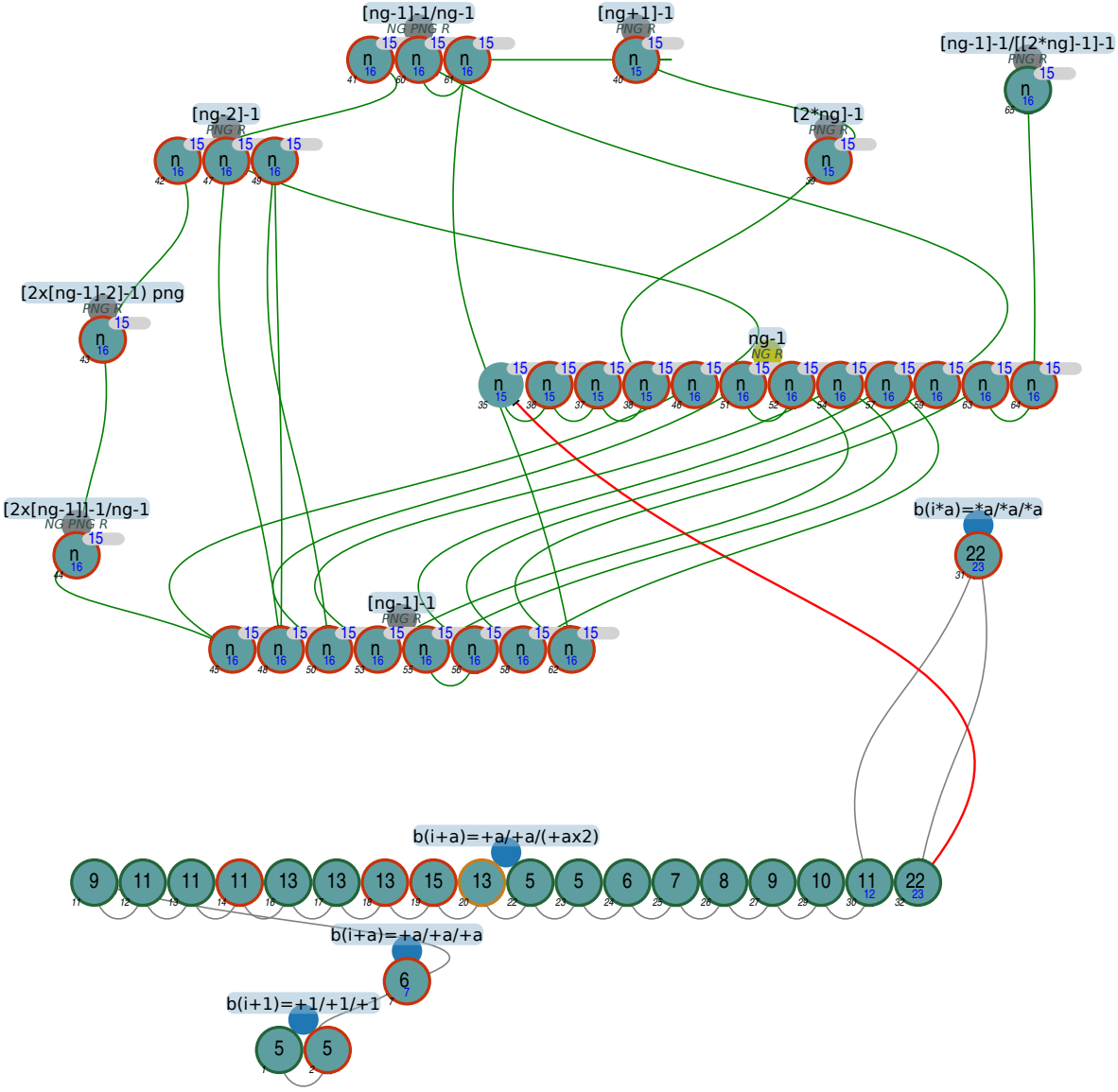


Figure 3.61 – Organisation de l'activité - 46i

Le groupe 46i va construire de nombreuses instances de scripts traçant et dénombrant des TS : en séance 3, des instances impaires jusqu'au TS13, puis les instances successives de TS5 à TS11, ainsi que le TS22, en séance 4. Ils se confronteront en séance 5 à la construction du script générique, mais en ne dépassant pas le stade de l'utilisation des nombres génériques.

Construction classique du TS5 (étapes 1-6) Le groupe 46i commence, comme quasiment tous les groupes, par dupliquer le script traçant un TS4 et le modifier en appliquant le TEA $+1/+1/+1$ (donc $TS(5) = \{4/3/7\}$), mais en oubliant (sans doute) de modifier b_6 :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 31, 5 \right)$$

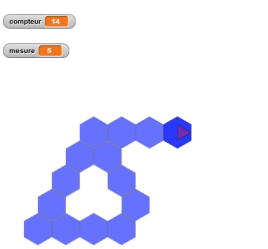
Contrairement au groupe 46g, le groupe 46i interprète bien les hexagones de couleurs plus foncée comme le signe d'un tracé invalide. Ces élèves procèdent alors à plusieurs exécutions incluant des pauses, afin sans doute d'identifier les erreurs. Cette exploration leur permet d'identifier l'oubli de modification de la boucle b_6, oubli qu'ils corrigent alors pour obtenir la rétroaction désormais classique du TEA $+1/+1/+1$:

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 32, 5 \right)$$

Les élèves vont alors modifier b_8 en lui ajoutant un $(b_8 \leftarrow 4)$, semblant considérer que chaque boucle trace un côté des triangles de base. La rétroaction leur permettra d'identifier le tracé effectif de la boucle b_8 (et indirectement celui de b_7)[44] :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 33, 5 \right)$$

En effet, grâce de nouveau à une exécution mise en pause, ils vont pouvoir vérifier que c'est bien le tracé de la dernière boucle qui rajoute un élément erroné (un chevauchement), puisque le tracé du côté montrant un nouveau chevauchement est valide jusqu'à la dernière partie du script :

$$TS_4(5)[] \rightsquigarrow \left(\text{  } \text{} , [], 5 \right)$$

44. Notons que le nombre d'hexagones dénombrés, 33, est bien celui attendu, et que ce nombre est connu pour le TS5. Cela ne semble pas être un critère de validité suffisant pour ces élèves — si tant est qu'il soit pris en compte à ce stade.

Ils vont, à la suite de ce résultat, remettre le nombre de répétitions de la boucle b_8 à son ancienne valeur, et augmenter celui de b_7 de un, obtenant ainsi le script valide pour TS5.

Construction du TS7 par le script $TS(6)$ (étapes 7-10)
Pour la construction du TS6, ce groupe va procéder différemment des autres groupes : si ces derniers, en majorité, semblent construire une nouvelle instance $p+1$ à partir de l'instance précédente p, en ajoutant un aux premières boucles, le groupe 46i semble en partie construire leur nouvelle instance du script attendu $TS(6)$ à partir de l'instance 4 (qu'ils ont dupliqué) ou 5 (qui est le script mitoyen) :

— Soit en ajoutant 3 au nombre de répétitions des boucles de $TS(4)$ (46i, S3, 6'24-6'59). Seules b_7 et b_8 seront modifiées autrement : $b_7^6 \leftarrow 8$, or $8 \neq b_7^4 + 3$, et $b_8^6 \leftarrow 4$, or $4 \neq b_7^4 + 3$.

— Soit en ajoutant 2 au nombre de répétitions des boucles de $TS(5)$ (46i, S3, 6'24-6'59). Seules b_7 et b_8 seront modifiées autrement : $b_7^6 \leftarrow 8$, or $8 \neq b_7^5 + 2$, et $b_8^6 \leftarrow 4$, or $4 \neq b_7^5 + 2$.

En fait, il semblerait que ce groupe ait affecté $b_1 \leftarrow 6$, peut-être en estimant que la première boucle doit tracer un côté complet du triangle de base, et donc 6 hexagones pour le TS demandé. Les autres modifications sont alors issues de la comparaison entre la boucle b_1 du script en cours de modification et les autres scripts : puisqu'il faut ajouter deux (respectivement trois) pour passer de b_1^5 (respectivement b_1^4) à la valeur de b_1 du script en cours, alors on ajoute deux (respectivement trois) partout. La démarche adoptée par les élèves pour les deux dernières boucles est différente, possiblement due la mobilisation d'une autre relation. Quoiqu'il en soit, ce groupe pense peut-être construire le script $TS(6)$, mais leur script trace en fait un TS7.
Ce script obtenu, {6/5/5/5/5/6/8/4} produira la rétroaction suivante (46i, S3, 7'27) :

$TS(6)[] \rightsquigarrow$ 

$, 46, 6$

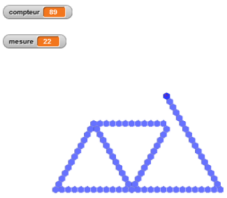
Il sera corrigé d'abord en ajustant b_7 (en deux essais), puis en adaptant b_8 (figure 3.62, p. 339). On peut constater que pour ce groupe, comme pour d'autres, le fait que le tracé sorte en partie de l'espace d'exécution n'est pas un critère d'invalidation, puisque suite à la rétroaction montrant une forme valide (figure 3.62c, p. 339), ce groupe va immédiatement passer à la construction d'une nouvelle instance.

Construction des TS9, TS11, TS13 et TS15 (étapes 11-20) Le groupe 46i va par la suite enchainer la création de scripts, dans un premier temps en dupliquant systématiquement le script $TS(4)$, afin de construire deux instances qui traceront un TS9 et un TS11. Pour les instances suivantes (traçant les TS11, TS13 et TS15), ils dupliqueront le dernier script créé. Ils produiront ainsi successivement les scripts $S_a = \{8/7/16\}$, $S_b = \{10/9/20\}$, $S_c = \{12/11/24\}$ et $S_d = \{14/13/28\}$. On voit ainsi que ce groupe ne construit pas des instances successives, mais des instances dont la mesure augmente de deux en deux. Il est tout à fait probable que les élèves du groupe 46i ne considèrent plus à ce moment la valeur de la mesure comme étant un critère de validation. Ils semblent plutôt réappliquer une organisation de leurs actions qui a abouti à un tracé valide. Ces scripts seront créés rapidement, mais cela ne se fera cependant pas sans erreurs.

Ainsi, pour la boucle b_6 du script traçant le TS11, le nombre de répétitions sera fixé à 9 (comme b_2 à b_5) et non à 10 (comme b_1). Cette erreur (déjà présente pour la construction du TS5) est corrigée immédiatement après la rétroaction. On peut ainsi penser que les boucles b_3 à b_5 sont construites non par référence à la même boucle d'une autre instance ($b_3^{11} = b_3^9 + 2$, relation inter-objectale), mais par référence à la boucle b_2 (ou la boucle précédente) de l'instance en cours ($b_3^{11} = b_2^{11}$ et $b_4^{11} = b_3^{11}$ ou $b_4^{11} = b_2^{11}$, relation intra-objectale).
Lors de la création du script S_c traçant le TS13, une nouvelle erreur se fera sur b_6, cette fois en lui affectant un nombre de répétitions de 14 au lieu de 12. Cette erreur se répercutera sur la boucle b_8 du même script, ainsi que sur le script suivant, qui sera construit à partir de la duplication du script S_c avant que celui-ci ne soit identifié comme erroné. Ici encore, il nous semble voir l'application d'une relation intra-objectale : s'il est difficile d'établir l'origine de l'erreur sur b_6 (14 au lieu de 12, peut-être une confusion avec la méthode permettant d'obtenir b_7, $b_7^{p+2} = b_7^p + 4$), on peut penser que l'action $b_8^{13} \leftarrow 13$ fait le lien entre b_8 et b_6 ($b_8^p = b_6^p - 1$). Le script S_c est copié avant d'avoir modifié la boucle b_7, et avant d'avoir été testé. Le groupe 46i modifie alors les boucles b_1 à b_6 de ce nouveau script, puis les boucles b_7 des deux scripts en cours de création :

— $b_{i\in[1..6]}^{15} = b_i^{13} + 2$
— $b_7^{13} = 24$ et $b_7^{15} = 28$

La boucle b_8 du dernier script sera répétée 16 fois (au lieu de 14), le nombre 16 semblant provenir de la boucle b_6 du même script, rejoignant l'idée d'une relation intra-objectale, mais différente de celle du précédent script ($b_8^p = b_6^p$ pour ce script, $b_8^p = b_6^p - 1$ pour le précédent). La rétroaction du script S_c sera stoppée dès l'erreur visible, mais non corrigée pour cause de fin de séance.

$$S_c[] \rightsquigarrow \left(\text{  } \text{} \quad , [], 22 \right)$$

Construction de multiples instances consécutives (étapes 22-27) En séance 4, le groupe 46i va commencer par enchainer la construction de nouvelles instances, cette fois-ci en suivant davantage les tâches prescrites.
Ils vont ainsi, en moins de neuf minutes, construire les instances demandées traçant les TS5, TS6, TS7, mais aussi celles non attendues traçant les TS8 et TS9. Comme les élèves du groupe 45f (voir p. 277), les élèves du groupe 46i ne semblent pas se préoccuper de la tâche prescrite, ils associent plutôt, par raccourci sémiotique, la lettre de la touche déclenchant l'exécution du script et l'instance associée par itération :

a	→	4
b	→	5
c	→	6
d	→	7
e	→	8
f	→	9

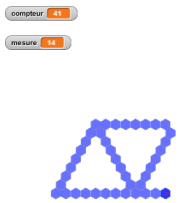
Ces cinq instances construites consécutivement le seront sans aucune erreur. Comme en séance 4, et comme pour d'autres groupes déjà évoqués, la dimension relative du TS tracé ne semble pas être une caractéristique de l'invalidation, pas plus que la valeur de la mesure affichée. Ainsi, les scripts traçant les TS8 et TS9 produiront les rétroactions suivantes considérées comme valides

par les élèves (puisque enchainant alors sur l'instance suivante) :

$$TS(8)[] \rightsquigarrow \left(\text{  } \text{} , 60, 11 \right)$$

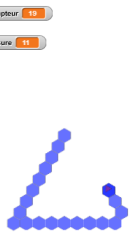
$$TS(9)[] \rightsquigarrow \left(\text{  } \text{} , 69, 22 \right)$$

Une demande perturbante (étape 28) Comme les élèves du groupe 46g, les élèves du groupe 46i semblent perturbés par l'invitation à entrer une valeur à l'exécution du script générique, sous l'entête duquel a été copié le script traçant le TS9. Ils vont ainsi lancer et arrêter l'exécution du script quatre fois de suite, sans entrer de valeur. Ils entreront ensuite la valeur 14, mais sans laisser le script s'exécuter jusqu'au bout :

$$TS_6(n)[14] \rightsquigarrow \left(\text{  } \text{} , [], 14 \right)$$

Cette rétroaction ne semble pas lever l'indécision quant à cette invite : les élèves passent alors à la construction des TS10, TS12 et TS23, non pas en modifiant le script attendu comme générique, mais en modifiant de nouveau les scripts exécutés par l'appui sur les touches « e » (tâche prescrite : TS11) et « f » (tâche prescrite : TS22). On peut remarquer que la valeur 14 entrée lors de l'invite est la position de la lettre « n » (touche lançant l'exécution du script générique) dans l'alphabet, contribuant à l'idée de raccourci sémiotique liant lettre et instance : pour ces élèves, le TS tracé devrait être « le quatorzième script », comme le « a » était le premier. Impossible, sans accès aux interactions langagières de ce groupe, de définir ce qu'est pour eux ce quatorzième script, mais en tout cas cela semble amener les élèves à invalider le raccourci sémiotique entre lettre et instance, puisque le nouveau script construit le sera sur un script correspondant à une lettre déjà utilisée. Avant cette nouvelle construction, il se sera écoulé dix minutes sans action de programmation, entre le test avec une entrée de 14, et un nouveau test (lui aussi interrompu) avec une entrée de 20, immédiatement suivie de la construction de ce nouveau script. La perturbation perdure...

Relation mesure et boucles (étapes 29-30) Une fois le script traçant le TS10 construit, sous l'entête du script dont la tâche prescrite est un TS11, les élèves vont le tester, de nouveau sans le laisser terminer le tracé :

$$TS(10)[] \rightsquigarrow \left(\text{  } \text{} \quad , [], 11 \right)$$

Ils modifient alors le script sur lequel ils travaillent, $\{10/9/20\}$, en $\{11/10/22\}$. Cette modification peut traduire deux activités différentes :

— soit les élèves sont certains de la validité de leur script, et ils passent à l'instance suivante ;

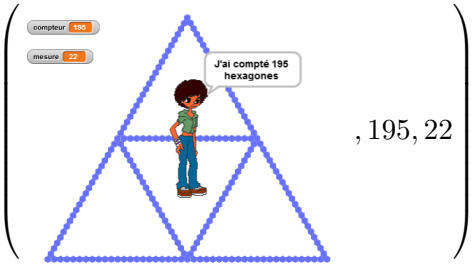
— soit ils constatent sur la rétroaction que le nombre d'hexagones tracés par la première boucle n'est que de neuf, alors que la tâche prescrite et la mesure affichée indiquent une valeur de onze : ils ajustent donc le nombre de répétitions de la boucle b_1 à la mesure en effectuant $b_1 = 11$.

C'est cette deuxième possibilité qui nous semble la plus probable, le groupe 46i enchainant ensuite sur la tâche prescrite de la construction du TS22 en commençant par l'action $b_1 = 22$. Ils semblent donc, à ce stade, et comme en début de séance 2 pour la construction du $TS(6)$, établir une relation d'égalité entre la valeur du nombre de répétitions de la boucle b_1 et la valeur de la mesure.

Construction d'une instance « éloignée » (étapes 31-32) En commençant à construire ce que le groupe 46i pense être un TS22 [45], les élèves commencent par effectuer les actions $b_1 \leftarrow 22$, $b_2 \leftarrow 20$, $b_3 \leftarrow 20$, etc. Les élèves semblent ainsi passer du script correspondant à leur idée d'un TS11 à ce nouveau script en multipliant les valeurs du nombre de répétitions des boucles par deux : comme $22 = 11 \times 2$, $b_i^{22} = b_i^{11} \times 2$, soit le TEA $*a/ * a/ * a$ [46]. Ce TEA, que d'autres groupes ont expérimenté, est sans doute lié à une représentation de la situation comme étant une situation de proportionnalité, ce qui n'est pas le cas. Le binôme ne va cependant pas avoir besoin de tenter le script pour se corriger : arrivés à l'action $b_6 \leftarrow 22$, ils modifient les boucles b_2 à b_5 avec une valeur identique (et exacte relativement à b_1, de 21, soit $b_1 - 1$). Il est possible que le fait mobilisé afin d'établir cette erreur est l'observation de la constante différence de un entre les valeurs du nombre de répétitions des boucles b_5 et b_6. Une autre possibilité étant que les élèves se soient ravisés pour finalement appliquer le TEA $+a/ + a/ + 2a$ déjà mobilisé en séance 3.

45. Rappelons que pour un script valide avec $b_1 = 22$, le TS tracé est le TS23.
46. En informatique, le l'astérisque dans une expression arithmétique désigne généralement l'opérateur de la multiplication, afin de ne pas confondre le signe « multiplié par » $\times$ et le signe « x ».

Le tracé obtenu est sans doute considéré comme valide, mais la fin de séance empêche de confirmer ce fait. Si tel est le cas, les élèves n'associent toujours pas la valeur de la mesure avec le nombre d'hexagones *effectivement* tracés sur les côtés des triangles de base — ceux-ci étant ici au nombre de vingt-trois.

$TS(22)[] \rightsquigarrow$ 

$, 195, 22$

Vers la généralisation (étapes 34-66) En début de séance 5, l'enseignante a rappelé le problème (construire un script traçant n'importe quel TS), le fonctionnement attendu du programme générique (la valeur entrée correspond à la mesure du TS tracé), et quelques faits partagés lors de la construction des différentes instances.
On trouve ainsi notamment :

— $b_1^8 = 8 - 1$, et plus généralement : la première boucle, « c'est un chiffre de moins que la mesure » (3.20, 249-254) ;

— la deuxième boucle, « c'est un de moins que la boucle une » (3.20, 255-258), $b_2^n = b_1^n - 1$

— la deuxième boucle, si on l'exprime « par rapport à la mesure », « c'est deux chiffres en dessous que la mesure » (3.20, 259-261) : $b_2^8 = 8 - 2$;

Les élèves du groupe 46i vont rapidement mobiliser ces faits. Ils vont passer quelques minutes à explorer ce qu'il se passe lorsqu'on entre les valeurs 15 ou 20 pour le script générique sur lequel on a copié le script permettant de tracer un TS4. Ensuite, une première tentative est faite, $b_1 \leftarrow$ (15 - 1) (ce que nous écrirons dans la suite de cette analyse $b_1 = 15 - 1$, dans le sens « le nombre de répétitions de la boucle b_1 est représenté par l'expression "15-1" »), testée avec une valeur entrée de 15 :

$TS(n)[15] \rightsquigarrow$ 

$, 35, 15$

Alors qu'ils avaient terminé la séance en laissant l'impression que, pour eux, $b_1 = mesure$, ce qui aurait dû produire $b_1 = 15$, les élèves mobilisent bien les faits partagés :

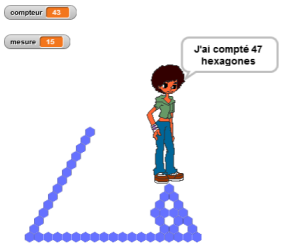
— la valeur entrée est la valeur de la mesure

— la première boucle, c'est la mesure moins un.

232	13 :55.760	Prof	sauf que moi je voulais un de mesure huit
233	13 :58.040	Prof	donc l'ordinateur devrait | lui -même | trouver ce qu'il faut mettre ici
234	14 :04.420	Prof	vous avez une idée déjà de si c'est un mesure huit qu'est-ce qu'on va mettre dans la première boucle ?
235	14 :10.350	Eleves	sept
236	14 :11.380	Prof	ça va être sept
237	14 :14.220	Prof	si c'est un de mesure huit
238	14 :17.940	Prof	là je devrais avoir sept
239	14 :20.840	Prof	dans la boucle d'après
240	14 :22.130	Eleves	six
241	14 :22.840	Prof	H ?
242	14 :23.160	Eleves	six
243	14 :23.880	Prof	six
244	14 :26.430	Prof	mais
245	14 :27.180	Prof	ça c'est pas à vous de le rechanger
246	14 :29.570	Prof	l'ordinateur il est capable de le faire tout seul
247	14 :32.130	Prof	de trouver de lui-même | ce qu'il va falloir mettre
248	14 :35.850	Prof	dans les boucles
249	14 :37.810	Prof	alors en fait là euh le principe
250	14 :41.250	Prof	général pour trouver la première boucle il suffit de faire quoi
251	14 :48.310	Eleves	bah c'est un chiffre de moins que
252	14 :50.240	Prof	c'est un chiffre de moins
253	14 :52.820	Eleves	que la mesure
254	14 :53.540	Prof	que la mesure
255	14 :54.520	Prof	dans la deuxième boucle c'est quoi ?
256	14 :58.750	Prof	H ?
257	14 :59.310	Eleves	un chiffre de moins que la boucle une
258	15 :02.050	Prof	c'est un de moins que la boucle une
259	15 :04.280	Prof	et est-ce qu'on peut dire par rapport à la mesure
260	15 :10.720	Eleves	c'est deux chiffres en moins
261	15 :11.860	Prof	c'est deux chiffres en dessous que la mesure
262	15 :13.940	Prof	si c'est huit c'est deux de moins ça fait six
263	15 :17.610	Prof	mais ça
264	15 :19.400	Prof	je pense que l'ordinateur est capable de les faire les calculs
265	15 :23.320	Prof	alors je ne sais plus si je vous avais montré quels sont les | les outils qui permettent de faire les calculs | sur snap

Transcription 3.20 – 46 Prof, S5 : faits partagés

Ils vont alors modifier la boucle b_2 de la même façon, $b_2 \leftarrow 15 - 1$, produisant :

$$TS(n)[15] \rightsquigarrow \left(\text{  } \text{} , 47, 15 \right)$$

Cette rétroaction semble être considérée comme valide puisque les élèves vont étendre la modification aux boucles $b3$ à b_5 : $b_{i\in[3..5]} \leftarrow 15 - 1$. Cela ne va pas être testé, mais lorsque les élèves vont commencer la modification de la boucle b_6, ils vont montrer une grande hésitation.
Comme on le voit dans la transcription des actions de programmation 3.1 les élèves vont affecter l'opérateur (◯ - ◯) à b_6, l'enlever, mettre (15 - 1), et hésiter de la même façon pour b_5. Jusqu'à la fin de la séance, le groupe 46i procédera ainsi aux essais suivants :

— $b_{i\neq 7} \leftarrow 15 - 1$,
— $b_{i\neq 7} \leftarrow 14 - 1$,
— $b_{i\neq 7} \leftarrow 16 - 1$,

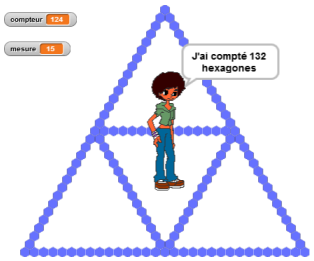
```
7 :26.265   (BOUCLE)b6i0(NEWVAL) : répéter ** fois (commandes)
7 :26.588   (BOUCLE)b6i0(DROPVAL) : répéter *[ − ]* fois (commandes)
7 :27.337   (BOUCLE)b6i0(NEWVAL) : répéter ** fois (commandes)
7 :29.166   STOP receiveKey(478)
7 :29.182   SNP STOP
7 :30.488   (BOUCLE)b5i0(NEWVAL) : répéter ** fois (commandes)
7 :32.304   (BOUCLE)b5i0(DROPVAL) : répéter *[15 − 1]* fois (commandes)
7 :36.153   (BOUCLE)b6i0(DROPVAL) : répéter *[ − ]* fois (commandes)
7 :38.814   (BOUCLE)b6i0(VAL) : répéter [*15* − ] fois (commandes)<<>>
7 :42.729   (BOUCLE)b6i0(VAL) : répéter [15 − *1*] fois (commandes)<<>>
7 :42.800   (BOUCLE)b6i0(NEWVAL) : répéter ** fois (commandes)
7 :44.633   (BOUCLE)b5i0(NEWVAL) : répéter ** fois (commandes)
7 :47.538   (BOUCLE)b5i0(NEWVAL) : répéter *[** − **]* fois (commandes)
7 :49.502   (BOUCLE)b5i0(VAL) : répéter [*15* − ] fois (commandes)<<>>
7 :50.648   (BOUCLE)b5i0(VAL) : répéter [15 − *1*] fois (commandes)<<>>
7 :53.743   (BOUCLE)b6i0(NEWVAL) : répéter *[** − **]* fois (commandes)
7 :56.830   (BOUCLE)b6i0(VAL) : répéter [*15* − ] fois (commandes)<<>>
7 :58.305   (BOUCLE)b6i0(VAL) : répéter [15 − *1*] fois (commandes)<<>>
8 :03.233   KEY_n
8 :03.251   START receiveKey(478)
8 :03.285   ASK <>
8 :05.136   ANSW <<15>>
8 :08.363   FIN receiveKey(478)
8 :08.373   SNP FIN478
8 :27.697   (BOUCLE)b7i0(DROPVAL) : répéter *[ − ]* fois (commandes)
8 :29.949   (BOUCLE)b7i0(VAL) : répéter [*15* − ] fois (commandes)<<>>
8 :30.839   (BOUCLE)b7i0(VAL) : répéter [15 − *1*] fois (commandes)<<>>
8 :34.467   (BOUCLE)b8i0(NEWVAL) : répéter *[** − **]* fois (commandes)
8 :36.781   (BOUCLE)b8i0(VAL) : répéter [*15* − ] fois (commandes)<<>>
8 :37.688   (BOUCLE)b8i0(VAL) : répéter [15 − *1*] fois (commandes)<<>>
8 :39.953   START receiveKey(478)
```

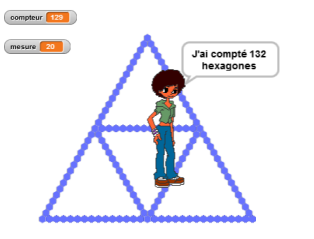
Trace 3.1 – 46i, S5 : Des hésitations

— $b_7 \leftarrow 30 - 1$,

— $b_7 \leftarrow 31 - 1$,

Ce mélange de représentation du Nombre Générique (NG) et perte de généricité (PNG, voir un premier exemple avec le groupe 46g, p. 325) sera davantage exploré dans les parties suivantes. Ils aboutiront, après vingt minutes d'essais, au script $\{16 - 1/15 - 1/31 - 1\}$. Ce script sera considéré comme valide pour une entrée de 15 et invalide pour une entrée de 20, puisque des modifications seront tentées après les deux rétroactions suivantes :

$$TS(n)[15] \rightsquigarrow \left(\text{  } \text{} , 132, 15 \right)$$

$$TS(n)[20] \rightsquigarrow \left(\text{  } \text{} , 132, 20 \right)$$

Une relation avec le compteur (étapes 67-70) Il est probable que, pour le groupe 46i, une des raisons invalidant la dernière rétroaction est que le nombre d'hexagones dénombrés (132) ne bouge pas. En effet, la forme du tracé étant finalement valide, ils vont envisager des modifications de l'initialisation du compteur :

— *compteur* ← 

— *compteur* ← 

Ces modifications, qui ne seront pas testées, laissent penser que les élèves cherchent ici une certaine expression de Snap ! qui permettrait au compteur de s'adapter à un nouveau tracé, renvoyant ainsi à ce qui était envisagé par le groupe 46g en début de séance 3 (voir p. 301).

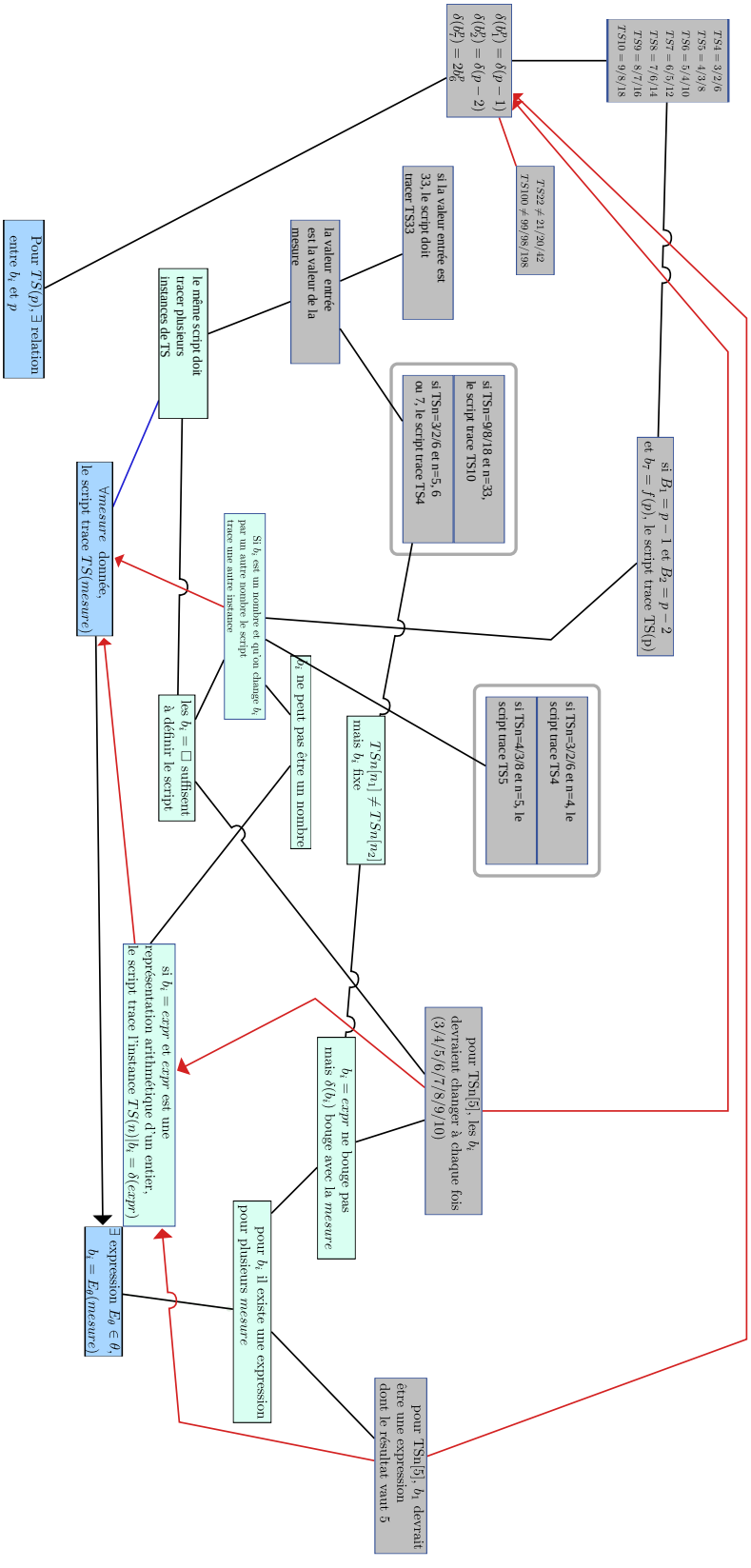


Figure 3.60 – Espace des faits-contraintes - 46g

(a) Premier ajustement $b_7 \leftarrow b_7+2$ **(b)** Deuxième ajustement $b_7 \leftarrow b_7+2$ **(c)** Ajustement $b_8 \leftarrow b_8+1$

Figure 3.62 – Rétroactions suite aux ajustements pour un TS6 (46i, S3)

Recherche de régularités

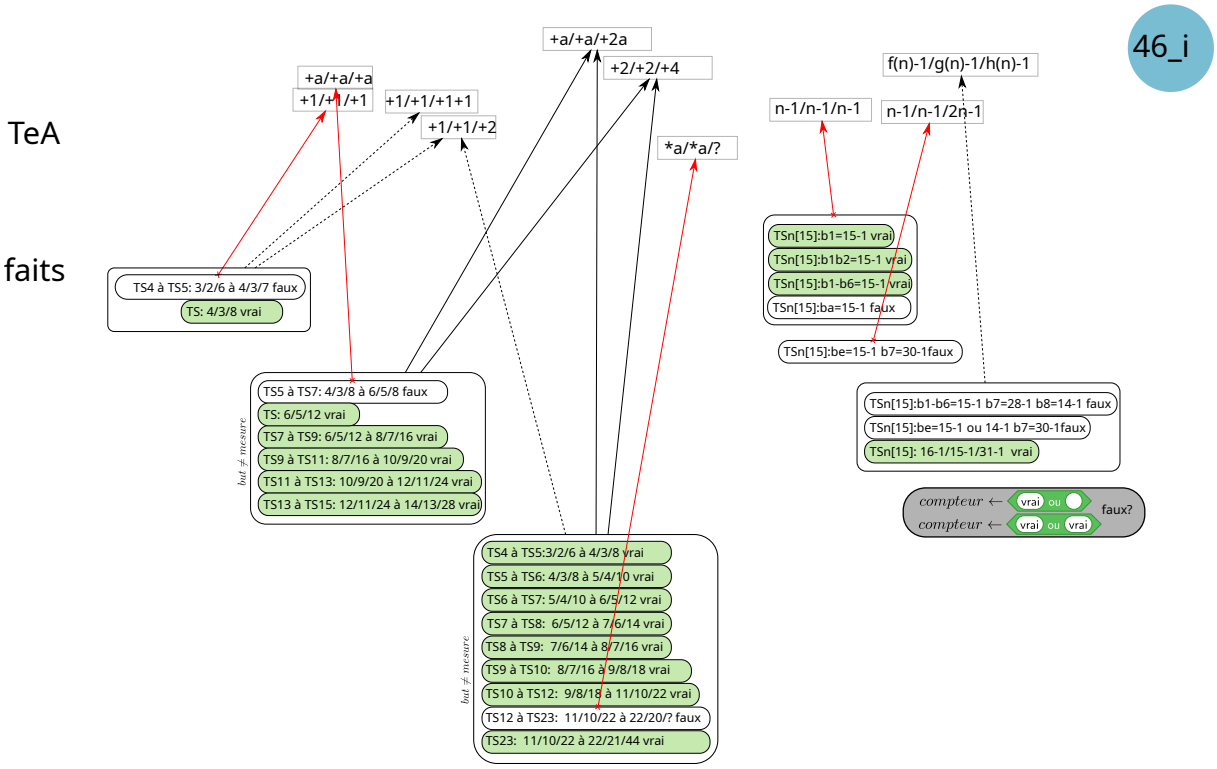


Figure 3.63 – Faits et TEA, groupe 46i

Le groupe 46i a produit de nombreux faits relatifs à la construction de diverses instances. Cependant, les éventuels TEA mis en œuvre ne seront pas mobilisés lors de la construction du script générique : ce sont les faits partagés avec la classe qui vont participer à l'organisation de l'activité de ces élèves.

Un TEA peu courant : $+a/+a/+2a$ En séance 3 et 4, les différentes instances construites semblent être une mise en œuvre de TEA $+a/+a/+2a$:

- $B_1^{p+a} = B_1^p + a$
- $B_2^{p+a} = B_2^p + a$
- $b_7^{p+a} = b_7^p + 2a$

Dans un premier temps, le groupe 46i semble mettre en œuvre une version non stabilisée du TEA $+a/+a/+a$ pour construire le $TS(6)$. Le groupe 45a a mis lui aussi en œuvre ce TEA, comme une extension du TEA $+1/+1/+1$ (voir p. 268). Il s'agit non plus d'ajouter un à chaque boucle, mais d'ajouter une certaine valeur a, dépendant du tracé voulu : $b_i^p = b_i^q + a$. Ici, les élèves semblent considérer que si le tracé doit être celui d'un TS6, le nombre de répétitions de la première boucle doit être de 6. Ensuite, en établissant une relation inter-objectale avec le script dupliqué $TS(4)$, on constate qu'on passe de $b_1^4 = 3$ à $b_1^6 = 6$, c'est-à-dire qu'on ajoute 3 à b_1 : les élèves vont commencer à ajouter 3 à toutes les boucles. Elles appliquent le TEA $+a/+a/+a$ pour $a = 3$ [47]. Ce TEA n'est cependant pas stable [48]. En effet, après un temps de

47. $a = 2$ si les élèves se réfèrent au script $TS(5)$.
48. ou n'est pas celui identifié.

réflexion légèrement plus long que pour les boucles précédentes [49], pour la boucle b_7 l'ajout sera seulement de deux. Une possibilité est que les élèves, suite aux faits construits lors de la création du TS5, s'intéressent à b_7 comme étant un élément problématique — puisqu'ils n'avaient pas trouvé immédiatement la bonne valeur. Ils ont ainsi sans doute comparé les valeurs du nombre de répétitions des boucles b_7 sur les deux scripts déjà existants et valides, ce qui aboutirait à constater que $b_7^5 = b_7^4 + 2$. En se référant alors à la relation entre $TS(4)$ et $TS(5)$, ils appliquent la même relation pour construire les dernières boucles du $TS(6)$ relativement au script $TS(5)$: $b_7^6 = b_7^5 + 2$ et $b_8^6 = b_8^5 + 1$. Nous considérons ici que c'est cependant bien le TEA $+a/+a/+a$ qui est à l'œuvre, car c'est celui qui semble expliquer les prochaines modifications de scripts. En effet, ce TEA va évoluer pour les instances suivantes en sa version valide $+a/+a/+2a$. Les premiers scripts étant construits à partir du $TS(4)$, on peut penser que c'est en mettant en relation ce script, leur TEA et leur but que ces élèves aboutissent aux TS9 et TS11 (46i, S3, 9'37-12'33) :

$$\begin{aligned}
b_1^4 = 3 \wedge b_1^6 = 5 &\implies b_1^{p+a} = b_1^p + a &&\implies b_1^{4+5} = b_1^4 + 5\\
b_2^4 = 2 \wedge b_2^6 = 3 &\implies b_2^{p+a} = b_2^p + a &&\implies b_2^{4+5} = b_2^4 + 5\\
b_7^4 = 6 \wedge b_7^6 = 10 &\implies b_7^{p+a} = b_7^p + 2 \times a &&\implies b_7^{4+5} = b_7^4 + 2 \times 5
\end{aligned}$$

Ce TEA a été rarement observé, seul le groupe 46f semble l'avoir mobilisé à deux reprises :

— pour passer de {6/5/12} à {11/10/20} ($a = 4$)

— pour passer de {9/8/18} à {21/20/42} ($a = 12$)

En revanche, des variantes non valides de ce TEA, impliquant une relation entre des instances de TS non consécutives, sont assez courantes, notamment celles qui étendent « naturellement » le TEA $+1/+1/+1$: $+a/+a/+a$ (groupes 46e, 46a, 46c, 46f, 45a, 45j et 45k) ou $*a/*a/*a$ (46f, 45f, 46m). Le groupe 46e étendra le TEA valide $+1/+1/+2$ en une version invalide $*a/*a/*a+2$ ($b_7^{p\times a} = b_7^p \times a + 2$).
Cependant, il est possible que les élèves appliquent une version moins générique de ce TEA : ils augmentent la mesure de deux à chaque nouveau script, ou plutôt *ils augmentent la valeur du nombre de répétitions de b_1 de deux à chaque script*. La construction et la mise en œvre du TEA seraient dans ce cas :

$$\begin{aligned}
b_1^4 = \boxed{3} \wedge b_1^6 = 5 = \boxed{3} + 2 &\implies b_1^{p+2} = \boxed{b_1^p} + 2 &&\implies b_1^{6+2} = \boxed{b_1^6} + 2\\
b_2^4 = \boxed{2} \wedge b_2^6 = 4 = \boxed{2} + 2 &\implies b_2^{p+2} = \boxed{b_2^p} + 2 &&\implies b_2^{6+2} = \boxed{b_2^6} + 2\\
b_7^4 = \boxed{6} \wedge b_7^6 = 10 = \boxed{6} + 4 &\implies b_7^{p+2} = \boxed{b_7^p} + 4 &&\implies b_7^{6+2} = \boxed{b_7^6} + 4
\end{aligned}$$

En effet, là où les autres groupes construisent des instances consécutives, ce qui correspond à la tâche prescrite aux élèves, le groupe 46i construit des instances de deux en deux : le TS7 puis le TS9, puis le TS11, puis le TS13, puis le TS15. On peut penser que pour ce groupe, le problème qu'ils se posent, leur but, est de construire à chaque fois un TS *plus grand*, de plus grande taille. Or, leur première tentative valide commençait par augmenter le nombre de répétitions de la boucle b_1 de deux. N'ayant pas de raison de faire bouger leur TEA, ni leur but, les élèves vont

49. Presque neuf secondes séparent les modifications de b_6 et b_7, contre moins de cinq secondes pour les autres (46i, S3, 6'24-6'59).

ainsi continuer à le mettre en œuvre. Une conséquence logique serait que ces élèves n'établissent pas de lien avec la valeur de la mesure, se contentant d'augmenter le nombre de répétitions des boucles de 2, ou de 4 pour b_7 ($+2/+2/+4$). Ce lien avec la mesure ne fera son apparition qu'en séance 4, lorsque les élèves tenteront de construire le TS11 comme demandé.
Lors de la séance 4, le groupe 46i commence cette fois par construire des instances consécutives, du TS4 au TS10, puis un saut de deux instances (TS12) suivi du TS23 (46i, étapes 23-32), avec des scripts testés valides. Il nous semble voir dans ces constructions une nouvelle application du TEA $+a/+a/+2a$. En effet, ces élèves dupliquent non pas le dernier script construit — ils seraient ainsi susceptibles de construire leur nouvelle instance à partir de l'instance précédente — mais le script traçant un TS4, ce qui suggère une application de leur TEA fonctionnel $+a/+a/+2a$ pour différentes valeurs de a : $b_i^5 = b_i^{4+1}$, $b_i^6 = b_i^{4+2}$, etc. Il est aussi envisageable de voir une application d'une instanciation de ce TEA pour $a = 1$. Ce TEA est peut-être aussi appliqué pour passer du TS12 au TS23.

Relation intra et/ou inter-objectale Nous avons fait le choix de représenter les TEA en nous basant sur des relations inter-objectales [50], puisque nous cherchons à établir comment les élèves parviennent — ou non — à généraliser un motif, et que ce motif se répète d'instance en instance. Cependant, les élèves du groupe 46i, tout comme ceux du groupe 46g et d'autres, semblent mobiliser aussi des relations intra-objectales. Il est probable que, suivant le but poursuivi et la charge cognitive impliquée par l'action, les élèves mobilisent soit les relations intra-objectales, soit les relations inter-objectales, suivant ce qui est le plus efficace et le moins coûteux.
Ainsi, pour construire une instance $p+a$ à partir de l'instance p existante, les élèves peuvent procéder comme suit (considérant que dans l'écrasante majorité des cas les élèves modifient les boucles dans l'ordre de leur apparition dans le script, éventuellement en différant le traitement de b_7). Dans un premier temps, le nombre de répétitions de b_1^{p+a} est établi en fonction de celui de b_1^p : $b_1^{p+a} = b_1^p + a$ (inter). Dans un deuxième temps, et de la même façon, b_2^{p+a} est établi en fonction de b_2^p : $b_1^{p+a} = b_1^p + a$ (inter). Et ainsi de suite pour les boucles b_4, b_5, b_6, b_7 et b_8, avec la variation $b_7^{p+a} = b_7^p + 2a$. Une autre possibilité serait, une fois b_1^{p+a} déterminée, de n'utiliser que des relations intra-objectales : $b_2^{p+a} = b_1^{p+a} - 1$, $b_{i\in[3..5]}^{p+a} = b_2^{p+a}$ — ou $b_{i\in[3..5]}^{p+a} = b_{i-1}^{p+a}$ —, $b_7^{p+a} = b_6^{p+a} \times 2$, et $b_8^{p+a} = b_6^{p+a} + 1$ ou $b_8^{p+a} = b_{i\in[2..5]}^{p+a}$. Tout mixte de ces possibilités est bien entendu possible, mais si l'activité de l'élève est organisée, il doit y avoir une régularité dans les mobilisations des différentes relations. Ce sont donc les ruptures, les erreurs qui peuvent nous donner des indices de ce qui sous-tend l'activité de l'élève. Ainsi, les erreurs sur les boucles b_6 qui se répercutent sur les boucles b_8 nous indiquent que ce groupe semble faire des relations inter-objectales (sans doute plus simples à traiter) pour les boucles autres que b_1 et b_7. Pour b_7 en revanche, à deux reprises la modification de cette boucle est faite de façon différée, après toutes les autres. Cela pourrait être un signe d'une relation différente, donc inter-objectale. Mais ce peut aussi être le signe d'une difficulté cognitive plus importante : le traitement des boucles autres que b_7 en intra-objectal est simple (égalité ou différence de un), celui de b_7 est plus complexe. Il s'agit dans ce cas, soit d'ajouter quatre lorsqu'on passe de l'instance p à l'instance $p+2$ (ce qui ne devrait pas poser de difficulté particulière à un élève de 4ème), soit d'ajouter par exemple 18 (après avoir établi que $13-4=9$ et $2\times 9=18$) pour le passage de l'instance 4 à l'instance 13, soit encore de multiplier par deux la valeur de b_6, donc effectuer par exemple 12×2 pour construire le TS13. La deuxième possibilité semble, pour ces cas, la plus coûteuse cognitivement parlant, c'est donc peut-être celle-ci qui est — ou était — à l'œuvre lorsque le groupe 46i a construit les instances des TS13 et TS15.

50. Voir p. 146.

Un TEA mis en tension La séance 5, lorsque les élèves sont confrontés à la nécessité d'une généralisation algébrique, viendra mettre en tension leur TEA avec les faits partagés par l'enseignante en début de séance, établissant une relation entre la mesure et la valeur du nombre de répétitions des boucles (« on fait moins un », par exemple). Ce fait, non construit par eux, sera difficilement mobilisable. C'est peut-être cela qui empêchera les élèves de s'appuyer sur les relations intra-objectales qui semblent établies. En début de séance, ce groupe affectera la même expression pour les boucles b_1 et b_2, ou b_5 et b_6, alors que des relations intra-objectales, impliquant une différence entre ces boucles, ont été constatées en séance 3 et 4. On verra néanmoins que dans cette phase de généralisation, la relation intra-objectale de b_7 semble mobilisée.

Généralisation algébrique

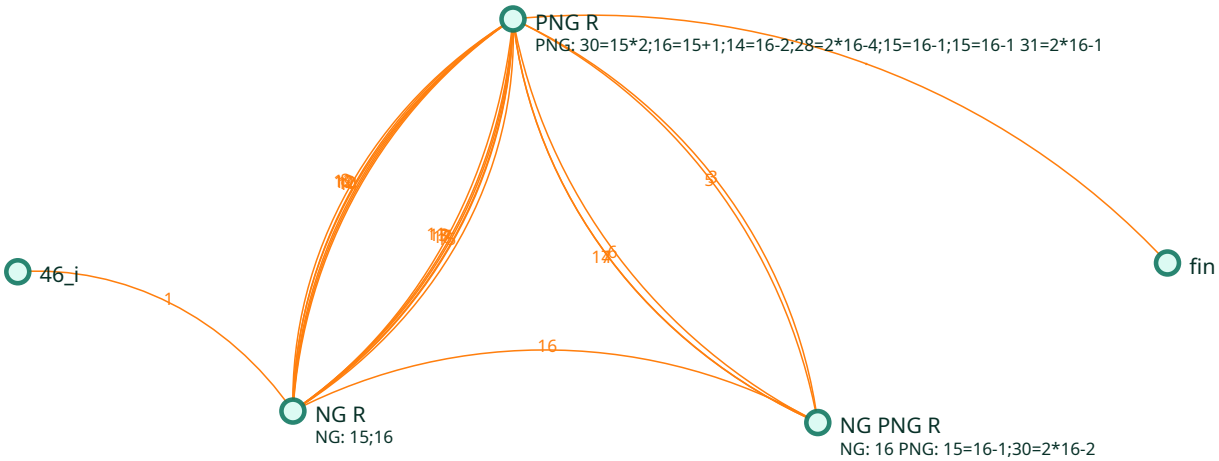


Figure 3.64 – Type de généralisation - 46i

Comme on le voit sur la figure 3.64 (p. 343), lors de la phase de généralisation, le groupe 46i passe et repasse par différentes représentations de la généricité. Ils vont ainsi, après avoir hésité assez longuement à définir $b_5 = 15 - 1$ et $b_6 = 15 - 1$, ou non, lancer une première exécution après avoir modifié les six premières boucles, et sans toucher aux deux dernières dans un premier temps. (script 3.7, p. 344). La rétroaction obtenue est la suivante :

$$TS(n)[15] \rightsquigarrow \left(\text{  } \text{} , 94, 15 \right)$$

Cette rétroaction, malgré la présence d'un trou, semble être considérée comme une validation de

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 6 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 2 fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

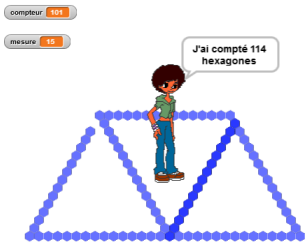
(a) Partielle

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [15 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [15 − 1] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(b) Totale

Script 3.7 – Première tentative de généralisation : NG-1 (46i, S5)

leurs actions, puisqu'ils vont compléter le script jusqu'à ce que toutes les boucles voient l'expression déterminant leur nombre de répétitions fixée à (15 - 1), obtenant (script 3.7b, p. 344) :

$$TS(n)[15] \rightsquigarrow \left(\text{  } \text{} , 114, 15 \right)$$

La mesure identifiée, partiellement Contrairement à ce qu'il s'est passé dans les autres séances, les élèves font ici une relation entre la mesure, la valeur entrée par l'utilisateur, et l'expression dénotant le nombre de répétitions des boucles. En effet, la valeur entrée, 15, est en cohérence avec l'expression de la première boucle, $15 - 1$. Les actions précédentes des élèves ne permettent pas d'identifier la raison de cette soudaine prise en compte de la mesure, nous supposons donc que celle-ci est issue des faits partagés en début de séance :

— « le principe général pour trouver la première boucle », « c'est un chiffre de moins que la mesure » (3.20, 249-254)

— si je rentre la valeur 8, je veux « celui de mesure huit » (46 Prof, S5, 13'01-13'18)

— « l'ordinateur se débrouille tout seul », il « devrait lui-même trouver ce qu'il faut mettre ici [en b_1] » (46 Prof, S5, 13'40-14'04)

— « mais ça [l'expression des boucles] c'est pas à vous de le rechanger »

On peut noter que les élèves appliquent *stricto sensu* ce qui leur a été indiqué :

— L'ordinateur trouve lui-même la valeur à mettre ($14 = 15 - 1$),

— c'est bien un hexagone de moins que la mesure voulue.

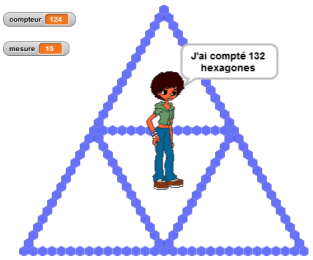
Cependant, on peut se demander si les élèves associent bien la mesure entrée et le nombre d'hexagones sur les côtés du triangle de base. En effet, les rétroactions de leurs deux premiers scripts montrent des côtés formés de 16 hexagones, et non 15, mais cela ne semble pas être considéré comme un critère d'invalidité. Le script final obtenu, dont le tracé est valide (script 3.8, p. 347) est d'ailleurs considéré comme valide par les élèves alors qu'il trace un TS16 pour une mesure entrée de 15. La mesure définit la boucle, mais ne parait pas être, pour ces élèves, une propriété du TS tracé. À ce stade, le paramètre *mesure* ne peut être que plus difficilement construit.

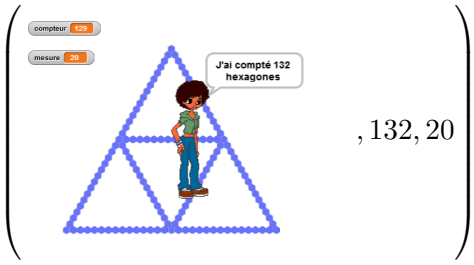
Le TEA revisité, un doublement de b_6, ou doubler un tracé Le tracé produit va amener les élèves à remplacer $b_7 = 15 - 1$ par $b_7 = 30 - 1$. Le nombre 15 a donc été multiplié par deux. Trois explications de cette action sont possibles :

1. dans leur TEA $+a/ + a/ + 2a$, il y a la notion de double qui apparait pour b_7 ;
2. les élèves font une relation intra-objectale, entre b_6 et b_7 (la seconde étant le double de la première), la notion de double apparait encore ;
3. visiblement b_7 ne trace qu'un côté du triangle de base, il faudrait tracer deux côtés, d'où de nouveau la notion de double.

Le lien entre mesure et boucle entrant en tension avec le TEA tel que formulé, nous écartons la première possibilité : le résultat produit par les élèves n'est certainement pas une mise en œuvre de leur TEA, même adapté à une situation nouvelle. Plus tard, les élèves vont laisser $b_6 = 15 - 1$, mettre $b_8 = 14-1$ et ajuster la rétroaction erronée en faisant $b_7 \leftarrow 28-1$. Cela nous semble indiquer que ce n'est donc pas b_6 qui est doublée (ou partiellement doublée), mais bien la longueur d'un côté.

Prévalence de la relation Durant toute la séance, seul le premier paramètre de l'opérateur (- 1) sera modifié, quelle que soit la boucle. Tout se passe comme si c'était cette relation (quelque chose moins un) qui permettait à l'ordinateur de se débrouiller tout seul, et qu'il fallait seulement trouver les bonnes valeurs à entrer pour le premier paramètre (c'est-à-dire celles permettant d'obtenir un tracé bien formé, sans trou ni chevauchement). Ceci est confirmé par leur activité en fin séance. En effet, après avoir finalement abouti à un script dont le tracé est valide (script 3.8, p. 347), les élèves vont invalider leur script en entrant une valeur de 20 :

$$TS(n)[15] \rightsquigarrow \left(\text{  } \text{} , 132, 15 \right)$$

$TS(n)[20] \rightsquigarrow$ 

$, 132, 20$

Mais au lieu de mettre en question les valeurs entrées comme paramètres de la relation 

, le groupe 46i va considérer que l'erreur vient du compteur (dont la valeur finale aurait dû changer). Pour ces élèves, les paramètres de la relation sont valides, puisque l'ordinateur se débrouille tout seul pour faire un tracé valide, et si la rétroaction est considérée comme erronée alors que le tracé est valide, cela doit provenir d'un autre aspect du script. On retrouve ici le compteur mobilisé comme caractéristique du tracé, comme pour le groupe 46g (p. 319).

Nombres génériques et perte de généricité : NG et PNG Le script obtenu par le groupe 46i en fin de séance (script 3.8, p. 347) est emblématique de la perte de généricité, qui rend moins probable l'identification de motifs, et donc une formulation générique avec un paramètre.
La mesure souhaitée est 15, les boucles b_2 à b_5, ainsi que la boucle b_8, utilisent ce 15 comme exemple générique et le rendent visible : $\boxed{15} - 1$. En rendant visible ce nombre dans toutes les boucles, le TEA $n - 1/n - 2/2n - 2$ aurait eu plus de probabilité d'être construit :

- $b_1 = \boxed{15} - 1$
- $b_2 = \boxed{15} - 2$
- $b_7 = \boxed{15} \times 2 - 2$ ou $b_7 = 2 \times (\boxed{15} - 1)$

Mais ce groupe, comme d'autres, a fait le choix de conserver la relation au détriment des paramètres : les autres boucles voient leur nombre de répétitions représenté par $16 - 1$ pour b_1 et b_6, ou $31 - 1$ pour b_7. Les nombres 16 et 31 font disparaitre le lien avec la mesure, le lien avec le nombre générique n'est plus visible, il y a ainsi une perte de généricité que nous nommons PNG (Perte du Nombre Générique). Une autre occurrence de ce PNG a déjà évoquée pour le groupe 46g (p. 325), et cette perte de généricité a été constatée dans de nombreux autres groupes[51]. Cet élément étant ainsi très fréquent, nous l'étudierons plus précisément ultérieurement (Entropie et types de généralisation, p. 457).

51. Parmi les groupes qui ont abordé la généralisation, ce PNG n'a pas été constaté sur six d'entre eux, et constaté sur douze.

```
Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [ 16 - 1 ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ 15 - 1 ] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [ 15 - 1 ] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [ 15 - 1 ] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [ 15 - 1 ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ 16 - 1 ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ 31 - 1 ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ 15 - 1 ] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || compteur |]] ||
hexagones |]]
```

Script 3.8 – Un script valide, qui aurait pu être générique (46i, S5)

Construction des nécessités

Le groupe 46i, malgré de nombreuses constructions de faits liés aux différentes instances créées, a finalement construit peu des nécessités du problème[52].
L'existence d'une relation entre la mesure entrée et les boucles n'est pas une construction, mais la mobilisation d'un fait partagé par le groupe classe. La relation liant la mesure entrée et la boucle b_1 est, elle aussi, un fait partagé : comme nous l'avons vu plus haut, les élèves du groupe 46i ne semblent pas prendre en compte la mesure attendue du TS pour concevoir leurs scripts, se contentant d'ajouter un certaine valeur au nombre de répétitions de la boucle b_1 en fonction du script sur lesquels ils se basent (par exemple en ajoutant deux à chaque fois).
En outre, les élèves semblent bien avoir construit la nécessité qu'un même script doit tracer plusieurs instances, ce qui implique de régler la tension « il faut changer les valeurs des b_i »-« l'expression donnant b_i ne bouge pas ». Ils utilisent ainsi un opérateur, qui fait que l'ordinateur doit « se débrouiller tout seul » pour trouver la bonne valeur.
Cependant, une nécessité importante du problème ne semble pas construite : lorsque la mesure entrée change, l'instance change, mais pour ces élèves, « l'instance change » ne semble pas liée au nombre d'hexagones par côté (la propriété *mesure* du TS) mais plutôt au nombre d'hexagones total d'un TS[53]. La mesure entrée définit le TS, c'est-à-dire le nombre d'hexagones tracés. Ainsi, pour ces élèves, si une expression doit concerner plusieurs mesures, elle n'est pas directement

52. Dans la figure 3.65, p. 352, nous représentons par des traits discontinus les enchainements logiques ou les tensions *qui auraient dû se faire*.
53. Nombre donné par la valeur finale de la variable *compteur*.

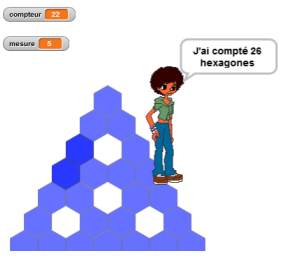
en lien avec le tracé effectif, mais plutôt avec une valeur attendue : si *compteur* = 33 à la fin du tracé, c'est que le tracé correspond au TS5 — alors que *compteur* = 33 *parce que le TS5 est tracé*. À ce stade, le groupe 46i ne peut donc construire que de façon incomplète, voire inexacte, la nécessité d'une expression dans Snap ! liant les boucles et la mesure. Par suite, la construction du paramètre n'est pas encore accessible.

Bilan

Ce groupe a rapidement été capable de construire différentes instances sans erreurs. Cependant, les TEA qui organisent leur activité lors de ces constructions semblent devenir des obstacles à une généralisation. En effet, la recherche, hésitante et quelque peu désordonnée, d'une relation permettant d'établir par calcul la valeur du nombre de répétitions d'une boucle, montre que ce groupe ne mobilise pas les relations pourtant établies lors de la construction des différentes instances, notamment les liens d'égalité ou de différence entre les familles de boucles. La prévalence de l'opérateur (- 1) sur toute forme de relation intra ou inter-objectale laisse penser que les aspects sémiotiques l'emportent ici sur les aspects arithmétiques ou les relations entre boucle et tracé. De plus, l'absence d'identification de la valeur de la mesure avec le nombre d'hexagones tracés sur les côtés des triangles de base empêche la création des nécessités aboutissant à la construction du paramètre.

3.4.3 Synthèse Groupement 3

Les groupes de ce groupement, comme le précédent, n'ont pas mobilisé le paramètre, mais ils montrent une exploration plus variée de la généralisation.
Du point de vue des rétroactions et de leur interprétation, le groupe 46g montre une particularité : les élèves de ce groupe ne prennent pas en compte le chevauchement comme un signe d'erreur, ou plutôt, ils ne considèrent pas qu'une couleur plus foncée pour un hexagone transmet une information invalidant le script. Cette non prise en compte du chevauchement sera présente y compris lors de la phase de généralisation du script.
Cela semble lié à leur difficulté à poser le problème : ces élèves restent sur le problème consistant à compléter les scripts proposés, c'est-à-dire à construire autant d'instances de TS que de scripts. Le problème de la généralisation ne se posera que grâce à l'intervention de l'enseignante, mais cela n'empêchera pas l'une des élèves, en séance 5, de persister à considérer que le but est de construire des formes valides, la rétroaction suivante étant ponctuée fièrement d'un « on a réussi » (46g, transcription S5,9'13) :

$$TS(n)[5] \rightsquigarrow \left(\text{  } \text{} , 26, 5 \right)$$

On peut ainsi se questionner sur le nombre de groupes qui ont produit de nombreuses instances avant de se poser le problème de la généralisation. L'intervention de l'enseignante a-t-elle été nécessaire, et les élèves voient-ils la place de ce problème au sein du problème d'optimisation donné au départ : trouver le plus grand TS traçable avec 1400, 1654 ou 1710 hexagones ? Le groupe 45k va d'ailleurs tenter de construire directement un TS1400, témoignant d'un autre problème posé.

Le compteur, comme la mesure, semble être considérés par ces groupes comme des propriétés essentielles du tracé, et comme critère de validation. Comme dans le groupement précédent, le groupe 46g envisage la possibilité que modifier l'initialisation du compteur ferait changer le tracé, qui s'adapterait en quelque sorte au nombre d'hexagones « fournis ». La cohérence compteur-mesure ne semble cependant plus prise en compte lors de la généralisation du script ; le problème n'est plus de construire des tracés bien formés, mais un script qui permet plusieurs tracés.
Les TEA permettant de construire les (nombreuses) instances produites par ces groupes, sont assez stables : $+1/+1/+2$ pour les groupes 46g et 45k, et son extension valide $+a/+a/+2a$ pour le groupe 46i. Cependant, le groupe 46b a construit les instances du TS5 au TS10 en appliquant le TEA $+1/+1/+1$, avant de les corriger, et les groupes 46g et 45k ré-appliqueront ce même TEA lors de la généralisation du script[54] : là encore, des faits construits dans le cadre d'un problème — construire des instances successives — n'ont pas été mobilisés dans un autre problème — construire le script générique.
Concernant la généralisation, deux de ces groupes ont montré une généralisation algébrique naïve, construisant directement le TS100 (46g) ou le TS1400 puis le TS11 (45k). Le groupe 46g a bien identifié le lien entre la mesure et la valeur de chacune des boucles (relation trans-objectale), et le lien entre la boucle $b7$ et la boucle b_6 (ou b_1) : $b_7 = 2 \times b_6$ (relation intra-objectale), en obtenant un script valide. Le groupe 45k quant à lui semble identifier le lien entre la mesure et la valeur des boucles des familles B_1 et B_2, mais n'associe pas la mesure et la valeur entrée : pour la construction d'un TS11, les valeurs entrées seront de 19, 24, 14, 23... Le groupe 46i semble lui être capable de construire n'importe quelle instance, mais toujours à partir d'une autre.
La généralisation du script, quant à elle, montrera des explorations de θ similaires au groupement précédent, avec quelques variations susceptibles de les amener vers la généralisation algébrique symbolique (Radford, 2003, p. 56).
Les groupes 46g et 45k vont envisager des instructions résolvant le problème :

— 

(pour les deux groupes) ;

— ou 

(pour le groupe 45k).

Le reste des explorations concernera des relations permettant de déterminer la valeur du nombre de répétitions des boucles en fonction de la mesure.
On retrouve ainsi des nombres potentiellement génériques (NPG), par exemple dans le groupe 46b (étapes 47-48), qui va tester $b_1 = 56 \times 7$, ou $b_1 = 3 \times 89$ et $b_2 = 2 \times 798$, ou encore $b_1 = 256$. Ces NPG sont ici encore, comme pour le groupe 45a du groupement 2, des exemples d'instanciation de signes génériques ou nombre-signes génériques : les élèves passent de :

— $b_1 =$ 

(testé plusieurs fois) à $b_1 = 56 \times 7$, instanciant un SG par un NPG ;

— $b_1 = 3 \times []$ et $b_2 = 2 \times []$ à $b_1 = 3 \times 1$ et $b_2 = 2 \times 1$, transformant un SG en NSG ;

— $b_1 = 3 \times 1$ et $b_2 = 2 \times 1$ à $b_1 = 3 \times 89$ et $b_2 = 2 \times 798$, instanciant un NSG par un NPG.

— $b_1 = 5 - 1$ à $b_1 = 5 - 66666666666$, pour le groupe 45l, instanciant aussi un NSG par un NPG.

Des NPG puissances de dix seront aussi expérimentées par ce groupe pour ces mêmes boucles : $b_1 =$ 

ou $b_1 =$ 

ou encore $b_1 =$ 

. Les « 100 » comme les 1 semblent ici porteur d'une forme de généricité que l'ordinateur devrait interpréter, il ne s'agit visiblement pas d'une opération permettant de déterminer un nombre de répétitions (d'autant plus que ces élèves font ces tests sur une instance de TS5).

54. Le groupe 46l cherchera à construire les TS5, TS21 et TS47, mais ne produira aucun script valide.

Trois de ces groupes vont utiliser un nombre générique dans une relation permettant de déterminer la valeur du nombre de répétitions des boucles en fonction de la mesure — ce qui n'était pas le cas dans le groupement 2. Ainsi, on retrouvera les expressions $b_1 = 15 - 1$ pour une entrée de 15, $b_1 = 5 - 1$ et $b_2 = 5 - 2$ pour une entrée de 5. Le groupe 45k testera plusieurs exemples ($b_1 = 45 - 1$ pour une entrée de 45, $b_1 = 28 - 1$ pour une entrée de 28 etc, étapes 80-85), soulignant par là ce caractère exemplaire : « si la mesure est 45 tu fais 45-1 ».
Les deux autres groupes vont envisager une nouvelle solution pour permettre à l'ordinateur de « se débrouiller tout seul » : faire varier les nombres au sein même de l'instance. Ainsi le groupe 46g envisagera $b_1 = 3$, $b_2 = 4$, $b_3 = 5$ etc, et le groupe 46l testera $b_1 = 11 - 1$, $b_2 = 16 - 6$, $b_3 = 14 - 4$ etc (scripts 3.9, p. 350). Ces solutions à la tension fixe-variable entre le script unique et les instances

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 4 fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter 5 fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter 6 fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter 7 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 8 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 9 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 10 fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(a) Sans relations (46g, S5)

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [11 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [16 − 6] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [14 − 4] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [19 − 9] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [12 − 2] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [13 − 3] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [17 − 7] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [18 − 8] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(b) Avec relations (46l, S5)

Script 3.9 – Généralisation par NGV

multiples, semblent se baser sur l'idée de fournir des faits à l'ordinateur pour qu'il en déduise les opérations à faire : il faut changer chacune des boucles lorsqu'on crée une nouvelle instance.
Le script 3.9b (p. 350) du groupe 46l a une spécificité originale : le dénoté de chacune des boucles est égal à 10, qui se trouve être la valeur de la mesure entrée (épisodes 31-36). Le nombre « 10 » est donc peut-être ici un nombre à caractère générique : si on remplaçait l'entrée par une certaine valeur n, pour chaque boucle b_i il s'agirait de trouver un nombre x_i et de remplacer l'expression de b_i en prenant comme premier opérande le dénoté de $n + x_i$ et comme deuxième opérande le dénoté de x_i.

Le NG « 10 » perdrait ainsi de sa généricité, il ne serait plus visible dans les écritures, mais implicite. On perd ici le nombre générique (PNG). Cette perte se retrouve aussi chez le groupe 46i : si le groupe 46b construit un script $\{5-1/5-2/5+3\}$ pour une entrée de 5, le groupe 46i construit le script $\{16-1/15-1/31-1\}$ [55]. Ces groupes établissent bien une relation entre la mesure et la boucle, et ces deux scripts sont considérés comme valides. Cependant, là où le groupe 46b laisse visible le NG, établissant à chaque fois un rapport visible entre ce NG et la boucle, le NG 16 du groupe 46i, visible dans l'expression de b_1, est perdu pour les autres boucles, qui n'évoquent que implicitement le lien avec ce NG : « 15-1 » c'est en fait « $[16-1]-1$ », et « 31-1 » c'est en fait « $[16\times 2-1]-1$ ». Ces élèves considèrent que la relation « moins un » est fixe, et que le premier opérande doit s'adapter. Cette perte de généricité, que l'on retrouvera dans les groupements suivants, est un obstacle à l'identification d'une représentation dans θ qui aurait comme dénoté le NG choisi. Notons que, si l'on remplace le NG par la mesure, l'expression de b_7 serait valide pour le groupe 46i quelle que soit la mesure ($b_7^n = 2n-2$), alors qu'elle ne serait valide que pour la mesure 5 pour le groupe 46b ($b_7^{n=5} = n+3$, mais $b_7^{n\neq 5} \neq n+3$). Ces deux groupes sont confrontés à deux obstacles différents sur la voie du paramètre.

55. Ce groupe commence par $b_1 = 15-1$, et tous ses tests se feront avec une entrée de 15, mais les corrections successives amèneront $b_1 = 16-1$, mais toujours avec une entrée de 15. Les élèves ici semblent perdre le lien avec la mesure, puisqu'ils construisent un TS16 et non plus un TS15, mais leur objectif est de construire un tracé valide avec ces expressions de relation.

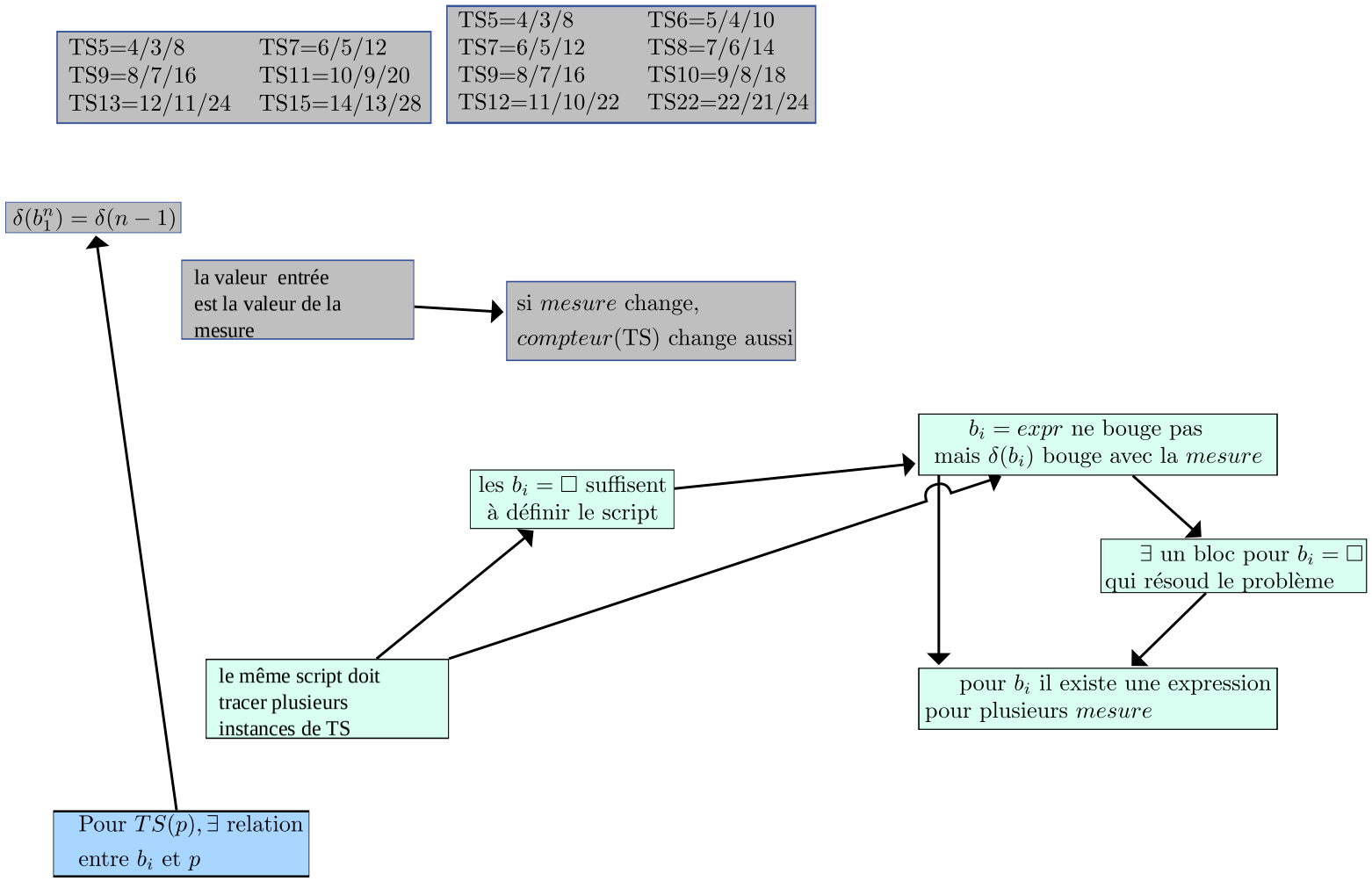


Figure 3.65 – Espace des faits-contraintes - 46i

3.5 Groupement 4

3.5.1 Groupe 45e

Le groupe 45e a eu des difficultés à construire les premières instances de scripts, mais ces élèves semblent avoir réussi à établir les relations entre le nombre de répétitions des boucles et la mesure, y compris sur la boucle b_7. C'est le premier groupe étudié qui a utilisé mesure, mais sans doute pas comme paramètre.

Description de l'activité

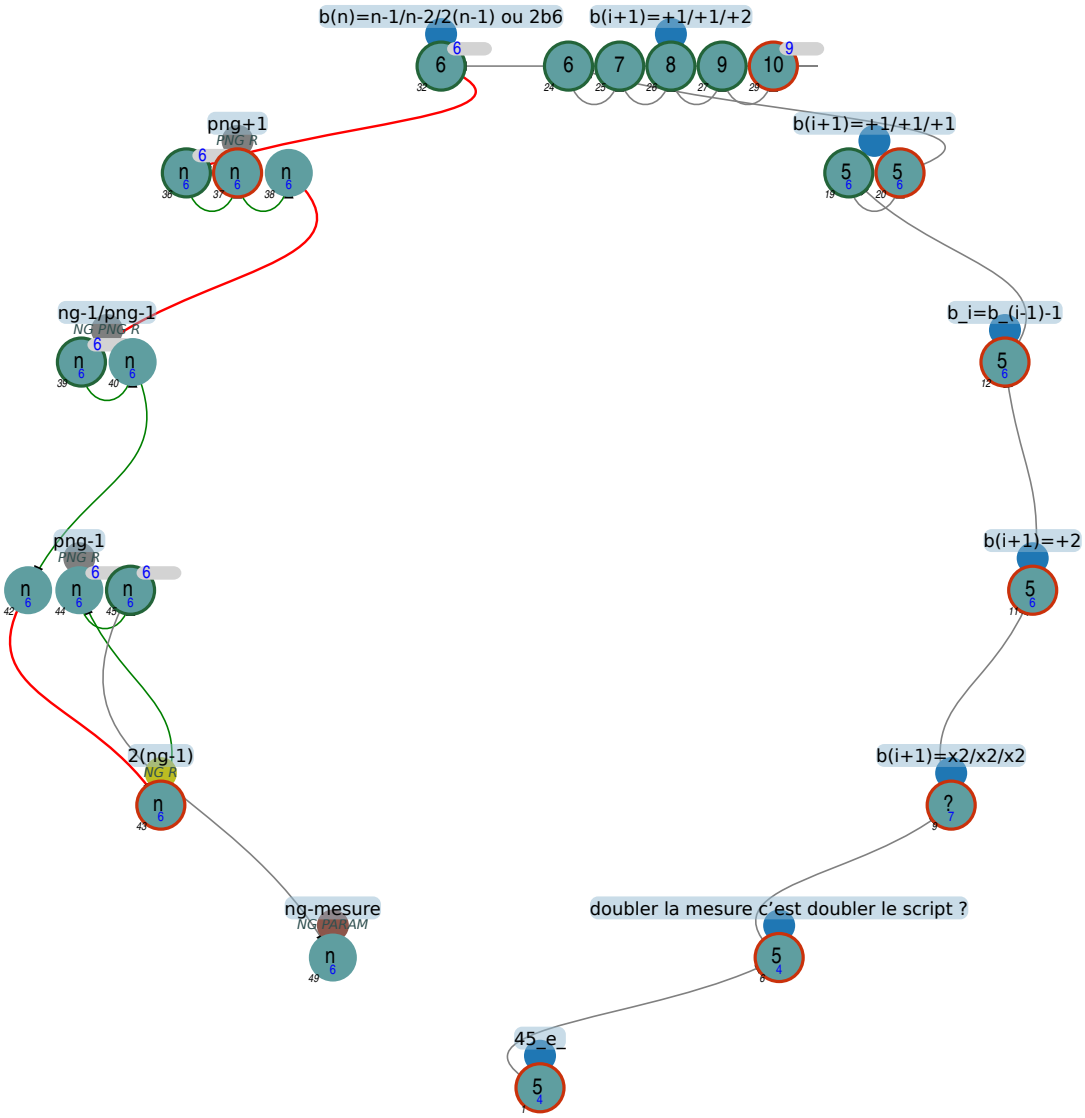
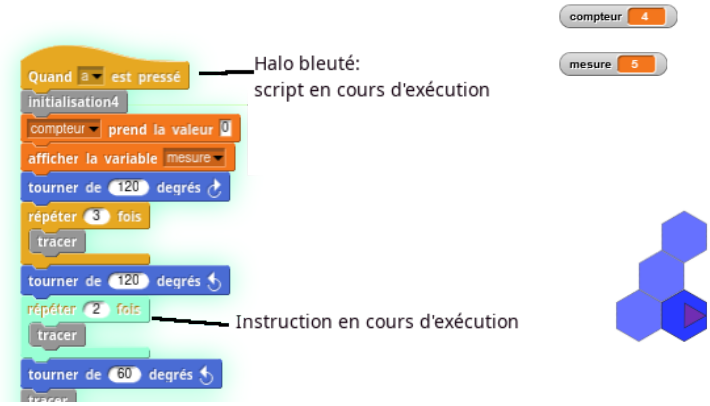


Figure 3.66 – Organisation de l'activité - 45e

Le groupe 45e mettra plus d'une séance à construire le script $TS(5)$, mais une fois cette tâche accomplie, ils construiront rapidement plusieurs instances successives avant de s'atteler au problème de la généralisation.

Construction d'une première instance Le groupe 45e va commencer classiquement par dupliquer le script traçant le TS4, comme demandé. Ces élèves vont alors lancer le script une première fois, puis une seconde fois en le mettant en pause à plusieurs reprises (45e, S3, 1'37-3'28). Ils semblent ainsi s'intéresser au fonctionnement de l'algorithme, et ces pauses leur permettent de faire le lien entre tracé et instructions. En effet, comme nous l'avons vu précédemment, une mise en valeur du script et des instructions est faite lors d'une pause ou d'un déroulement pas-à-pas (figure 3.67, p. 354). Ils vont ensuite de nouveau dupliquer le script $TS(4)$ et vont ajouter cette



(a) $TS(4)$ en pause sur la boucle b_2 **(b)** Espace d'exécution correspondant à la pause

Figure 3.67 – Utilisation de la pause pour comprendre le script

copie à leur script en cours : ils « doublent » en quelque sorte le script. La rétroaction étant manifestement le signe d'un script invalide (45e, S3, 6'03), ces élèves vont supprimer la partie en trop et la déplacer sous l'entête du script devant tracer un TS6, script qui sera alors testé.
Le groupe 45e va alors modifier les boucles en construisant le script $TS(5) = \{8/6/12\}$ (étape 9) aboutissant à la rétroaction (45e, S3, 8'48) :

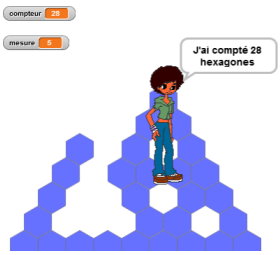
$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 46, 5 \right)$$

Ils passent ainsi :

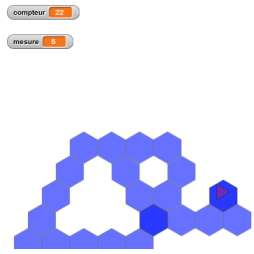
- de $b_1^4 = 3$ à $b_1^5 = 6$,
- de $b_2^4 = 2$ à $b_2^5 = 4$,
- ...
- de $b_7^4 = 6$ à $b_7^5 = 12$,
- et de $b_8^4 = 2$ à $b_8^5 = 4$.

Après avoir doublé le script, ces élèves doublent les valeurs du nombre de répétitions des boucles. Le tracé est invalide, tant par sa forme que par la mesure.

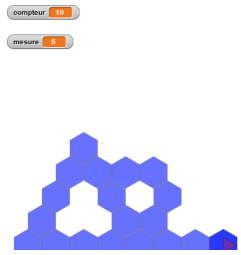
Les élèves vont revenir à une version correcte du script, en supprimant leur script et en re-dupliquant le script traçant le TS4. Ils effectueront alors les actions $b_1^5 \leftarrow 5$ et $b_2^5 \leftarrow 4$ (étape 11). On peut penser qu'ils associent ici b_1 à la valeur de la mesure, et comme ils ont augmenté b_1^4 de deux, ils font de même pour b_1^5. Leur test renvoie (45e, S3, 12'37) :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , 28, 5 \right)$$

Cette rétroaction semble valider leurs modifications, puisqu'ils vont ensuite modifier $b_3^5 \leftarrow 3$, obtenant ainsi, après une pause :

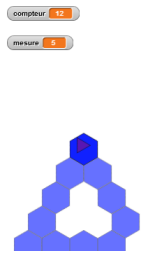
$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , [], 5 \right)$$

Ils vont ensuite tester divers ajustements (étapes 13-14), ne leur permettant pas d'obtenir une rétroaction valide en fin de séance :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , [], 5 \right)$$

.

En début de séance 4, le groupe 45e va reprendre son script et le diviser : les élèves vont déplacer les instructions suivant la boucle b_3 (script 3.10, p. 356). Cette façon de procéder est typique de la pensée informatique : on décompose le problème en problèmes plus simples. En modifiant de nouveau $b_3 \leftarrow 3$, les élèves obtiendront une rétroaction qui leur paraitra valide :

$$TS(5)[] \rightsquigarrow \left(\text{  } \text{} , [], 5 \right)$$

En effet, celle-ci montre un triangle de base qui est celui attendu, avec une mesure de cinq. Les élèves du groupe 45e vont alors rajouter la partie de script coupée, et la modifier en accord avec ce qui leur parait valide. Comme ils passent de $b_1^4 = 3$ à $b_1^5 = 5$, de $b_2^4 = 2$ à $b_2^5 = 3$, et de $b_3^4 = 2$ à $b_3^5 = 3$, ils vont propager ces modifications à b_4, b_5, et b_6. Ainsi, ils effectueront $b_4 \leftarrow 3$, $b_5 \leftarrow 3$,

Block_478
Quand b est pressé
initialisation5
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter 5 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 3 fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter 2 fois (commandes)
.....tracer

Script 3.10 – 45e, S4 : Diviser pour régner

puis $b_6^5 = 5$ (étape 20). Ils semblent montrer ainsi qu'ils ont identifié les familles de boucles B_1, B_2 et B_7 (dont le traitement est ici repoussé). Ainsi, la construction de b_6^p dépend de b_1^p et non de b_6^{p-1}, une relation intra-bjectale est ici établie entre les boucles d'une même instance. La rétroaction invalide amènera le groupe à faire plusieurs ajustements avant d'aboutir à une version valide du script $TS(5)$ (étapes 21-23). Leur but, tracer un TS5, était identifié à la boucle b_1, mais cette dernière version du script devrait invalider cette identification.

Construction d'instances successives (étapes 24-30) Le groupe 46e va alors construire les instances traçant les TS6, TS7, TS8, TS9 et TS10. Ils feront une seule erreur pour la boucle b_8 du TS10, rapidement corrigée. Les quatre premières instances seront construites en environ cinq minutes (45e, S4, 5'35-10'24). Après une inactivité (du moins en terme d'actions sur l'EPGB) de quelques minutes, le TS10 sera construit sur le script censé être générique. Ils exécuteront ce script avec une valeur entrée de 9, ce qui semble montrer qu'à ce stade, pour ces élèves, leur but est identifié à la première boucle : ici $b_1 = 9$ et ils pensent construire un TS9.

Ajustement du but En séance 5, les élèves vont modifier le script générique pour qu'il trace un TS6, et cette fois-ci leur but ne semble plus assimilé à la première boucle : ils construisent le script {5/4/10} (étape 32) et le testent avec une valeur entrée de 6 (45e, S5, 1'29), ce qui est bien la mesure du TS tracé. Il est probable que cet ajustement provienne des faits partagés par la classe en début de séance.

Vers la généralisation (étapes 33-49) Une fois validée, cette instance sera de nouveau testée, mais cette fois avec une valeur entrée de 15, produisant la rétroaction suivante (45e, S5, 1'31) :

$$TS(n)[15] \rightsquigarrow \left(\text{  } \text{} , 42, 15 \right)$$

Les élèves semblent alors confrontés à la tension fixe-variable entre l'unicité du script et la multiplicité des instances qu'il peut tracer, et passeront le reste de la séance à chercher à généraliser leur instance. Ils testeront $b_1 =$ (5 + 1) (que nous noterons par la suite $b_1 = 5 + 1$), après avoir envisagé $b_1 = 4+1$. Après avoir envisagé $b_2 = 4+1$, les élèves du groupe 45e affecteront les expressions (6 - 1) (respectivement (5 - 1)) aux boucles de la famille B_1 (respectivement B_2), ce qui est valide pour une mesure entrée de 6. Après avoir envisagé $b_7 = 11 - 1$, les élèves testent leur script avec $b_7 =$ (6-1 × 2). Ce script produira le résultat suivant (45e, S5, 26'24), qui invalidera (à tort) leur action :

$$TS(n)[6] \rightsquigarrow \left(\text{  } \text{} , 44, 6 \right)$$

L'affectation $b_7 \leftarrow$ (11 - 1) renverra, elle, tous les signes d'une validité du script. Les élèves relanceront alors le script, en choisissant une mesure de 12 (45e, S5, 27'16) :

$$TS(n)[12] \rightsquigarrow \left(\text{  } \text{} , 42, 12 \right)$$

Ils considéreront cette rétroaction comme invalidant leur script, puisqu'ils termineront la séance en envisageant $b_1 \leftarrow$ (6 - mesure) (étape 49, 45e, S5, 29'36) à la place de $b_1 =$ (6 - 1), mais n'auront pas l'occasion de tester le script obtenu.

Recherche de régularités

Le groupe 45e a mis en œuvre divers TEA avant d'appliquer avec succès le TEA $+1/+1/+2$ permettant de passer d'une instance à la suivante. Contrairement à beaucoup de groupes, ces élèves semblent avoir identifié la relation entre b_7 et la valeur de la mesure, mais un faux-fait construit sera sans doute un obstacle à la généralisation.

Un TEA lié à l'agrandissement ? Le groupe 45e va commencer par mettre en œuvre des variations du TEA $+1/+1/+1$. Ils vont ainsi doubler la valeur du nombre de répétitions de toutes les boucles, passant d'un TS4 à un TS7, ce qui est possiblement une mise en œuvre du TEA noté $*a/*a/*a$ pour $a = 2$, soit de façon plus formelle $b_i^{p \times a} = b_i^p \times a$. Ensuite, après avoir identifié leur but (la mesure du TS à construire) à la boucle b_1, ces élèves vont envisager d'augmenter le nombre de répétitions des boucles b_1 et b_2 de deux, soit une manifestation du TEA $+a/+a/+a$ pour $a = 2$ ($b_i^{p+a} = b_i^p + a$). Ces TEA font penser aux TEA liés à l'agrandissement de figures. Ainsi, $+a/+a/+a$ évoque la croyance selon laquelle quand on agrandit un côté d'une figure, passant d'une longueur l_1 à une longueur l_2, tous les côtés seront agrandis en leur

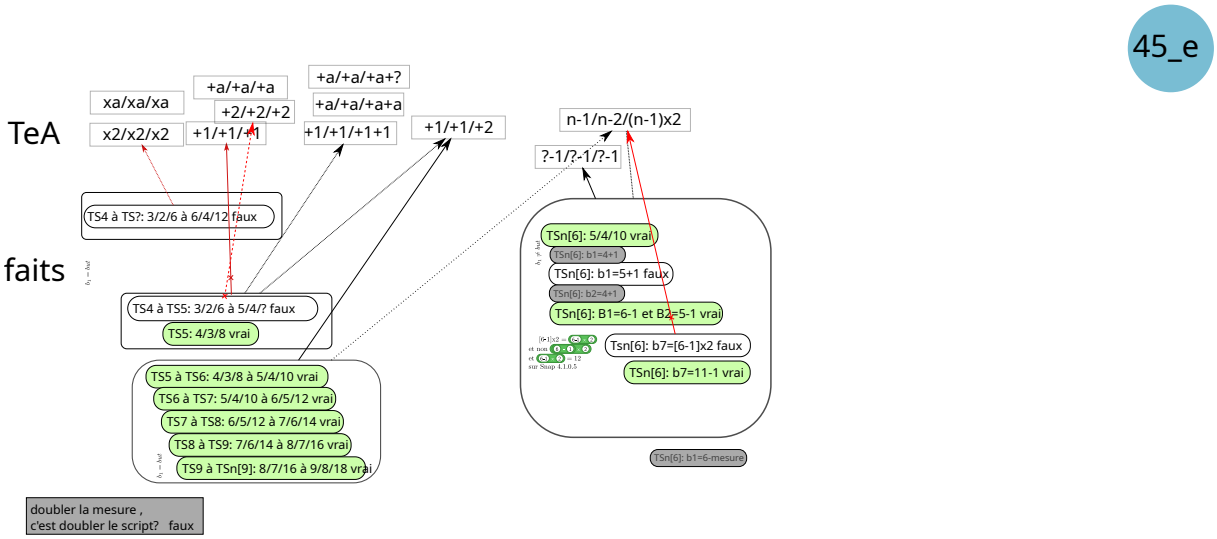


Figure 3.68 – Faits et TEA, groupe 45e

ajoutant la même longueur d telle que $l_2 = l_1 + d$ [56]. Le version $*a/*a/*a$ est un TEA valide en cas d'agrandissement de figure, mais il n'est pas adapté ici, puisque ce sont les valeurs du nombre de répétitions des boucles qui changent, et il n'y a pas de relation de proportionnalité entre les mêmes boucles d'instances différentes.Ces TEA ont aussi été mis en œuvre par d'autres groupes :

— pour $+a/+a/+a$, les groupes 46e (étape 9), 46i (étape 7), 46k (étapes 16-17), 46f (étapes 18-21), 45j (étapes 8, 16, 42) ou encore 45m (étape 12) ;

— pour $*a/*a/*a$ (et variantes), les groupes 46e (étape 11, avec $b_7^{i+a} = b_7^i + 2a$), 46i (étape 31), 46m (étape 22), 46f (étape 9), ou 45l (étape 12).

Ces TEA sont rapidement invalidés. Selon Zazkis et Liljedahl (2002b, p. 380), cette méthode basée sur la proportionnalité a déjà été constatée dans des généralisations de motifs linéaires (c'est-à-dire dont l'instance n est déterminable par l'expression $an + b$), chez des élèves de 8 à 13 ans — à partir d'un motif géométrique (Stacey, 1989) — comme chez de futurs enseignants d'école primaire (Zazkis et Liljedahl, 2002a)— à partir d'un motif uniquement arithmétique —.

Construction du TEA $+1/+1/+2$ Après la mise en œuvre du TEA $+2/+2/+2$ en fin de séance 3, pour les boucles b_1 et b_2, les élèves du groupe 45e se retrouvent avec le script $S = \{5/4/2/2/2/3/6/2\}$. On peut représenter la suite des modifications successives apportées par le groupe 46e jusqu'à aboutir au script valide du $TS(5)$ de la façon suivante — des modifications simultanées [57] étant placées dans une même colonne, la séparation marquant le changement de séance :

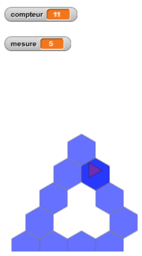
56. Voir la situation du Puzzle de Brousseau (1981, p. 69).
57. Par simultanées on entend faisant partie de la même étape, c'est-à-dire des modifications non séparées par des tests.

$$
\begin{array}{rcl ccc|cccccc rcl}
b_1^5 = & b_1^4 + 2 & & & & & & & & & -1 & & b_1^5 = & b_1^4 + 1 \\
b_2^5 = & b_2^4 + 2 & & & -1 & & & & & & & & b_2^5 = & b_2^4 + 1 \\
b_3^5 = & b_3^4 & & +1 & -1 & & +1 & & & & & & b_3^5 = & b_3^4 + 1 \\
b_4^5 = & b_4^4 & & & & & & +1 & & & & & b_4^5 = & b_4^4 + 1 \\
b_5^5 = & b_5^4 & & & & & & +1 & & & & \Longrightarrow & b_5^5 = & b_5^4 + 1 \\
b_6^5 = & b_6^4 & & & & +1 & & +1 & -1 & & & & b_6^5 = & b_6^4 + 1 \\
b_7^5 = & b_7^4 & & & & & & & & +2 & & & b_7^5 = & b_7^4 + 2 \\
b_8^5 = & b_8^4 & & & & & & & +1 & & & & b_8^5 = & b_8^4 + 1
\end{array}
$$

Comment, à partir de ces faits premiers, les élèves peuvent-ils construire le TEA $+1/+1/+2$ qu'ils mobiliseront ensuite sur plusieurs instances ? La succession de modifications n'est sans doute pas mémorisée, et l'EPGB ne garde pas trace de ces modifications : comment serait-il possible que les élèves, sur deux séances de surcroit, arrivent à en déduire les relations entre les boucles de l'instance 4 et celles de l'instance 5 ?

La modification de S en $TS(5)$ valide se fait de façon progressive. Si l'on ne considère que les modifications faites en séance 4, en faisant l'hypothèse raisonnable que les élèves n'ont ni mémorisé ni noté les modifications faites cinq jours plus tôt, les élèves commencent par diviser le script en deux, afin de ne tester que la première partie (script 3.10, p. 356). Ce script, à l'exécution, produit la rétroaction suivante :

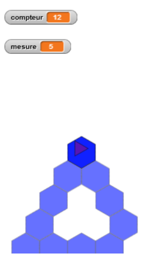
$S[] \rightsquigarrow \Big($



$, [], 5 \Big)$

Les élèves modifient alors $b_3 \leftarrow 3$ (soit une augmentation d'une itération) : ils cherchent à « terminer » le tracé du premier triangle de base, dont le sommet du haut semble manqué. C'est une vision erronée, puisqu'un hexagone supplémentaire sera tracé, il y aura donc deux hexagones tracés sur le sommet de ce premier triangle. Cependant, cela n'est pas rendu visible par la nouvelle rétroaction obtenue :

$S[] \rightsquigarrow \Big($



$, [], 5 \Big)$

En effet, à la place d'un hexagone surnuméraire représenté par un bleu foncé, les élèves voient ici le curseur que nous avons estimé utile de représenter. La rétroaction est donc considérée comme valide, le premier triangle de base semble être tracé, les élèves rassemblent alors les deux morceaux de script et répercutent la modification faite sur la dernière boucle testée : le nombre de répétitions

de la boucle b_3 a été augmenté de un, il sera fait de même pour les boucles suivantes, sauf b_7.

$$S[] \rightsquigarrow \left(\text{  } \text{} , 32, 5 \right)$$

Cette fois-ci, l'hexagone surnuméraire au sommet du premier triangle de base est visible, mais les élèves, peut-être parce qu'ils considèrent cette construction comme acquise, vont tenter de corriger d'autres erreurs visibles. Les élèves semblent identifier le tracé de la boucle b_6 comme étant trop long de un hexagone, et ils diminuent cette boucle d'une itération. Faisant cela, il est possible que les élèves projettent mentalement ce que devrait faire b_8 *si le côté tracé par* b_7 *était complet*, et identifient alors qu'une itération supplémentaire doit être faite (figure 3.69, p. 361). Si notre hypothèse est exacte, cela signifierait que :

— les élèves ont identifié les côtés tracés par chacune des boucles ;

— les élèves considèrent que si une boucle est répétée p fois, le tracé correspondant sera un côté de p hexagones ;

— les élèves ont identifié le tracé de la boucle b_7, et décalent son traitement car c'est une boucle particulière.

Ainsi, cette correction peut laisser penser que ces élèves sont dans un registre explicatif algorithmique, parce qu'ils identifient le sens des instructions et parce qu'ils mettent en œuvre un aspect de la pensée informatique en décomposant, de nouveau, le problème. Cette compréhension de l'algorithme étant sans doute issue des explorations du déroulement du script en séance 3.
La rétroaction semble confirmer leurs actions, puisqu'ils s'attellent alors à modifier, de manière valide, b_7. Ils obtiennent ainsi :

$$S[] \rightsquigarrow \left(\text{  } \text{} , 34, 5 \right)$$

Comme nous l'avons déjà évoqué dans la partie Rétroaction dynamique et différée (p. 179), cette rétroaction finale peut être produite par une erreur sur b_8 ou une erreur sur b_1. Le tracé dynamique, ou les rétroactions précédentes, semblent avoir permis aux élèves d'identifier la source de l'erreur : b_1 fait une itération de trop.
Ainsi, ces élèves ont possiblement abouti au TEA $+1/+1/+2$ non parce qu'ils ont établi des rapports entre des nombres, mais parce qu'ils ont identifié le sens des instructions : ils savent quelle boucle trace quoi. Cette connaissance sera confirmée lors de la construction du script générique, qui montrera que ces élèves ont bien identifié le sens de la boucle b_7. Lors de cet épisode, les élèves semblent avoir construit des faits tiers, raisons des erreurs et réussites liées à leurs actions. Non seulement la localisation de l'erreur est faite, mais sa cause est elle aussi identifiée.

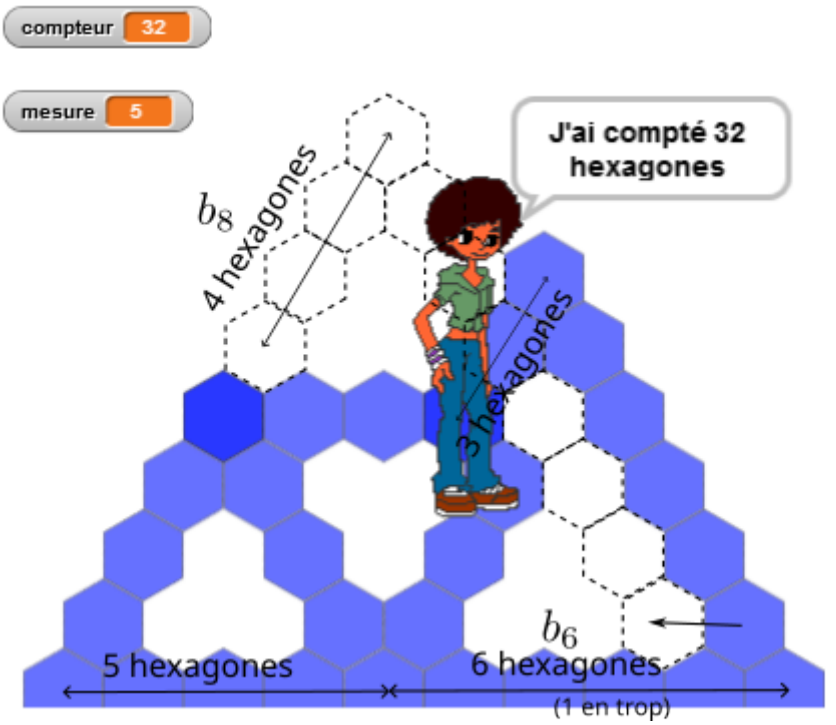


Figure 3.69 – Une explication possible de modifications (45e, S4)

Construction d'un faux fait Lors de la phase de construction du script générique, les élèves commencent par construire directement le script permettant de construire un TS6, en modifiant d'abord les valeurs du nombre de répétitions des familles de boucles B_1 et B_2, puis en traitant de nouveau b_7 à part. Cette suite d'actions valide prendra moins d'une minute (45e, S5, 0'49-1'24), ce qui semble confirmer la bonne compréhension de l'algorithme. Ils vont ensuite tenter de généraliser ce script (voir la partie suivante). Après quelques modifications, mais peu de tests — ce qui est un indice que les élèves anticipent et mobilisent leurs connaissances —, ils aboutissent au script 3.11a (p. 362). Si l'on omet b_8 qui n'a pas encore été modifiée, ce script $\{6-1/5-1/11-1\}$, non testé, est valide pour une mesure de 6. En effet, $\delta($(6 - 1)$) = 5$, $\delta($(5 - 1)$) = 4$, et $\delta($(11 - 1)$) = 10$; or $\{5/4/10\}$ est le script permettant de tracer un TS6. Les élèves semblent ne pas se satisfaire de l'expression de b_7, qu'ils transformeront en $b_7 \leftarrow$ (6-1 × 2) (script 3.11b, p. 362). Considérant que $\delta((6-1)\times 2) = 10$, le script devrait être valide, or la rétroaction infirme ce fait :

$$TS(n)[6] \rightsquigarrow \left(\text{  }, 44, 6 \right)$$

La démarche des élèves est valide, malheureusement dans la version de Snap ! utilisée pour le dispositif, $\delta($(6-1 × 2)$) \neq \delta((6-1)\times 2)$ car $\delta($(6-1)$) = 6$! Nous développerons ce point, fortement lié aux registres de représentations sémiotiques, dans la partie suivante.

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [6 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [5 − 1] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [5 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [5 − 1] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [5 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [6 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [11 − *1*] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 4 fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(a) NG et PNG

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [6 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [5 − 1] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [5 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [5 − 1] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [5 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [6 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [6-1 × 2] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [5 − 1] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(b) NG et faux-fait

Script 3.11 – 45e, S5 : Premières généralisations (NG)

Généralisation algébrique

Pour généraliser, le groupe 46e va partir d'une instance qui servira d'instance générique. Ils mobiliseront des nombres génériques (NG) mais aussi des nombres ayant perdu leur généricité (PNG), et aboutiront à l'utilisation de la variable mesure, mais sans doute pas dans un sens de paramètre.

Nombre générique et perte de généricité Ainsi, comme vu ci-dessus, les élèves de ce groupe vont partir d'une version instanciée du script générique traçant un TS6. Ils vont ensuite modifier les expressions du nombre de répétitions des boucles en utilisant un opérateur, pour que « l'ordinateur se débrouille pour faire les calculs ». Ils commencent par envisager $b_1 \leftarrow 4 - 1$[58] : ici, il y a une première recherche d'une expression arithmétique dont le résultat est la valeur du nombre attendu de répétitions de b_1 (5). Mais pour cette instance, la mesure est de 6, et ils semblent à ce moment de nouveau considérer que la valeur du nombre de répétitions de la boucle b_1 est identique à la mesure entrée, donc remplace leur expression par $b_1 \leftarrow 5 + 1$ — donc une expression dont le résultat est égal à la valeur de la mesure. La rétroaction invalide le script, et les élèves envisagent de modifier aussi la deuxième boucle, tout en s'assurant qu'elle reste valide relativement à la première : ils effectuent donc $b_2 \leftarrow 4 + 1$, respectant bien $\delta(b_2) = \delta(b_1 - 1)$, ce qui semble être un fait construit par ces élèves. Ils ne testent pas cette version du script,

58. Autre écriture de $b_1 \leftarrow$ (4 + 1), et différente de $b_1 \leftarrow 5$.

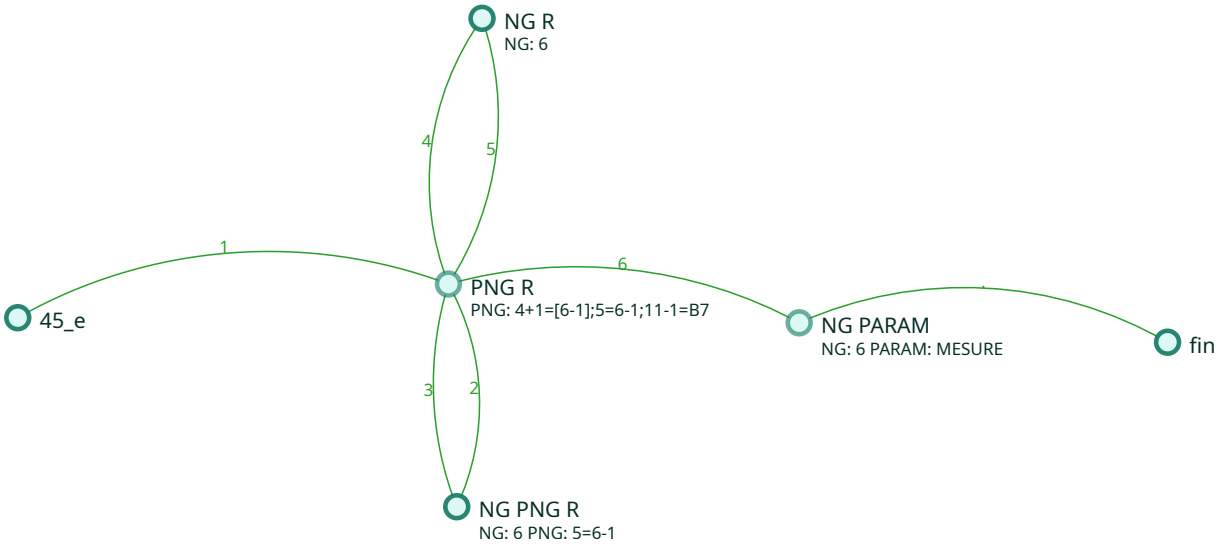


Figure 3.70 – Type de généralisation - 45e

l'invalidant peut-être par un retour du lien entre b_1 et la mesure : « moins un », fait possiblement construit par les élèves, et partagé en début de séance. Ils remplaceront l'expression de b_2 par $5-1$, puis feront de même pour b_3. Ainsi, tout se passe comme si le nombre 5 prenait un caractère d'exemple : on lui ajoute quelque chose pour b_1, on lui enlève quelque chose pour b_2. On pourrait écrire $b_1 = \square + 1$ et $b_2 = \square - 1$. Encore une fois sans nécessiter de lancer une exécution, les élèves du groupe 45e vont corriger b_1, sans doute invalidant leurs hypothèses en mettant en relation les faits premiers $b_1 = 5 + 1 = 6$, $b_2 = 5 - 1 = 4$, $4 \neq 6 - 1$ et le fait tiers (ou la nécessité locale) $b_2 = b_1 - 1$. Ils modifient alors $b_1 \leftarrow 6 - 1$. Il est difficile de dire si ce nombre 6 est un nombre générique, correspondant à la valeur de la mesure. Il peut aussi être le nombre qui, si on lui enlève un, donne un nombre supérieur de un à la valeur du nombre de répétitions de la boucle b_2 (soit le nombre p tel que $p - 1 = (5 - 1) + 1$). Un test sera fait avec cette version du script, et cela validera leurs actions. Ces élèves vont alors propager leurs modifications sur l'ensemble des boucles des familles B_1 et B_2, en entrant dans un premier temps $b_7 \leftarrow 11 - 1$.
On voit ici que, même si 6 est un nombre générique, on perd la généricité pour les boucles B_2 et B_7 : $b_2 = 5 - 1 = [6 - 1] - 1$ et $b_7 = [6 \times 2 - 1] - 1$: le lien entre l'expression du nombre de répétition des boucles B_2 et la mesure n'est plus visible, et de même pour b_7. Le nombre 5 et le nombre 11 sont ici considérés comme des PNG, des nombres ayant perdu la généricité du NG 6.

Une question sémiotique : l'expression de b_7 L'expression testée par les élèves pour la boucle b_7, qui parait justifiée d'un point de vue arithmétique, se trouve invalidée pour une question sémiotique : $\delta($(6-1) × (2)$) \neq \delta((6-1) \times 2)$ car $\delta($(6-1)$) = 6$. L'expression utilisée par les élèves pour exprimer le nombre de répétitions de b_7 pour une mesure de 6 ne dénote pas dans Snap ! ce que les élèves pense être dénoté. Pour eux, $\delta($(6-1) × (2)$) = 10$, mais dans Snap !, $\delta($(6-1) × (2)$) = 12$. On se retrouve ici avec un souci sémiotique, mais ce souci n'a pas pour origine un manque de connaissances des élèves, mais un fonctionnement non attendu, et problématique de l'EPGB.

En effet, dans les opérateurs de Snap !, seules les expressions correspondant à l'écriture d'un nombre en chiffres sont possibles, mais la version utilisée [59] utilise une vérification du type erronée. Un test est fait sur les caractères entrés, mais il n'y a pas de vérification faite pour déterminer s'il s'agit d'une écriture valide de nombre. Ainsi, comme les chiffres, le caractère « . » et le caractère « - » sont autorisés pour pouvoir écrire des décimaux — car « -2 » ou « 1.05 » sont des écritures valides—, la chaine « 6-1 » est considérée comme valide, de même que « –.-8– » par exemple. Ce point est corrigé dans la version actuelle [60].
D'un point de vue plus formel, la grammaire du langage simplifié — c'est-à-dire ici en ne tenant compte que des expressions de nombres et non de combinaisons avec des variables ou d'autres opérateurs— L_{v4} accepté par notre version de Snap ! concernant les paramètres des opérateurs est la suivante [61] :

<Chiffre> : := 0|1|2|3|4|5|6|7|8|9
<Caractere> : := []|<Chiffre>|"."|"-"
<Nombre> : := <Caractere>|<Caractere><Nombre>

Pour la dernière version de Snap !, la grammaire (simplifiée) du langage L_{v8} pour les paramètres des opérateurs est :

<Chiffre> : := 0|1|2|3|4|5|6|7|8|9
<Signe> : := []|"-"
<Entier> : := <Chiffre>|<Chiffre><Entier>
<Nombre> : := <Signe><Entier>["."<Entier>]["e"["-"]<Entier>]

Ainsi :

— "123" $\in L_{v4}$ et "123" $\in L_{v8}$
— " $-$ 123.45" $\in L_{v4}$ et " $-$ 123.45" $\in L_{v8}$
— mais "2. $-$ 5.. $-$ " $\in L_{v4}$ et "2. $-$ 5.. $-$ " $\notin L_{v8}$ [62]

L'expression « 6-1 » dans L_{v4} est donc bien acceptée comme une écriture qui correspond à un nombre (et non comme l'écriture d'une opération arithmétique). En interne, l'évaluation de cette expression est faite par la fonction javascript `parseFloat()`, qui prend en entrée une chaine de caractères et renvoie, s'il existe, le *premier nombre reconnu* de la chaine de caractère. Ainsi, `parseFloat("6-1")` renvoie le nombre 6, tout comme `parseFloat("6suividautres85caractères")`. Il n'y a pas pour cette fonction d'exigence que la totalité de la chaine de caractères représente un nombre, par exemple « 1584621 », et le contrôle de typage à l'entrée d'un paramètre des opérateurs est insuffisant pour écarter des écritures qui ne sont pas des écritures de nombres entiers ou décimaux [63].
Ainsi, les élèves auraient dû utiliser l'expression (6 - 1) × 2, et si le contrôle du typage avait été plus rigoureux — c'est-à-dire si, comme dans la version 8.2.5, on n'aurait pas pu rentrer autre chose que l'écriture d'un nombre —, c'est sans doute ce qu'ils auraient été amenés à faire. Ici, ils ont utilisé une expression qui, si on accepte le mélange d'écritures arithmétiques et d'expressions de θ, est interprétable comme une solution valide. Mais (6-1), dans θ, équivaut sémantiquement à (6). L'EPGB est à l'origine de la construction d'un faux-fait, d'un fait qui *aurait dû être vrai*. Cette expression invalidée, les élèves vont revenir sur leur première idée, (11 - 1), qui perd en généricité.

59. Snap ! 4.1.0.5
60. Snap ! 8.2.3 au 22/06/2023.
61. Nous ne reprenons pas ici les conventions d'écritures utilisées dans (Drouhard et Panizza, 2012) pour exprimer une grammaire générative, leur préférant une convention typique de l'informatique (BNF).
62. et dans la version 4.1.0.5 de Snap !, δ((2.-5..) + (1)) = 3.
63. Notons que dans la dernière version étudiée (8.2.3), la notation scientifique est acceptée, ainsi « 1e2 » est reconnu comme 100, mais l'unicité du caractère "e" n'est pas vérifiée, ni l'unicité du signe "-" qui suivrait cette lettre. La chaine « 1.eee2e3-e52e5–e5 » est donc acceptée (et renverra le nombre 1).

Cette expression que les élèves ont testé pour b_7 montre trois points importants :

— les élèves mettent en relation le nombre d'itérations et la mesure (ici par l'intermédiaire d'un nombre générique, 6) ;

— les élèves semblent avoir compris le sens de la boucle b_7 et ce qu'elle trace *ou/et* ils ont établi la relation entre b_7 et b_1 (ou b_6) ;

— les changements de registre de représentation sémiotique, lorsqu'un registre admet des écritures d'un autre registre, mais sans leur donner le même sens, sont un obstacle à la généralisation

En effet, le nombre 6 est de nouveau mobilisé dans l'écriture, et correspond à la valeur de la mesure entrée. S'il s'agissait juste de trouver une expression donnant dix [64], il est plus probable que les élèves auraient envisagé 5×2, fait numérique bien connu et immédiatement accessible. Le choix d'utiliser ce nombre 6 est intentionnel et prend un caractère d'exemple, donc de nombre générique. De plus, cette expression peut signifier que les élèves sont capables d'établir une relation entre la boucle b_7 et ce qu'elle trace (figure 3.71, p. 366) : il s'agit de tracer un côté du triangle de base moins un hexagone, et ce deux fois. Dans ce cas, les élèves « 6-1 » désigne l'un des traits noirs de la figure 3.71 (p. 366) : il y a une relation de référence à l'objet tracé par la boucle, mais aussi une relation de référence à la mesure.

> La relation de référence d'un signe ou d'une combinaison de signes à un objet résulte d'une opération discursive de désignation. C'est de cette manière que les lettres en algèbre, les mots, ou les constructions syntagmatiques de mots dans un énoncé, réfèrent à un objet.(Duval, 2006a, p. 52)

L'expression « $(6-1) \times 2$ » représente ainsi à la fois la valeur du nombre de répétitions que doit faire la boucle b_7 lorsque la mesure est 6, et le fait que ce qui est tracé par $6-1$ répétitions est un côté d'un triangle de base (qui est lui-même représenté par 6 hexagones, le 6 étant la propriété nommée « mesure » du TS) moins un hexagone qu'il ne faut pas tracer. Les élèves manipulent ainsi des représentations sémiotiques désignant un même objet, mais dans différents registres : algorithmique (l'expression entrée comme paramètre du bloc 

, et ce que fait cette boucle), géométrique (le tracé effectif et ce qu'il représente par rapport à la mesure), et algébrique (puisque mettant en relation une propriété de l'objet, la mesure, représentée de façon générique, avec le nombre de répétions des boucles).
Il est aussi possible (et les deux hypothèses ne sont pas exclusives) que les élèves aient identifié la relation entre b_7 et b_6 (ou b_1) : le nombre de répétitions de la boucle b_7 est le double de celui de la boucle b_6. Dans ce cas, les élèves auraient substitué une représentation rhétorique de b_6 par une représentation symbolique désignant b_6, et utilisée dans une expression. On passe de « on fait le double de ce que fait la boucle b_6 » — expression rhétorique — à « on fait le double de l'expression qui permet de déterminer le nombre de répétitions de b_6 », soit « le double de $6-1$ » — expression syncopée —, puis à « on fait $(6-1) \times 2$ » — expression symbolique.
Dans les deux cas, on voit que les élèves sont en cours d'élaboration d'un registre de représentation algébrique. S'ils avaient disposé des faits « $b_1 = \boxed{6} - 1$ est valide » et « $b_7 = (\boxed{6} - 1) \times 2$ est valide », la mobilisation du fait partagé avec la classe (ou construit par les élèves) précisant que 

contient la représentation de la valeur entrée par l'utilisateur, aurait pu aboutir à la représentation

64. La valeur attendue du nombre de répétitions pour b_8^6.

$b_7 =$ (mesure - 1) × 2. La nécessaire existence d'une expression de θ reliant les boucles et la mesure est établie, et cela aurait pu conduire à la nécessaire existence d'une représentation de la mesure dans θ. Malheureusement, le faux-fait invalidant l'expression des élèves pour b_7 aboutira à un nouvel éloignement de la généricité avec le retour de l'expression $b_7 = 11 - 1$.
L'expression arithmétique (« 6-1 »), est possible comme paramètre dans un opérateur de Snap !, mais ce à quoi elle réfère, c'est-à-dire sa dénotation, n'est pas identique à ce à quoi réfère (6-1) dans Snap ! La possibilité de mélanger une expression du registre de représentation sémiotique de Snap ! ((○ × 2)) et une expression arithmétique non interprétée comme telle, conduit à un faux-fait qui est un obstacle à la généralisation, puisque obligeant ici les élèves à utiliser une expression dans laquelle la généricité du nombre 6 n'apparait plus.

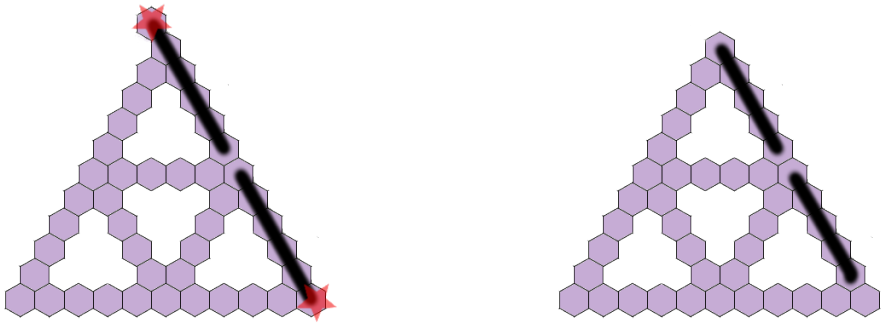


Figure 3.71 – Représentation du sens de $b_7 = (n - 1) \times 2$

Ce qui ne change pas : la relation Les élèves obtiennent un script valide pour une certaine valeur. Les expressions dénotant le nombre de répétitions des boucles sont toutes de la forme $\square - 1$, mais les valeurs représentées par $\square$ ne sont pas les mêmes suivant les boucles (6,5 ou 11). Ainsi, ce qui semble être générique, ce qui ne bouge pas suivant les boucles — c'est-à-dire pour une même instance —, c'est la relation $\square - 1$: si elle est générique, elle doit donc bouger si la mesure bouge — si l'instance change —, c'est sans doute pourquoi les élèves vont envisager $b_1 \leftarrow$ (6 - mesure). C'est une façon d'exprimer un schéma général : pour une certaine mesure « $b_1 = 6 -$ quelque chose qui a un certain rapport avec la mesure » et pour une autre mesure « $b_1 = 6 -$ autre chose qui a un même rapport avec la mesure ». Ici, (mesure), pour les élèves, n'est pas une représentation d'un objet dénotant la mesure de l'instance. C'est un bloc, une expression de θ qui est censée être interprétée par l'ordinateur comme la marque de ce qu'il doit changer en fonction de l'instance.

Construction des nécessités

Lorsque les élèves entament le problème de la généralisation, ils commencent par établir que lorsque la valeur entrée change (avec une entrée de 6, puis une de 15), le tracé devrait changer. Comme le même script doit tracer plusieurs instances, et que la valeur entrée définit la mesure du TS tracer, la nécessité que ce script doit tracer le TS demandé semble bien construite. Cela implique notamment qu'il existe une relation entre les valeurs du nombre de répétitions des boucles et la mesure de l'instance voulue. Les élèves du groupe 45e semblent avoir identifié ces relations (y compris pour b_7), non seulement pour une certaine instance, mais aussi, probablement, quelle que soit la mesure : le nombre générique 6 est remplaçable par un autre nombre pour trouver b_1, et une fois b_1 trouvée, les élèves ont établi la différence de une répétition de moins pour la boucle b_2. La formulation de la relation de b_7 avec la mesure, problématique d'un point de vue sémiotique, entre en tension avec les relations établies : comment se fait-il que $b_7^6 = 10$ ou $b_6 = 11 - 1$ donne un tracé valide, alors que $b_7 = (6 - 1) \times 2$ (c'est en tout cas ce que voulaient signifier les élèves) ne fonctionne pas ?
Cela semble aboutir à devoir régler une tension fondamentale : pour $n = 6$, $b_1^{n=6} = 6 - 1 \wedge b_2^{n=6} = 5 - 1 \wedge b_7^{n=6} = 11 - 1$ donne un tracé valide, mais ne change pas si on change la mesure : le résultat de chacune de ces expressions est un nombre uniquement valide pour l'instance de mesure 6. Réduire cette tension signifie qu'il faut trouver autre chose qu'une simple représentation arithmétique, il faut trouver une expression qui dépende de la mesure, et ces élèves ont bien identifié que 

dénotait la valeur entrée par l'utilisateur, c'est-à-dire la mesure. Il doit donc exister, nécessairement, une expression dans Snap ! qui dépend de 

, et qui permet de représenter les relations établies quelle que soit la mesure. Les élèves envisagent donc la possibilité $b_1 =$ 

. Ici, les élèves semblent être en bonne voie pour aboutir à la solution, et ils semblent bien être en train de construire le concept de paramètre. Il est possible que s'ils avaient pu tester, et invalider, leur proposition, un raisonnement associant un sens à l'expression testée aurait pu aboutir à une explication de l'invalidité — puisqu'on enlève pas la valeur de la mesure à la valeur 6, même pour la mesure 6. Ceci aurait pu les ramener vers des expressions de relations en fonction de la mesure ($b_1^n = n - 1$ par exemple), et non en fonction de la valeur effective d'une autre boucle ($b_2 = \delta(b_1) - 1$), ou d'un nombre spécifique répondant au problème ($b_7 = a - 1$ pour a tel que $a - 1 = 10$).
Cependant, on peut aussi imaginer que les tensions engendrées par le faux-fait construit sur l'expression de b_7 sera un obstacle important.

Bilan

Le groupe 45e est un groupe qui semble avoir mis en œuvre ce qui s'apparente à une pensée informatique : ils ont exploré l'algorithme donné de façon organisée, y compris en le divisant pour traiter séparément les problèmes. Ces élèves semblent avoir compris le sens de cet algorithme, ce qu'il fait, ce que trace chacune des boucles. Ils ne sont pas sur une recherche de relation entre des nombres, mais plutôt sur une recherche de relation sur ce que trace chacune des boucles en fonction de la mesure. C'est ce qui leur permet d'être un des rares groupes à avoir identifié la relation permettant de trouver b_7 pour une certaine mesure.
On peut regretter qu'un faux fait issu de l'EPGB vienne s'opposer à leur construction du problème.

3.5.2 Groupe 46m

Description de l'activité

Le groupe 46m, absent lors de la séance précédente (séance 2), va construire une partie des instances demandées. La recherche du script générique prendra des formes différentes, mais débouchera malgré tout sur l'utilisation du paramètre *mesure*. La construction de la boucle b_7, pourtant bien identifiée, ne sera pas concrétisée par une formulation générique valide.

Construction du TS5 et du TS6 (étapes 1-5) Le groupe 46m commence, lui aussi par dupliquer le script $TS(4)$ et ajouter une répétition à chaque boucle. La boucle b_7 n'est pas immédiatement identifiée comme erronée, les élèves modifiant la boucle b_6 en ajoutant une répétition : $b_6 \leftarrow 5$. La rétroaction produite semble leur poser question, puisqu'ils vont supprimer l'ensemble des blocs du script $TS(5)$ avant de dupliquer de nouveau le $TS(4)$. Ils modifieront alors les boucles de façon valide, en ajoutant une répétition à toutes le boucles, sauf pour b_7 qui verra son nombre de répétitions incrémenté de deux.
Les élèves du groupe 46m dupliquent alors de nouveau le script $TS(4)$ pour créer le $TS(6)$. Ils commencent par ajouter, de nouveau, une répétition aux deux premières boucles (46m, S3, 9'51-10'02) avant de se reprendre et à définir les boucles en fonction du script $TS(5)$, en ajoutant une itération à chaque boucle, obtenant ainsi le script {5/4/9} (46m, S3, 10'05-10'28). La rétroaction produite sera ainsi :

$TS(6)[] \rightsquigarrow$ 

$, 41, 6$

Elle sera correctement interprétée et b_7 sera immédiatement corrigée pour aboutir à un TS6 valide.

Construction d'instances successives (étapes 6-19) Le groupe 46m va ensuite, comme beaucoup, construire plusieurs instances successives, allant au-delà de ce qui était demandé : ils vont ainsi construire les instances des TS7, TS8, et TS9. Ces trois instances vont être construites sans doute en appliquant le TEA $+1/+1/+2$. Cependant, une erreur sur b_6 sera faite pour le TS8 et sera ensuite répercutée sur le TS9. Ainsi, la boucle b_6 prendra comme nombre de répétitions les valeurs successives suivantes :

$$b_6^6 = 5 \rightarrow b_6^7 = 6 \rightarrow b_6^8 = 9 \rightarrow b_6^9 = 10$$

Il est difficile d'identifier la source de la première erreur sur la boucle b_6, puisque cette modification est une rupture avec les actions précédentes des élèves. Cependant, cela nous donne des informations sur les relations sur lesquelles s'appuient les élèves.

— Comme $b_6^8 = 9$ et $b_7^8 = 14$, on constate que la boucle b_7 n'est pas vue comme le double de la boucle b_6 de la même instance : les élèves construisent b_7 en se basant sur une relation inter-objectale entre instances successives.

— De même, comme l'erreur sur b_6^8 est répercutée sur b_6^9, cela confirme cette relation inter-objectale : en cas de relation intra-objectale, les élèves se baseraient soit sur $b_6^p = b_1^p$, soit sur $b_6^p = b_5^p + 1$, ce qui n'est pas le cas lorsqu'ils construisent b_6^9.

Les élèves, lors de la construction des scripts traçant les TS7 et TS8, semblent avoir confiance dans leur méthode. En effet, ils modifient l'intégralité du script devant tracer un TS8 et poursuivent, sans faire de test, par la modification du script devant tracer un TS9 (46m, S3, 13'23-14-28). On peut noter qu'ils modifient le script $TS(7)$ « de haut en bas », c'est-à-dire en modifiant d'abord la boucle b_1, puis b_2 et ainsi de suite jusqu'à b_8. Le script $TS(8)$ sur lequel ils enchainent, sera modifié « de bas en haut », en commençant par b_8, puis b_7 etc. Le groupe 46e ne va pas constater la non validité de ces scripts, puisque la fin de la séance ne leur en donne pas le temps. La boucle b_1^9 ne sera par ailleurs pas modifiée.
En séance 4, les élèves du groupe 46m testent leur $TS(9)$:

$$TS(9)[] \rightsquigarrow \left(\text{  } \text{} , 69, 22 \right)$$

Plusieurs signes d'une invalidité sont présents :

- il y a des trous,
- le TS est mal formé (il y a des décalages),
- la valeur de la mesure affichée (22) ne correspond pas à la mesure du TS tracé (9),
- le nombre d'hexagones dénombrés n'est pas cohérent avec la mesure affichée (pour un TS22, il faut 186 hexagones) — mais ce fait n'est pas accessible pour les élèves,
- la dimension du TS tracé, relativement à l'espace d'exécution, est trop faible — les élèves modifient le script censé tracer un TS22.

En revanche, le nombre d'hexagones dénombrés est cohérent pour un TS9 : la non-modification de b_1 compense l'erreur sur b_6. Cette cohérence n'est cependant pas identifiable par les élèves, qui n'ont pas construit de TS9 sur papier, et ne peuvent pas encore calculer le nombre d'hexagones d'un TS donné [65].
Cette double erreur sur b_1 et b_6 va rendre difficile la correction : les élèves s'y reprendront à dix fois avant de reconstruire un TS9 valide (étapes 10-19, figure 3.74, p. 376). L'activité de ces élèves manifeste une hésitation quant aux boucles à modifier, tantôt b_6, tantôt b_7, tantôt b_8. Les modifications faites ne semblent pas pour autant être le fruit d'une procédure essais-erreurs :

- l'étape 10 modifie b_7 pour obtenir une forme valide ;
- l'étape 11 corrige la non-modification de b_1,
- l'étape 12 visait sans doute à compenser le décalage dû à la boucle b_6 (mais celle-ci n'est pas identifiée comme source de l'erreur)
- les étapes 13, 14 et 15 procèdent de la même intention, mais ici les élèves semblent avoir des difficultés à traiter les erreurs multiples,
- l'étape 16 vise à corriger le trou issu de b_6,
- ce qui à l'étape 17 et 18 aboutit à tenter de reconstruire une forme valide,
- enfin, à l'étape 19, les élèves traitent, pour la première fois, les trois erreurs simultanément, ce qui leur permettra d'aboutir au TS9 valide.

65. Pour rappel, le nombre d'hexagones d'un TS de mesure p est $9p - 12$.

Les élèves de ce groupe ne procèdent donc pas comme le groupe 45e, ils ne décomposent pas le problème et cherchent à le régler globalement. Malgré la complexité de cette méthode, les élèves parviennent à identifier la triple erreur, sa localisation et la correction que cela implique : ils semblent bien construire ici des faits tiers. On ne peut écarter la possibilité d'une intervention extérieure, mais l'intervalle de temps entre la rétroaction de l'étape 18 et les modifications de l'étape 19 (28", 46m, S4, 11'36-12'54) parait trop court.

Construction du TS22 Après quelques minutes sans action sur l'EPGB (46m, S4, 13'40-19'14), le groupe 46m va modifier leur dernier script afin qu'il trace, comme demandé, un TS22. Cette construction prendra un peu plus de trente secondes (46m, S4, 19'14-19'46) et sera valide : $TS(22) = \{21/20/42\}$.

Vers la généralisation (étapes 21-22)
Le script $TS(22)$ est alors copié sous l'entête du script générique. Il est alors testé avec une valeur entrée de 100, puis de 88, mais sans laisser l'exécution se terminer : le déroulement est lent et peu visible. Le groupe 46m semble alors faire une relation entre la valeur entrée par l'utilisateur pour le script générique et la mesure du TS tracé : ils cherchent alors à construire l'instance TS85. Plus précisément, ils cherchent à transformer l'instance du TS22 en une instance *proche* de 88. Ils multiplient alors par quatre les valeurs du nombre de répétitions des boucles de leur script :

— $b_1^p = b_1^{22} \times 4 = 21 \times 4 = 84$;
— $b_2^p = b_2^{22} \times 4 = 20 \times 4 = 80$;
— $b_7^p = b_7^{22} \times 4 = 42 \times 4 = 168$;

Nous ne précisons pas la mesure du TS que les élèves veulent construire, car ils ne semblent pas chercher à faire un lien entre mesure et boucle lors de cet épisode. En effet, si cela avait été le cas, ils auraient testé le script obtenu avec une valeur de 85, or ils ont lancé une nouvelle exécution en entrant une valeur de 88. Cependant, la construction du TS22, correspondant à la tâche prescrite, semble montrer que, pour ce script, le groupe 46m a bien établi la différence de un entre la mesure voulue (22) et le nombre de répétitions de la boucle b_1 (21).

Vers la généralisation : recherche d'un bloc (étapes 25-30) En séance 5, les élèves du groupe 46m vont dupliquer le $TS(4)$ sous l'entête du script générique, et le tester avec une valeur entrée de 8 :

$TS(n)[8] \rightsquigarrow$ 

$, 24, 8$

Cette rétroaction semble invalider le tracé, puisque les élèves, à partir de ce moment, vont chercher à généraliser le script pour une valeur entrée qui sera systématiquement de 8, jusqu'à la fin de la séance.
Les élèves semblent alors partir à la recherche d'un bloc qui permettrait de régler la tension fixe-variable entre le script aux expressions immuables et la capacité que devrait avoir ce script à tracer un TS correspondant à la mesure, mesure qui n'est pas fixe. Ils vont ainsi tester différents opérateurs booléens, à chaque fois pour la boucle b_1 et seulement elle (tableau 3.2,

p. 377). Les instructions répéter op.bool. fois ne provoquent pas d'erreur, les booléens, qui sont des valeurs soit « vrai », soit « faux », sont représentés par des nombres, dans Snap ! comme dans d'autres langages. La valeur « vrai » est ainsi représentée par le nombre « 1 », la valeur « faux », par le nombre « 0 ». Ainsi, répéter 4 = 0 fois est équivalent à répéter 0 fois et répéter 4 = 4 fois est équivalent à répéter 1 fois.
Les élèves de ce groupe testeront aussi l'opérateur booléen vrai et vrai dans une instruction « tourner », mais il est probable qu'il s'agisse d'une erreur de manipulation : l'instruction « tourner » est juste au-dessus de la boucle b_1, qui voit son nombre répétition laissé vide (script 3.12, p. 371).

Block_478

```
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de [ true et true ] degrés à droite
répéter fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 2 fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter 2 fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter 2 fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter 2 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 6 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 2 fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || compteur |]] || hexagones |]]
```

Script 3.12 – Un opérateur booléen et mal placé (46m, S5)

Le groupe 46m semble avoir exploré les opérateurs disponibles qui ne sont pas des opérateurs arithmétiques : ils ont testé presque tous les opérateurs booléens disponibles dans la situation (voir figure 3.75, p. 378). Puisque cela n'a pas été concluant, les élèves vont commencer à tester les opérateurs arithmétiques.

Vers la généralisation : recherche d'un bloc faisant des calculs (étapes 31-39) Le groupe 46m va ainsi commencer par tester l'opérateur produit. Ils testeront successivement, toujours pour la seule boucle b_1 et pour une valeur entrée de 8, les opérateurs (2 × 2), (2 × 4), (4 × 2), puis de nouveau (2 × 2) :

$$TS(n)[8] \rightsquigarrow \left(\text{  } \text{} , 25, 8 \right)$$

Les élèves ne semblent pas le remarquer, mais la figure a subi une rotation par rapport à la figure normalement tracée, puisqu'à la suite de leur modification de l'instruction Tourner de (120) à droite en début de script, ils ont non pas corrigé, mais supprimé cette instruction. La boucle b_1 commence donc sont tracé à l'horizontal de gauche à droite. Cela sera davantage visible lors de leur tentative suivante.
Suite à ces essais non concluants, les élèves de ce groupe vont envisager de modifier plusieurs boucles. Ils commencent ainsi par $b_1 \leftarrow$ (4 × 4) et $b_2 \leftarrow$ (2 × 2) :

$$TS(n)[] \rightsquigarrow \left(\text{  } \text{} , 39, 8 \right)$$

Ce résultat peut paraitre difficilement interprétable si on considère la rétroaction d'un point de vue statique. On peut cependant légitimement penser que les élèves observent le déroulement du script, et constatent ainsi que la boucle b_1 trace trop d'hexagones (ce qui fait sortir le tracé de l'espace d'exécution). Comme le reste du tracé parait valide, ils vont affecter à b_1 la même expression qu'à b_2 : $b_1 \leftarrow$ (2 × 2). Ils semblent aussi s'apprêter à étendre cette action aux autres boucles, puisqu'ils préparent les boucles b_3 à b_8 avec l'opérateur (×) (script 3.13, p. 373).

```
Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de degrés à droite
répéter [ 2 × 2 ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ 2 × 2 ] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [ × ] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [ × ] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [ × ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ × ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ × ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
⇒ répéter *[ × ]* fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || compteur |]] ||
hexagones |]]
```

Script 3.13 – L'opérateur "×" généralisé (46m, S5)

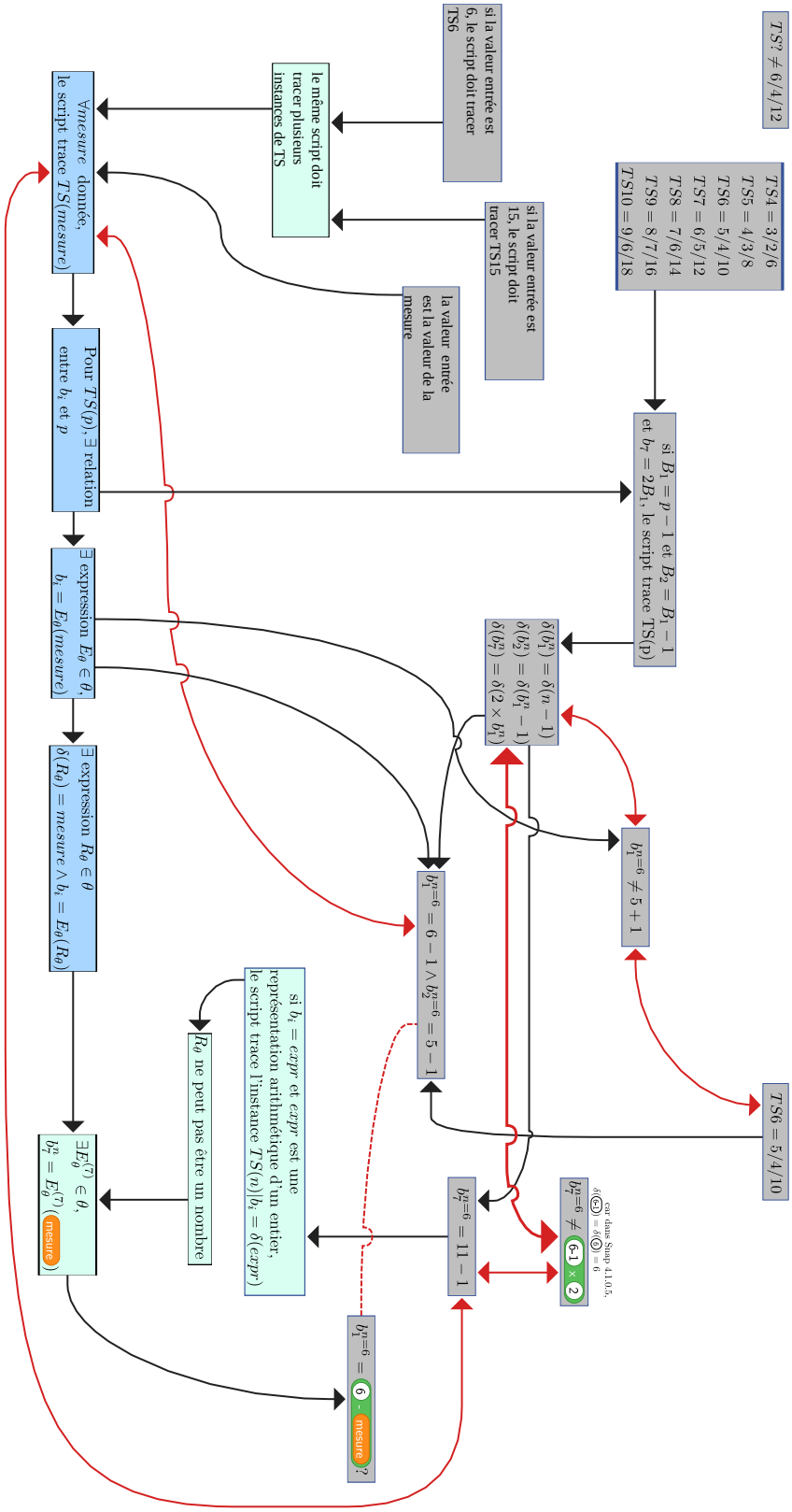


Figure 3.72 – Espace des faits-contraintes - 45e

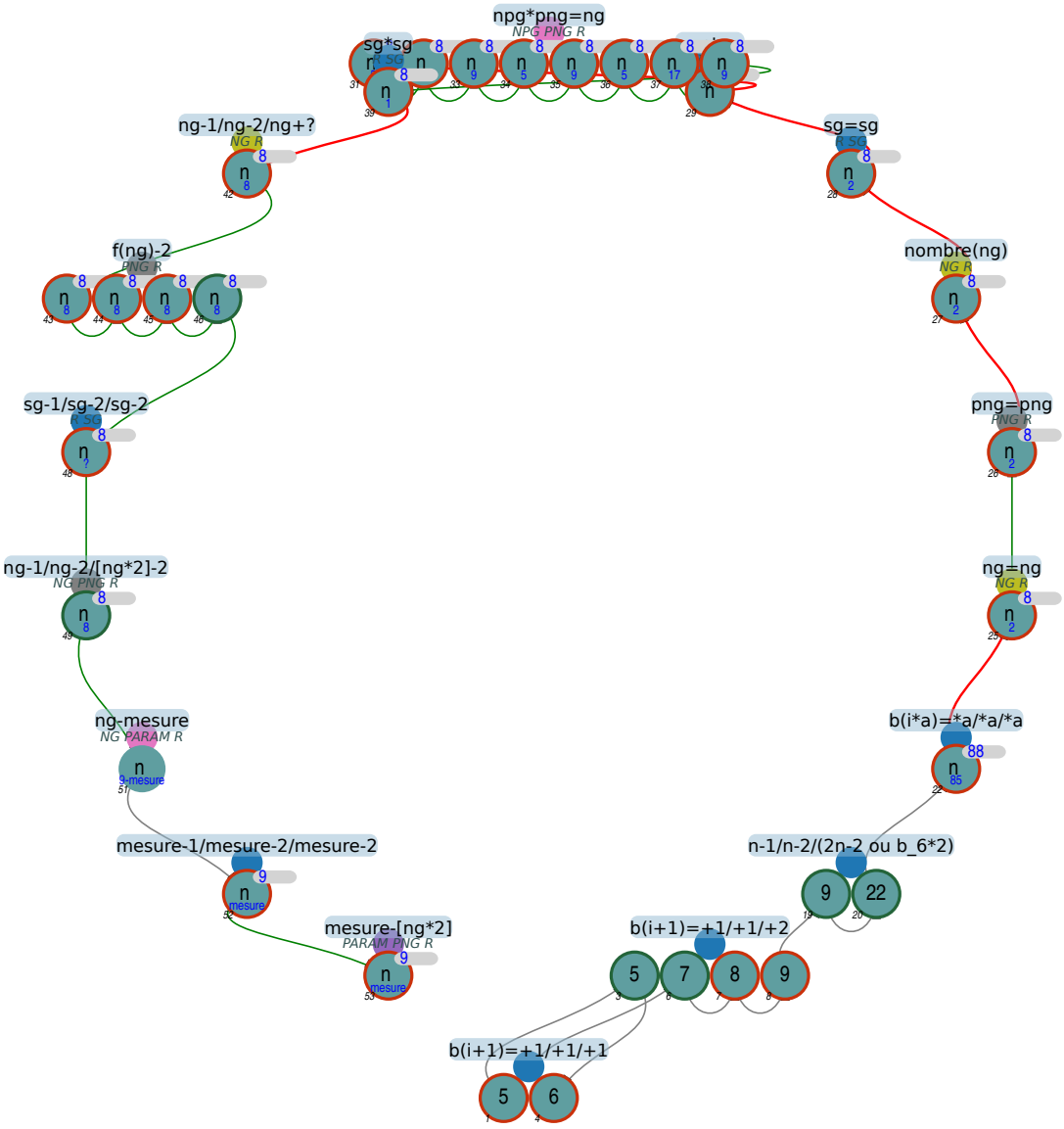


Figure 3.73 – Organisation de l'activité - 46m

(a) étape 10 **(b)** étape 11 **(c)** étape 12

(d) étape 13 **(e)** étape 14 **(f)** étape 15

(g) étape 16 **(h)** étape 17 **(i)** étape 18

Figure 3.74 – Rétroactions successives lors de la correction du TS9 (46m, S4)

Expression	Rétroaction
8 est identique à 8 ?	
4 = 2	
5 est un(e) nombre ?	
vrai ou vrai	

Tableau 3.2 – Opérateurs booléens et rétroaction (46m, S5)

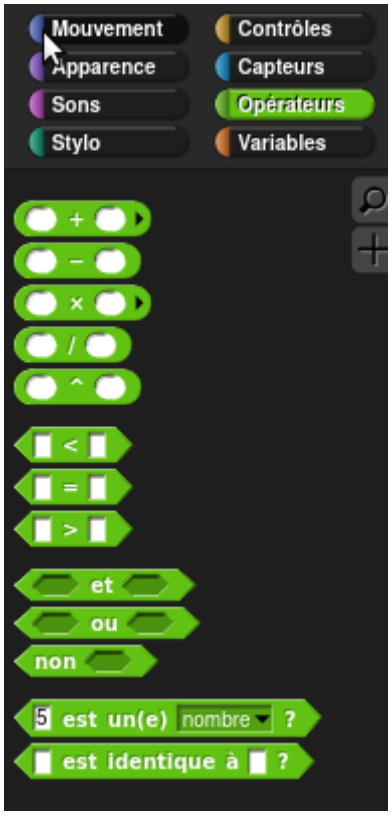


Figure 3.75 – Opérateurs disponibles dans la situation

Vers la généralisation : recherche d'une formalisation de la relation (étapes 41-47) Ce script néanmoins ne sera pas testé. Ces élèves vont reprendre une copie du $TS(4)$ comme base pour leur script générique, et vont le lancer avec une entrée de 4 — qui renvoie une rétroaction valide —, puis de 5, — qui est sans doute vue comme invalide puisque ne traçant que quatre hexagones sur les côtés des triangles de base. Ils vont alors mobiliser, soit les faits partagés en début de séance, soit leurs propres faits construits : « pour trouver le nombre de répétitions de la boucle b_1 on prend la valeur de la mesure moins un » et « pour trouver le nombre de répétitions de la boucle b_1 on prend la valeur de la mesure moins deux ». On peut noter que cette formulation confirme les relations intra-objectales (voire trans-objectales ici) que font ces élèves : b_2 n'est pas exprimée en fonction de b_1. Ils fixent ainsi les boucles de la famille B_1 à (8 - 1), celles de B_2 à (8 - 2), et laisseront b_7 en suspens : $b_7 \leftarrow$ (8 + ◯) (script 3.14a, p. 380). Ce script leur renverra, sans doute de façon attendue par les élèves :

$$TS(n)[8] \rightsquigarrow \left(\text{  } \text{} , 56, 8 \right)$$

Après avoir envisagé, mais non testé d'exprimer b_7 en ajoutant un certain nombre, non plus à 8, mais à dix, les élèves vont finalement décider de changer d'opérateur. Ils envisageront successivement $b_7 \leftarrow 10 - 2$, $b_7 \leftarrow 5 - 2$, $b_7 \leftarrow 14 - 2$, avant d'aboutir à un script produisant un tracé valide pour une mesure de 8 en utilisant $b_7 \leftarrow 16 - 2$ (script 3.14b, p. 380).
Ce script est alors testé pour des valeurs entrées de 9, puis de 5, ce qui confirme que le TS tracé reste identique.

Vers la généralisation : comment généraliser ? (étapes 47-53) Le groupe 46m va alors modifier le script en supprimant ce qui est pour nous le nombre générique 8, et le nombre ayant perdu de sa généricité 16 : $b_1 \leftarrow$ (◯ - 1), $b_2 \leftarrow$ (◯ - 2) et $b_7 \leftarrow$ (◯ - 2) (script 3.15a, p. 381). Sans faire de test, ils vont remplacer les places vides par les valeurs attendues, 8 pour les boucles B_1 et B_2, 16 pour b_7. Cette nouvelle occurrence du script 3.14b (p. 380) sera de nouveau testée pour une valeur entrée de 8, et validée.
Après avoir envisagé $b_1 \leftarrow$ (8 - mesure) (46m, S5, 25'55), les élèves lui préfèrent rapidement $b_1 \leftarrow$ (mesure - 1). Ils vont ensuite propager ces modifications, remplaçant les places vides par la variable mesure (script 3.15b, p. 381). L'exécution de ce script renverra, testé avec une valeur cette fois différente de 8 :

$$TS(n)[9] \rightsquigarrow \left(\text{  } \text{} , 60, 9 \right)$$

Block_478

Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [8 − 1] fois (commandes)
......tracer
tourner de 120 degrés à gauche
répéter [8 − 2] fois (commandes)
......tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [8 − 2] fois (commandes)
......tracer
tourner de 120 degrés à droite
répéter [8 − 2] fois (commandes)
......tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [8 − 2] fois (commandes)
......tracer
tourner de 120 degrés à gauche
répéter [8 − 1] fois (commandes)
......tracer
tourner de 120 degrés à gauche
répéter [8 +] fois (commandes)
......tracer
tourner de 120 degrés à gauche
répéter [8 − 2] fois (commandes)
......tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(a) b_7 en attente

Block_478

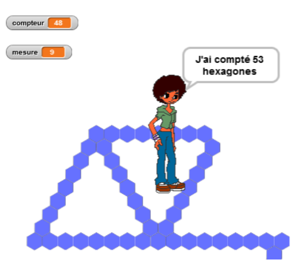
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [8 − 1] fois (commandes)
......tracer
tourner de 120 degrés à gauche
répéter [8 − 2] fois (commandes)
......tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [8 − 2] fois (commandes)
......tracer
tourner de 120 degrés à droite
répéter [8 − 2] fois (commandes)
......tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [8 − 2] fois (commandes)
......tracer
tourner de 120 degrés à gauche
répéter [8 − 1] fois (commandes)
......tracer
tourner de 120 degrés à gauche
répéter [16 − 2] fois (commandes)
......tracer
tourner de 120 degrés à gauche
répéter [8 − 2] fois (commandes)
......tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(b) Valide pour un TS8

Script 3.14 – Expressions des relations (46m, S5)

Une ultime correction sera faite et testée avant la fin de la séance, $b_7 \leftarrow$ mesure - 16, qui renverra :

$TS(n)[9] \rightsquigarrow$ (



$, 53, 9$)

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [− 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [− 2] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [− 2] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [− 2] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [− 2] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [− 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [− 2] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [− 2] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(a) Des marque-places...

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 2] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [*mesure* − 2] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [*mesure* − 2] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [*mesure* − 2] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 2] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 2] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(b) ... à la mesure

Script 3.15 – Vers le script générique (46m, S5)

Recherche de régularités

Résistance du TEA $+1/+1/+1$ Comme pour la plupart des autres groupes, le TEA $+1/+1/+1$ est mis en œuvre en premier. La nécessité d'ajuster les actions, en modifiant d'abord b_6 puis b_7, ne suffit pas à invalider totalement ce TEA, qui est valide selon les élèves lors de la construction du TS6. Une nouvelle modification de b_7 semble permettre l'émergence du TEA $+1/+1/+2$, qui sera appliqué pour le TS7 et TS8, ainsi que pour le TS9 mais avec une erreur sur b_6. Une fois encore, comme le groupe 45d par exemple, les TEA résistent tant qu'ils ne sont pas remplaçables, pour les élèves, par un autre TEA : tout se passe comme si les nouveaux faits ne validaient ou n'invalidaient pas un TEA ou un autre, mais plutôt qu'ils modifient la probabilité, pour les élèves, que ces TEA soient valides ou invalides.

Une généralisation algébrique naïve mais non stabilisée Les élèves parviennent à construire directement le script traçant un TS22, ce qui laisse supposer qu'ils ont établi des relations, non formalisées, entre l'instance (la mesure) et les nombres de répétitions des boucles. Il n'y a notamment pas d'hésitation sur la boucle b_7, qui sera modifiée en 8 secondes là où les autres le seront en 4 à 6 secondes (46m, S4, 19'14-19'46). Les élèves de ce groupe semblent avoir trouvé un moyen de trouver la valeur du nombre de répétitions de b_7. Cependant, lorsqu'ils vont chercher à construire une instance s'approchant de la dimension d'un TS88, c'est le TEA $*a/*a/*a$ qui est mobilisé, or celui-ci est en opposition avec les relations permettant de déterminer les

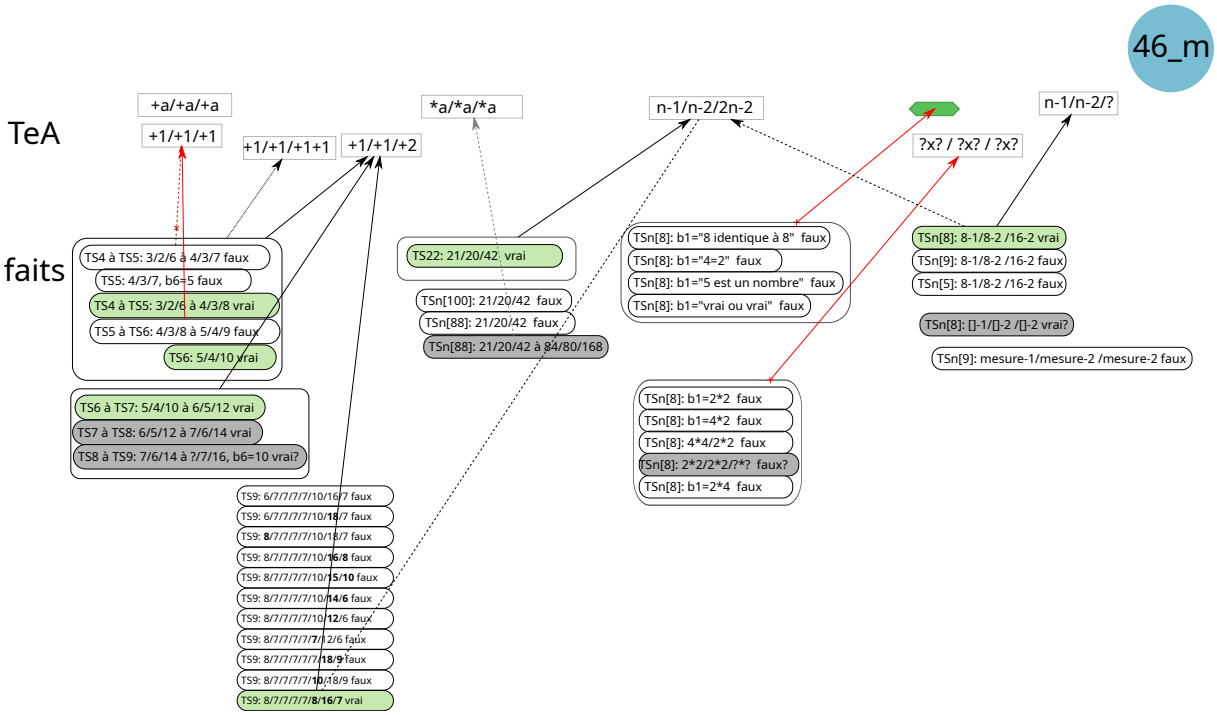


Figure 3.76 – Faits et TEA, groupe 46m

valeurs du nombre de répétitions des boucles de la famille B_2, puisque si $b_1 = 84$ on devrait avoir $b_2 = 83$. Là encore, le TEA permettant de déterminer directement les valeurs pour le TS22 n'est pas stabilisé, et a été remplacé ici par un TEA qui semblait plus opportun, peut-être car moins coûteux. Ce TEA considérant qu'il faut multiplier toutes les boucles par la même valeur a été rencontré chez plusieurs groupes (voir le groupe 45e, , p. 357). Pour les élèves de ce groupe, il a une particularité qui le rend en partie valide : le rapport entre la boucle b_7 et les boucles de la famille B_1 est respecté : si $b_7 = 2b_1$, $a \times b_7 = 2(a \times b_1)$. Ainsi, si ces élèves mobilisaient le TEA qui leur a permis de construire le TS22, il y aurait cohérence pour les boucles B_1 et b_7. Le script obtenu n'ayant pas été testé jusqu'au bout, il est difficile de savoir si les élèves le considèrent toujours comme valide ou non.

Cette construction du TS22, issue d'une généralisation naïve, apparait juste après la construction, laborieuse, de l'instance traçant un TS9. La construction de cette instance a nécessité les actions suivantes :

$$
\begin{array}{lllllllllllll}
b_1^9 = & 6 & & +2 & & & & & & & & & \\
b_2^9 = & b_2^8 + 1 & & & & & & & & & & & \\
b_3^9 = & b_3^8 + 1 & & & & & & & & & & & \\
b_4^9 = & b_4^8 + 1 & & & & & & & & & & & \\
b_5^9 = & b_5^8 + 1 & & & & & & & & & & & \\
b_6^9 = & 10 & & & & & & & -3 & & +3 & -2 & \\
b_7^9 = & b_7^8 + 2 & +2 & & -2 & -1 & -1 & -2 & & +6 & & -2 & \\
b_8^9 = & b_8^8 + 1 & & & +1 & +2 & -4 & & & +3 & & -2 &
\end{array}
$$

Comme pour le groupe 45e (, p. 358), nous nous demandons comment cette suite d'action, elle aussi sur deux séances, aurait pu aboutir sur un TEA, de surcroit générique (puisque permettant de construire directement une certaine instance).

La dernière rétroaction, avant la triple modification qui permettrait d'aboutir à un TS valide est la suivante :

$$TS(9)[] \rightsquigarrow \left(\text{  } \text{} , 75, 22 \right)$$

La dernière correction implique donc que les élèves ont identifié :

1. que b_6 traçait deux hexagones de trop ;
2. que cela implique que le côté b_7 est trop long de deux hexagones ;
3. et par suite que b_8 deviendrait trop long de deux hexagones.

Ainsi, cette correction parait impliquer que les élèves identifient bien les boucles, ce qu'elles tracent et le rapport entre ce qu'elles tracent et leur nombre de répétitions : comme le groupe 45e, cette exploration semble les avoir amenés à comprendre l'algorithme de tracé, ce qui pourrait expliquer, en partie, pourquoi ils arrivent ensuite à construire une instance directement.
Il n'est pas impossible que la correction du TS9 et la construction du TS22 soient issus d'interventions extérieures. Cependant, cela parait peu probable : d'une part, ce groupe est en bout de rangée, et leur seul voisin, le groupe 46l, n'a construit ni le TS9 ni le TS22 ; d'autre part, si l'enseignante étaient intervenue, elle n'aurait pas apporté les solutions directement (en tout cas si on se fie à ce qu'elle dit et fait avec les groupes filmés). Ici, une certaine forme d'errance due à la difficulté de traiter simultanément de multiples erreurs, semble avoir amené les élèves à s'intéresser à l'algorithme et non seulement aux nombres.

Généralisation algébrique

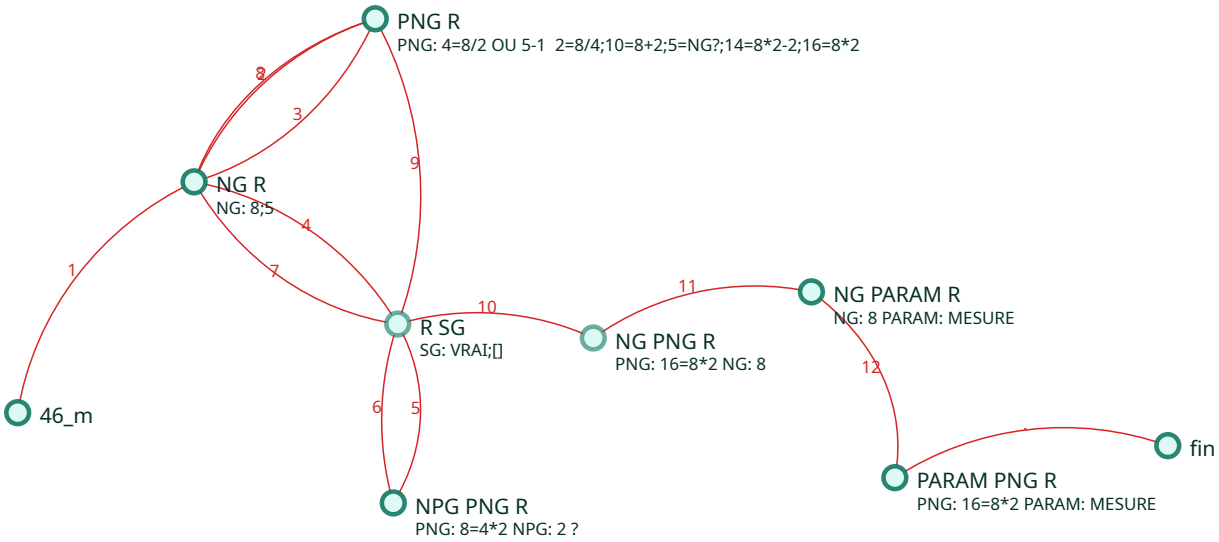


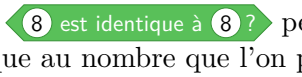
Figure 3.77 – Type de généralisation - 46m

Le groupe 46m parait avoir exploré de nombreuses possibilités de généraliser le script, mobilisant des nombres génériques, des nombres ayant perdu leur généricité, des signes génériques, tout cela en recherchant une formulation des relations génériques entre nombre de répétitions et valeur de la mesure voulue. Ils finiront par utiliser la variable *mesure*, mais sans que le concept de paramètre soit complètement construit.

Recherche d'une expression de θ Une fois que les élèves ont manifesté leur prise en compte de la nécessité qu'un même script trace plusieurs instances de TS, en fonction de la mesure, le groupe 46m va commencer par chercher un bloc, une expression de θ, qui pourrait permettre à l'ordinateur de « lui-même trouver ce qu'il faut mettre ici [le nombre de répétitions des boucles] » (46 Prof, S5, 233), c'est-à-dire un bloc qui représenterait la relation entre la mesure et le nombre d'itérations des boucles. Les élèves vont faire tous leurs tests en conservant l'exemple donné par l'enseignante en début de séance : le TS de mesure 8 (46 Prof, S5, 220-266). Comme suggéré par l'enseignante, ils s'intéressent aux blocs de la famille des opérateurs (figure 3.75, p. 378). Cependant, ils ne cherchent pas à utiliser un bloc permettant de faire des calculs : ils font des tentatives sur des blocs d'opérateurs booléens, dont on peut imaginer que les élèves ne connaissent pas le sens. Or, certains de ces blocs peuvent laisser penser qu'ils ont une portée générique :

— le bloc 

pourrait signifier la recherche d'un « nombre », qui remplacerait le nombre « 5 » [66] ;

— le bloc 

peut évoquer « voici ce que l'on ferait si le nombre entré était identique au nombre que l'on prend en exemple (8) » ;

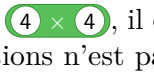
— ce dernier bloc pouvant être dérivé en 

, auquel les élèves préfèrent 

, suggérant en plus l'idée d'un calcul —ici il faut que $4 \times 2 = 8$, 8 étant la valeur exemple.

Les élèves sont donc en train de chercher un signe de θ, qui pourrait traduire leur pensée. Ils se trouvent confrontés à la contrainte de rendre explicite les actions, de « faire-faire » (Samurçay et Rouchier, 1985) à un dispositif d'exécution. Les expressions utilisées par les élèves contiennent sans doute pour eux une grande part d'implicite ou d'interprétation : à l'ordinateur de comprendre ce que les élèves attendent de lui. Cette contrainte n'est pas aussi forte lorsqu'il s'agit de communiquer avec un humain [67].

Recherche d'un calcul dans θ Puisque ces blocs ne fonctionnent pas, le groupe 46m teste les opérateurs produit. Ici encore, ils semblent s'intéresser à la recherche d'une expression qui « pourrait fonctionner ». Ils ne s'intéressent pas au sens littéral des opérateurs. En effet, lorsque les élèves testent $b_1 \leftarrow$ 

ou $b_1 \leftarrow$ 

, il est difficile d'imaginer qu'ils seraient incapables de voir que le résultat de ces opérations n'est pas compatible avec les valeurs attendues pour un TS8. Ceci est renforcé par les deux tests consécutifs $b_1 \leftarrow$ 

et $b_1 \leftarrow$ 

: on peut légitimement imaginer que le fait numérique $4 \times 2 = 2 \times 4$ est construit par des élèves de cet âge. Ils ne cherchent donc pas une opération donnant un certain résultat attendu, mais une expression d'un calcul que l'ordinateur interpréterait correctement, en fonction de leur but.

66. Notons que le nombre « 5 » n'est ici pas choisi par les élèves, c'est la valeur par défaut du premier paramètre de cet opérateur.

67. Le groupe 46i a, lui aussi, utilisé un opérateur booléen, mais uniquement dans l'affectation du compteur (p. 337).

On peut imaginer que les élèves ont invalidé ces possibilités au moment où ils ont cherché à généraliser l'expression (2 × 2) à toutes les boucles (script 3.13, p. 373). La prise en compte des faits construits préalablement sur la construction des TS, notamment ceux impliquant une différence entre b_1 et b_2, a pu aboutir au constat qu'une opération arithmétique avec des écritures de nombres donne toujours le même résultat, et donc que b_1 et b_2, à l'exécution, traceront autant d'hexagones l'une que l'autre.

Représentation des relations boucle-mesure : NG et SG et paramètre Le groupe 46e va par la suite mobiliser les faits construits par eux et/ou partagés par la classe : pour un TS8, la boucle 1 fait 7, on fait moins un par rapport à la mesure, et la boucle 2 fait 6, on fait moins deux par rapport à la mesure (46 Prof, S5, 234-262). Ces faits seront traduits par des expressions de θ : $b_1 \leftarrow$ (8 - 1) et $b_2 \leftarrow$ (8 - 2). Pour b_7, la relation n'est pas totalement formalisée par les élèves : $b_7 \leftarrow 8+?$. Le 8 est ici un nombre générique, nombre exemple utilisé par l'enseignante, et nombre mobilisé pour tous les tests de ce groupe. Ce nombre générique était déjà présent lors des explorations précédentes : (8 est identique à 8 ?). Dans cette expression, il est possible que les deux signes « 8 » n'aient pas le même sens pour les élèves, l'un ayant un statut générique, et l'autre un statut d'instance précise.
Si l'on omet la boucle b_7, les élèves de ce groupe passent par quatre étapes successives, représentées dans le tableau suivant (3.3) :

	b_1	b_2
1	(8 - 1)	(8 - 2)
2	(◯ - 1)	(◯ - 2)
3	(8 - 1)	(8 - 2)
4	(mesure - 1)	(mesure - 2)

Tableau 3.3 – passage du NG à la mesure, b_1 et b_2 (46m, S5)

Dans l'étape 1, ils formalisent la relation entre la mesure et le nombre de répétitions des boucles pour un exemple générique, « 8 ». Les étapes 2 et 3 — sans test entre l'étape 2 et l'étape 3 — semblent être une simulation de ce qu'il « devrait se passer » si le script était générique. Dans l'étape 2, on peut penser que les élèves ont remplacé le nombre générique par un signe générique, un espace vide ◯, qui marque une place à prendre (*placeholder*) : c'est cet espace qui doit « bouger » lorsque l'instance change. Une nouvelle instance implique que la valeur entrée, la valeur de la mesure, vient prendre la place laissée vide. C'est ce que simule ce groupe en enchainant sur le remplissage de tous ces espaces vides par une valeur « 8 » : d'une manière ou d'une autre, l'ordinateur devrait faire cette substitution [68]. Ce « 8 » entré n'est plus ici un nombre générique, il est le dénoté de la valeur entrée par l'utilisateur pour la mesure, il est une une instanciation de la mesure. Ces rôles différents pris par un même nombre étaient déjà constatés dans le recueil de problèmes Héroniens *Geometrica*, par exemple sur le problème n°38 (Vitrac, 2005, p. 12).
Il est probable que ce soit cette simulation qui amène les élèves à mobiliser la mesure :

— il y a une expression E_θ de θ qui met en relation la mesure et le nombre de répétitions des boucles ;

— dans cette expression, un des paramètres dénote la valeur de la mesure, par exemple 8 ;

68. Le groupe 46e effectuera la même simulation, mais sur une seule boucle (p. 424).

— mais si on laisse la valeur « 8 », le tracé ne change pas si on choisit d'entrer une autre mesure, par exemple 9 ;

— donc, dans cette expression, il existe une expression R_θ dans θ que l'ordinateur va substituer à la valeur de la mesure entrée, une expression dont la dénotation est la valeur de la mesure ;

— la valeur de la mesure entrée est stockée dans la variable mesure [69], $\delta($mesure$) = 8$ si $mesure = 8$;

— mesure est donc une expression de θ qui dénote la mesure ;

— donc il est probable que $R_\theta =$ mesure, et donc que, pour b_1, $E_\theta^{(1)} =$ mesure - 1.

Le signe générique vide suffirait pour un dispositif d'exécution humain, à la condition de lui avoir indiqué au préalable qu'il faut remplacer l'espace libre par la valeur de la mesure entrée. C'est le fait que le dispositif d'exécution est incapable d'interpréter ce SG qui implique nécessairement l'existence d'un moyen, dans θ, de faire cette substitution.

Représentation des relations boucle-mesure : PNG, SG et paramètre Cependant, la boucle b_7 va introduire une contrariété, une tension. Les élèves ne l'ont pas exprimée à partir du nombre générique « 8 », du moins pas directement. Ils fixent ainsi $b_7 = 16 - 2$. Cette expression est valide pour un TS8, puisqu'elle est équivalente à $b_7 = 2 \times 8 - 2$. Si l'on complète les étapes amenant les élèves du nombre générique à la mesure, le tableau 3.3 (p. 385) devient :

	b_1	b_2	b_7
1	8 - 1	8 - 2	16 - 2
2	◯ - 1	◯ - 2	◯ - 2
3	8 - 1	8 - 2	16 - 2
4	mesure - 1	mesure - 2	mesure - 2

Tableau 3.4 – passage du NG à la mesure, b_7 (46m, S5)

On peut voir que le marque-place ◯ indique bien un espace à prendre, mais aussi qu'il n'est pas une représentation univoque : suivant les boucles, il est censé prendre soit la valeur de la mesure (8), soit le double de cette mesure (16). Pour reprendre les termes de Serfati (1997, p. 45) [70], ce marque-place est à la fois un *signe* et une *marque* : un signe non-univoque, on l'a dit, et une « marque dans le texte » de ce qu'il faut changer. Ce signe non-univoque est problématique dans le cas d'un dispositif d'exécution automatique : autant on pourrait faire comprendre à un humain que, pour une mesure de 8, il faut mettre 8 à tels endroits marqués et 16 à tel autre, autant ceci est impossible pour un dispositif automatisé, si on ne dispose que d'un signe [71]. Pour un dispositif d'exécution automatisé, un signe, dans le langage de ce dispositif, est univoque. Ou presque : dans un cas comme celui-ci, on pourrait modifier la variable *mesure*, son sens, suivant l'état du programme. On pourrait ainsi ajouter avant la boucle b_7

69. Fait partagé en groupe classe et rappelé en cours de séance.
70. En parlant de l'inconnue chez Diophante.
71. Notons cependant qu'il existe des signes non univoques dans certains langages, des « tags ». Par exemple, en Rust, on peut écrire `println!("Une variable ici {} et une autre là {}.", a, b)`, de même en Python `'une variable ici %s et une autre là %s.' % (a,b)` ou `'une variable ici  et une autre là .'.format(a,b)`. Cela signifie qu'il faut fournir deux valeurs, mais dans le cas de notre dispositif automatisé seule une valeur (la mesure) est connue.

l'instruction mettre mesure à mesure × 2 : la variable mesure représente alors deux côtés d'un triangle de base, ou le double de la mesure entrée. Après b_7, on rétablit le sens originel : mettre mesure à mesure / 2. Quoiqu'il en soit, il faut rendre explicite le passage du NG 8 au PNG 16, lors de l'étape 3. À ce stade, les élèves ne semblent pas considérer le caractère non-univoque comme un problème. mesure, pour les élèves, est donc une marque de ce qu'il faut changer, et non pas uniquement le signe représentant la valeur de la mesure du TS à tracer. mesure n'est donc encore construit comme étant un paramètre pour les élèves.
La rétroaction permettant d'invalider le tracé, les élèves vont envisager $b_7 \leftarrow$ mesure - 16. Ils cherchent peut-être ici à marquer le caractère « différent » de mesure pour b_7 : il faut faire la même chose que ce qu'on ferait pour trouver 16 lorsque la mesure est 8. On retrouve une variante de la solution envisagée par le groupe 45e en fin de séance 5 (6 - mesure). L'opérateur arithmétique de la soustraction n'est pas reconnu comme tel : il s'agit d'une expression de θ qui prend un autre sens. Ici encore, le PNG semble être un obstacle à la généralisation. Cependant, ce PNG (et le NG 8 associé) n'est pas mobilisé avec le même sens que celui donné par les élèves du groupe 46i. Ainsi, dans le script 3.8 (p. 347), les élèves fixent $b_1 = 15 - 1$, $b_2 = 16 - 1$ et $b_7 = 31 - 1$. Le nombre générique est ici 15, mais les élèves semblent considérer que c'est la relation « $? - 1$ » qui est générique : les nombres 6 et 31 sont trouvés en cherchant quel pourrait être le nombre qui, si on lui enlève un, donnerait la valeur du nombre de répétitions de la boucle. Pour les élèves du groupe 46m, dans $b_7 = 16 - 2$, ce n'est pas la relation qui est générique (puisqu'elle change), mais bien le nombre « 16 », comme en témoigne le passage au SG vide ensuite. Ce nombre n'est pas établi comme étant celui qui permet de trouver le bon résultat, mais comme une transformation du nombre générique permettant de trouver le bon résultat, connaissant cette relation, ici $16 = 2 \times 8$. Le PNG est un obstacle, mais le dépassement de cet obstacle permettrait de bien construire le fait que mesure est un signe et non seulement une marque. On peut ainsi imaginer qu'avec plus de temps, les élèves, qui ont constaté l'invalidité de ce signe, en seraient venus à rendre explicite ce qui permet de passer de 8 à 16, 8 × 2, et à substituer ce nouveau signe à 16 : 16 - 2 deviendrait alors 8 × 2 - 2. Cela aboutirait, en reproduisant les étapes du tableau 3.4 (p. 386), à la démarche suivante (tableau 3.5) :

	b_1	b_2	b_7
1	8 - 1	8 - 2	8 × 2 - 2
2	○ - 1	○ - 2	○ × 2 - 2
3	8 - 1	8 - 2	8 × 2 - 2
4	mesure - 1	mesure - 2	mesure × 2 - 2

Tableau 3.5 – passage idéal du NG à la mesure

On peut aussi penser que ces élèves, qui arrivent à doubler le nombre exemple avant d'intégrer ce résultat dans une expression, sont limités par des aspects instrumentaux : comment représenter la structure profonde de $(8 \times 2) - 2$ dans le langage θ ? D'autres élèves semblent avoir été confrontés à cette difficulté.

Construction des nécessités

Les élèves du groupe 46m ont rapidement posé le problème de la construction du script générique. Les tests faits avec plusieurs valeurs entrées différentes, suivies de modifications du nombre de répétitions des boucles, montrent qu'ils ont construit la nécessité de changer ces valeurs, et seulement elles, et qu'ils mettent en tension le caractère immuable des expressions numériques définissant le nombre de répétitions avec la multiplicité des tracés que doit faire le script.
Pour résoudre cette tension, il doit donc exister un bloc qui permet à l'ordinateur de définir les nombres d'itérations en fonction de la mesure entrée, soit une expression traduisant la relation existant entre ces valeurs. Cette expression, si elle est formulée avec des nombres génériques, reste fixe : il doit y avoir un signe qui exprime la généricité, qui indique à l'ordinateur ce qu'il doit remplacer dans les expressions. Comme la variable mesure dénote la valeur entrée, l'expression doit manipuler cette représentation de la mesure.
Toutes les nécessités du problème semblent ainsi construites, mais il en est une que nous avions laissée implicite, et qui n'est pas encore construite par ces élèves. En effet, si un paramètre est un objet dont la représentation est manipulée dans une expression comme si cet objet était connu, cela implique qu'à l'évaluation de l'expression, c'est-à-dire lorsque qu'on instancie le problème pour trouver une solution à cette instance, la représentation doit être univoque : c'est un signe qui dénote un seul et même objet pour une instance donnée. Passer de la marque d'un espace à prendre au paramètre nécessite cette unicité, momentanée, de l'objet dénoté.

Bilan

Le groupe 46m a pu construire assez rapidement des instances successives. Une erreur et un oubli pour la construction d'un script traçant le TS9 rendront la correction du script difficile : de multiples erreurs sont difficiles à interpréter, sauf si l'on a la compréhension de l'algorithme de tracé. Il est possible que ces élèves — qui n'étaient pas présents lors de la séance 2, censée permettre aux élèves non seulement de faire une première rencontre avec une variable informatique, mais aussi de s'intéresser à la structure du script, à l'algorithme de tracé —, pour dépasser ces difficultés, se soient vus contraints de comprendre le script pour le corriger. Cela leur a permis de ne pas se contenter de chercher des rapports entre des nombres, et donc de construire une instance quelconque, mais aussi d'établir une relation, non formalisée, entre le nombre de répétitions de la boucle b_7 et la valeur de la mesure. La tentative de formalisation pour un cas générique utilisera cependant un PNG (16 au lieu de 2×8) qui montrera que pour eux, le signe mesure est essentiellement un moyen graphique de marquer un espace à prendre lors de l'instanciation du problème. Il pose ainsi la question de savoir si les groupes qui ont mobilisé cette variable dans une expression la considère comme une *marque* ou comme un *signe* univoque.

3.5.3 Synthèse Groupement 4

Le groupement 4 concerne les groupes 45e, 46m (détaillés) et 45l, qui ont mobilisé le paramètre mesure après une exploration variée.
Les groupes 45e et 46m ont construit, avec succès, un nombre conséquent d'instances : du TS5 au TS10 pour le premier, du TS5 au TS9 et le TS22 pour le deuxième. En revanche, le groupe 45l n'a tenté de construire que trois instances avant la généralisation du script : les TS5, TS10 et TS27, aucune n'aboutissant à un tracé valide. Pour construire ces instances les groupes 45e

et 45l ont commencé par le TEA $+1/+1/+1$, qu'ils ont ensuite étendu en mettant en œuvre $*a/*a/*a$. Le groupe 46m a lui aussi appliqué ce dernier TEA, mais seulement pour la recherche de généralisation sur un TS85, alors que, comme le groupe 45e une fois avoir invalidé ces TEA, il semblait mobiliser de façon stable le TEA valide $+1/+1/+2$. Le TEA dépend du problème posé. Le problème de la recherche d'un script générique est bien posé chez ces élèves, mais avec une identification de la propriété *mesure* variable : le groupe 45e semble ainsi associer la mesure à la valeur du nombre de répétitions de la boucle b_1, avant de mobiliser des faits (construits ou partagés en classe) établissant la relation « on fait moins un ». Le groupe 46m quant à lui fait bien le lien entre la mesure et la boucle b_1 ($b_1 = 21$ pour le TS22, $b_1 = 8 - 1$ pour une entrée de 8), malgré une hésitation sur la relation [valeur entrée]-[dimension de la figure tracée]. Pour le groupe 45l, ce n'est pas stable non plus, avec une entrée de 27 pour $b_1 = 26$, mais aussi une entrée de 30 pour cette même valeur. Ces hésitations viennent de la construction des instances et de l'EPGB : pour construire une instance à partir de la précédente, on n'a pas besoin de connaitre la valeur de la mesure, et le lien entre la mesure, les hexagones dénombrés et le tracé n'est pas mis en valeur par la situation. En effet, le message renvoyé aux élèves en fin de tracé est « J'ai compté [] hexagones ». La valeur de la mesure est certes visible, mais il aurait été plus pertinent, pour faciliter l'abduction (Radford, 2008, p. 2), d'afficher comme message « Pour un TS de mesure [], j'ai compté [] hexagones ». Ceci aurait peut-être permis aux élèves d'identifier la propriété mesure pour chaque instance. En effet, comment passer d'une suite $p_1, p_2, p_3 ..., p_k$ d'instances sur laquelle on repère un schéma, à un moyen de déterminer directement p_n, si on n'associe pas l'instance p_i et la propriété i ? Lors de généralisations de motifs non informatisés, on peut aussi avoir cette limite. On constate cependant que les faits partagés avec le groupe classe permettent de préciser cette association [mesure]-[boucles].
Comme cela a déjà été observé dans les précédents groupements, un groupe va explorer le langage pour trouver une expression, un bloc, une instruction, qui résoudrait le problème de la tension fixe-variable, entre expression immuable et dénoté du nombre de répétitions des boucles variable. Ce groupe, le 46m, va tester méthodiquement les blocs d'opérateurs booléens avant de chercher du côté d'une opération « magique ». Les faits qu'ils ont construits relativement aux instances déjà créées, peut-être combinés aux faits partagés en classe liant mesure et premières boucles, les amèneront à envisager l'expression 

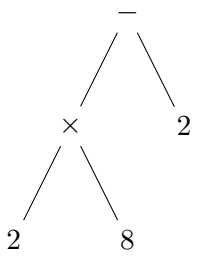
pour b_1 et une mesure de 8.
Cette utilisation du NG 8, déjà observée est aussi visible chez le groupe 45e ($b_1 = 6 - 1$ pour l'instance 6). Comme pour le groupement 2, on observe aussi une perte de ce nombre générique : le groupe 45e définira $b_1 = 6 - 1$ et $b_2 = 5 - 1$ pour l'instance 6, et le groupe 46m définira $b_1 = 8 - 1$, $b_2 = 8 - 2$ et $b_7 = 16 - 2$ pour l'instance 8. Le « 5 » ($5 = 6 - 1$) et le « 16 » ($16 = 2 \times 8$) sont des PNG. Il faut noter que le groupe 46m était passé d'abord par $b_7 = 8 + []$, ce qui conservait la visibilité du NG, mais traduisant aussi leur incapacité à formaliser la relation entre b_7 et la valeur de la mesure. Cette difficulté à formaliser b_7 est une constante, mais on voit dans ces groupes trois niveaux :

1. Le groupe 45l ne fait aucune relation entre la mesure et b_7, il explore, par essai-erreur, différentes combinaisons d'opérateur ayant comme premier opérande la mesure et comme deuxième opérande un certain nombre.

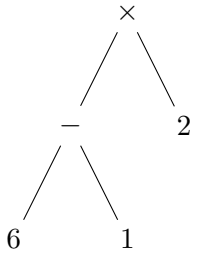
2. Le groupe 46m fait sans doute une relation implicite entre b_7 et la mesure, il constate que $b_7 = 2 \times 8 - 2$, mais ne peut exprimer cette relation dans θ.

3. Le groupe 45e a, lui, établi le lien entre b_7 et b_6 ($b_7 = 2 \times b_6$, soit $b_7^6 = (6 - 1) \times 2$), mais l'expression de cette relation dans θ produira un tracé erroné, invalidant leur formulation et les amenant à quitter cette représentation mobilisant un NG au profit d'un PNG : $b_7^6 = 11 - 1$.

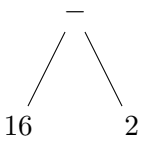
Pour les niveaux deux et trois, il y a un problème d'origine instrumentale. Le groupe 46m voudrait représenter la structure profonde suivante :



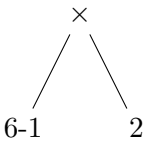
Le groupe 45l voulant représenter la structure :



Les élèves ne semblent pas identifier la possibilité qu'un opérande soit lui-même un opérateur, ce qui les amènera à représenter :



soit dans Snap ! (16 - 2) et



soit dans Snap ! (6-1 × 2).

Cette dernière expression dénotant la valeur 12 et non 10 comme attendue. Pour ce dernier groupe, l'incapacité à imbriquer des opérateurs est en plus impactée par la production d'un faux fait. On peut remarquer que les deux groupes qui ont établi une relation entre b_7 et la mesure, sont passés par une étape d'analyse du script et d'identification des boucles et de ce qu'elles tracent. S'ils ont établi une relation, c'est sans doute parce qu'ils mettent du sens derrière l'expression représentant le nombre d'itérations de la boucle b_7 : on trace un côté de deux fois la mesure, et on enlève les deux hexagones en trop, ou on trace deux fois [un côté d'une fois la mesure auquel on enlève un hexagone en trop]. En revanche, le groupe 45l ne met pas de sens sur l'expression de b_7 : ils obtiennent $b_7 =$ (mesure × 1.95). Cette dernière expression n'a pas de sens dans la situation, mais elle est valide pour certaines valeurs [72] : c'est un nouveau faux-fait.

Si le groupe 45l semble ne se baser que sur les faits partagés en classe, les groupes 45e et 46m vont être amenés à utiliser le signe (mesure) comme désignation de « ce qui bouge » entre les instances — après avoir utilisé le signe générique [] pour le groupe 46m. Pour le groupe 45e, les relations sont de la forme $b_i = f(b_i) - 1$, et ils semblent supposer que les $f(b_i)$ restent fixes d'une instance à l'autre, et ce qui change c'est le NSG « 1 », ils envisagent donc l'utilisation de (6 - mesure) pour b_1. Pour le groupe 46m, ce qui change d'une instance à l'autre, ce sont les premiers opérandes. Ainsi, « 8 − 1 », « 8 − 2 » et « 16-2 » seront réécrits avec le SG [] : « []-1 »,

72. $\forall n \in \mathbb{N}, 20 < n \leq 40 \implies 2n - 2 = n \times 1.95$.

« []-2 » et « []-2 », respectivement. Ce qui aboutira à « mesure-1 », « mesure-2 » et « mesure-2 ». Dans le premier cas, mesure indique ce qui change avec l'instance, de façon univoque : la valeur « 1 » pour l'instance 6. Mais dans le second cas, mesure ne dénote pas un objet de façon univoque : pour une même instance 8, $\delta($mesure$) = 8$ ou $\delta($mesure$) = 16$. Les trois groupes ont utilisé mesure, mais ils n'ont pas encore construit totalement le concept de paramètre.

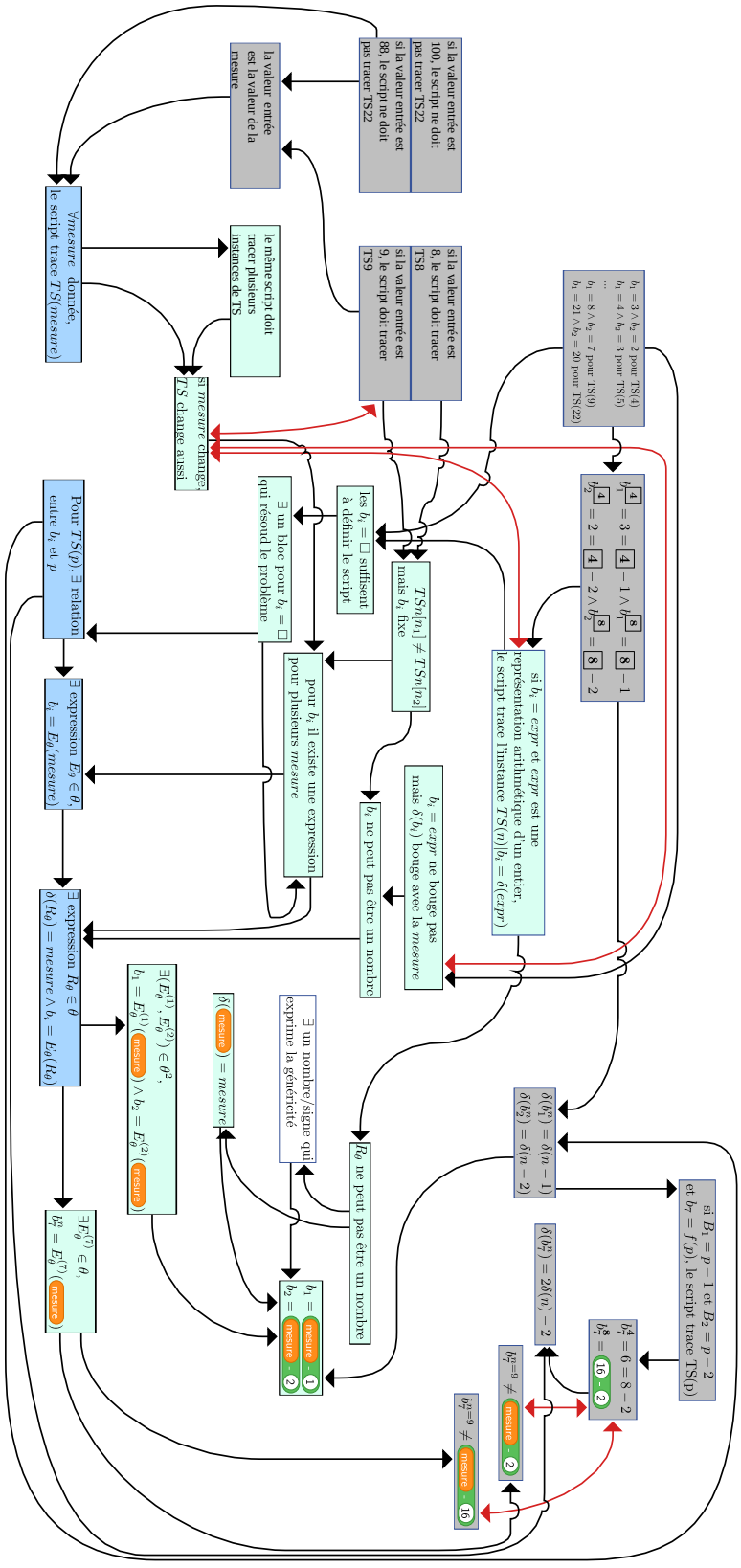


Figure 3.78 – Espace des faits-contraintes - 46m

3.6 Groupement 5

3.6.1 Groupe 45m

Description de l'activité

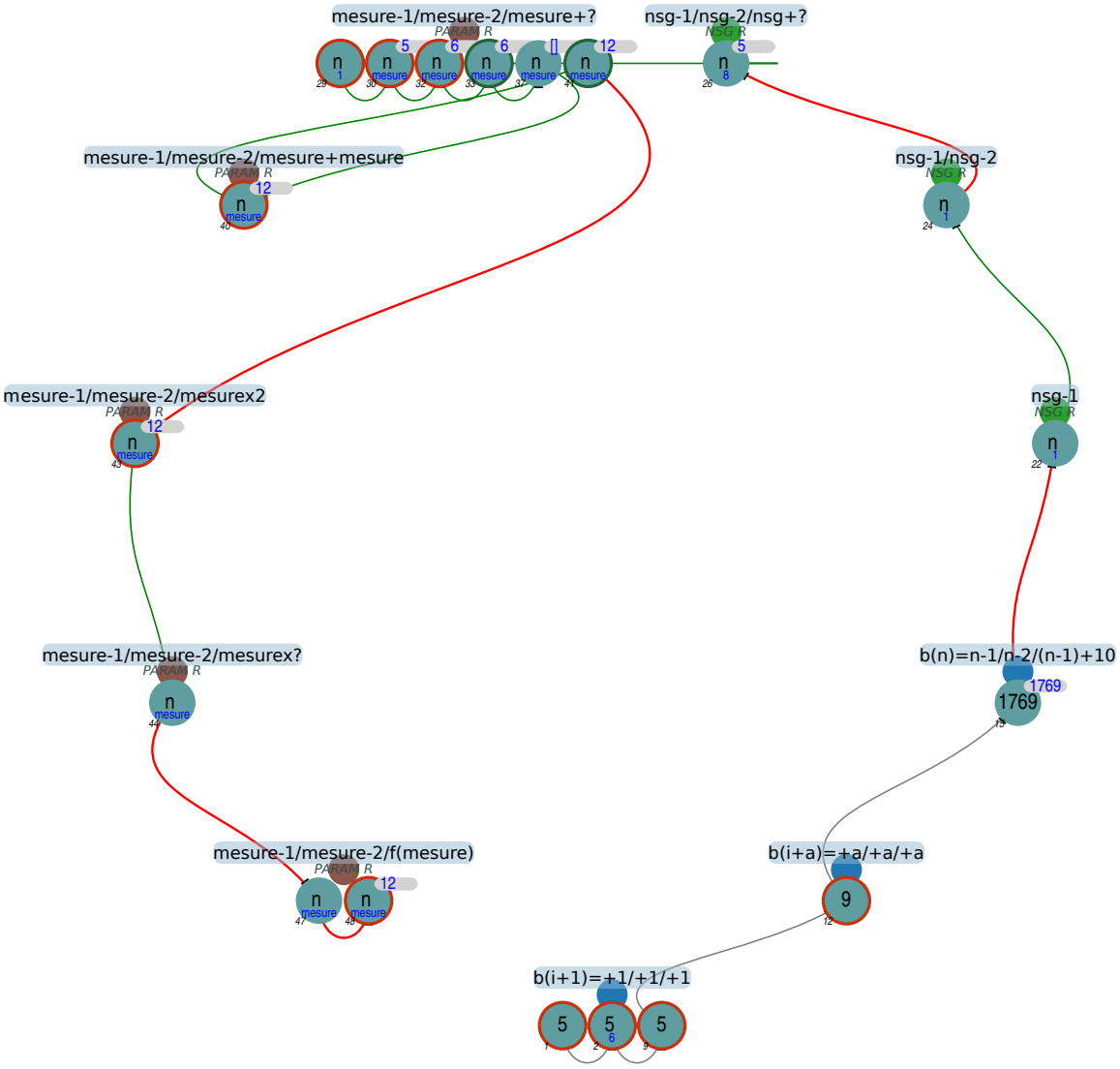
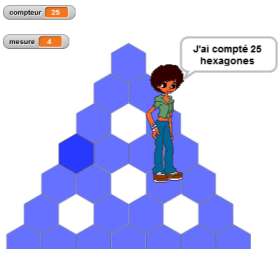


Figure 3.79 – Organisation de l'activité - 45m

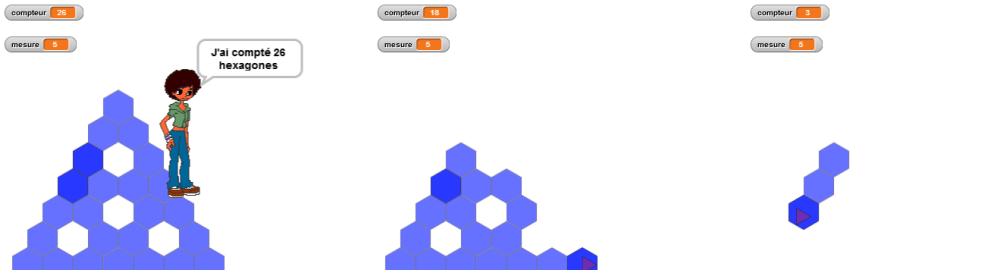
Le groupe 45m a un parcours atypique : après avoir été peu efficace en séance 3, ne parvenant pas à construire une instance valide, ils vont construire les instances traçant les TS5, puis TS9 et TS1769 en séance 4. Lorsque le problème de la généralisation du script se posera en séance 5, ils passeront par un nombre-signe générique (« 0 »), qui se verra substitué par la variable *mesure*. Comme de nombreux groupes, ils réussiront à formuler le lien entre les familles de boucles B_1 et B_2 et la mesure, mais ne parviendront pas à une expression valide de b_7.

Construction du TS5 (étapes 1-11)

En séance 3, le groupe 45m ne commence pas par dupliquer le script du $TS(4)$. Ils modifient directement ce dernier afin de tracer un TS5. Contrairement à la plupart des groupes, ils commencent par ne modifier que la boucle b_1, en augmentant son nombre de répétitions de un. Après quelques tests infructueux de lancement avec le drapeau vert, comme le groupe 45f (, p. 275), ils obtiendront le résultat suivant (45m, S3, 2'58) :

$$TS(4)[] \rightsquigarrow \left(\text{  } \text{} , 25, 4 \right)$$

Ce script est alors dupliqué sous l'entête du script dont la tâche prescrite est de tracer un TS5. On peut se demander s'ils considèrent le tracé comme valide ou invalide : est-il vu comme valide, et donc le groupe passe à l'instance suivante ; ou est-il vu comme invalide et donc ces élèves, qui se sont rendus compte qu'ils ne modifiaient pas le script attendu, dupliquent puis corrigent ? Ils vont alors ajouter de nouveau une itération à la boucle b_1. Trois tests seront lancés, le deuxième étant arrêté manuellement après la boucle b_5, et le troisième en cours d'exécution de la boucle b_1 (figure 3.80, p. 394) : cela laisse penser qu'ils considèrent le tracé comme erroné. Ces



(a) Script complet **(b)** b_6 en cours **(c)** b_1 en cours

Figure 3.80 – Tests du script {5/2/2/2/2/3/6/2} (45m, S3)

rétroactions montrent que la boucle b_1 trace trop d'hexagones par rapport aux autres boucles, mais le groupe 45m ne l'interprète pas ainsi : ils augmentent de nouveau de un le nombre de répétitions de b_1. Il est difficile d'établir la raison de cette interprétation. Peut-être les élèves estiment-ils, suite à la rétroaction de la figure 3.80b (p. 394), que la boucle b_1 devrait déjà avoir tracé l'ensemble du grand côté gauche du TS. Le script est testé deux fois de suite, dont la première en plein écran (45m, S3, 6'38 et 7'17).

$$TS(5) \rightsquigarrow \left(\text{  } \text{} , 27, 5 \right)$$

Ce résultat semble invalider leur dernière action, qu'ils annuleront en diminuant cette fois le nombre de répétitions de b_1, obtenant une rétroaction identique à la figure 3.80a (p. 394). Le script sera exécuté deux fois avant la fin de la séance.
En début de séance 4, de façon plus classique cette fois, les élèves du groupe 45m dupliquent le TS4 et augmentent de une itération toutes les boucles. Ils ajusteront la boucle b_7, justement considérée comme erronée, d'abord en lui enlevant une itération — puis en l'augmentant de deux, obtenant un TS5 valide.

Construction du TS9 (étapes 12-13) Le groupe 45m va alors construire une nouvelle instance en préparant le script $S = \{8/7/12\}$. Si les élèves se sont basés sur le script précédent, ils ont donc ajouté quatre itérations à chacune des boucles. Comme le script est sous l'entête censée tracer un TS6, le tracé sort de l'espace d'exécution (45m, S3, 5'17) :

$$S \rightsquigarrow \left(\text{  } \text{} , 65, 6 \right)$$

Les élèves de ce groupe vont alors correctement identifier la localisation de l'erreur et sa raison, ils corrigeront $b_7 \leftarrow 16$ et obtiendront un tracé dont la forme et le nombre d'hexagones sont valides, mais dont le tracé sort de l'espace d'exécution (45m, S3, 5'50).

$$S \rightsquigarrow \left(\text{  } \text{} , 69, 6 \right)$$

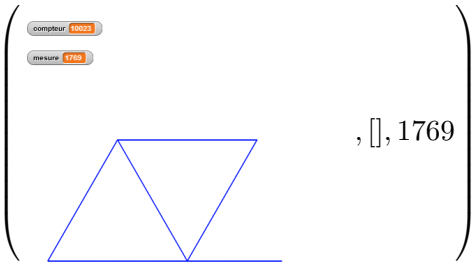
Même si les élèves ne disposent de suffisamment d'informations visuelles pour déterminer la validité du tracé, ils semblent considérer celui-ci comme valide puisqu'ils passent à une nouvelle instance. À ce stade, les élèves n'ont pas forcément fait le lien entre la mesure du TS voulu, la mesure affichée, et le nombre de répétitions des boucles : rien ne nous dit qu'ils soient conscients de construire le TS9. Ils construisent, en tout cas, un TS « plus grand ».

Construction d'une certaine instance : TS1769 (étapes 14-18) Le groupe 45m va alors construire une instance encore plus grande qui devrait tracer un TS1769, et leur activité montre qu'ils associent leur but à la mesure, et au moins la boucle b_1 à ce but. En effet, ils commencent par exécuter le script générique et, comme pour d'autres groupes, l'invite demandant d'entrer une valeur semble les avoir perturbé : ils mettront 12 secondes avant de se décider pour la valeur « 1769 » (45m, S4, 6'22-6'34). Ils vont alors copier leur script $TS(9)$ sous l'entête du script générique, puis modifier les boucles. Ils passeront ainsi :

— les boucles de la famille B_1 de 8 (pour l'instance 9) à 1768 (pour l'instance 1769) ;
— les boucles de la famille B_2 de 7 (pour l'instance 9) à 1767 (pour l'instance 1769) ;

— et la boucle b_7 de 16 (pour l'instance 9) à 1778 (pour l'instance 1769) [73] ;

Ils lanceront alors ce script, en entrant comme valeur de la mesure demandée, une nouvelle fois, « 1769 » (45m, S4, 10'51). Les élèves, n'ayant pas le temps de tester ce script en fin de séance 4, le feront en début de séance 5, de nouveau avec une entrée de 1769 (45m, S5, 3'16). Ils arrêteront l'exécution avant que la boucle b_7 ne commence, estimant sans doute que la durée du tracé est rédhibitoire : le script est en cours d'exécution depuis presque six minutes (45m, S5, 3'16-9'06). La portion de tracé effectuée semble leur suffire pour valider le script, puisqu'ils se baseront ensuite dessus.

$TS(n)[1769] \rightsquigarrow$ 

$, [], 1769$

La valeur entrée avant de créer le script et la valeur entrée après avoir créé le script sont cohérentes avec les valeurs du nombre de répétitions des boucles, tout du moins pour les boucles B_1 et B_2 : on a bien $b_1^{1769} = 1769 - 1 = 1768$ (45m, S5, 3'16,). L'origine de la valeur 1769 n'est pas identifiée. On peut seulement signaler qu'elle est du même ordre de grandeur que le nombre d'hexagones ciblés dans le problème qui est donné à la classe :« Si on dispose de 1400 hexagones, quel est le plus grand triangle de Sierpinski que l'on peut obtenir ? Si on dispose de 1654 hexagones ? Si on dispose de 1710 hexagones ? ».
Le fait de modifier b_1 après avoir choisi une valeur pour la mesure est un indice que le lien se fait dans le sens $mesure \rightarrow b_1$, et non l'inverse : autrement dit, ces élèves déterminent la valeur du nombre de répétitions de b_1 en fonction de la mesure. L'inverse aurait aussi été possible : une fois le script écrit, la valeur de la mesure à entrer est déterminée par les élèves en fonction du nombre d'itérations de b_1. Si l'on considère que les élèves ont déterminé la mesure avant le nombre d'itérations, alors on considère aussi que ces élèves font une généralisation algébrique naïve : ils cherchent à construire une certaine instance.

Vers la généralisation (épisodes 19-29) Le groupe 45m va alors se poser le problème de la généralisation, en commençant par effacer l'expression déterminant le nombre de répétitions de la boucle b_1. Les élèves vont alors modifier leur script pour tenter de le rendre générique. Ils vont ainsi effectuer les actions suivantes :

— pour $b_i \in B_1, b_i \leftarrow$ (0 - 1) ;
— pour $b_i \in B_2, b_i \leftarrow$ (0 - 2) ; dans un premier temps, ils oublieront de compléter b_4 ($b_4 \leftarrow$ (0 - ○)) ;
— $b_7 \leftarrow$ (0 + 9).

Ces élèves sont ainsi en recherche d'une formalisation générique permettant de déterminer à l'exécution le nombre de répétitions de chacune des boucles. Le signe « 0 » a ici un statut de nombre-signe générique : il indique ce qui devrait être remplacé par la valeur entrée, c'est un marque-place. En outre, ce nombre, le zéro, est très particulier, et il n'a pris un statut de nombre que tardivement dans l'histoire. Les groupes 46k, 46c, 46i ou encore 45k utiliseront aussi ce signe,

73. Voir (p. 398) pour des hypothèses sur l'origine de cette valeur.

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [0 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [0 − 2] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [0 − 2] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [0 − 2] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [0 − 2] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [0 − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [0 + 9] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [0 − 2] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(a) NSG

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 2] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [*mesure* − 2] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [*mesure* − 2] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [*mesure* − 2] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* + 9] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 2] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(b) mesure

Script 3.16 – Vers la généralisation (45m, S5)

et le groupe 46e en commentera la raison (NSG, p. 424). Le script 3.16a (p. 397) obtenu [74] sera alors testé avec une valeur de 5, ce qui produira le résultat suivant (45m, S5, 13'55) :

$$TS(n)[5] \rightsquigarrow \left(\text{  } \text{} , 11, 5 \right)$$

74. Les élèves ont été confrontés à un défaut du dispositif de captation. Des événements n'ont pas été enregistrés (à partir de l'événement 89807, 13'03). Les élèves, sur notre suggestion, ont sauvegardé puis rechargé leur travail. Cette sauvegarde a permis de confirmer que la boucle b_4 avait été elle aussi définie à (0 - 2) avant l'exécution.

Le statut erroné de leur tracé, bien visible, les amènera à modifier le script en s'interrogeant sur la première boucle. Ils effaceront ainsi l'expression déterminant le nombre de répétitions de cette dernière : $b_1 \leftarrow$ 

. Puis, un peu moins de deux minutes plus tard, ils inséreront, avant la boucle b_1, un bloc qu'ils ont créé : 

. Précisons bien que le *nom* de ce bloc, de cette instruction, est *la chaine de caractères* « nombre précédent - 1 », il n'y a pas d'opérateur ici [75]. Ces deux modifications ne sont pas testées.
Après avoir annulé cette dernière modification, ils vont remplacer tous les signes « 0 » par 

, obtenant le script 3.16b (p. 397). Ils exécuteront leur script avec une valeur de 5, puis, constatant que le tracé sort de l'espace d'exécution, une valeur de 6 :

$$TS(n)[6] \rightsquigarrow \left(\text{ 

} , 47, 6 \right)$$

Vers la généralisation : recherche de b_7 (étapes 30-49) Les élèves chercheront alors à corriger b_7, en effectuant les actions et tests suivants (tableau 3.6) :

b_7	**mesure**	Valide ?
$mesure + 6$	6	Non
$mesure + 4$	6	Oui
$mesure + 4$	12	Non
$mesure + 8$	12	Non
$mesure + mesure$	12	Non
$mesure + 10$	12	Oui
$mesure + 10$	24	Non
$mesure \times 2$	12	Non
	12	Non
	12	Non

Tableau 3.6 – Recherches pour b_7 (45m, S5)

La fin de séance empêchera le groupe de poursuivre ses recherches.

Recherche de régularités

Des TEA non stabilisés Le groupe 45m commence par appliquer le TEA $+1/+1/+1$, d'abord uniquement sur b_1, puis, en début de séance 4, sur toutes les boucles. Pour passer du TS5 au TS9, ils mettront en œuvre une variante plus étendue de ce TEA : ils ajouteront 4 à chaque nombre de répétitions des boucles, soit le TEA $+a/+a/+a$. Ce TEA est présent aussi chez les groupes 45a ou 45e (, p. 357). Une correction rapide, ajoutant de nouveau 4 (donc a pour ce cas) sera apportée.

75. En outre, les élèves n'ayant pas la possibilité de modifier les blocs créés, cette instruction est une instruction qui ne fait rien, une *NOP* (« no opération »).

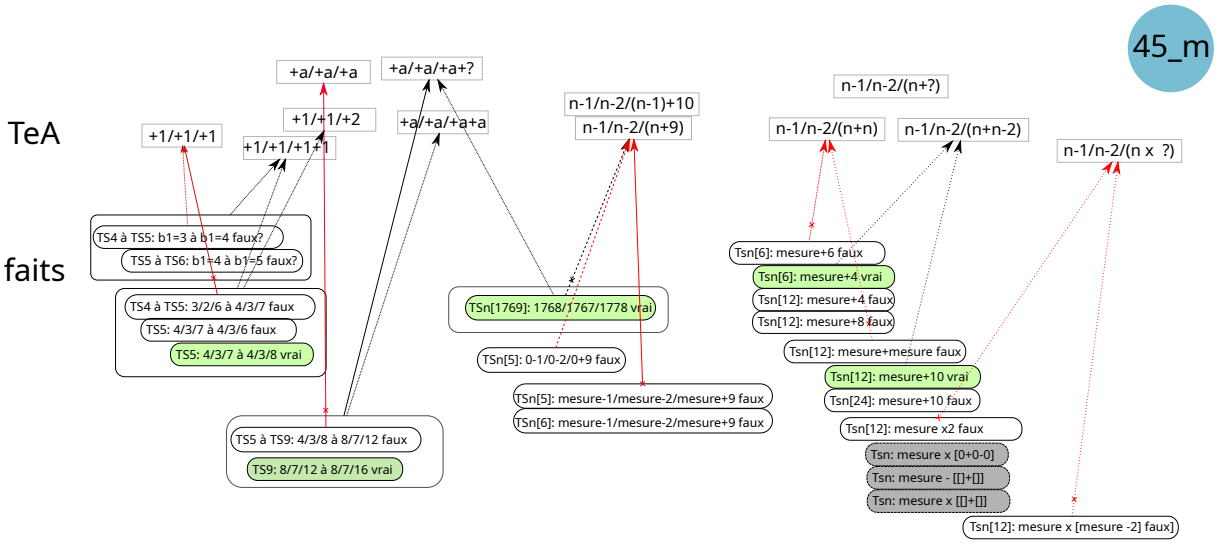


Figure 3.81 – Faits et TEA, groupe 45m

Un TEA très proche est susceptible de diriger l'activité des élèves lorsqu'ils construisent l'instance traçant le TS1769. Passant de 8 à 1768 pour les boucles B_1 et de 7 à 1767 pour les boucles B_2, et de 16 à 1778 pour b_7, ces élèves semblent mettre en œuvre le TEA $+a/+a/+a+2$: les faits qu'ils ont construits peuvent laisser penser qu'il faut ajouter une certaine valeur à chacune de boucles, et « un peu plus » pour b_7. Cependant, il est aussi possible que ces élèves, cherchant ici à construire une instance particulière, envisagent une démarche plus générique, même si elle n'est pas formalisée. Comme on l'a vu, les élèves ont établi un lien entre la mesure du TS et la première boucle, et ont choisi la mesure 1769 avant même de commencer à construire leur script : ils sont dans une phase de généralisation algébrique naïve, et sont capables de trouver b_1 et b_2 à partir de la mesure. En revanche, aucune régularité n'est établie au niveau des faits concernant b_7 : les corrections apportées étaient d'abord de 1 pour le passage TS-TS5, puis de 4 pour le passage TS5-TS9. Pour le TS1769, soit les élèves ajustent par anticipation en ajoutant 2 — mais on peut se demander d'où vient cette valeur —, soit ils envisagent une relation qui leur parait possible entre la mesure et le nombre d'itérations de b_7. Dans ce dernier cas, deux relations nous semblent possibles :

1. $b_7^{1769} = 1769 + 9$;
2. $b_7^{1769} = b_1^{1769} + 10$

Les relations déja existantes entre les boucles b_7 et b_1 d'une même instance sont $b_7 = b_1 + 4$ et $b_7 = b_1 + 8$. Ainsi, l'hypothèse 2 nous semble plus probable, étant plus proche d'un motif régulier que l'hypothèse 2. Cependant, rien ici ne semble stabilisé, et les élèves vont ensuite réécrire $b_7^{1769} = 0{+}9$. L'opérateur « +9 » pourrait être une formalisation de leur TEA, en cas d'hypothèse 1 valide. Mais il pourrait aussi être le résultat d'une recherche de formalisation de ce qui leur parait être valide [76] : ils recherchent une relation entre 1769 et 1778, et $1778 = 1769 + 9$ est une formulation possible.

76. Rappelons qu'ici $b_7 = 1778$ est erroné, mais il semble que les élèves le considère comme valide, n'ayant pas laissé le tracé se déroulé jusqu'au bout.

Il est ici difficile de se prononcer entre l'application d'un TEA permettant de passer d'une instance à une autre, avec des relations inter-objectales, ou un TEA fruit d'une généralisation algébrique naïve qui cherche à se formaliser, avec des relations trans-objectales et intra-objectales. Il est possible que la construction de cette instance lointaine soit la manifestation d'un mouvement de la construction d'une instance à partir d'une autre vers la construction directe, en fonction de la mesure, d'une instance particulière.

Peu de faits, mais un faux-fait Quoiqu'il en soit, le groupe 45m a produit peu de faits relatifs à la construction de b_7, et cela se verra lorsqu'ils vont avancer vers la généralisation algébrique formalisée. Parmi ces faits, l'un d'entre eux est un faux fait : lors que la création du TS1769, les élèves ont arrêté l'exécution avant que la boucle b_7 ne soit exécutée, obtenant la rétroaction :

$$TS(n)[1769] \rightsquigarrow \left(\text{compteur } 10023 \quad \text{mesure } 1769 \quad , [], 1769 \right)$$

Cette rétroaction donne effectivement tous les signes d'une validité en cours. Mais dans ce script, $b_7^{1769} = 1778$, alors qu'on devrait avoir $b_7^{1769} = 3536$. Du point de vue du chercheur ou du dispositif, « ce script est invalide » est un fait premier, mais du point de vue de l'élève, c'est « ce script est valide » qui est un fait premier. Ce faux fait est notamment mobilisé dans l'écriture $b_7 = 0 + 9$, l'ajout de la valeur 9 provenant de la relation $b_7^{1769} = 1778 = 1769 + 9$. Il sera un obstacle à la détermination d'un schéma permettant de déterminer le nombre de répétitions de b_7.

Généralisation algébrique

Figure 3.82 – Type de généralisation - 45m

Représentation des relations boucle-mesure : NSG et paramètre Le groupe 45m, contrairement au groupe 46m, ne passe pas par un nombre générique pour exprimer les relations entre nombre d'itérations des boucles et mesure. Ils partent de l'instance particulière qui parait valide (TS1769), et cherchent à exprimer les relations non pas par rapport à cette instance, mais par rapport à n'importe quelle instance. Ils utilisent alors le nombre-signe générique « 0 » pour signifier ce qu'il faut changer, comme les élèves du groupe 46m ont utilisé l'espace vide : $b_1 =$ (0 - 1). Cependant, là où ces derniers ont simulé ce qui devrait se passer lorsqu'on entre une certaine mesure, les élèves du groupe 45m ont testé le résultat produit. Ici, le « 0 » n'est pas qu'une marque indiquant ce qu'il faut changer, c'est aussi un signe supposé permettre à l'ordinateur de

« se débrouiller tout seul ». Ce statut nous parait être confirmé par l'instruction nombre précédent - 1, dont nous parlerons par la suite. Ici, il s'agit bien d'expliquer à l'ordinateur qu'il faut prendre « le nombre précédent » : cela non seulement indique une place dans l'expression, mais a aussi une fonction de désignation, univoque, d'un objet — c'est *cette valeur-là* qu'il faut utiliser. Les élèves cherchent donc :

— à exprimer les relations entre mesure entrée et boucles, relation qui prend la forme $\square - 1$ pour b_1 par exemple,

— à trouver une expression R_θ de θ qui permet de dénoter la valeur de la mesure entrée,

— et à exprimer cette relation E_θ en manipulant R_θ, comme si cette dernière était connue.

Puisque $R_\theta =$ "0" donne un résultat invalide, les élèves vont mobiliser le fait, partagé ou construit, « la valeur de la mesure entrée est stockée dans mesure », et remplacer « 0 » par mesure. Ils peuvent alors constater la validité de cette solution pour les familles B_1 et B_2, mais pas pour b_7.

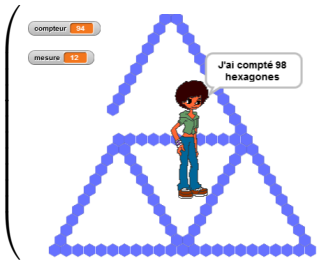
Représentation des relations boucle-mesure : recherche de b_7 Contrairement au groupe 46m, les élèves du groupe 45m n'ont pas construit suffisamment de faits concernant b_7 pour leur permettre d'identifier un schéma. En outre, un des faits construits est un faux-fait : ce binôme considère que leur script traçant un TS1769 est valide, donc que $b_7^{1769} = 1778 = 1769 + 9$ est vrai. Comment, à partir de ces faits (tableau 3.7), pourraient-ils identifier un schéma qui s'étendrait à toutes les mesures ?

mesure	b_1	b_7	**expression de b_7**
5	4	8	
9	8	16	
1769	1768	1778	$mesure + 9$
6	5	10	$mesure + 4$
12	11	22	$mesure + 10$

Tableau 3.7 – Faits concernant $b7$ considérés vrais (45m, S5)

Vers la généralisation : recherche d'un bloc Comme les élèves du groupe 46m, les élèves du groupe 45m semblent, à deux occasions au moins, être à la recherche d'un bloc qui résoudrait la tension fondamentale entre expression immuable et nécessité de plusieurs tracés. Deux blocs ont ainsi été créés et utilisés : nombre précédent - 1 et 0+0-0. Ces deux expressions de θ sont la manifestation de la recherche d'une façon d'expliquer à l'ordinateur, en quelque sorte, ce qu'il doit faire pour « lui donner des instructions pour qu'il puisse changer bah la valeur de ces nombres correctement » (45 Prof, S5, 12'33).

Dans nombre précédent - 1, le « nombre précédent » qu'évoquent ici les élèves est sans doute le nombre entré « juste avant » : c'est une façon de dire que l'ordinateur doit prendre le dernier nombre entré et lui enlever un, pour trouver b_1. On retrouve ici encore une trace de pensée magique : les élèves semblent estimer que cela va suffire pour que l'ordinateur fasse ce qu'on attend de lui. La seconde expression, 0+0-0, mobilise de nouveau le signe « 0 » comme nombre-signe générique, mêlé au paramètre *mesure*. Lorsque les élèves font $b_7 \leftarrow$ mesure × 0+0-0, ils sont en train de chercher à formaliser le fait que, pour trouver le nombre d'itérations de b_7, il faut faire un peu moins que le double de la mesure : ils ont déjà testé $mesure + mesure$, et $mesure \times 2$ juste avant cette expérimentation, avec un tracé de b_7 trop long. Avec $mesure \times 2$, ils ont ainsi obtenu :

$TS(n)[12] \rightsquigarrow$ 

$, 98, 12$

Ici, cette expression pourrait exprimer « Avec la mesure, additionner [quelque chose] à [quelque chose] et enlever [quelque chose] ». Une version valide serait alors : « Avec la mesure, additionner [la valeur de la mesure] à [la valeur de la mesure] et enlever [deux] ». Les [quelque chose] — les « 0 » dans le bloc créé par les élèves — ne sont pas univoques : comme pour le groupe 46m, ce groupe utilise une *marque* et non un *signe* (voir note 71, p. 386). Il est possible que (mesure) soit univoque pour ces élèves, mais ils ont besoin d'un autre objet traduisant une variation : les faits leur montrent que les écritures des nombres présents dans les expressions de b_7 valides ne sont jamais les mêmes ($b_7^6 = mesure + 4$, $b_7^{12} = mesure + 10$ ou le faux-fait $b_7^{1769} = mesure + 9$). Dans ce cas, on peut penser que les élèves sont en train de construire le concept de paramètre, mais il leur manque la relation entre b_7 et ce paramètre pour aboutir. Comme pour d'autres groupes, on constate aussi ici que cette relation est cherchée entre des nombres, mais sans mettre au travail le sens de ces nombres dans le contexte algorithmique du tracé : les élèves essayent d'ajuster la formule pour un certain TS, mais ne sont pas à la recherche d'un schéma répétitif dans l'algorithme (ou le tracé). On peut toutefois noter qu'en fin de séance ces élèves envisagent des expressions de b_7 qui sont proches du double de la mesure, et la boucle b_7 trace un nombre d'hexagones proche du double de la mesure. Le bloc (0+0-0) peut ainsi laisser penser qu'il s'agit de faire le double de la mesure « moins quelque chose », et l'expression testée ensuite, (mesure × mesure - 2) était peut-être censée s'écrire (mesure + mesure - 2), ce qui aurait été valide. Il y a peut-être ici, comme pour le groupe 46m, un problème d'ordre instrumental : comment transcrire la structure profonde représentée par $\square + \square - \square$ dans θ ? Snap ! ne manipule que des opérateurs arithmétiques unaires ou binaires (d'arité 1 ou 2) [77], il est impossible d'écrire (( ) + ( ) - ( )). Cela impose de définir la structure profonde, pour écrire soit (( ) + ( ) - ( )) (donc $(\square + \square) - \square$), soit (( ) + ( ) - ( )) (donc $\square + (\square - \square)$). Sans les interactions langagières entre les élèves, il est cependant impossible de se prononcer.

Construction des nécessités

Le groupe 45m se base sur peu de faits, mais semble construire les nécessités attendues. Le problème de la généralisation est rapidement posé, les élèves cherchent très rapidement à formuler les relations, qu'ils ont donc sans doute déjà établies, entre boucles et mesure du TS voulu. Les nécessités locales semblent être mobilisées sans qu'il y ait eut besoin de construire d'autres faits. Elles apparaissent ainsi comme des conditions du problème qui sont liées à la position du problème, et non à la production de faits. Ils recherchent donc une expression de θ qui représenterait cette relation.

77. La version 8.2.3 permet cependant d'augmenter le nombre d'opérandes de certains opérateurs, permettant ainsi des écritures telles que (( ) + ( ) + ( )) ou (( ) × ( ) × ( )).

Les élèves semblent chercher une représentation R_θ de la mesure qui soit un nombre et à la fois autre chose, un signe de généricité. Le zéro, ce nombre qui n'en est pas tout à fait un, et qui est « un multiple de tout » comme le dit une élève du groupe 46e (transcription 3.30, p. 425), est ainsi envisagé comme signe générique. Cette possibilité invalidée, ils vont rapidement mobiliser le seul signe de θ qui leur est accessible et qui a un rapport reconnu avec la mesure, mesure. La relation entre b_7 et la mesure n'est pas identifiée par les élèves, et sans doute pas identifiable s'ils ne défactualisent pas $b_7^{1769} = 1778$, c'est-à-dire s'ils ne finissent pas par considérer ce fait comme erroné. Ils produiront ainsi de nombreux nouveaux faits concernant le rapport entre b_7 et la mesure, sans aboutir, même s'ils se rapprochent de l'expression attendue.
Les élèves du groupe 45m construisent ainsi de nombreux faits en explorant les expressions possibles, mais en n'y mettant que peu de sens.

Bilan

Le groupe 45m, après un début peu efficace en séance 3, a rapidement posé le problème de la généralisation, mais sans avoir établi suffisamment de faits concernant b_7. La construction du paramètre semble avancée : les élèves ont bien établi la nécessité d'une expression de θ manipulant une représentation R_θ de la mesure. Cette représentation parait univoque, considérant le lien effectué par les élèves avec *le nombre précédent*, la valeur qui vient d'être entrée en début de script — en ce sens, c'est bien une valeur unique. Une fois encore, la recherche de relation en se basant uniquement, ou presque, sur les nombres, ne permet pas aux élèves d'avancer sur b_7. La proximité qu'ils semblent entrevoir entre b_7 et le double de la mesure, qui les amène à des expressions proches de celles attendues, peut être issue tout à la fois des nombres ou de l'observation de la partie tracée par b_7.

3.6.2 Groupe 46e

Le groupe 46e, dont une captation vidéo et audio est disponible, montre la construction progressive des nécessités aboutissant au paramètre. Il permet en outre de comprendre la difficulté d'établir une relation pour b_7 en ne se basant que sur des nombres, c'est-à-dire sans faire jouer les changements de cadre (Douady, 1984).

Description de l'activité

Les élèves du groupe 46e vont construire peu à peu les instances demandées. La construction du script générique les amènera rapidement vers le paramètre, avec une longue recherche pour établir la relation entre la boucle b_7 et la mesure.

Construction des différentes instances demandées : du tracé vers les nombres (étapes 1-22) Le groupe 46e commence, lui aussi, par dupliquer le script $TS(4)$ puis augmenter la valeur du nombre de répétitions de chacune des boucles de un, avant d'ajuster la boucle b_6. L'erreur constatée est rapidement corrigée (« non c'est c'est l'autre d'après », 46e, transcription S3, 3'20), et b_7 est ajustée pour obtenir un TS5. Les élèves vont ensuite s'attacher aux tâches suivantes, en utilisant le raccourci sémiotique déjà évoqué : « maintenant faut faire le "c" et le "d" et le "e" » (46e, transcription S3, 6'56). La construction de l'instance suivante voit de nouveau l'augmentation du nombre de répétitions de chaque boucle de un : E2 propose « on rajoute une fois je crois » (46e, transcription S3, 7'12), et E1 fait les modifications. Arrivée à la boucle b_7,

elle met en doute la pertinence de ce choix : il faudrait faire « plus deux, parce que là [passage de TS4 à TS5] on avait fait plus deux tout à l'heure » (3.21, 256-262). E2 n'étant pas convaincue, le test se fera sans cette modification, puis b_7 sera correctement ajustée. Le script traçant le TS7 sera fait en appliquant cette fois une augmentation de un pour chaque boucle, et de deux pour b_7. Pour la construction du script traçant un TS11 (étapes 9-10 et 21-22), les élèves de ce groupe vont ajouter quatre à chaque nombre de répétitions des boucles : en comptant sur ses doigts, E1 cherche à passer de la mesure du dernier script tracé (7), à la mesure voulue (11) (3.22, 352-354), et en conclut « il faut faire plus quatre ». Cette modification s'appliquera aussi à b_7 (alors que l'augmentation pour cette boucle aurait dû être de huit). Avant même la fin de l'exécution du test, le groupe 46e veut commencer à construire le script traçant un TS22 (étapes 11-20). E1, constatant que « celui [le script] d'à côté fera [...] vingt-deux », en déduira, en montrant le script $TS(11)$: « du coup c'est fois deux à chaque fois » (46e, transcription S3, 13'02-13'06). La fin de l'exécution de $TS(11)$ produisant une invalidation de leur script (46e, S3, 13'10), cette modification du script $TS(22)$ sera repoussée.

$$TS(11)[] \rightsquigarrow \left(\text{  } \text{} , 83, 11 \right)$$

Une fois la boucle b_7^{11} ajustée en lui ajoutant deux (ce qui est erroné), le binôme modifie les valeurs du nombre de répétitions des boucles comme évoqué : $b_1^{22} = b_1^{11} \times 2 = 20$, $b_2^{22} = b_2^{11} \times 2 = 18$, etc (46e, S3, 14'08-14'58). Pour b_7 en revanche, une autre méthode est appliquée, puisque au lieu de la valeur 36 attendue (18×2), le groupe 46e effectue l'action $b_7^{22} \leftarrow 38$. Ces élèves ont dû ajouter deux à leur première proposition pour la boucle b_7^{11} : « attends douze treize quatorze quinze seize plus deux oui dix-huit » (46e, transcription S3, 13'27-13'30). Ils vont de nouveau ajouter deux pour le script $TS(22)$: $b_7^{22} = b_7^{11} \times 2 + 2 = 38$. En effet, lors de la modification de b_7 pour construire le TS22, E1 souligne dans un premier temps que « mais là du coup on sait pas » (46e, transcription S3, 14'39). E2 va alors commencer par multiplier 18 par 2 : « attends, dix-huit

249	08 :15.533	VALEURS	(BOUCLE)b7i0(VAL) : répéter *9* fois (commandes)<<8>>
250	08 :17.469	T1	mais là il faudrait peut-être changer de
251	08 :18.913	T1	deux fois
252	08 :18.913	G1	2 avec les doigts
253	08 :19.434	T1	il faudrait faire
254	08 :19.434	G1	montre l'écran
255	08 :19.911	T1	fois deux
256	08 :19.911	G1	deux avec les doigts
257	08 :20.867	T1	parce que euh
258	08 :20.867	G1	montre tracé
259	08 :21.492	T1	'fin plus deux
260	08 :21.492	G1	deux avec les doigts
261	08 :22.937	T1	parce que là on avait fait plus deux tout à l'heure
262	08 :22.937	G1	montre alternativement b7 sur TS4 et TS5
263	08 :25.738	T2	mouai
264	08 :25.984	T1	on verra après mais on
265	08 :27.842	T1	déjà on essaye de faire ça
266	08 :28.666	VALEURS	(BOUCLE)b8i0(VAL) : répéter *4* fois (commandes)<<3>>

Transcription 3.21 – 46e, S3 : Un doute sur b_7

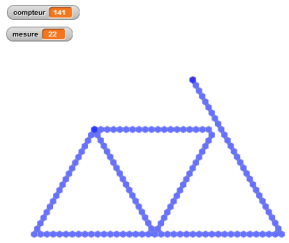
351	11 :07.773	T1	du coup il faut qu'on change là cette fois ci
352	11 :09.995	T1	faut faire euh huit neuf
353	11 :09.995	G1	compte sur ses doigts
354	11 :12.569	T1	attend | neuf dix onze
355	11 :12.569	G1	compte sur ses doigts
356	11 :15.020	T1	faut faire plus quatre
357	11 :16.440	T1	six | sept huit neuf dix
358	11 :16.440	G1	compte sur ses doigts
359	11 :21.693	T1	cinq plus euh cinq plus quatre
360	11 :21.978	VALEURS	(BOUCLE)b1i0(VAL) : répéter *10* fois (commandes)<<6>>
361	11 :27.282	VALEURS	(BOUCLE)b2i0(VAL) : répéter *9* fois (commandes)<<5>>
362	11 :27.445	T1	neuf
363	11 :29.105	VALEURS	(BOUCLE)b3i0(VAL) : répéter *9* fois (commandes)<<5>>
364	11 :29.655	T1	neuf
365	11 :30.954	VALEURS	(BOUCLE)b4i0(VAL) : répéter *9* fois (commandes)<<5>>
366	11 :31.365	T1	neuf
367	11 :32.577	VALEURS	(BOUCLE)b5i0(VAL) : répéter *9* fois (commandes)<<5>>
368	11 :32.847	T2	dix
369	11 :34.811	VALEURS	(BOUCLE)b6i0(VAL) : répéter *10* fois (commandes)<<6>>
370	11 :36.662	T1	douze treize quatorze quinze seize
371	11 :36.662	G1	compte sur ses doigts
372	11 :41.523	VALEURS	(BOUCLE)b7i0(VAL) : répéter *16* fois (commandes)<<12>>
373	11 :42.421	T1	euh neuf du coup
374	11 :44.113	G1	regarde E2
375	11 :45.125	T2	oui voilà c'est bon
376	11 :46.002	T1	non avant | après
377	11 :46.002	G1	prend la souris
378	11 :48.366	T1	on va là
379	11 :49.026	VALEURS	(BOUCLE)b8i0(VAL) : répéter *9* fois (commandes)<<5>>
380	11 :49.095	EXECUTION	START receiveKey(475)
381	12 :00.526	T1	attend on a un problème
382	12 :02.392	EXECUTION	PAUSE receiveKey(475)
383	12 :02.921	T1	stop
384	12 :03.902	T1	on a oublié de prendre un truc
385	12 :06.180	T2	quoi
386	12 :07.476	T1	prendre la valeur
387	12 :08.699	T1	là
388	12 :08.775	VARIABLE	(i3) ** prend la valeur *0* [new loc :bottom]
389	12 :09.310	T2	ah oui
390	12 :10.508	T1	c'est pour ça que ça va trop loin regarde
391	12 :10.508	G1	montre compteur
392	12 :11.368	VARIABLE	(i3) *compteur* prend la valeur 0 [val_inputChanged <<>>]
393	12 :11.368	VALEURS	b0i3(VAL) : *compteur* prend la valeur 0<<>>
394	12 :12.317	T1	yen n'a pas soixante-seize déjà

Transcription 3.22 – 46e, S3 : TS11 et compteur

plus dix-huit c'est | seize, trente-six » et E1 complètera « trente-six plus deux fait trente-huit » (46e, transcription S3, 14'41-14'50). Malgré cela, E1 à un doute, puisqu'elle affirme de suite, et avec raison, « mais euh sauf qu'il va y avoir un problème ». Cette affirmation est sans doute un écho de la constatation de la persistance de l'erreur sur la boucle b_7 du script $TS(11)$, constatée en cours de modification du script $TS(22)$ (46e, S3, 14'22) :

$$TS(11)[] \rightsquigarrow \left(\text{  } \text{} , 85, 11 \right)$$

L'erreur cependant ne se situe pas qu'au niveau de b_7 (46e, S3, 16'09) :

$$TS(22)[] \rightsquigarrow \left(\text{  } \text{} , [], 22 \right)$$

Malgré l'envie de E1 de terminer, la fin de séance l'empêche de prolonger la réflexion, seule une modification $b_6^{22} \leftarrow 18$ est faite, mais non testée.

Correction des TS11 et TS22 : construction dynamique et algorithme (étapes 14-22)
Les vingt premières minutes de la séance 4 seront consacrées à la correction des scripts permettant de tracer les TS11 et TS22, scripts erronés repris de la séance 3.
Le groupe 46e va s'appuyer sur la structure de l'algorithme afin d'identifier la localisation des erreurs. En début de séance 3, la nécessité de modifier l'angle de rotation, les 

, est évoquée (3.23, 70) puisque « c'est plus grand » (3.23, 78). À ce stade, les élèves ne considèrent pas la conservation des angles par homothétie. Le doute est néanmoins présent : doit-on augmenter l'angle de rotation ou le diminuer ? (3.23, 77-79). E2 essaye de visualiser l'angle nécessaire en utilisant sa main comme axe (3.23, 80-86 et figure 3.85, p. 412), mais comme le doute persiste, ces élèves vont essayer « de mettre un nombre de plus » (3.23, 83). Comme ils vont parvenir à construire le TS5, E2 va constater « en fait les degrés ça change pas hein », ce qui sera confirmé par E1 : « oui ça change jamais » (46e, transcription S3, 7'57).
Le fait que les angles ne doivent pas changer est établi, mais les élèves sont régulièrement dans un cadre géométrique, et dynamique (algorithmique). Ainsi, en début de séance 3, E1 déplore « en fait c'est que là il tourne au mauvais moment » (46e, transcription S3, 5'42) : il y a une localisation spatiale de l'erreur (« là »), mais aussi temporelle (« au mauvais moment »). Cette évocation de la rotation est aussi visible en fin de séance 3 : E1 explique à l'enseignante que le problème vient du fait que « il se referme trop tôt » (46e, transcription S3, 17'12), en parlant de l'enchainement des tracés fait par b_7 et b_8, ce que l'enseignante reformule en « ça

70	02 :50.390	T1	il faut juste changer les nombres qui sont là
71	02 :53.637	T1	mais faut pas euh faut pas tourner dans le même sens non plus
72	02 :53.637	G1	moulinets avec les mains vers l'avant
73	02 :56.606	T1	faut faut pas tourner du même degré
74	02 :56.606	G1	montre instruction (début)
75	02 :58.773	EXECUTION	FIN receiveKey(481)
76	02 :59.724	T2	ben faut que
77	03 :00.360	T1	faut tourner un peu plus
78	03 :02.095	T2	ah ben oui parce que c'est plus grand
79	03 :02.780	T1	ou un peu moins
80	03 :05.095	T2	tu tournes euh un peu moins de
81	03 :05.095	G2	main verticale qui bascule à l'horizontal
82	03 :07.194	T1	mais au pire on essaye juste euh enlever les mêmes XXX
83	03 :09.774	T1	on fait 4 au pire on essaye de mettre un nombre de plus
84	03 :12.132	VALEURS	(BOUCLE)b1i0(VAL) : répéter *4* fois (commandes)<<3>>
85	03 :12.755	T2	nan faut tourner un peu plus
86	03 :12.755	G2	rotation main

Transcription 3.23 – 46e, S3 : Une question de rotation

descend trop vite » (figure 3.86, p. 413). Ces rotations et le moment où elles s'opèrent sont donc reconnues comme des étapes importantes, des points de repère. Le groupe 46e va s'en servir en séance 4 pour tenter de comprendre les erreurs de leur première tentative de $TS(22)$. Ainsi, après plusieurs modifications n'ayant pas permis d'aboutir à un tracé valide, ces élèves vont mettre en pause le script (figure 3.87, p. 413) et tenter de localiser le problème (donc de construire des faits tiers). L'élève E1 tente d'identifier la boucle erronée en suivant les « tourner », puisque le souci provient de « quand ça a tourné » entre b_2 et b_3 (3.24, 51-53). Elle cherche alors à suivre le déroulé de l'algorithme en fonction du moment des rotations (c'est-à-dire de leur place dans la séquence d'instructions formant le script), en écartant la rotation initiale, et visiblement en rassemblant la double rotation entre b_2 et b_3 en une seule (rotation 2, figure 3.87, p. 413). Pour E1, l'erreur se situe sur la troisième boucle puisqu'elle identifie trois rotations, et en déduit donc qu'il faut donc diminuer b_3 de un (3.24, 63-68). E1 voit ainsi le tracé non comme une figure géométrique statique, mais comme une figure dynamiquement créée par la suite des instructions du script : elle travaille alors sur l'algorithme.
En procédant par étapes, le groupe 46e parviendra ainsi à corriger le script $TS(22)$, puis le script $TS(11)$.

Vers la généralisation (étapes 23-71) En fin de séance 4, le groupe 46e va aborder le problème de la création du script générique. Les élèves vont dupliquer leur dernier script (traçant un TS22), et E1 tentera d'aider E2 à poser le problème. Ce point sera étudié plus précisément dans la partie 3.6.2, p. 435. En fin de séance, l'utilisation de la variable (mesure) sera expérimentée, dans une instruction (répéter mesure fois). Malheureusement, E1, lors des explications données à E2, a entré une chaine vide lors de l'invite (46e, transcription S4, 22'05), ce qui produit une erreur non interprétable par les élèves, et surtout persistante : quoiqu'ils fassent, l'erreur reste visible (figure 3.88, p. 414). Il leur faudra sauvegarder et recharger leur programme pour supprimer cette erreur. Mais ceci aura un effet sur le déroulement de l'activité : la variable (mesure), peut-être parce qu'elle a été associée par les élèves aux erreurs récurrentes, n'est pas remobilisée. Elle sera remplacée néanmoins par une autre variable, déjà envisagée par d'autres groupes (45a par exemple), (taille_hexagone). Cette solution sera rapidement écartée : si E2 pense que l'erreur vient du fait que seule la première boucle a été modifiée (les autres ayant des chaines vides comme représentation du nombre de répétitions), E1 confirme, mais précise qu'elle voulait tester uniquement cette variable (3.25), et en conclut, après un tracé invalide, « bah alors on va pas utiliser ça » (46e, S4, 31'39). En séance 5, le groupe 46e mobilisera les faits partagés en classe, en utilisant dès le début de la séance la variable (mesure), utilisée avec un opérateur : (mesure - 1).
De façon similaire au groupe 46i (voir 3.4.2, p.334), les élèves affecteront dans un premier temps cette expression pour toutes les boucles : $b^n_{i\in[1..8]} = n - 1$. Elles corrigeront rapidement pour aboutir à un script valide excepté pour la boucle b_7 (script 3.17, p. 410). On peut noter que les élèves font l'hypothèse $b^n_7 = n\boxed{+2}$, ce qui pourrait être une extension de leur dernier TEA considéré comme valide ($+a/+a/+a\boxed{+2}$ ou $*a/*a/*a\boxed{+2}$).
La boucle b_7 sera ensuite étudiée, en testant un nombre conséquent d'expressions :

- (mesure + 4), qui s'avérera valide pour $TS(n)[6]$ mais invalide pour $TS(n)[8]$;
- (mesure * 2) ;
- (mesure / 2) ;
- (mesure - 0.5) ;
- (mesure - 3) ;
- (mesure + 1) ;

34	01 :26.569	T1	à XXX on avait réglé le problème mais c'était le onze qu'on n'avait pas réglé
35	01 :26.569	G1	montre script TS11
36	01 :30.509	T2	ah oui
37	01 :43.385	T2	c'est long
38	01 :46.141	T1	oui c'est long
39	01 :55.158	T1	ah non on a un problème
40	01 :56.944	T2	ah oui c'est à cause de quoi
41	01 :57.339	EXEC	PAUSE_button
42	01 :57.356	EXECUTION	PAUSE receiveKey(472)
43	02 :01.855	T1	on est rendu où ?
44	02 :02.924	T1	là il a tracé une fois
45	02 :02.924	G1	montre avec la souris
46	02 :05.557	T2	mesure 22
47	02 :06.398	T2	ok ça
48	02 :08.086	T1	deux fois
49	02 :12.734	T1	mais XXX
50	02 :15.437	T1	là ya trois trucs tracés
51	02 :22.832	T2	attend c'est quand ça a tourné
52	02 :24.460	T2	par là
53	02 :24.460	G2	tracé b2-b3 avec le doigt
54	02 :26.054	T1	alors ça tourne
55	02 :26.267	T2	on est là
56	02 :26.267	G2	montre l'écran
57	02 :27.221	T1	déjà ça a tourné une fois
58	02 :27.221	G1	montre sur script
59	02 :28.859	T2	oui
60	02 :29.409	T1	deux fois
61	02 :29.409	G1	montre sur script
62	02 :29.938	T2	oui
63	02 :31.551	T1	ben là ça fait une fois
64	02 :33.204	T1	ça fait deux fois | trois fois
65	02 :33.204	G1	montre script en descendant
66	02 :35.822	T1	et | c'est avant
67	02 :38.385	T1	il faut qu'on en mette un de moins
68	02 :40.365	T1	donc ça fait | dix-sept

Transcription 3.24 – 46e, S4 : Se repérer dans l'algorithme

740	31 :02.384	T2	ah c'est parce que t'as pas changé ça
741	31 :02.384	G2	montre le reste du script
742	31 :04.975	T2	du coup ça fait juste une ligne droite
743	31 :04.975	G2	geste du tracé b1
744	31 :15.915	T1	je sais mais je voulais regarder ce que ça faisait que ça
745	31 :20.759	T2	je sais pas
746	31 :21.119	EXEC	STOP_button(all)
747	31 :21.141	EXECUTION	STOP receiveKey(478)
748	31 :26.326	EXECUTION	START receiveKey(472)
749	31 :28.340	EXEC	STOP_button(all)
750	31 :28.354	EXECUTION	STOP receiveKey(472)
751	31 :37.012	T2	oh la la
752	31 :39.778	T1	bah alors on va pas utiliser ça
753	31 :39.778	G1	(supprime "taille_hexagone)
754	31 :42.629	T2	non
755	31 :43.573	T2	c'est une très mauvaise idée
756	31 :46.170	T2	et enlève euh celui(ci)
757	31 :46.170	G2	montre menu variable

Transcription 3.25 – 46e, S4 : Écarter *taille_hexagone*

— mesure + 2 ;
— mesure - 0.25 ;
— mesure - 0.5 ;
— mesure - 0.75 ;
— mesure + 6, valide pour un TS8 ;
— mesure + 4, invalide pour un TS8 ;
— mesure + 10 ;
— mesure - mesure ;
— mesure + mesure ;
— et pour finir mesure + mesure - 1 ;

La dernière tentative est proche d'un des résultats attendus : mesure + mesure - 2. La fin de la séance ne leur permettra pas d'aller plus loin.

Block_478

```
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [ mesure − 1 ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ mesure − 2 ] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [ mesure − 2 ] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [ mesure − 2 ] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [ mesure − 2 ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ mesure − 1 ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ mesure + 2 ] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [ mesure − 2 ] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || compteur |]] ||
hexagones |]]
```

Script 3.17 – Deuxième script générique (46e, S5)

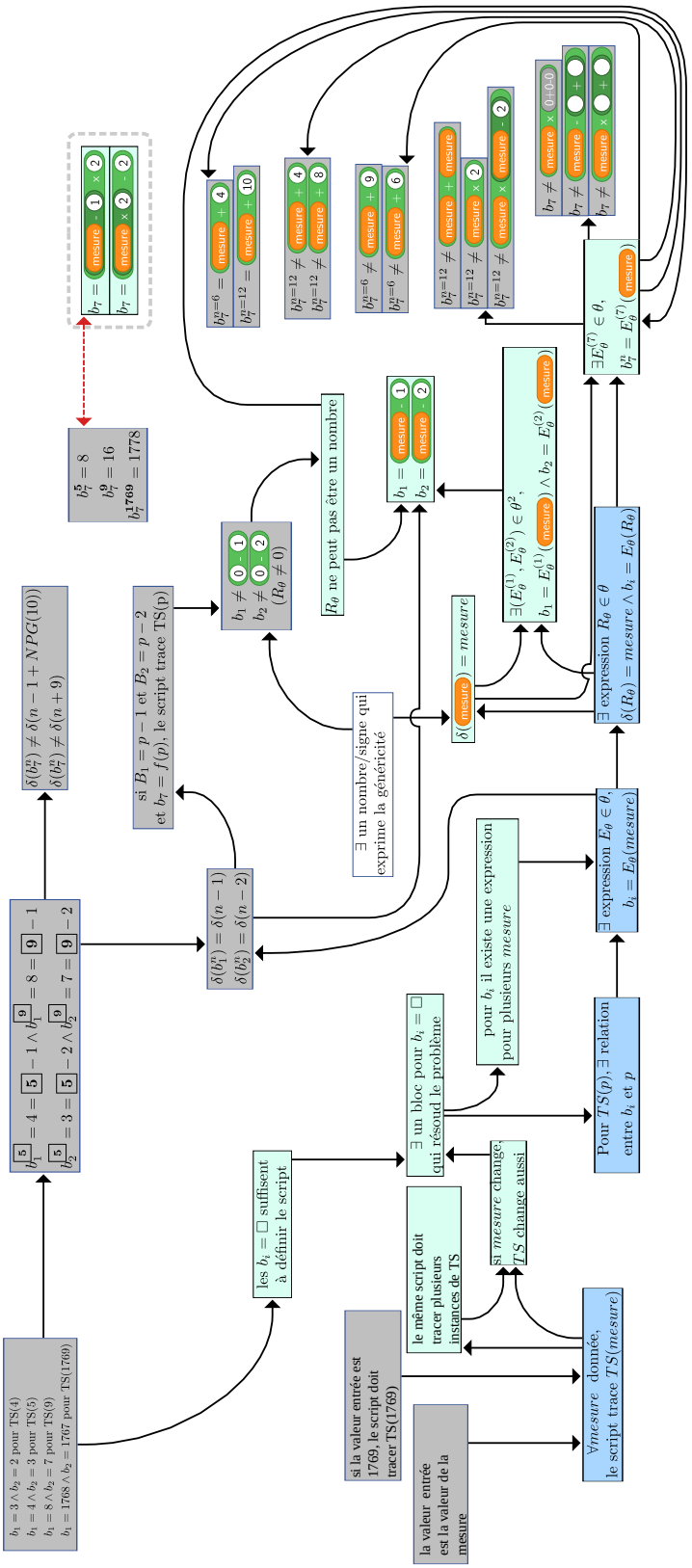


Figure 3.83 – Espace des faits-contraintes - 45m

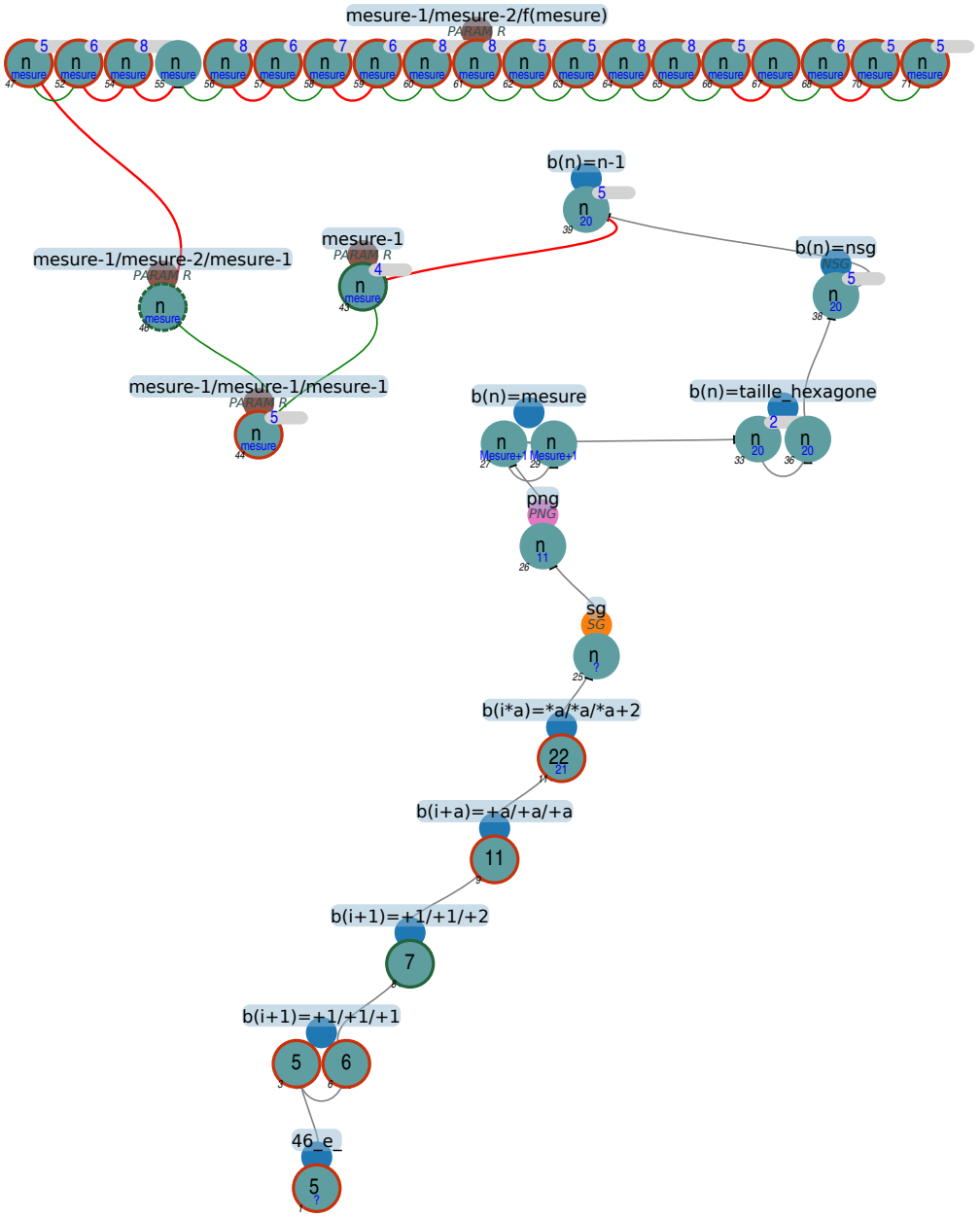


Figure 3.84 – Organisation de l'activité - 46e

Figure 3.85 – Un angle entre ça et ça... (46e, S3)

Figure 3.86 – « il se referme trop tôt » ou « il redescend trop vite(46e, S3) »

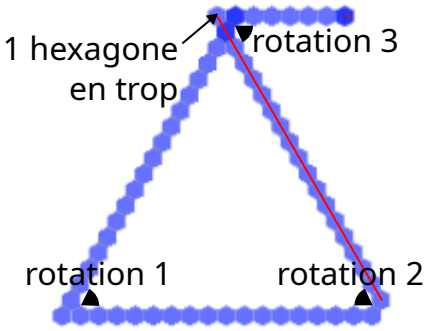


Figure 3.87 – Pause sur le $TS(22)$ (46e, S4)

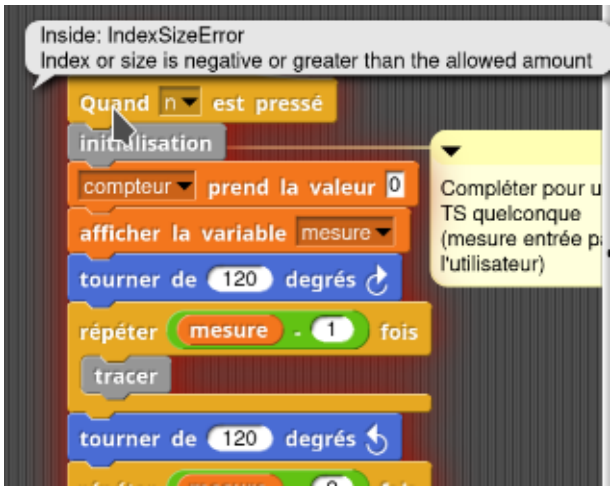


Figure 3.88 – Une erreur sibylline (46e, S4)

Recherche de régularités

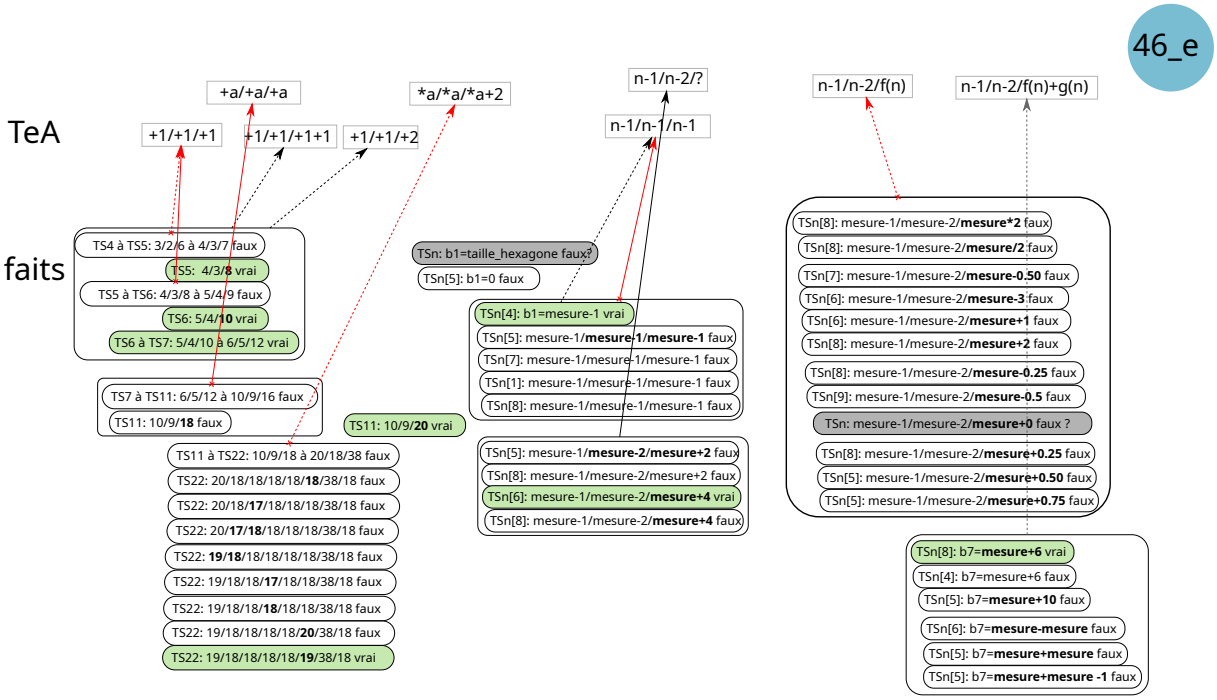


Figure 3.89 – Faits et TEA, groupe 46e

Les élèves du groupe 46e vont construire de nombreux faits invalidant leurs actions. On peut constater (figure 3.89, p. 415) que les TEA ne sont pas stabilisés, avec notamment une longue recherche pour la construction du TS22 [78]. Si les relations $B_1 = mesure - 1$ et $B_2 = mesure - 2$ sont assez rapidement établies, la relation pour b_7 prendra des formes assez variées.

Un cheminement classique dans les TEA… Pour construire de nouvelles instances, les TEA du groupe 46e évoluent de façon similaire à ce qui a déjà été observé :

1. Le TEA $+1/+1/+1$ est appliqué.
2. Au bout de deux instances invalides [79] — mais avec E1, qui dès la construction du TS5, voyait un souci au niveau de leur TEA concernant b_7 —, le groupe 46e passe au TEA valide $+1/+1/+2$.
3. Pour faire un saut de quatre instances (passer de la mesure 7 à la mesure 11), ce groupe généralise son TEA initial : on passe de $+1/+1/+1$ à $+a/+a/+a$. Le groupe 46e ajuste alors cette évolution du TEA en $+a/+a/+a+2$. Ce dernier TEA étant par ailleurs erroné, mais la rétroaction le rendant visible ne se fera qu'en fin de séance.
4. Ce TEA se généralise de nouveau, en $*a/*a/*a+2$, lui aussi invalide.

78. Pour ces tentatives, les boucles modifiées sont en gras.
79. Cette forme de « résistance » des TEA aux invalidations sera étudiée plus précisément dans la section 3.7.6 (p. 453).

... Mais des ajustements successifs empêchant leur évolution En séance 4, le groupe 46e va procéder à sept ajustements de leur $TS(22)$ erroné de fin de séance 3 [80], ajustements concernant toutes les boucles (sauf b_5) (étapes 14-20). Ces ajustements successifs rendent très difficile l'évolution du TEA : comment les élèves peuvent-ils se rendre compte que, finalement, leur TEA $*a/*a/*a+2$ est totalement invalide, puisque toutes les transformations appliquées ont été modifiées ? Comment pourraient-ils identifier un schéma (qui pourrait faire évoluer leur TEA valide $+1/+1/+2$ en $+a/+a/+2a$, par exemple) dans leur transformation du $TS(11)$ en $TS(22)$? Si on suit les modifications faites, voici un pseudo-TEA instancié qui aboutirait à leur tracé valide [81] :

$$
\begin{array}{llllllll}
b_1^{11\times 2} = b_1^{11} \times 2 & & & -1 & & & & \\
b_2^{11\times 2} = b_2^{11} \times 2 & & & -1+1 & & & & \\
b_3^{11\times 2} = b_3^{11} \times 2 & & -1 & +1 & & & & \\
b_4^{11\times 2} = b_4^{11} \times 2 & & & & & -1+1 & & \\
b_5^{11\times 2} = b_5^{11} \times 2 & & & & & & & \\
b_6^{11\times 2} = b_6^{11} \times 2 & -2 & & & & & & +2-1 \\
b_7^{11\times 2} = b_7^{11} \times 2 + 2 & & & & & & & \\
b_8^{11\times 2} = b_8^{11} \times 2 & & & & & & &
\end{array}
$$

Mais seul le tracé est valide, puisque ici le TS tracé est un TS20 et non un TS22. Si l'on suit les résultats des modifications des élèves, en simplifiant ce qui peut l'être, ce pseudo-TEA représente le passage de a à $2a-2$, ici pour $a = 11$. Mais ce TEA est-il pertinent ?
Pour b_2, on a $b_2^{2a-2} = 2b_2^a$, ce qui est valide. En effet, pour un TS de mesure a, $b_2^a = a-2$, donc :

$$
\begin{aligned}
b_2^{2a-2} &= (2a-2)-2 \\
&= 2a-4
\end{aligned}
$$

Or pour les élèves :

$$
\begin{aligned}
b_2^{2a-2} &= b_2^a \times 2 \\
&= (a-2) \times 2 \\
&= 2a-4
\end{aligned}
$$

Les boucles de la famille B_2 seraient donc valides. De même pour b_1 : $b_1^a = a-1$, donc :

$$
\begin{aligned}
b_1^{2a-2} &= (2a-2)-1 \\
&= 2a-3
\end{aligned}
$$

Et pour les élèves :

$$
\begin{aligned}
b_1^{2a-2} &= b_1^a \times 2 - 1 \\
&= (a-1) \times 2 - 1 \\
&= 2a-3
\end{aligned}
$$

80. Plus un ajustement en fin de séance 3 ! $b_6 = 20 - 2 = 18$ (46e, étape 12).
81. Les actions successives sont placées dans des colonnes successives.

Ce pseudo-TEA serait donc aussi valide pour les boucles de la famille B_1.
On peut remarquer que ce TEA est loin d'être évident à identifier, et particulièrement si on ne garde pas trace des modifications faites, ce qui est le cas pour ce groupe.
De plus, si l'on vérifie la validité de ce pseudo-TEA pour la boucle b_7, on se rendra compte qu'il n'est valide que pour *ce cas* particulier ($a = 11$) et *parce qu'il y a une erreur* sur b_7^{11}.
En effet, on a $b_7^a = 2a - 2$, donc on devrait avoir :

$$\begin{aligned} b_7^{2a-2} &= 2(2a-2) - 2 \\ &= 4a - 6 \end{aligned}$$

Or, avec les actions des élèves :

$$\begin{aligned} b_7^{2a-2} &= 2b_7^a + 2 \\ &= 2(2a-2) + 2 \\ &= 4a - 2 \end{aligned}$$

Ici, le tracé est néanmoins valide, car pour $a = 11$ et $b_7^a = 18$ (version des élèves, invalide), on a :

$$\begin{aligned} b_7^{2\times 11a-2} &= 2b_7^{11} + 2 \\ &= 2 \times 18 + 2 \\ &= 38 \end{aligned}$$

Or, on devrait bien avoir :

$$\begin{aligned} b_7^{2\times 11-2} &= 4 \times 11 - 2 \\ &= 38 \end{aligned}$$

En outre, toutes ces transformations sont des corrections sur des nombres, et même si les élèves semblent mettre du sens sur les différentes boucles en identifiant leur tracé, ces modifications de boucles sont internes au script : il s'agit « [d'en mettre] un de moins » (46e, transcription S4, 2'38), ou « d'en rajouter un là » (46e, transcription S4, 3'32). Il n'y a pas de relation faite entre les nombres obtenus et ceux disponibles pour les autres instances. En fait, cela induit des relations intra-objectales.
Ainsi, les actions des élèves pour modifier $TS(11)$ ne sont valides que parce qu'il y a une erreur dans le script de départ. De plus, l'identification d'un schéma généralisable dans ces actions est pour le moins complexe. Enfin, les modifications faites induisent des relations intra-objectales, entre nombres. Tout cela rend toute forme de généralisation peu accessible.
Les élèves parviennent donc à construire des instances, mais sans pouvoir établir de règles d'action stables. Que ce soit en procédant par ajustements successifs d'un script comptant un certain nombre d'erreurs, ou en procédant par essais-erreur, le schéma reliant les instances est rendu moins visible.

Comprendre le script pour construire des faits Durant la construction successive d'instances en séance 3 et 4, les élèves du groupe 46e vont avancer dans la compréhension de l'algorithme de tracé et de la rétroaction. Quatre aspects seront traités par ces élèves :

— le lien compteur-TS tracé ;
— l'algorithme comme preuve ;
— la nécessité de modifier la structure, ou non ;

41	01 :56.376	T1	non mais attend
42	01 :57.930	T1	si on met B
43	01 :58.987	EXECUTION	START receiveKey(481)
44	02 :02.529	T1	oh ça le compte
45	02 :03.628	T2	mouou
46	02 :04.964	T1	sauf qu'à la fin ça va faire vingt-quatre
47	02 :08.558	T1	non mais attend on regarde
...			
64	02 :39.526	T1	mais oui mais là ça va faire le 24
...			
68	02 :46.798	T1	là c'est la même forme mais
69	02 :48.748	T1	ça va mettre 34 à la fin
...			
95	03 :30.347	T1	après faut qu'on compte les | machins
96	03 :32.742	T2	ouai
...			
99	03 :39.192	T1	attend XXX je vais compter sur mon
100	03 :39.192	G1	se lève et semble compter sur sa feuille
101	04 :05.408	E1	erreur de comptage !
102	04 :05.408	T1	36 il en faut 36
103	04 :07.735	T1	faut retenir XXX
104	04 :11.674	G1	note un résultat sur sa main
105	04 :16.514	T1	j'lai écrit (rire)
106	04 :19.094	T1	et maintenant il faut qu'on compte
107	04 :22.452	T1	trois
108	04 :24.865	T1	bah ça fait trois
109	04 :26.298	T1	sept huit neuf dix
110	04 :26.298	G1	compte sur ses doigts (3)
111	04 :27.915	T1	onze douze treize quatorze
112	04 :27.915	G1	compte sur ses doigt (4)
113	04 :28.557	T2	pourquoi tu comptes ça ?
114	04 :30.082	T1	quinze seize | dix-sept
115	04 :30.082	G1	compte sur ses doigts 2+1 de plus
116	04 :31.279	T1	dix-huit dix-neuf vingt
117	04 :31.279	G1	compte sur ses doigts (3)
118	04 :32.359	T1	vingt et un vingt deux vingt trois
119	04 :32.359	G1	compte sur ses doigts (3)
120	04 :33.754	T1	vingt quatre vingt cinq vingt six
121	04 :33.754	G1	compte sur ses doigts (3)
122	04 :38.322	T1	vingt-sept vingt-huit vingt-neuf
123	04 :38.322	G1	compte sur ses doigts (3)
124	04 :40.026	T1	trente trente-et-un trente-deux trente-trois
125	04 :40.026	G1	compte sur ses doigts (4)
126	04 :42.261	T1	j'crois ya un problème
127	04 :43.489	T1	attend au pire on fait
...			
135	05 :02.516	T1	normalement ça fait euh | trente six XXX
136	05 :02.516	G1	regarde ce qui est noté sur sa main
137	05 :05.202	T1	si c'est bien fait
...			
145	05 :26.958	T1	mais en fait là ça en fait deux alors qu'il en faut trente six
146	05 :26.958	G1	montre l'écran
147	05 :29.014	T2	en fait il faut décaler les trois comme ça
148	05 :29.014	G2	montre les 3 de b8 avec trois doigts et les déplace virtuellement à la bonne place
149	05 :31.440	T1	nan c'est qui manque des carrés
...			
196	06 :55.102	T1	yavait trente trois ça veut dire que j'ai mal compté
197	06 :55.102	G1	efface le nombre écrit sur sa main
...			
383	12 :02.921	T1	stop
384	12 :03.902	T1	on a oublié de prendre un truc
385	12 :06.180	T2	quoi
386	12 :07.476	T1	prendre la valeur
387	12 :08.699	T1	là
388	12 :08.775	VARIABLE	(i3) ** prend la valeur *0* [new loc :bottom]
389	12 :09.310	T2	ah oui
390	12 :10.508	T1	c'est pour ça que ça va trop loin regarde
391	12 :10.508	G1	montre compteur
392	12 :11.368	VARIABLE	(i3) *compteur* prend la valeur 0 [val_inputChanged <<>>]
393	12 :11.368	VALEURS	b0i3(VAL) : *compteur* prend la valeur 0<<>>
394	12 :12.317	T1	yen n'a pas soixante-seize déjà

Transcription 3.26 – 46e, S3 : Le rôle du compteur

— la mesure comme propriété du TS tracé.

En séance 3, comme on peut le voir sur la transcription 3.26, les élèves évoquent à de nombreuses reprises le « compteur », souvent mis en lien avec le fait de « compter » — ce qui est bien le rôle de la variable *compteur*. Il sera un peu moins évoqué en séance 4, et on en verra quelques occurrences en séance 5. Ce compteur, qui a été découvert dans la séance précédente, lors de laquelle les élèves devaient justement « apprendre à compter » au script traçant un TS4, va être en premier lieu une propriété définissant un TS, puis un critère de validation et une indication de l'état du programme, et enfin une propriété mise en relation avec le tracé et le script.
En début de séance 3, E1 définit le TS par la valeur du compteur attendue, c'est-à-dire par le nombre d'hexagones formant le TS, nombre indiqué par la valeur finale de la variable *compteur* : « sauf qu'à la fin ça va faire vingt-quatre » (3.26, 46) devient « mais oui mais là ça va faire *le* vingt-quatre » (3.26, 34). « 24 » dénote à la fois le nombre d'hexagones total et *le* TS particulier construit avec ce nombre d'hexagones. C'est sans doute ce qui explique les tentatives de modification du compteur afin de modifier le TS faites par certains groupes (46g p. 301, 46i p. 337).
Dans un deuxième temps, le compteur va être lié à l'état du programme — ou du tracé—, tandis que les TS deviennent dénommés par la valeur de leur mesure. Concernant l'état du programme (ou du tracé) et son état final, E1 prend en compte l'état final attendu : « ça va en mettre 34 à la fin » (3.26, 69). Cet état final est un critère de validité :

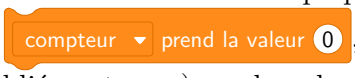
— E1 indique qu'il « faut qu'on compte les machins [les hexagones] » (3.26, 95), et, après être allé compter effectivement les hexagones sur le TS5 présent dans son cahier, elle complète — ayant mal dénombré — qu'« il en faut 36 » (3.26, 102). Le nombre d'hexagones correspond à un certain TS, et il est un objectif à atteindre.

— Elle précise plus tard « normalement ça fait euh trente-six si c'est bien fait » ((3.26, 135-137)). La validité du tracé ou du programme est liée à l'atteinte de l'objectif en termes de nombre d'hexagones.

— Après un tracé erroné dénombrant trente-deux hexagones, E1 constate que « mais en fait là ça en fait deux [le 2 de 32] alors qu'il en faut trente-six ». Alors que E2 s'intéresse au tracé (« en fait il faut décaler les trois comme ça », 3.26, 147 et figure 3.90, p. 419), E1 considère que ce n'est pas un problème de tracé, « c'est qui manque des carrés [des hexagones] » (3.26, 149). Ce n'est pas le tracé qui est un critère de validité, c'est le nombre d'hexagones indiqués comme étant tracés.

— Après avoir obtenu un TS5 valide, elle reconnaitra « yavait trente-trois ça veut dire que j'ai mal compté » et effacera le nombre cible qu'elle avait marqué sur sa main (3.26, 196-197). C'est une fois encore le nombre d'hexagones qui est un critère de validité.

Figure 3.90 – « Il faut décaler les trois comme ça » (46e, S3)

Ce nombre d'hexagones cible est mis directement en lien avec l'algorithme de tracé. Ainsi, dans les interventions 107 à 127 de l'extrait 3.26, E1 compte non pas les hexagones qui sont tracés à l'écran, mais la somme des valeurs du nombre des répétitions des boucles. Elle commence par b_8 (3), ajoute b_7 (7) et remonte ainsi jusqu'à b_1. Cependant, le script n'étant pas entièrement visible sur l'espace d'exécution, elle va devoir le déplacer, et cela va l'amener à perdre ses repères et à compter deux fois la boucle b_2. Elle trouve ainsi au total trente-trois hexagones, ce qui pour elle est un problème puisqu'elle avait établi qu'il en fallait trente-six. Ici, E1 utilise l'algorithme comme preuve de la validité ou non du tracé, sans que ce tracé soit nécessaire. Notons que E1 cumule des erreurs pour aboutir à une conclusion vraie. Elle ne prend pas en compte les deux hexagones qui sont tracés en dehors des boucles, la boucle b_7 trace un hexagone de moins que pour un script valide : ces trois hexagones non comptabilisés sont compensés par le comptage surnuméraire de $b_2 = 3$. Le nombre obtenu, 33, est bien ainsi le nombre d'hexagones d'un TS5. Mais E1 ne considère pas cela comme valide, puisqu'elle en avait dénombré trente-six. Ainsi, sa conclusion indiquant qu'elle pense que le tracé sera erroné est juste, tout en se basant sur des prémisses fausses. De faux faits peuvent amener à construire des faits valides. En séance 4, E1 procèdera de façon légèrement différente, en comparant les hexagones tracés au nombre de répétitions des boucles (46e, transcription S4, 10'45-11'46), faisant ainsi le lien entre l'algorithme et le tracé. Cette compréhension du lien entre compteur, hexagones tracés et boucles, permettra au groupe 46e d'identifier une erreur de duplication sans avoir à terminer l'exécution. En dupliquant un script avant modification, ces élèves ont omis l'instruction 

, qui initialise la variable *compteur*. E1 le constate (« stop on a oublié un truc ») car la valeur du compteur lui parait trop élevée à ce stade du tracé (« c'est pour ça que ça va trop loin regarde [...] yen n'a pas soixante-seize déjà ») (transcription 3.27). On peut voir que pour ce groupe, ou tout au moins pour E1, la valeur du compteur en cours d'exécution est une représentation de l'état du programme. Cette identification du rôle du compteur permet donc de construire des faits tiers, sans nécessiter une exécution du script.
Un peu plus tard, comme on l'a vu précédemment (Construction dynamique et algorithme, page 406), les élèves du groupe 46e vont analyser l'algorithme, non plus en termes de nombre de répétitions et de compteur, mais plutôt en termes de tracé dynamique, en repérant les moments du script impliquant un changement de direction dans le tracé.
Dans le même temps, la compréhension de l'algorithme par ce groupe passe aussi par le mouvement de la propriété caractéristique des scripts traçant les TS. Si au début le compteur désignait le TS (« le vingt-quatre »), cette façon de dénommer les TS va évoluer. Ainsi, pour désigner le TS11, E1 dira « et là il faut il faut qu'on fasse euh un truc onze » (46e, transcription S3, 10'35). On retrouvera une formulation similaire en séance 4 (« on avait réglé le problème mais

383	12 :02.921	T1	stop
384	12 :03.902	T1	on a oublié de prendre un truc
385	12 :06.180	T2	quoi
386	12 :07.476	T1	prendre la valeur
387	12 :08.699	T1	là
388	12 :08.775	VARIABLE	(i3) ** prend la valeur *0* [new loc :bottom]
389	12 :09.310	T2	ah oui
390	12 :10.508	T1	c'est pour ça que ça va trop loin regarde
391	12 :10.508	G1	montre compteur
392	12 :11.368	VARIABLE	(i3) *compteur* prend la valeur 0 [val_inputChanged <<>>]
393	12 :11.368	VALEURS	b0i3(VAL) : *compteur* prend la valeur 0<<>>
394	12 :12.317	T1	yen n'a pas soixante-seize déjà

Transcription 3.27 – 46e, S3 : Initialisation du compteur

c'était le onze qu'on n'avait pas réglé », 46e, transcription S4, 1'26). En séance 4, lorsque E1 demande où elles en sont rendues, E1 répond « mesure 22 » (46e, transcription S4, 2'05). Un peu plus tard, E1 constate « déjà ça va beaucoup plus vite que le vingt-deux » en précisant le lien entre « 22 » et la mesure : « là c'est mesure onze là mesure vingt-deux » (46e, transcription S4, 16'21-16'27). Ce passage du compteur à la mesure comme propriété caractéristique du TS aboutira à la position du problème de la création d'un script générique. En effet, une fois le $TS(22)$ construit, E2 affirme que « maintenant faut qu'on fasse mille euh chais pu combien » (3.28, 449) — c'est-à-dire trouver le TS que l'on peut construire avec mille quatre cent hexagones, le problème initial posé aux élèves —. E1 réfute en affirmant « non maintenant faut qu'on fasse n'importe quelle | mesure » (3.28, 451). Elle précise ensuite, en donnant un exemple en entrant une valeur de 8 à l'invite : « il faut que | il m'en fasse huit 'fin un de mesure huit, mais sauf que là ça va m'en faire un | un de mesure vingt-deux » (3.28, 467-471). Elle conclut « donc il faut trouver un programme pour aller pour toutes les mesures » (3.28, 475-477). Ainsi, identifier le TS à sa mesure permettrait de poser le problème, et peut-être aussi d'identifier des relations entre la mesure et le nombre de répétitions des boucles, de construire de nouveaux faits. On voit que ce groupe, avant de passer au script générique :

— identifie le script et le tracé à sa *mesure*,

— fait le lien entre nombre de répétitions des boucles et tracé,

— fait le lien entre tracé et localisation de l'instruction dans la séquence que constitue le programme,

— et a identifié la non-nécessité de modifier la structure du programme.

On verra cependant que lors de la phase de généralisation, les élèves vont s'attacher davantage aux nombres qu'au tracé, ce qui les empêchera de construire l'expression permettant de déterminer le nombre de répétitions de la boucle b_7.

Généralisation algébrique

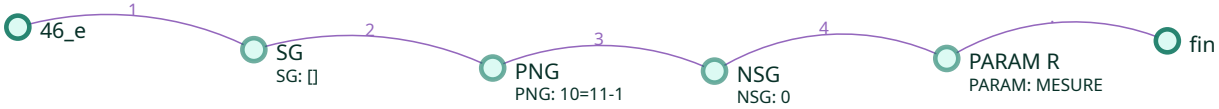


Figure 3.91 – Type de généralisation - 46e

Le groupe 46e, lors de la phase de généralisation algébrique, va explorer l'utilisation d'un nombre-signe générique avant de passer au paramètre. L'analyse des transcriptions montre aussi qu'un signe générique a été utilisé.

Indice sémiotique Une fois le $TS(22)$ et $TS(11)$ corrigés, le groupe 46e s'attelle au script générique, afin de faire « n'importe quelle mesure » (3.28, 451). Dans un premier temps, E2 envisage de modifier le bloc d'initialisation : « comment on change ça ? » (3.28, 455). En montrant le bloc d'initialisation du $TS(22)$, elle explique : « bah parce que là c'est là qu'il y a vingt-deux » (3.28, 460). Elle complète alors en montrant le bloc d'initialisation du script générique $TS(n)$, « mais pour celui on sait pas » (3.28, 462). E2 identifie bien la mesure comme propriété caractéristique du TS, et elle cherche ce qui change entre les scripts relativement à la mesure. Or, les seuls indices donnés par des signes sont :

— les commentaires précisant la tâche prescrite,

449	20 :04.138	T2	maintenant faut qu'on fasse mille euh chais pu combien
450	20 :04.802	EXEC	STOP_button(all)
451	20 :07.159	T1	non maintenant faut qu'on fasse n'importe quelle | mesure
452	20 :10.467	T2	ah d'accord
453	20 :13.702	T2	bah |
454	20 :16.585	T1	eux XXX
455	20 :17.357	T2	comment on change ça
456	20 :17.357	G2	doigt sur "initialisation" de TSn (ou commentaire)
457	20 :18.565	STRUCTURE	DUPLIC_846-821(846)
458	20 :20.256	VARIABLE	(i3) compteur prend la valeur 0 [drop loc :bottom duplic 1218565]
459	20 :22.425	T1	pourquoi
460	20 :23.505	T2	bah parce que là c'est là qu'il y a vingt-deux
461	20 :23.505	G2	montre "initialisationmax22"
462	20 :25.400	T2	mais pour celui-là on sait pas
463	20 :27.696	T1	mais justement c'est en fait parce que
464	20 :33.115	EXEC	KEY_n
465	20 :33.141	EXECUTION	START receiveKey(478)
466	20 :33.193	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
467	20 :34.579	T1	et là par exemple si je mets huit
468	20 :36.194	ENTRÉE	ANSW <<8>>
469	20 :36.746	T1	il faut que | il m'en fasse huit 'fin un de mesure huit
470	20 :36.746	G1	montre tracé en gros
471	20 :40.005	T1	mais sauf que là ça va m'en faire un | un de mesure vingt-deux
472	20 :40.732	EXEC	STOP_button(all)
473	20 :40.742	EXECUTION	STOP receiveKey(478)
474	20 :44.023	T2	ah oui
475	20 :46.731	T1	donc il faut trouver un programme
476	20 :49.511	T1	pour aller pour toutes les mesures
477	20 :49.511	G1	geste englobant avec la souris

Transcription 3.28 – 46e, S4 : position du problème générique

— les nombres présents dans l'écriture des blocs d'initialisation(figure 3.92, p. 422).

E2 semble donc envisager la possibilité que initialisation# spécifie que le script va tracer un TS de mesure $\delta(\#)$ [82]. En soi, ce n'est pas inexact : initialisation4 doit correspondre au tracé d'un TS4, et initialisation11 doit correspondre à un TS de mesure 11. Le signe « # » est finalement un rappel de la tâche prescrite. Mais E2 considère qu'il *faut changer* le signe « # » pour que le script lui-même change, comme si l'ordinateur allait de lui-même interpréter l'instruction initialisation#, où « # » dénote une valeur écrite avec des chiffres, comme « Trace un TS de mesure $\delta(\#)$ ». On peut y voir une fois de plus cette « pensée magique » qui laisse penser que l'ordinateur va faire ce qu'on attend de lui, puisque c'est ce qu'il se passe en général. Le signe « # » présent dans l'intitulé des blocs serait donc une sorte d'indice sémiotique, pour l'élève comme pour l'ordinateur. L'absence de signe dénotant un nombre dans le bloc d'initialisation du script générique renforce sans doute cette idée : puisqu'il faut faire « n'importe quelle mesure », il faut indiquer cette mesure dans le bloc d'initialisation.

Figure 3.92 – Les différents blocs d'initialisation : un indice sémiotique

82. Nous utilisons toujours $\delta(S)$ pour indiquer le dénoté du signe S.

496	21 :23.953	T1	mais euh c'est quel bloc qu'il faut mettre ?
497	21 :28.001	T2	je sais pas
498	21 :30.025	T2	j'ai pas compris qu'est-ce qu'il faut faire en fait
499	21 :32.032	T2	il faut changer euh
500	21 :33.508	T1	par exemple regarde là | quand on met N
501	21 :36.291	EXEC	KEY_n
502	21 :36.310	EXECUTION	START receiveKey(478)
503	21 :36.361	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
504	21 :37.311	T1	elle nous demande quel chiffre on veut
505	21 :40.294	T1	par exemple tu vois là ya une mesure
506	21 :40.294	G1	déplace l'affichage de "mesure"
507	21 :43.949	T1	avec une huit | du coup ça faire un | un de mesure huit
508	21 :48.334	T1	là si je mets vingt | vingt-deux ça va | enfin il faut que ça fasse | le truc de mesure vingt-deux
509	21 :54.891	T1	si je marque trois
510	21 :57.005	T1	faut qu'ça fasse le même (truc) | mais du coup fait sur XXX c'est le même programme
511	21 :58.529	T2	mais du coup c'est les | c'est les chiffres là
512	21 :58.529	G2	balaye en aller-retour les instructions du script avec le doigt
513	22 :05.651	ENTRÉE	ANSW <<>>
514	22 :05.676	ERREUR	ERR_Inside : IndexSizeError : Index or size is negative or greater than the allowed amount
515	22 :07.446	T1	oui et XXX qui en XXX
516	22 :08.270	EXEC	STOP_button(all)
517	22 :08.288	EXECUTION	STOP receiveKey(478)
518	22 :20.421	T1	mais du coup euh
519	22 :25.494	T1	je sais pas
520	22 :26.247	T2	ouai
521	22 :28.976	T1	ça c'est bon c'est bon | ça c'est bon
522	22 :31.675	T1	c'est là le nombre qu'il faut changer
523	22 :31.675	G1	clic (?)
524	22 :33.787	T2	XXX
525	22 :35.843	T1	faut réfléchir
526	22 :36.440	T2	si c'est huit euh si c'est hui c'est |
527	22 :40.946	T2	ah mais c'est dur huit
528	22 :46.114	T1	si c'est huit on l'a pas fait le huit
529	22 :48.592	T2	ah oui
530	22 :49.422	T1	le sept XXX
531	22 :51.590	T1	mais euh tu vois c'est juste faut changer ça
532	22 :51.590	G1	met rpt à vide
533	22 :52.294	T2	et c'est déjà ce qu'on n'a pas fait normalement
534	22 :55.735	VALEURS	(BOUCLE)b1i0(VAL) : répéter ** fois (commandes)<<19>>
535	22 :55.787	T1	ça
536	22 :57.842	T1	ça
537	22 :57.879	VALEURS	(BOUCLE)b2i0(VAL) : répéter ** fois (commandes)<<18>>
538	22 :59.184	VALEURS	(BOUCLE)b3i0(VAL) : répéter ** fois (commandes)<<18>>
...			
550	23 :06.462	T1	mais | on marque quoi à la place ?

Transcription 3.29 – 46e, S4 : Changer les expressions des boucles

Un signe générique (SG) comme marque place E1 va alors expliciter le problème à E2 : il faut faire un programme qui aille « pour toutes les mesures » (46e, transcription S4, 20'49). Pour cela il faut changer quelque chose (3.29, 499), mais E1 ne sait pas quoi : « mais euh c'est quel bloc qu'il faut mettre ? » (3.29, 496). E1 explique ensuite à E2 que si l'utilisateur entre des mesures différentes, cela trace malgré tout le même programme (3.29, 500-510). E2 comprend alors que ce qu'il faut changer « du coup c'est les chiffres là » (3.29, 511-512), c'est à dire les expressions numériques déterminant le nombre de répétions des boucles, ce que E1 reformule : « tu vois il faut juste changer ça » (3.29, 531), en remplaçant le nombre de la première boucle par un espace vide : 

. E2 reste sur les instances, puisqu'elles ont « déjà » fait ce remplacement par d'autres nombres (3.29, 533), mais E1 comprend que c'est l'expression qu'il faut changer pour que

cela fonctionne quelle que soit la mesure. Elle supprime alors toutes les expressions des boucles et pose la question : « mais on marque quoi à la place ? » (3.29, 550). Elle utilise ici cet espace laissé vide comme un signe générique, un marque-place, qui indique que c'est ce signe qui doit changer en fonction de la mesure, « pour toutes les mesures possibles ». Un peu plus tard, elle donnera à E2 un exemple de la tension fixe-variable : si on entre 10 comme nombre de répétitions pour une boucle, on ne sait pas trop quel TS cela va tracer, mais ce qui est certain c'est que « ça va pas le [faire] pour tous » (3.30, 578-581), revenant à son idée qu'il doit bien y avoir une expression de θ qui résout le problème :« je pense qu'il y a un bloc qui existe » (3.30, 583) [83].

Un Nombre-Signe Générique, ou un nombre pour les représenter tous Si E1 est bien dans le registre sémiotique du « langage de la théorie » θ, cherchant une expression de θ qui résoudrait le problème, E2 reste quant à elle dans le registre sémiotique du problème, ou plus précisément ici celui des nombres. Elle a bien identifié la tension entre l'expression du nombre de répétitions des boucles, qui est fixe, et la nécessité de tracer un TS correspondant à la mesure, qui est variable, mais cherche à la résoudre dans le domaine des nombres. Elle cherche un nombre qui les représenterait tous, « un multiple de tout ça » (3.30, 558-560). Commençant à énumérer les instances, elle reconnait ne pas réussir à identifier ce multiple (3.30, 569-573), et finit par proposer un signe, « un truc qui finit par zéro, parce que zéro c'est un multiple de tout » (3.30, 575-576). Ce « truc », ce nombre tout à la fois signe générique et nombre, que nous avons nommé NSG (Nombre-Signe Générique) lorsqu'il s'agissait d'un chiffre, et NPG (Nombre Potentiellement Générique) sinon, a déjà été observé chez certains groupes (45e, 45m, 45k, 46k...). E1 exclut cette possibilité, puisque pour une expression « 10 », ça ne tracera qu'un certain nombre d'hexagones, qui ne changera pas si on change la mesure. Cependant, en fin de séance, une tentative avec répéter 0 fois sera faite, mobilisant le NSG 0. En fin de séance 5, E2 reviendra sur cette idée d'« un nombre qui finit par zéro ». Lorsque E1 cherche de nouveau à « trouver un nombre pour tous » (46e, transcription S5, 26'50), c'est-à-dire un nombre x tel que $\forall p \in \mathbb{N}, b_7^p = p + x$, E2 proposera « et si tu mets six sur un triangle de huit ça marche pas non plus » puis « et si tu mets dix sur le triangle de de tout ça marche » (46e, transcription S5, 27'02-27'05), ce qui sera invalidé par un test.

Une expression générique Lorsque E1 pense qu'il existe un bloc permettant de résoudre le problème, elle exprime le fait que pour elle, il doit nécessairement exister une expression de θ qui représente la valeur du nombre d'itérations des boucles pour toutes les instances de la mesure. Cette expression est instanciée ou évaluée à l'exécution, lorsque la mesure est connue, ce que simule en quelque sorte E1 lorsqu'elle remplace répéter ◯ fois par répéter 10 fois avant de revenir à répéter ◯ fois. répéter 10 fois, pour l'élève, est une instruction spécifique (Komis, Touloupaki et Baron, 2017) [84], mais répéter ◯ fois a ainsi sans doute pour elle le statut d'une expression paramétrée, comme l'est répéter n fois. Cette dernière ne devient spécifique qu'à

83. Pour ce passage, la seule utilisation des traces de programmation amenait à penser que le signe vide était ici utilisé non pas comme SG, mais comme une étape de préparation avant modification, puisque le script obtenu n'a pas été testé.

84. Voir partie Analyse a priori.

558	23 :18.458	T2	ça veut dire qu'il faut euh un multiple ?
559	23 :20.838	T2	et euh | de tout ça
560	23 :24.527	T2	c'est quoi les mesures déjà ya vingt-deux ya onze
561	23 :28.765	T1	ya vingt-deux onze
562	23 :31.886	T1	sept six cinq quatre tr
563	23 :32.097	T2	sept
564	23 :34.068	T1	cinq quatre
565	23 :35.229	T1	quatre
566	23 :37.030	T1	quatre
567	23 :37.665	T1	cinq six sept onze
568	23 :37.665	G1	dénombre sur ses doigts
569	23 :39.163	T2	ben il y a
570	23 :39.163	G2	dénombre sur ses doigts
571	23 :40.305	T1	vingt-deux et euh
572	23 :42.507	T2	fois |
573	23 :46.527	T2	ah je sais pas
574	23 :49.090	T2	heum
575	23 :51.406	T2	bah un truc qui finit par un zéro
576	23 :55.955	T2	par ce que zéro c'est un multiple de tout
577	24 :01.370	T1	ah non ça | oui mais euh
578	24 :04.721	T1	oui mais si on marque par exemple euh dix ça va le faire euh chais pas combien de fois
579	24 :08.480	VALEURS	(BOUCLE)b1i0(VAL) : répéter *10* fois (commandes)<<>>
580	24 :08.752	EXEC	KEY_
581	24 :08.840	T1	fin ça va pas euh (finir) pour tous
582	24 :08.992	EXEC	KEY_
583	24 :12.156	T1	je pense qu'il y a un bloc qui existe
584	24 :14.386	VALEURS	(BOUCLE)b1i0(VAL) : répéter ** fois (commandes)<<10>>

Transcription 3.30 – 46e, S4 : Un nombre pour les représenter tous

l'exécution, en fonction de la valeur que prendra la variable n.[85] Le groupe 46e va alors explorer les différentes catégories de blocs — « ya quoi comme truc la dedans XXX » (46e, transcription S4, 24'59) demande E1 en parcourant ces catégories —, sans doute à la recherche de ce qui transformerait l'instruction spécifique en instruction paramétrée.

85. Notons que dans cette version Snap !, le typage des instructions empêche les élèves d'écrire par exemple répéter mesure fois. En revanche, il est possible d'écrire dire mesure. Nous n'avons pas été confronté à ce cas, mais cela pose un souci didactique : comment différencier dire mesure et dire mesure ?

615	25 :55.005	T1	ptêtre si on met ça comme ça
616	25 :55.191	VALEURS	(BOUCLE)b1i0(NEWVAL) : répéter *mesure* fois (commandes)
617	25 :59.152	T2	hmm hmm
618	26 :00.785	T1	ça non ?
619	26 :01.523	T2	je sais pas
620	26 :01.999	VALEURS	(BOUCLE)b1i0(NEWVAL) : répéter ** fois (commandes)
621	26 :03.411	VALEURS	(BOUCLE)b1i0(DROPVAL) : répéter *mesure* fois (commandes)
622	26 :03.445	T2	peut-être on essaye
623	26 :04.395	VALEURS	(BOUCLE)b1i0(NEWVAL) : répéter ** fois (commandes)
624	26 :05.838	T1	oui mais sauf que 'tends regarde
625	26 :08.527	T1	là on n'a pas le même | truc | que dans la mesure
626	26 :08.527	G1	montre ? avec la souris
627	26 :11.716	E1	lance TS22
628	26 :11.716	T1	si on commence celui-là
629	26 :12.215	EXECUTION	START receiveKey(472)
630	26 :12.286	ERREUR	ERR_Inside : IndexSizeError : Index or size is negative or greater than the allowed amount

Transcription 3.31 – 46e, S4 : Un doute sur la mesure

Le paramètre *mesure* Rapidement la variable mesure [86] est utilisée pour b_1. Cependant, E1 met en doute la validité de ce bloc : peu après avoir effectué l'action $b_1 \leftarrow$ mesure, elle l'annule, puis la refait et l'annule de nouveau, tout en marquant son doute avec un « je sais pas » (3.31, 619). Elle explique ensuite ce qui la fait douter : « oui mais sauf que 'tends regarde, là on n'a pas le même | truc | que dans la mesure » (3.31, 624-626). La captation vidéo n'étant pas assez précise pour identifier ce qui est montré « là », nous ne pouvons que faire des hypothèses :

— E1 compare la valeur de la première boucle des scripts et la mesure voulue : ainsi par exemple $b_1^{11} = 10$ et donc $b_1^{11} \neq mesure$. Dans ce cas, cela signifierait que cette élève conçoit mesure comme une dénotation de la valeur de la mesure, puisqu'elle est capable d'établir que dans le cas particulier où $mesure = 11$, $b_1 \neq mesure$. E1 aurait donc construit le concept de paramètre pour cette situation. L'expression « dans la mesure » évoquant par ailleurs un espace de stockage, en reprenant une formulation utilisée lors de l'institutionnalisation de la variable en fin de séance 3 : « une variable informatique est un espace de stockage, une mémoire, permettant de conserver une donnée. »

— Il est aussi possible que E1 soit perturbée par l'affichage du contenu actuel de la variable mesure : mesure [87]. Cependant, il est difficile d'établir avec quoi elle compare ce contenu. Ça n'est sans doute pas avec les boucles du script en cours, puisqu'elles sont toutes avec le SG ◯. Et pourquoi comparer avec les boucles d'un autre script, puisque ce serait une autre valeur de la mesure ?

Il est donc probable que E1 ait conceptualisé mesure comme étant un objet dénotant une certaine valeur non connue *a priori*, mais identique à celle dénotée par l'entrée de l'utilisateur, tout en constatant l'impossibilité que ce bloc suffise, puisque le dénoté de mesure ne correspond pas au dénoté du nombre de répétitions de la boucle b_1 d'un TS de mesure δmesure.
Malheureusement, les erreurs dues à cette entrée vide vont semble-t-il invalider l'exploration de cet objet. Le groupe 46e va poursuivre avec un autre objet proche, taille_hexagone, variable définie dans le bloc d'initialisation. Il aurait été souhaitable de ne pas rendre cette variable visible ou accessible par les élèves, mais les seules possibilités avec Scratch ou Snap ! sont d'afficher ou non la valeur de la variable sur l'espace d'exécution (figure 3.93, p. 426). Toutes les variables créées sont disponibles dans la catégorie « variables », qui est celle utilisée à ce moment par ce groupe.

Figure 3.93 – Les variables, cachées ou non, sont toujours accessibles (46e, S4)

Généralisation effective de B_1 et B_2 En début de séance 5, immédiatement après les faits partagés en plénière, E1 va affecter l'expression mesure - 1 à toutes les boucles. Cette affectation sans différenciation a déjà été constatée chez d'autres groupes, comme le 46i. Ici, E1 commence par dupliquer le script du TS4 sur le script générique, puis effectue $b_1 \leftarrow$ mesure - 1, et lance le test avec une valeur entrée à l'invite de 4 (script 3.18a, p. 427). Il y a bien cohérence entre le script choisi et la mesure entrée, et comme le tracé est valide, E1 en déduit sans doute

86. À ce stade, pour les élèves, il ne s'agit que d'un bloc particulier.
87. Suite à l'entrée d'une chaine vide à l'invite, et qui va entraîner les erreurs évoquées dans la partie activité.

que ce bloc, cette instruction particulière résolvant le problème, qu'elle cherchait en séance 4, est l'instruction $b_1 \leftarrow$ mesure - 1. Elle étend donc sa solution, valide pour b_1, à l'ensemble des boucles (script 3.18b, p. 427). Le résultat est erroné :

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 2 fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter 2 fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter 2 fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter 2 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 3 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 6 fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter 2 fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(a) Premier test du bloc cherché

Block_478
Quand n est pressé
initialisation
compteur prend la valeur 0
afficher la variable mesure
tourner de 120 degrés à droite
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 60 degrés à gauche
tracer
tourner de 60 degrés à gauche
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à droite
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 60 degrés à droite
tracer
tourner de 60 degrés à droite
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 1] fois (commandes)
.....tracer
tourner de 120 degrés à gauche
répéter [*mesure* − 1] fois (commandes)
.....tracer
final
dire [regroupe [| [regroupe [| J'ai compté || *compteur* |]] || hexagones |]]

(b) Extension à l'ensemble du script

Script 3.18 – Premières tentatives de généralisation (46e, S5)

$$TS(n)[5] \rightsquigarrow \left(\text{  } \text{} , 34, 5 \right)$$

La mobilisation par E1 de la variable *mesure* est le prolongement de son exploration en fin de séance 4. L'utilisation de l'opérateur et de la relation « moins un » provient quant à elle sans doute des faits partagés en début de séance, puisque c'est la première action de E1 suite à cette plénière. Il est aussi possible que l'expression identique choisie pour b_2, $b_2 \leftarrow$ mesure - 1 — fait également partagé en plénière — n'ait pas le même sens : si la première expression signifie que pour trouver la valeur de b_1 on fait moins avec la mesure, la deuxième signifierait que pour trouver b_2 on fait moins avec b_1 — et la relation « moins un » entre b_1 et b_2 est un fait qui a été partagé en groupe classe. Dans ce cas, le signe mesure serait un indice de ce qu'il faut changer,

mais ne serait pas univoque, puisque référent soit la valeur de la mesure soit la valeur de b_1. E2, quant à elle, ne semble pas avoir construit le sens de la variable *mesure*. Elle commence par se demander « qu'est ce que c'est c'te mesure ? » (46e, transcription S5, 1'13) lorsque E1 l'utilise (« faut mettre ça [...] mesure », 46e, transcription S5, 1'08-1'12). Elle propose ensuite l'expression « plus un » avec comme justification « le dessin » (46e, transcription S5, 1'26), mais ne semble pas mettre de sens derrière cette expression : plus tard, après la rétroaction invalidant le script 3.18b (p. 427), elle commentera seulement « c'est pas moins peut-être » (46e, transcription S4, 3'00), montrant par là qu'elle reste sur la relation mais sans chercher à faire de lien avec le tracé, ni avec les faits préalablement construits — notamment que pour tous les scripts valides $b_1 \neq b_2$. E1 va remobiliser les faits partagés (figure 3.94, p. 430) pour construire les faits $b_1^n = n - 1$ et $b_2^n = n - 2$ (3.32, p. 429). Tout d'abord elle explique que :

— Si $mesure = 8$, $b_1^8 = 7$: « par exemple pour la mesure huit, après quoi la boucle d'en dessous elle fait sept » (3.32, 170-174) ;

— si $mesure = 5$, $b_1^5 = 4$: « la mesure cinq la boucle d'en dessous elle fait quatre » (3.32, 176)

Elle en conclut : « c'est différent » (3.32, 177). Mais qu'est-ce qui est différent pour cette élève ? La mesure et la boucle 1, ou la boucle 1 et la boucle 2 ? On peut penser que la différence existant en la valeur de la mesure et le nombre de répétitions de la boucle 1, est un premier fait que construit E1, mais que la différence entre les deux premières boucles se construit à la suite, les éléments « 8-7-6 » pour « mesure - boucle 1 - boucle 2 » étant les uns au-dessus des autres (figure 3.94, p. 430). En effet, elle oppose leur choix $b_i = mesure - 1$ et absence de changement : « sauf que là du coup ça fait tout le temps la mesure euh... » (3.32, 178-179). En mobilisant toujours les faits au tableau — et ceux partagés à l'oral — E1 établit que « du coup, il faudrait faire euh moins un » (3.32,180-181) pour b_1, action qu'elle rétablit alors. Contrairement à la première occurrence de l'action $b_1 = mesure - 1$, qui désignait pour E1 une opération à faire à l'ordinateur pour « qu'il se débrouille tout seul », ici l'expression $mesure - 1$ est bien vue comme une expression mettant en relation la valeur de la mesure et le nombre de répétitions de la boucle b_1. D'ailleurs, E1 enchaine en établissant qu'il faut modifier b_2, en cherchant ce qu'« il faudrait mettre » (3.32, 183) à la place de la mesure : elle remplace $b_2 =$ [mesure - 1] par $b_2 =$ [◯ - 1]. En effet, il y a la même relation entre la mesure et la boucle 1 qu'entre la boucle 1 et la boucle 2 : ici E1 se base sur une relation intra-objectale, liant les boucles d'une même instance entre elles. Comme elle ne voit pas par quoi remplacer *mesure* [88], elle finit par émettre la possibilité « au pire » que ce soit moins deux (3.32 191-192), identifiant qu'entre b_2 et la mesure on « fait moins deux » (3.32, 193). En anticipant l'égalité des deux expressions $b_1 =$ [mesure - 1] et $b_2 =$ [mesure - 1], et en en déduisant — avec peu de certitudes néanmoins — que $b_2 =$ [mesure - 2], E1 construit, ou confirme, le caractère univoque de [mesure] : c'est une expression qui a pour seul et unique dénoté la valeur de la mesure entrée, et ceci même si cette valeur change.
Les élèves vont alors poursuivre la modification du groupe en identifiant les relations entre la mesure et le nombre de répétitions de la boucle : par exemple pour b_4, elles mobilisent $b_4^4 = 2$, ce qui aboutit à $b_4 \leftarrow mesure - 2$ (3.32, 211-213). En procédant de la même façon pour les autres boucles, exceptée b_7 qui est traitée à part, les familles de boucles B_1 et B_2 sont correctement généralisées.

88. Si on veut représenter la relation $b_2 = b_1 - 1$, il faudrait alors remplacer [mesure] par [mesure - 1] : $b_2 \leftarrow$ [mesure - 1 - 1]. Ceci serait la manifestation d'une capacité à établir des substitutions dans une expression.

170	04 :25.147	T1	moi pour moi le plus il marche pas parce que regardes
171	04 :25.147	G1	montre trace écrite classe
172	04 :27.401	T1	la prof elle a dit | par exemple pour la mesure huit
173	04 :27.401	G1	montre trace écrite classe
174	04 :30.793	T1	après quoi la boucle d'en dessous elle fait sept
175	04 :34.096	T2	ha
176	04 :39.948	T1	la mesure cinq la boucle d'en dessous elle fait quatre
177	04 :43.284	T1	c'est différent
178	04 :44.767	T1	sauf que là | du coup
179	04 :46.233	T1	ça fait tout le temps la mesure euh |
180	04 :48.724	T1	du coup il faudrait faire euh
181	04 :51.912	T1	moins un
182	04 :52.725	VALEURS	(BOUCLE)b1i0(DROPVAL) : répéter *[mesure − 1]* fois (commandes)
183	04 :53.518	T1	et là il faudrait pas mettre mesure mais il faudrait mettre euh |
184	04 :53.781	VALEURS	(BOUCLE)b2i0(NEWVAL) : répéter [** − 1] fois (commandes)
185	04 :56.343	AFFICHAGE	AFFBL_variables
186	04 :56.501	T1	celui-là euh
187	04 :59.009	T1	les nombres de la boucle
188	05 :01.500	T2	les nombres de la boucle c'est-à-dire euh
189	05 :04.410	T2	ouai tu mets euh
190	05 :04.410	G2	montre trace écrite classe
191	05 :06.204	T1	ou au pire on fait moins deux
192	05 :09.145	VALEURS	(BOUCLE)b2i0(VAL) : répéter [− *2*] fois (commandes)<<1>>
193	05 :09.474	T1	puisque là elle fait moins deux
194	05 :10.778	VALEURS	(BOUCLE)b2i0(NEWVAL) : répéter [*mesure* − 2] fois (commandes)
195	05 :11.408	T2	ouai
196	05 :12.227	T2	vas-y on essaye avec moins deux
197	05 :16.045	VALEURS	(BOUCLE)b3i0(NEWVAL) : répéter ** fois (commandes)
198	05 :17.022	T1	et là c'est moins deux aussi
199	05 :20.386	VALEURS	b0i1(VAL) : mesure − *2*<<1>>
200	05 :20.664	T2	pourquoi tu vas [rire]
201	05 :21.295	VALEURS	(BOUCLE)b3i0(DROPVAL) : répéter *[mesure − 2]* fois (commandes)
202	05 :22.631	T1	oui chais pas | euh | par contre attends
203	05 :27.139	T1	je vais dupliquer
204	05 :28.063	STRUCTURE	DUPLIC_398-646(398)
205	05 :28.434	T1	parce que je veux regarder un truc en même temps
206	05 :29.923	EXEC	KEY_space
207	05 :30.917	T1	mais après euh | j'pense que c'est ça | tu vas être avec moi enfin euh
208	05 :31.054	EXEC	KEY_right arrow
209	05 :35.326	T1	toi tu sais pas mais | moi pour moi ça à l'air logique
210	05 :38.031	T2	mais c'est c'est logique
211	05 :40.112	T1	là deux deux
212	05 :41.808	T1	là ya deux donc on fait m'| moins deux
213	05 :46.791	VALEURS	(BOUCLE)b4i0(VAL) : répéter [mesure − *2*] fois (commandes)<<1>>

Transcription 3.32 – 46e, S5 : le « moins deux »

Le problème de la boucle b_7 : faits tiers et faits seconds Pour cette boucle b_7, E1 cherche toujours une relation entre le nombre de répétitions et la mesure, et en se basant sur le fait $b_7^4 = 6$, elle établit $b_7^4 = 4 + 2$ et en déduit $b_7 \leftarrow$ mesure + 2 (3.33). Ceci est bien entendu valable seulement si $mesure = 4$, mais les autres faits construits qui aboutiraient à cette conclusion (par exemple $b_7^5 = 8$) ne sont pas mobilisés.
On peut noter que E2 cherche à suivre le raisonnement de E1, mais l'interprète de façon inexacte et sans construction du paramètre. En effet, pour la boucle b_7, E2 propose « ça c'est moins six » (3.33, 246-247). Elle semble se baser une nouvelle fois sur les signes et non sur le sens de la relation. Les faits suivants (ici seconds) sont construits par E2, lorsque E1 met côte-à-côte

	mesure 4	8	mesure 5	mesure 6
boucle 1	3	7	4	5
boucle 2	2	6	3	4
boucle 3	2	6	3	4

...

Figure 3.94 – Des faits partagés au tableau noir : 7'38, bleu : 14'04

(46e, S5)

230	06 :02.084	T1	faut juste le déplacer
231	06 :04.608	T1	je vais le mettre face à l'autre
232	06 :06.756	T1	là ya trois donc on fait moins un
233	06 :08.788	T1	là ya deux donc on fait moins deux
234	06 :10.500	T1	moins deux moins deux moins deux
235	06 :14.368	T2	moins deux
236	06 :15.778	T2	moins trois
237	06 :17.310	T1	là du coup là c'est moins un
238	06 :18.933	T2	hein ?
239	06 :19.039	VALEURS	(BOUCLE)b5i0(VAL) : répéter [mesure – *2*] fois (commandes)<<1>>
240	06 :20.121	T2	moins un ?
241	06 :21.957	T1	et du coup
242	06 :23.677	T1	là c'est bah pour moi
243	06 :31.636	T1	et là c'est euh
244	06 :41.275	T1	celle là XXX pas
245	06 :43.751	VALEURS	(BOUCLE)b8i0(VAL) : répéter [mesure – *2*] fois (commandes)<<1>>
246	06 :44.291	T2	ça c'est
247	06 :47.446	T2	moins six
248	06 :49.347	T1	ben quatre
249	06 :51.274	T1	là c'est la mesure de quatre
250	06 :53.036	G1	semble compter sur ses doigts
251	06 :55.585	T1	six c'est euh plus deux
252	06 :58.126	T1	du coup là il faudrait faire plus deux | mesure plus deux

Transcription 3.33 – 46e, S5 : relation entre b_7 et $mesure$

les scripts $TS(4)$ et $TS(n)$ (3.33, 230-234) :

$$b_2^4 = \boxed{2} \longrightarrow b_2 = mesure - \boxed{2}$$
$$b_3^4 = \boxed{2} \longrightarrow b_3 = mesure - \boxed{2}$$
$$b_4^4 = \boxed{2} \longrightarrow b_4 = mesure - \boxed{2}$$
$$b_5^4 = \boxed{2} \longrightarrow b_5 = mesure - \boxed{2}$$

Le motif identifié par E2 serait donc $b_i = \square \longrightarrow b_i = mesure - \square$. Comme $b_6^4 = \boxed{3}$, E2 pense tout d'abord que $b_6 = mesure - \boxed{3}$ (3.33, 236). Elle ne peut saisir le raisonnement qui pousse E1 à affirmer pour b_6 « là du coup là c'est moins un » (3.33, 237-240). Sans explication complémentaire lui permettant d'identifier un autre motif dans ces faits, E2 poursuivra son raisonnement pour

b_7 : puisque $b_7^4 =$ 6, on devrait avoir $b_7 = mesure -$ 6. Elle envisage ainsi :

$$b_6^4 = 3 \longrightarrow \qquad b_6 = mesure - 3$$
$$b_7^4 = 6 \longrightarrow \qquad b_7 = mesure - 6$$

E1 n'a pas construit et mis en relation les mêmes faits en mettant côte-à-côte les deux scripts :

$$b_1^{④} = 3 \wedge 3 = ④ \text{- 1} \longrightarrow b_1 = mesure \text{- 1}$$
$$b_2^{④} = 2 \wedge 2 = ④ \text{- 2} \longrightarrow b_2 = mesure \text{- 2}$$
$$b_3^{④} = 2 \wedge 2 = ④ \text{- 2} \longrightarrow b_3 = mesure \text{- 2}$$
$$b_4^{④} = 2 \wedge 2 = ④ \text{- 2} \longrightarrow b_4 = mesure \text{- 2}$$
$$b_5^{④} = 2 \wedge 2 = ④ \text{- 2} \longrightarrow b_5 = mesure \text{- 2}$$
$$b_6^{④} = 3 \wedge 3 = ④ \text{- 1} \longrightarrow b_6 = mesure \text{- 1}$$
$$b_8^{④} = 2 \wedge 2 = ④ \text{- 2} \longrightarrow b_8 = mesure \text{- 2}$$

$$b_7^{④} = 6 \wedge 6 = ④ \text{+ 2} \longrightarrow b_1 = mesure \text{+ 2}$$

E1 construit des faits tiers, des faits raisonnés qui s'appuient sur l'existence d'une relation entre la mesure(④) et les nombres de répétitions des boucles (□). Ne se basant que sur le script $TS(4)$ et les faits précédents construits à partir de ce script, la relation établie pour b_7 parait valide. E2 construit des faits seconds, relations entre faits premiers, qui entrent en tension avec ceux construits par E1. Ces deux élèves construisent ainsi des faits différents, ils ne peuvent se comprendre. E1 va alors chercher à expliquer son raisonnement (3.34). Sa première tentative reprend les motifs qu'elle a identifié : « regarde là le trois c'est le un et le deux c'est le deux et le six c'est plus deux » (3.34, 289-291), ce qui ne semble guère convaincre E2. En reprenant le codage des motifs identifiés par E1, cela pourrait s'écrire :

le 3 c'est le (-) 1
le 2 c'est le (-) 2
le 6 c'est le + 2

Dans cette première explication, E1 met en relation des signes semblables (des chiffres) mais qui n'ont par le même sens : 3 représente un nombre de répétitions pour une boucle, (-) 1 représente la relation entre le nombre de répétitions d'une boucle et la mesure. E1 va rendre explicite cette relation, d'abord à partir du $TS(4)$:

« ④ moins combien est égal à 3 ? moins un, ④ - 1 est égal à 3 » (3.34, 297-300).

Elle exprime ensuite que cette relation est générique en prenant un autre exemple, pour $mesure = 5$:

« donc euh chais pas ⑤ moins c'est égal à 4 donc tu fais moins un » (3.34, 302-303).

285	07 :42.867	T2	celui là ça fait euh | trois
286	07 :42.867	G2	montre script b8
287	07 :45.957	T2	du coup c'est
288	07 :46.850	T1	plus (trois)
289	07 :47.948	T1	regardes là le trois c'est le | un
290	07 :50.653	T1	et le deux c'est le deux
291	07 :52.628	T1	et le six c'est plus deux
292	07 :54.185	T2	ah (ok)
293	07 :56.365	T1	'fin j'pense mais euh ça m'la pas mis
294	07 :58.889	T1	donc euh
295	08 :03.405	T2	et pourquoi t'en a mis qu'un ?
296	08 :05.536	T1	bah parce que euh
297	08 :09.650	T1	quatre | moins combien est égal à trois ?
298	08 :14.034	T1	moins un
299	08 :15.271	T2	oui
300	08 :16.271	T1	quatre moins un est égal à trois
301	08 :17.926	T1	donc euh
302	08 :19.508	T1	euh chais pas cinq moins c'est égal à quatre
303	08 :21.614	T1	donc tu fais moins un
304	08 :23.654	T1	et euh pour euh quatre moins
305	08 :24.702	EXECUTION	START receiveKey(388)
306	08 :25.576	EXECUTION	FIN receiveKey(388)
307	08 :27.064	T1	quatre moins combien pour aller à deux c'est moins deux
308	08 :29.572	T1	donc on met moins deux là
309	08 :31.177	T2	ouai
310	08 :34.021	T1	et là
311	08 :35.913	T1	donc là c'est six donc c'est plus | plus deux
312	08 :39.651	T1	donc là j'ai mis plus deux
313	08 :41.863	T1	'fin comment
314	08 :43.173	T2	six plus deux
315	08 :43.338	T1	c'est ça mais apparemment c'est faux donc euh
316	08 :45.238	T2	six plus deux
317	08 :48.156	T1	nan
318	08 :48.991	T1	nan quatre plus deux
319	08 :50.458	T1	c'est égal à six
320	08 :50.645	T2	ah oui
321	08 :51.941	EXECUTION	START receiveKey(478)
322	08 :51.981	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
323	08 :53.538	T1	mais pourquoi ça marche pas ça ?
324	08 :54.911	ENTRÉE	ANSW <<8>>
325	08 :56.554	T1	oh ça marche
326	08 :56.855	EXECUTION	FIN receiveKey(478)
327	08 :57.446	T1	ah non
328	08 :58.519	T1	mais c'est au début que ça marche et à la fin ça bug
329	09 :01.895	T1	un deux trois quatre cinq six sept huit
330	09 :01.895	G1	compte b1 avec la souris

Transcription 3.34 – 46e, S5 : « le trois c'est le un et le deux c'est le deux »

Pour E1, le ⑤ a ici valeur de nombre à caractère générique, qui montre bien qu'elle associe la variable *mesure* à un nombre *potentiel* avec lequel elle pourrait être substituée dans l'écriture $mesure - 1$. E2 ne semble pas pour autant avoir compris ce raisonnement, puisque lorsque E1 aborde la boucle b_7 en affirmant « donc la c'est six donc c'est plus deux » (3.34, 311), E2 semble comprendre « six plus deux » (3.34, 314 et 316). E2 formule une expression qui utilise le nombre de répétitions ($b_7^4 = 6$) comme paramètre, et non la mesure : elle reste sur les nombres et sur les relations entre les nombres de répétitions (donc des relations intra ou inter-objectales), là où E1 semble avoir établi des relations entre la mesure et le nombre de répétitions (donc des relations trans-objectales). Il est probable que E2 soit aussi confuse à cause de l'utilisation, dans les exemples donnés par E1, du nombre quatre, qui désigne des objets différents : celui-ci est dans la première explication la valeur de la mesure(④), et, dans l'application exemplaire suivante, un nombre de répétitions (4).

Le problème de la boucle b_7 : sens et dénotation Le script obtenu (script 3.17, p.410) est alors testé avec deux valeurs différentes pour la mesure entrée :

$TS(n)[5] \rightsquigarrow$ 

$, 32, 5$

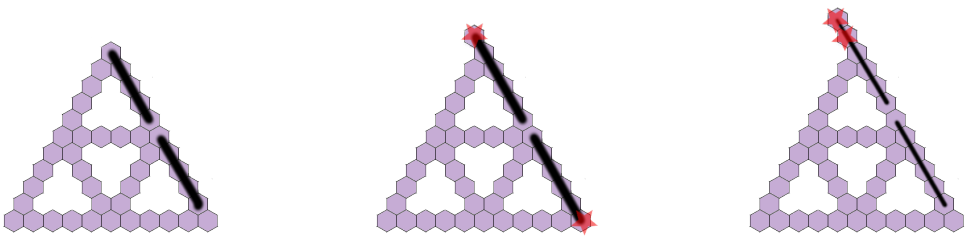
$TS(n)[8] \rightsquigarrow$ 

$, 56, 8$

Après avoir envisagé que l'erreur pouvait provenir de b_8, le groupe 46e va explorer de nombreuses possibilités pour la boucle b_7 :

— mesure + 4, qui s'avérera valide pour $TS(n)[6]$ mais invalide pour $TS(n)[8]$;
— mesure * 2, mesure / 2 ;
— mesure - 0.5, mesure - 3 ;
— mesure + 1, mesure + 2 ;
— mesure - 0.25, mesure - 0.5, mesure - 0.75 ;
— mesure + 6, mesure + 4, mesure + 10 ;
— mesure - mesure, mesure + mesure, et pour finir mesure + mesure - 1 ;

Cette longue suite d'essais laisse penser que les élèves ne font de lien entre la boucle b_7 et ce qu'elle trace (figure 3.95, p. 433). En effet, les élèves restent sur des relations entre des nombres en



(a) $b_7^n = 2(n-1)$ **(b)** $b_7^n = 2n-2$ ou $2(n-1)$ **(c)** $b_7^n = 2n-2$

Figure 3.95 – Différents sens pour les expressions de b_7

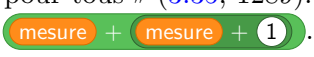
essayant de s'adapter au tracé, mais sans prendre en compte le sens des expressions relativement au tracé. La boucle b_7 doit tracer l'équivalent de deux côtés de triangles de base, auquel il faudrait

enlever deux hexagones (figure 3.95b ou 3.95c, p. 433), ou bien tracer l'équivalent de deux fois ce que trace la boucle b_1 (figure 3.95b ou 3.95a, p. 433). Mais le groupe 46e ne semble que s'ajuster aux différentes rétroactions : « il en manque un » (46e, transcription S5, 9'41), « euh ben il en manque euh il tourne » (46e, transcription S5, 9'48), « il faut pt'être faire moins un » (46e, transcription S5, 10'51), « ah ben oui | il a pas tracé assez long » (46e, transcription S5, 11'09), « faudrait pt'être ne faire plus quatre » (46e, transcription S5, 12'00), « avec un autre nombre euh genre huit là » (46e, transcription S5, 12'34)... Elles cherchent une expression dont la dénotation produirait un tracé valide, mais elles ne cherchent pas le sens de ce que trace la boucle b_7.
Cependant, durant un court moment (46e, transcription S5, 10'18-10'21), E1 semble envisager le tracé de b_7 comme étant « deux fois la boucle ». En effet, elle constate « mais c'est que là il fait qu'une seule boucle et là il en fait deux fois la boucle ». Comme elle montre ces localisations dans le script avec le curseur de la souris, nous ne pouvons pas identifier ces localisations sur l'image. On peut cependant raisonnablement penser que le « là il en fait deux fois la boucle » s'applique à la boucle b_7, qui est le problème à résoudre pour les élèves. Cette boucle b_7 semble tracer deux côtés du triangle de base, ce qui est sans doute le sens donné par E1 : une boucle trace un côté, mais b_7 doit en tracer deux. Cependant, aucun fait n'a été préalablement construit concernant b_7 : dans les séances précédentes, les séries d'ajustements successifs n'ont pas pu permettre d'identifier des schémas concernant cette boucle, et dans la séance 5, seul le script traçant un TS4 est disponible, en sus des faits partagés au tableau, mais qui ne concernent pas b_7. Le groupe 46e ne peut donc s'appuyer sur d'autres faits pour établir une relation entre la valeur de la mesure et le nombre de répétitions de b_7 [89]. Et en eut-il, aurait-il pu identifier la relation en se basant uniquement sur les nombres ? Une élève de ce niveau de classe, si elle n'a pas l'habitude de chercher des relations entre des nombres, peut elle identifier un schéma en ne se basant que sur les faits suivants ?

— pour $mesure = 4$, $b_7^4 = 6$,
— pour $mesure = 5$, $b_7^5 = 8$,
— pour $mesure = 6$, $b_7^6 = 10$,
— pour $mesure = 7$, $b_7^7 = 12$,
— pour $mesure = 11$, $b_7^{11} = 20$,
— pour $mesure = 22$, $b_7^{22} = 42$,

En revanche, l'analyse du tracé *que devrait faire* b_7, associée à ce que représente *mesure* pour un TS, doit permettre de construire cette relation.
Ce groupe va néanmoins tenter de construire des faits relatifs à b_7 en reprenant les réussites et échecs de leurs explorations :

— $b_7^4 = 6$, mais exprimé « on a fait le triangle de six et c'était XXX quatre » (3.35, 844), ce qui pourrait aussi évoquer $b_7^6 = mesure + 4$;
— $b_7^6 = mesure + 4$ (3.35, 994) ;
— mais $b_7^8 \neq mesure + 4$ (3.35, 1278-1280 et 1290-1295) ;
— et $b_7^8 = mesure + 6$ (3.35, 1282-1285), vérifié par un test ;

E1, constate alors que le nombre que l'on doit ajouter à la mesure change en fonction de la mesure, et en déduit, comme elle l'avait dit en séance 4, « il faudrait un nombre pour tous » (3.35, 1289). C'est sans doute ce qui l'amènera à proposer, en toute fin de séance, $b_7 \leftarrow$ 

.

89. E2 souligne d'ailleurs l'importance de garder une trace des faits construits : « faut qu'on retienne les nombres qu'on a fait parce que sinon XXX XXX [rire] » (46e, transcription S5, 15'15).

844	25 :39.743	T1	on a fait le triangle de six et c'était XXX quatre
845	25 :40.230	VALEURS	(BOUCLE)b7i0(VAL) : répéter [mesure + **] fois (commandes)<<0.75>>
846	25 :43.251	T1	pour le triangle de XXX | triangle de Sierpinski de mesure six
847	25 :43.251	G1	écrit sur le cahier
...			
994	25 :50.442	T1	en fait | la boucle euh l'avant dernière c'était plus six mesure plus six || euh non c'était plus euh quatre
995	25 :50.442	G1	écrit sur le cahier
...			
1277	26 :03.236	T2	ouai
1278	26 :05.309	T1	mais pour le huit | pour euh le triangle de Sierpinski de mesure huit
1279	26 :05.309	G1	écrit sur le cahier
1280	26 :14.792	T1	la mesure | si on faisait plus quatre ça marchait pas
1281	26 :14.792	G1	écrit sur le cahier
1282	26 :18.054	T1	mais mesure plus six | j'crois ça marchait
1283	26 :25.393	VALEURS	(BOUCLE)b7i0(VAL) : répéter [mesure + *6*] fois (commandes)<<>>
1284	26 :26.918	ENTRÉE	ANSW <<8>>
1285	26 :40.840	T1	tu vois là ça marche
1286	26 :42.888	T2	wow
1287	26 :43.109	EXECUTION	FIN receiveKey(478)
1288	26 :44.076	T1	mais c'est parce que c'est plus six
1289	26 :50.625	T1	mais faudrait trouver un nombre pour tous
1290	26 :54.452	T1	parce que regarde si j'mets quatre
1291	26 :56.509	T1	pour le triangle de huit
1292	26 :57.653	VALEURS	(BOUCLE)b7i0(VAL) : répéter [mesure + *4*] fois (commandes)<<6>>
1293	26 :57.747	EXECUTION	START receiveKey(478)
1294	26 :57.812	ENTRÉE	ASK <<Quelle est la mesure du triangle de Sierpinski que tu veux dessiner ?>>
1295	26 :59.918	T1	ça marche pas
1296	27 :00.078	ENTRÉE	ANSW <<8>>

Transcription 3.35 – 46e, S4 : recueil de faits sur b_7

Le problème de la boucle b_7 : trouver une expression Il est intéressant de noter que E1, qui a établi ce que représente la variable *mesure* dans sa relation avec le nombre de répétitions des boucles, semble perdre le sens des expressions arithmétiques utilisées. Les expressions $mesure/2$ ou $mesure + 0.5$ ou encore $mesure - 0.25$ ne semblent pas ainsi pas faire sens pour ces élèves :

— soit il n'y a aucun lien de fait entre répétitions et tracé (que signifierait « tracer 0.25 hexagones ? »)

— soit ces élèves veulent exprimer une relation plus complexe mais ne savent pas comment le faire dans θ : 

pourrait ainsi signifier $mesure + 0.25 \times mesure$, sans que les élèves sachent comment représenter cette structure profonde (Drouhard et Panizza, 2012) dans Snap !, qui oblige à insérer un opérateur comme paramètre d'un autre opérateur : 

.

Cette structure profonde et le sens de la *mesure* seront néanmoins mobilisés en fin de séance avec l'ultime tentative des élèves : $b_7 \leftarrow$ 

Construction des nécessités

Dans ce groupe, il semble que les nécessités aboutissant à la construction du concept de paramètre aient été au moins en partie construites. Le problème été rapidement et bien posé : il faut construire un script qui permet de tracer plusieurs instances de TS, et ce en fonction de la mesure entrée. La nécessité de trouver « un bloc qui existe » pour résoudre la tension fixe-variable, entre les expressions des boucles qui ne changent pas, et les dénotés de ces expressions qui changent en fonction de la mesure entrée, est construite. Elle amène les élèves à chercher d'abord « un nombre pour tous ». E1 a bien établi que si les expressions du nombre de répétitions des boucles étaient une représentation d'un entier (un nombre écrit en chiffres), cela ne traçait que l'instance

correspondant à ces expressions. Il existe donc une expression qui change en même temps que la mesure change — puisque ces élèves, notamment E1 ont bien mis en relation mesure, script, et tracé. Cette expression,  , est peut-être induite par les faits partagés en début de séance 5, mais le lien est ensuite fait entre l'expression, la représentation de la mesure par la variable  , et les valeurs du nombre de répétitions des boucles, pour les deux premières familles de boucles B_1 et B_2, c'est-à-dire toutes les boucles sauf b_7.
En revanche, faute d'avoir mis du sens sur les valeurs du nombre de répétitions des boucles, et faute de faits en nombres suffisants, le groupe 46e doit se limiter à la recherche de relations entre des nombres, et cette relation est complexe pour la boucle b_7.

Bilan

L'espace des faits-contraintes représenté figure 3.96 (p. 442) représente la construction du problème pour le groupe 46e. Cependant, on a pu constater dans l'analyse des interactions que, si les nécessités aboutissant à la construction du concept de paramètre semblent bien présentes chez E1, elles sont beaucoup plus incertaines chez E2. Ainsi, E1 évolue rapidement vers un REX algébrique ou algorithmique, tandis que E2 semble rester sur un REX arithmétique et numérique : seul le monde des nombres écrits en chiffres et de leur relation est envisagé comme explication des phénomènes ou domaine d'exploration des possibles. En témoigne la persistance du zéro (un nombre « qui finit par zéro »), ou encore l'idée qu'il fallait rajouter le nombre correspondant à l'instance à la fin de la chaine de caractère définissant le bloc d'initialisation.
E1 a bien identifié, pour B_1 et B_2, le schéma reliant mesure et nombre de répétions des boucles : en s'appuyant sur plusieurs faits relatifs à ces boucles, elle a été capable de considérer la mesure comme élément essentiel de la généricité, et de mobiliser une représentation de la mesure dans θ. Ce passage vers la représentation de la mesure,  , reste cependant obscur, faute d'explication de la part de l'élève. On peut soupçonner que la représentation affichée de la mesure, en plus des faits partagés en plénière et de l'institutionnalisation de la variable *compteur* en fin de séance 2, ait eu un rôle fondamental :

- la variable informatique a été définie comme « un *espace* de stockage, une mémoire, permettant de conserver une donnée »,
- en début de séance 4, est partagé le fait que la valeur entrée par l'utilisateur est affichée dans le signe rendant compte de la valeur des variables : « vous avez la mesure qui s'affiche » (46 Prof, S4, 11'07) [90],
- le signe  , affiché sur l'espace d'exécution, évoque cette idée d'espace contenant,
- E1 constate que « là on n'a pas le même | truc | que *dans* la mesure », le « dans » évoquant de nouveau cet espace contenant.

Ainsi, E1 aurait construit, ou tout au moins commencé à construire, le concept de paramètre en s'appuyant sur ce concept d'« espace contenant » de la variable informatique, associé à une représentation dans θ. Elle manipule cette représentation dans une expression, comme si la valeur était connue : c'est un paramètre. Dans la séance 6, lorsque l'enseignante reprécisera le problème (trouver une façon de calculer le nombre d'hexagones d'un TS rapidement), E1, après avoir pris un exemple (3 là et 3 là etc...), évoque de suite $mesure - 1$ mais en précisant que « ça marche pas » (méthode non explicitée), puis « faut faire un truc par rapport à la mesure » (21'04). Le

90. Avec l'autre classe, l'enseignante précisera d'ailleurs que la valeur entrée « s'affiche *dans* mesure » ou « se met directement *dans* mesure » (45 Prof, S4, 25'50).

groupe trouvera comme méthode « on compte le nombre d'hexagones on fait x3 et après moins 3 (ceux qu'on répète) », avec comme exemple $9 \times 3 - 3 = 24$ (figure 3.97, p. 443). Même si E1 envisage d'utiliser la mesure, le fait de partir sur le nombre d'hexagones d'un triangle de base (avec la méthode « on compte ») ne rend plus visible cette mesure (faible entropie). Un autre groupe, partant du même principe, a néanmoins abouti à $((mesure \times 3 - 3) \times 3) - 3$, ce qui est une formulation valable. Il faut noter que si E1 envisageait l'utilisation de la mesure, les deux autres élèves n'en étaient pas rendu à ce stade, ce qui pose une nouvelle fois le problème d'une avancée simultanée de tous les élèves dans des travaux en binôme [91].

3.6.3 Synthèse Groupement 5

Le groupement 5 est constitué de groupes ayant manipulé mesure, avec une exploration plus directe que le groupement 4.
Pour les trois groupes 45m, 46e et 45b, la construction des premières instances montre de façon surprenante des TEA peu stabilisés :

— pour le groupe 45m, qui va construire les instances des TS5, TS9 et TS1769, le TEA $+1/+1/+1$ semble résistant, et amène son extension plus générale $+a/+a/+a$, lors de la création du TS9 ; cependant ce groupe parviendra à construire directement les familles de boucles B_1 et B_2 du TS1769, pour une valeur entrée de 1769 ;

— pour le groupe 46e, qui construira les instances demandées des TS5,TS6, TS7, TS11 et TS22, le TEA $+1/+1/+1$ semble là aussi résistant. Après une application du TEA valide $+1/+1/+2$ pour le TS7, l'instance plus lointaine du TS11 sera construite avec le TEA $+a/+a/+a$, et le TS22 avec $*a/*a/*a+2$;

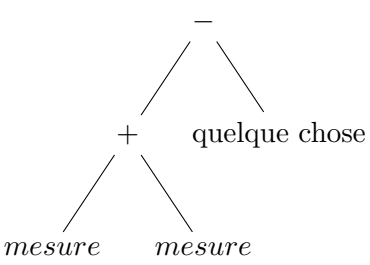
— pour le groupe 45l, là encore le TEA $+1/+1/+1$ semble résistant : les élèves de ce groupe ont, pour le TS7, appliqué le TEA $+1/+1/+2$, puis ont été capables de construire six instances différentes et éloignées, du TS40 au TS158. Cependant, lors de la phase de généralisation, la nouvelle construction du TS7 sera une mise en œuvre du TEA $+1/+1/+1$. Là encore, les élèves ne mobilisent pas les mêmes TEA pour des problèmes qui leur paraissent différents.

Les élèves de ces trois groupes sont donc capables de créer directement un script pour une certaine mesure, en associant correctement les valeurs des boucles B_1 et B_2 à la mesure. Seule le groupe 45b s'avère capable de faire cette association pour la boucle b_7, lors de la construction de certaines instances spécifiques.
En effet, si le problème de la généralisation du script a rapidement été posé par ces élèves, cela n'a pas été le cas pour le groupe 45l. Celui-ci a créé, directement et sans erreur, les instances du script traçant et dénombrant les TS40, TS58, TS108, TS148, TS158 et TS157. Ce groupe n'est pas à la recherche d'une généralisation, il cherche à résoudre le problème initial posé à la classe : quel est le plus grand TS que l'on peut construire avec 1400 hexagones [92]. Malgré le temps de construction de chaque script, ces élèves, qui ont construit ces instances en séance 4, n'ont pas semblé construire la nécessité d'un script générique. Et pour cause : chaque nouvelle instance créée prend entre trente et quarante secondes. Cela n'est pas assez coûteux pour envisager d'abandonner une solution qui fonctionne. La pertinence du problème initial est de nouveau discutable.
En séance 5, l'enseignante ayant rappelé que le problème de la séance était de généraliser le script, ces trois groupes ont recherché ou évoqué une instruction spécifique du langage θ qui permettrait de résoudre le problème :

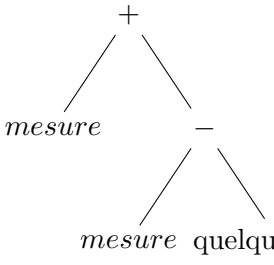
91. Ce groupe, en outre, n'a échangé qu'en présence de l'enseignante.
92. Le TS157 est construit avec 1401 hexagones.

— le groupe 46e s'est demandé si les écritures de nombres présentes dans les noms des blocs d'initialisation définissaient la mesure du TS tracé ;
— le groupe 45m a exprimé la relation entre b_1 et la valeur de la mesure entrée avec un bloc créé et nommé [nombre précédent-1] ;
— le groupe 45l a envisagé une solution plus radicale, un bloc nommé [faire le programme tout seul]. Si ce groupe sait créer n'importe quel programme traçant et dénombrant un TS, le faire faire par l'ordinateur est identifié comme un problème différent.

La nécessité d'employer [mesure] a pris des chemins différents pour ces trois groupes.
Le groupe 45b a utilisé une expression avec un nombre générique : $b_1 = 5 - 1$, en modifiant directement le script $TS(5)$ (étape 29), puis, immédiatement après, un signe générique vide [] est entré pour toutes les boucles, et testé (étape 30). Un peu plus tard, ils affecteront $b_1 = 13 - 1$ (étape 47) et, sans faire de test, ni étendre cette utilisation du NG aux autres boucles, ils transformeront l'expression en $b_1 = mesure - 1$ (étape 48). Ils étendront ensuite cette utilisation de $mesure$ avec $B_1 \leftarrow mesure - 1$, $B_2 \leftarrow mesure - 2$, et $b_7 \leftarrow mesure + 3$. Cette dernière expression, erronée, semble contradictoire avec la capacité des élèves à construire n'importe quelle instance spécifique, sans doute, comme on l'a déjà vu, en utilisant la relation intra-objectale $b_7 = 2 \times b_6$. Ici, il est possible que soit les faits construits lors de la construction du problème initial ne soient pas mobilisables lors de la construction du problème de généralisation, soit la relation trans-objectale induite par $\boxed{b_6} = mesure - 1$ obère l'utilisation d'une relation intra-objectale ($b_7 = 2 \times \boxed{b_6}$), suivie d'une substitution par la relation trans-objectale $2 \times \boxed{(mesure - 1)}$, comme le groupe 45e a été capable de le faire.
Le groupe 46e a envisagé et écarté l'utilisation de nombre-signes génériques (0, « parce que zéro c'est un multiple de tout ») ou nombres potentiellement génériques (10) puisque cela ne créée qu'une seule instance spécifique. Le SG vide [] a été utilisé, avec une simulation de ce qu'il devrait se passer lorsque le programme s'exécute, et ce groupe a envisagé l'utilisation de [mesure] à la place de ce SG, ayant identifié [mesure] comme contenant la valeur entrée. Cette solution a été écartée non parce que la relation boucle-mesure est inexacte, mais parce que l'EPGB a produit une erreur d'exécution : nouveau faux-fait. C'est sans doute la mobilisation des faits partagés en groupe classe qui les amènera à tester [mesure - 1], d'abord pour toutes les boucles, laissant augurer la possibilité d'une dénotation non univoque de [mesure - 1], utilisée à la fois pour dénoter la relation permettant de passer de la mesure à la valeur de la boucle b_1 et pour dénoter la relation permettant de passer de la boucle b_1 à la mesure. Le constat de ce caractère équivoque a été fait et a débouché sur l'utilisation de [mesure - 2] pour les boucles de la famille B_2.
Le groupe 45m, lui, se basera sur des relations exprimées avec un nombre-signe générique (0) : $B_1 = 0 - 1$, $B_2 = 0 - 2$ et $b_7 = 0 + 9$, en cohérence avec ce qu'ils pensent être un script valide, mais qui ne l'est pas ($\{1768/1767/1778\}$), autre faux-fait. Notons qu'ils ont bien construit l'expression de b_7 en se référant à la valeur de la mesure ($1778 = 1769 + 9$), ce qui laisse penser que ce SG est univoque. Ne fonctionnant pas, il sera remplacé par [mesure] ($\{mesure - 1/mesure - 2/mesure + 9\}$).
Ces deux derniers groupes n'ont pas réussi à identifier la relation entre b_7 et la mesure, à cause notamment du faux-fait concernant le script TS1769 (45m), et d'une recherche de relation uniquement basée sur les nombres par b_7 (46e), alors même que lors de la création des instances différentes ils se sont intéressés à l'organisation du tracé, à la logique de l'algorithme. Le groupe 45m semblait proche de faire une relation entre b_7 et ce qu'elle trace, mais, comme pour le groupement précédent, la représentation dans Snap ! d'une structure profonde a été compliquée. Ils ont ainsi voulu représenter la structure :



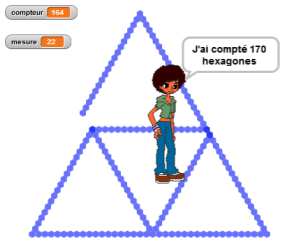
ou



par mesure × 0+0-0. Cette expression pourrait ainsi être interprétée comme « appliquer le schéma “0+0-0” à la mesure ». Ils arriveront cependant à explorer la possibilité d'emboiter des opérateurs, d'utiliser un opérateur comme opérande, mais on peut se demander quelle est la signification de ces opérateurs pour les élèves.

Le groupe 45h[93] est le seul groupe à avoir abouti à la formulation générique valide $b_7 =$ mesure + mesure − 2. Nous ne pouvons déterminer les faits construits par eux en séance 3 et 4, suite à la double utilisation de l'identifiant de groupe 45_h, combinée à une erreur d'affectation du programme de base de la séance 4. Cependant, il nous parait intéressant de compléter ce bilan par une analyse rapide de la séance 5 de ce groupe.

Lors de cette séance 5, après une période d'exploration des tracés des différents scripts ($TS(4)$, $TS(5)$, $TS(6)$, $TS(11)$ et $TS(20)$), le groupe 45h va s'attacher à corriger un des scripts fournis, celui qui devrait tracer un TS20 (sur le script dont la tâche prescrite est de tracer un TS17, TS19 ou TS21). Celui-ci est erroné, issu sans doute du doublement du $TS(11)$: on a $b_1 = 20$ et $b_6 = 18$, $B_2 = 18$ et $b_7 = 40$. Ce script sera exécuté deux fois, produisant la rétroaction :

$$TS(20)[] \rightsquigarrow \left(\text{  } \text{} , 170, 22 \right)$$

Ces élèves vont alors corriger partiellement le script, de façon valide, en effectuant $b_6 \leftarrow 19$ et $b_7 \leftarrow 38$. L'exécution du script donne le résultat représenté figure 3.98.

93. Plus précisément « un des groupes » 45h, voir Groupement 5 (p. 231).

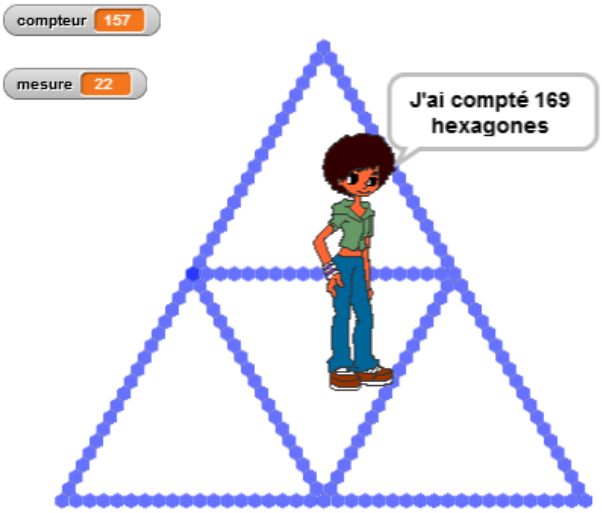
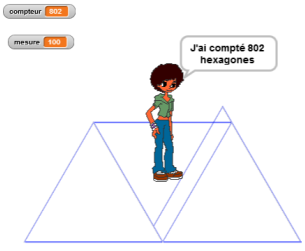


Figure 3.98 – Rétroaction du TS20, erreur b_1 (45h, S5a, 3'22)

Le tracé est ainsi erroné : pour un TS20, on devrait avoir $b_1 = 19$, et le nombre d'hexagones devrait être de 168. Cependant, les élèves semblent considérer le script comme étant valide, puisqu'il change de but, en s'attachant à construire le script générique. Ils commenceront par modifier le script $TS(4)$ en effectuant $b_1 \leftarrow 4 - 1$. Le 4 est ainsi un NG, et le script prend aussi figure de script à caractère générique. La rétroaction étant considérée comme valide, les élèves vont modifier toutes les boucles en fixant leur nombre de répétitions à (mesure − 1), comme l'a fait le groupe 46e. Le résultat est invalide (45h, transcription S5a, 19'43) et les élèves vont corriger en affectant (mesure − 2) à toutes les boucles B_2. La recherche de b_7 (45h, transcription S5a, 18'27-20'33) aboutira à $b_7^{14} =$ (mesure + 12). Un test avec une valeur entrée de 100 montrant le caractère erroné de cette solution les élèves vont immédiatement corriger $b_7^{100} =$ (mesure + 98) : ils sont ainsi capables de déterminer le nombre de répétitions de b_7 pour une certaine valeur de la mesure, mais la formulation met en tension le caractère immuable des valeurs 12 ou 98 et celui variable de la mesure entrée. Cependant, sans tester cette dernière correction, le groupe 45h va effectuer $b_7 \leftarrow$ (mesure + mesure), testé et immédiatement corrigé en $b_7 =$ (mesure + mesure − 2). Ainsi, ce groupe semble avoir construit le fait « pour trouver b_7 on prend la mesure et on lui ajoute le résultat de la mesure moins deux » uniquement en corrigeant un script erroné qu'ils n'ont pas conçu. Ce fait s'appuie sur les relations entre des nombres et non sur le tracé, puisqu'il est impossible de déterminer qu'il manque 98 hexagones à partir de la rétraction suivante :

$$TS(n)[10] \rightsquigarrow \left(\text{  } \text{} , 802, 100 \right)$$

Ils ne s'appuient donc que sur le fait $b_7^{14} = mesure + 12$, peut-être complété par $b_7^{20} = 38$, s'ils ont identifié qu'on a $b_1 = p - 1$ pour un TS de mesure p — fait qui a été partagé en groupe classe sur des exemples —. Ici, tout se passe comme si les élèves avaient travaillé à partir de caricatures (Orange, 2012, p. 102), et les faits construits sont, semble-t-il, mobilisés de façon au moins aussi pertinente que s'ils avaient entièrement conçu l'ensemble des scripts. C'est une façon alternative de faire construire des faits par les élèves, mais si ces caricatures sont anticipées, cela permet de définir à l'avance les faits possiblement construits : il faudrait étudier davantage cette possibilité.

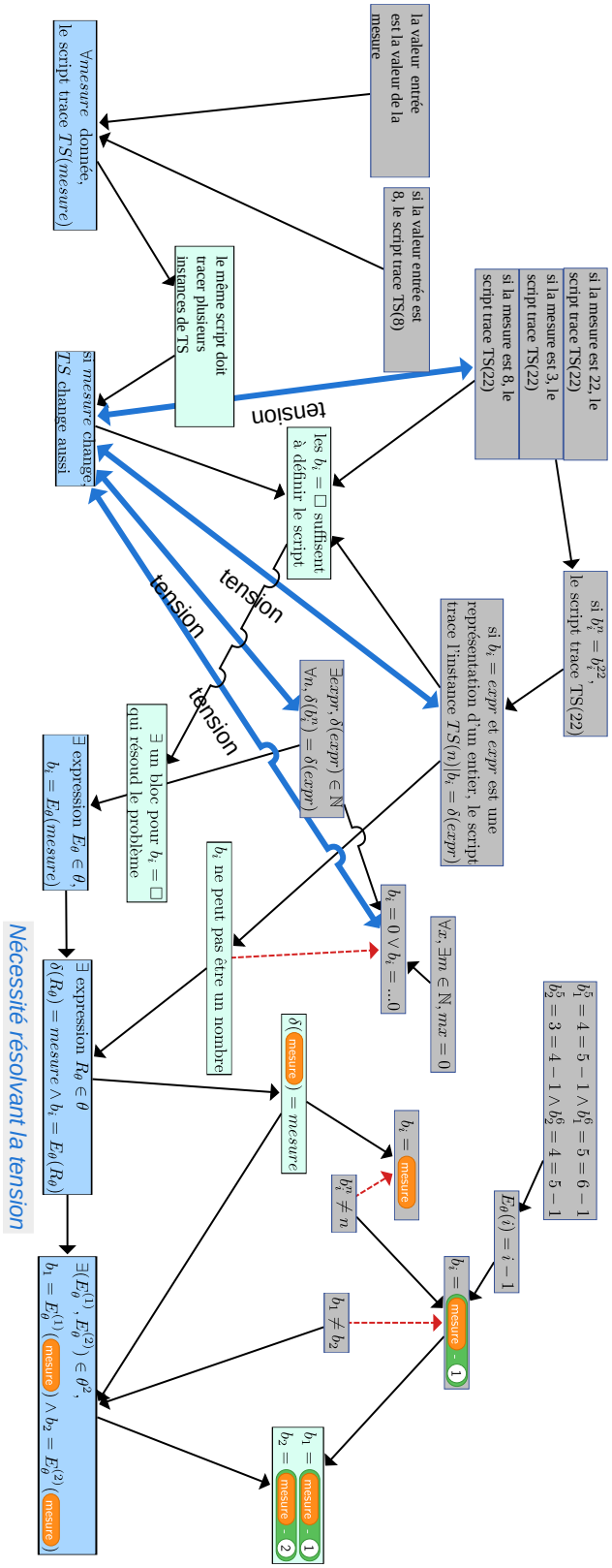


Figure 3.96 – Espace des faits-contraintes - 46e

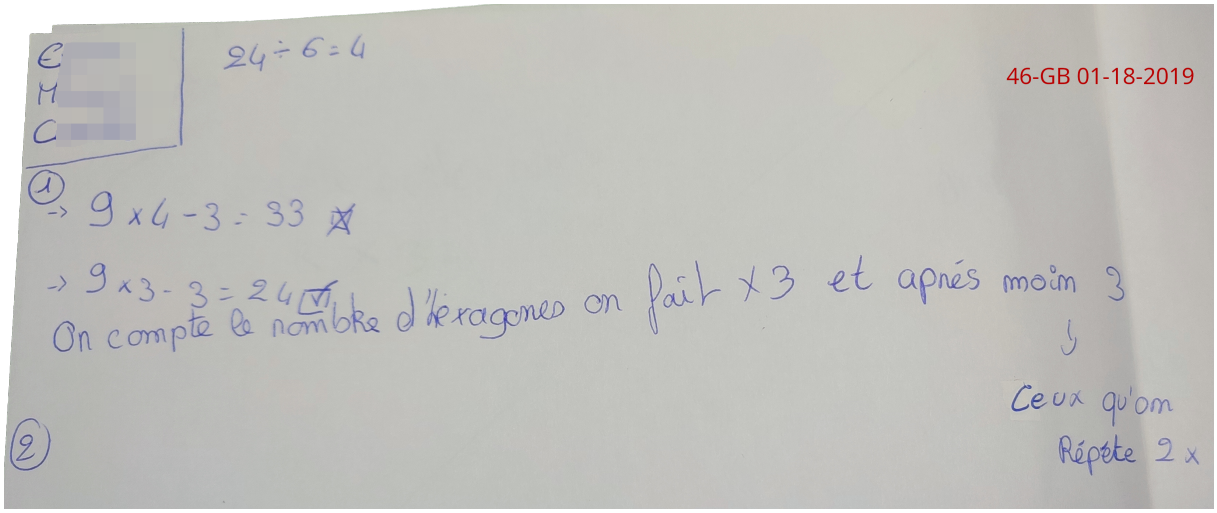


Figure 3.97 – Méthode trouvée en séance 6 (46e GB, S6)

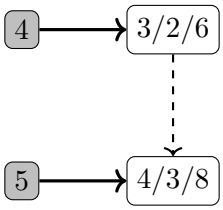
3.7 Conclusion du chapitre

Suite à cette analyse de différents groupes, la construction de la nécessité du paramètre dans notre situation amène à faire des remarques concernant les rétroactions et faits construits, la position du problème chez certains groupes, les TEA construits lors de la généralisation arithmétique, la question de la généralisation algébrique naïve et, finalement, comment les élèves en arrivent, ou pas, à généraliser algébriquement. Nous nous intéresserons aussi à une question qui semble traverser les groupes et les activités, que nous relions avec le concept d'entropie de l'information (Shannon et Weaver, 1964).

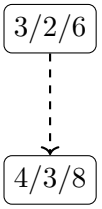
3.7.1 Rétroaction et faits premiers/seconds

En ce qui concerne la rétroaction et les faits premiers ou seconds construits, on peut constater en premier lieu, sur l'ensemble des groupes, que la rétroaction est généralement bien interprétée, et mise en relation avec les scripts. On trouve peu de cas manifestes d'essais-erreurs non organisés. La question de l'interprétation du chevauchement cependant se pose pour certains groupes, ou certaines instances. Malgré une explicitation faite en groupe classe, quelques groupes n'ont pas semblé considérer la couleur plus foncée d'un hexagone comme étant la représentation d'une information spécifique : il y a — au moins — un hexagone surnuméraire, et donc le tracé devrait être considéré comme invalide. La forme est bien celle d'un TS, mais il a parfois fallu l'intervention de l'enseignante ou d'un autre élève pour expliciter la non-validité du script, puisqu'il doit aussi dénombrer les hexagones, et pas seulement réaliser un tracé valide. En outre, pour des valeurs élevées de la mesure, cette rétroaction devient non signifiante : la dimension des hexagones tracés ne permet pas de différencier un hexagone surnuméraire d'un hexagone valide. Cet aspect n'est cependant pas fondamental, puisque les élèves devaient d'abord tracer des instances spécifiques pour lesquelles le chevauchement est bien visible.
Le chevauchement comme signe du caractère erroné du script doit être mis en relation avec l'objectif du script. Le script n'a pas pour objectif de tracer un TS bien formé, mais de dénombrer les hexagones formant un TS d'une certaine instance. Le tracé n'est qu'un moyen permettant le dénombrement. Ainsi, la valeur finale du compteur doit être prise en compte, du moins lors de la construction des premiers faits permettant d'identifier une régularité. Les élèves ont connaissance du nombre d'hexagones nécessaires pour tracer les premières instances, ces faits ayant été établis à partir de figures tracées (TS4 à TS7, ainsi que le TS tracé par les élèves). Ce critère de validation, nécessaire comme manifestation d'hexagones surnuméraires ou manquants, est parfois pris en compte. Pour certains groupes, il est reconnu comme une propriété essentielle différenciant les différentes instances, au même titre que la mesure.
La dimension relative du tracé par rapport à l'espace d'exécution a été peu considérée comme la manifestation d'une erreur. Les seules exceptions identifiées sont liées à une dimension relative « manifestement » non adaptée : lorsque le tracé du TS est « trop » petit, ou lorsqu'il sort « trop » de l'espace d'exécution. Le « trop », dans les deux cas, étant variables selon les groupes. Cette non prise en compte est néanmoins justifiée, puisque la dimension des hexagones n'impacte ni le tracé, ni le nombre d'hexagones. Son lien avec la mesure, puisque la dimension des hexagones est calculée en fonction de la valeur de la mesure, est cependant source de confusion : la valeur entrée, ou la mesure, désigne deux propriétés de deux objets différents : d'une part un nombre d'hexagones sur un côté d'un triangle de base formant un TS, d'autre part la dimension d'une figure hexagonale dessinée par le script. La tâche prescrite concernant le script censé tracer un TS20, TS21 ou TS22 renforce en outre cette dualité, puisque la valeur de la variable mesure est initialisée à 22, alors que le tracé pourrait être celui d'un TS de mesure différente. Nous avons

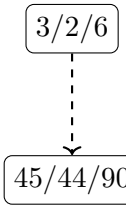
donc envisagé de modifier la situation pour clairement différencier la valeur de la mesure de la valeur du zoom. Cela introduirait un deuxième paramètre, mais on peut se demander si la manipulation (l'affectation) de ce deuxième paramètre a un réel intérêt pour la construction du concept, ou bien si elle est un risque d'amener les élèves à ne pas mettre de sens ou de caractère de nécessité pour la variable *mesure*. En effet, un paramètre « zoom » ne serait utilisé que comme stockage d'une certaine valeur, mais sans que sa représentation soit manipulée dans une expression, ce qui est une nécessité en algèbre.
Enfin, et ceci est lié à la non-prise en compte du « zoom », la valeur de la mesure ne semble pas être suffisamment prise en compte, ou factualisée, avant la généralisation. Les élèves construisent des instances nouvelles, mais sans forcément que la mesure soit associée au tracé ou au nombre d'hexagones. Or, c'est un élément essentiel pour établir des régularités et faire une généralisation algébrique (naïve ou symbolique). Le lien entre la valeur entrée et le nombre de répétitions des boucles est établi, mais le lien entre la mesure du TS et le nombre de répétitions des boucles l'est moins, et souvent grâce aux faits partagés en classe ($mesure = 5 \implies b_1 = 4$).
Cette absence d'association mesure-script est un obstacle à la généralisation. En effet, pour la mesure, il faudrait construire des faits que l'on peut représenter comme suit :



Cette représentation, qu'elle soit réellement représentée ou non, est essentielle pour établir les faits $b_1 = \delta(mesure) - 1$ ou $b_2 = \delta(mesure) - 2$. Cependant, si l'on représente de façon similaire les faits effectivement construits par de nombreux groupes, on aurait plutôt :



ou, si une instance lointaine est créée,



On peut ainsi établir des relations inter-objectales ou intra-objectales, mais sans relation avec la mesure. Sur quels indices peut-on se baser pour identifier des relations trans-objectales ? L'enseignante, à plusieurs reprises, insistera sur ce lien (voir 3.7.6, p. 458), mais ces faits ne prennent sens que lorsque les élèves en ont besoin. Comme nous l'avons évoqué, une possibilité serait de rendre davantage disponibles ces faits, en modifiant par exemple le message de fin de script en y incluant systématiquement la valeur de la mesure.

D'autre part, contrairement par exemple à une généralisation de motif « ordinaire » (non automatisée), il n'y pas d'utilisation des traces des faits construits d'une séance à l'autre : même si les élèves pourraient aller rechercher leurs scripts, un nouveau programme de base à chaque séance les en empêche en partie. Or, la disponibilité des faits constitués par les différents scripts valides créés est nécessaire pour identifier des régularités. On pourrait envisager, soit de ne faire qu'un seul programme de base pour cette phase de généralisation [94], soit incorporer au programme de base d'une séance les scripts construits dans la séance précédente (par le groupe ou par la classe).

3.7.2 Position du problème

Les problèmes que les élèves doivent poser dans cette situation découlent du problème de départ, trouver le plus grand TS possible constructible avec un certain nombre d'hexagones. Rappelons ici ces problèmes (2.1.6, p. 151).

1. Quel est le plus grand TS donné que l'on peut construire avec p hexagones ? C'est la question de départ, le problème principal qui rend nécessaire la résolution d'autres problèmes. Forme de la solution : une mesure n telle que le nombre d'hexagones nécessaires pour tracer le TS de mesure n soit inférieur ou égal à p et que le nombre d'hexagones nécessaires pour tracer le TS de mesure $n+1$ soit supérieur à p.
2. Comment trouver la mesure n la plus grande, telle que le TS de mesure n utilise moins de p hexagones ? Forme de la réponse : une méthode permettant de trouver la mesure.
3. Comment savoir combien d'hexagones sont nécessaires pour construire un TS de mesure n ? Forme de la solution : une méthode permettant de tracer et dénombrer un TS de mesure n quelconque.
4. Comment construire un script dans Snap ! traçant et dénombrant tout TS, sa mesure n étant donnée ? Forme de la solution : un script traçant et dénombrant un TS dont la mesure n est entrée par l'utilisateur lors de l'exécution.
5. Comment modifier un script traçant et dénombrant un certain TS de mesure n pour qu'il trace un TS de mesure m ? Forme de la solution : une méthode permettant de construire un script traçant et dénombrant un TS d'une mesure donnée m.
6. Comment modifier un script traçant et dénombrant un certain TS de mesure n pour qu'il trace un TS de mesure $n+1$? Forme de la solution : une méthode permettant transformer un script traçant un TS de mesure n en un script traçant un TS de mesure $n+1$.
7. Comment modifier un script traçant et dénombrant une certaine instance d'un TS pour qu'il trace et dénombre une autre instance ? Forme de la solution : une méthode permettant de trouver les changements à apporter au script.
8. Quels sont les éléments déterminant le tracé du TS dans le script traçant et dénombrant un certain TS ? Forme de la solution : les instructions du script qu'il faut modifier / les instructions du script qu'il faut ajouter.
9. Comment modifier les valeurs du nombre d'itérations des boucles d'un script traçant et dénombrant un certain TS pour qu'il trace et dénombre un autre TS ? Forme de la solution : des expressions du langage précisant le nombre d'itérations des boucles.

94. La différence entre les deux variantes étant la suppression de la pause systématique entre chaque tracé d'hexagone, afin d'accélérer l'exécution. Il serait tout à fait possible de supprimer cette pause, puisqu'un mode pas-à-pas (ou ralenti) est disponible dans Snap !

10. Comment modifier un script traçant un certain TS pour qu'il dénombre les hexagones nécessaires au tracé de ce TS ? Forme de la solution : les instructions du script qu'il faut modifier / les instructions du script qu'il faut ajouter.

Le dernier problème, qui ne concerne normalement que la séance 2 et non la phase de généralisation étudiée ici, a été néanmoins posé épisodiquement par certains groupes : 45b (épisode 43), 45f (épisode 1), 45n (épisodes 9, 12, 17, 18), 46g (épisodes 1 à 4) et 46k (épisode 29).
Certains élèves semblent par ailleurs rester sur le problème 1, cherchant à construire une instance de TS qui résolve le problème. Dans ce cas, ils ne posent pas le problème de la généralisation du script (problème 4), mais naviguent entre les problèmes 5 à 7 et le problème 1. Comme on l'a vu, les faits issus de la construction de ce problème 1 ne semblent pas ou peu mobilisables pour la construction du problème 4. Cela n'empêche pas les élèves, une fois le problème 3 posé, d'avancer dans sa construction, mais cela nécessite la construction de nouveaux faits, ou la reconstruction de faits anciens (qui ont été défactualisés, qui ont perdu leur statut de fait en changeant de position du problème). De plus, dans ce cas, la position du problème 4 nécessite une nouvelle intervention extérieure (de l'enseignante ou d'autres élèves).
D'autre part, certains élèves ont posé et trouvé une solution concernant le problème 2 : il faut construire une certaine instance d'une certaine mesure, et si le nombre d'hexagones de cette instance est inférieur à p, il faut construire une instance plus grande. Cela induit la création d'instances multiples, éventuellement successives, mais n'est pas suffisant pour poser le problème de la généralisation, qui là encore nécessite une intervention extérieure.
Enfin, certains élèves posent le problème non pas de construire et dénombrer des instances et comparer avec la valeur donnée dans le problème 1, mais simplement de construire diverses instances successives. Dans ce cas, la mesure et le nombre d'hexagones dénombrés (la valeur de la variable *compteur*) ne sont pas nécessaires : il suffit d'obtenir des tracés valides. Les faits construits dans le cadre de ce problème sont peu mobilisables pour la généralisation. De nouveau, ces élèves nécessitent une intervention extérieure pour poser le problème de la généralisation.
Ainsi, nous pensions que la situation, en elle-même, permettrait aux élèves de poser et construire l'ensemble des problèmes évoqués ci-dessus, mais cela n'a pas été le cas. Néanmoins, à part les élèves du groupement 1, les autres élèves ont fini par poser le problème de la généralisation du script, notamment grâce aux interventions de l'enseignante auprès des différents groupes ou lors de la plénière. Pourrait-on modifier la situation pour que la nécessité de la généralisation soit réellement construite ? Par exemple, factualiser la démarche du problème 2 permettrait d'identifier les contraintes : créer de nombreux scripts. Ces contraintes ont été partagées avec le groupe classe, mais la plupart des élèves ne semblent pas les avoir mobilisées comme des faits. Se confronter au coût de la construction des différentes instances, évoquée par l'enseignante à propos du groupe ayant construit les instances du TS4 au TS17 (46d) pourrait aider à construire cette nécessité de la généralisation. On pourrait ainsi ajouter des valeurs au problème initial : par exemple, trouver le plus grand TS possible pour 188 hexagones forcerait les élèves à aller jusqu'à l'instance TS22, ou 400 hexagones pour une instance TS46. Un travail de caricature (Orange, 2012, p. 102) sur les méthodes de construction — instances successives, saut de quelques instances, chercher une certaine instance — pourrait amener la position du problème de la généralisation du script. Cependant, cela impose une contrainte en termes de durée et de nombre de séances. Faut-il laisser le temps aux élèves de construire la nécessité permettant de poser le problème de la généralisation du script, ou faut-il didactiser ce problème en le posant avec les élèves, quitte à accompagner cette position pour ceux qui le nécessitent ?

Une autre solution serait de ne pas passer par le tracé informatisé, mais de chercher à formaliser le problème 2. Il s'agirait alors d'emmener les élèves vers la création complète du script générique résolvant la famille d'instances du problème initial (algorithme 3.7.1, p. 448, en pseudo-langage). Cependant, si cela nécessite bien de trouver une expression de θ manipulant une représentation R_θ de la mesure afin de déterminer le nombre d'hexagones d'un TS donné, nous y voyons plusieurs difficultés. D'une part, cela nécessite des compétences algorithmiques beaucoup plus élaborées, mêlant condition, itération, variable dans un rôle de compteur et variable dans un rôle de paramètre, ou encore identification de la solution à partir de la valeur du compteur. D'autre part, on supprime ici l'intérêt de la rétroaction pour construire des faits. En revanche, ce type de problème pourrait être un prolongement intéressant de la situation.

Résultat : Mesure du TS le plus grand pour une certaine entrée
pas de contrôle de l'entrée ($n > 5$);
`lire` (n);
$x \leftarrow 2$;
tant que $9x - 12 \leq n$ **faire**
  $x \leftarrow x + 1$;
fin
`dire` ("le plus grand TS possible avec "+n+"hexagones est le TS"+$x - 1$);

Algorithme 3.7.1 : Résolution du problème principal

3.7.3 TEA

Concernant plus spécifiquement la construction d'instances successives et les TEA organisant cette activité, on a pu relever quelques éléments.
De façon attendue, à l'exception des groupes 45f et 46g, la mobilisation majoritaire du TEA $+1/+1/+1$ suivie du TEA valide $+1/+1/+2$ a bien été constatée. Cependant, la réutilisation d'un TEA dont une production a été invalidée, soit la nécessité pour certains groupes de construire plusieurs faits invalidants avant de ne plus mobiliser un certain TEA est davantage questionnante. Comme cela traverse les groupes et semble se conjuguer avec certains faits rendant plus difficile la généralisation, nous traiterons ce point de façon plus détaillée ultérieurement (3.7.6, p. 453).
On peut noter que la rapidité d'apparition du TEA valide ou sa stabilité ne semblent pas être corrélés à la réussite de l'activité et la construction du paramètre. Il faut cependant que les élèves aient construit suffisamment de faits, ce qui n'est pas le cas de tous les groupes. Certains groupes ont construit peu de faits, mais les faits partagés semblent rarement suffisants pour être mobilisables. Il semblerait qu'une condition de mobilisation de ces faits partagés soit qu'ils répondent au problème posé par les élèves. Ainsi, si les élèves ont posé le problème de la généralisation, les faits partagés en séance 5 sont directement mobilisables. Sinon, ils ne deviendront mobilisables que lors de la position du problème, et à la condition qu'ils soient encore disponibles : les relations mesure-boucle écrites au tableau ne sont pas des faits pour certains groupes au moment où ils sont partagés, mais ils peuvent le devenir lorsqu'ils deviennent pertinents, c'est-à-dire en rapport avec le problème des élèves.

De même, l'instabilité des TEA semble, elle aussi, liée au problème, puisque les faits construits dans un problème (créer des instances) ne sont pas/plus mobilisés dans un autre (construire le script générique) : un TEA « invalidé » dans un problème ne l'est donc pas forcément dans un autre. Les faits et TEA dépendent du problème posé par les élèves, mais tous les élèves ne posent pas le même problème au même moment : comment dès lors partager des faits avec l'ensemble de la classe ? Faut-il que tous les élèves soient rendus à un moment donné à poser le même problème, ou faut-il mettre en place des conditions permettant la mobilisation ultérieure de faits partagés ?

3.7.4 Généralisation algébrique naïve

Concernant la généralisation algébrique naïve, soit la construction directe d'une instance lointaine, sans expliciter cette construction, on a pu constater qu'elle était résistante, notamment en raison des difficultés concernant la boucle b_7. Il est probable que cette difficulté soit issue, du moins en partie, du faible jeu de cadres mis en œuvre. Pour les élèves, la construction d'une certaine instance consistait essentiellement en la recherche de relations entre des nombres, et non d'une explication permettant de justifier pourquoi telle boucle trace tant d'hexagones. Les faits algorithmiques concernant le tracé ont été peu construits par les élèves, et non partagés avec la classe. Là encore, des caricatures, en fin séance 3 ou début séance 4, donnant à voir la structure du tracé, la logique de l'algorithme, aurait pu aider à lever ces difficultés. Une version du script changeant la couleur des hexagones pour chaque boucle a été envisagée. Elle aurait pu faciliter l'identification des actions des boucles, mais elle compliquait la visualisation des hexagones surnuméraires, et a donc été écartée. En revanche, amener les élèves à faire ces identifications parait nécessaire.
D'autre part, une généralisation algébrique naïve (GAN) ne suffit pas à amener les élèves sur la voie de la généralisation algébrique symbolique. Dans le tableau 3.8 (p. 450), on peut ainsi constater que le groupe 45b, s'il est capable de construire n'importe quelle instance de TS, exprime la relation permettant de déterminer b_7 sous la forme d'une addition de quelque chose à la mesure. De même, les groupes 46g et 46i ont manifesté une capacité implicite, ou explicite (groupe 46g) à déterminer b_7, mais ces faits, de nouveau semblent défactualisés. Ici encore, une hypothèse est que le fait soit très fortement lié au problème : trouver une méthode permettant de déterminer b_7 lors de la construction d'instances successives, ce n'est pas du même ordre que trouver une méthode permettant de déterminer b_7 lors de la construction du script générique. Enfin, la généralisation algébrique naïve ne semble pas être nécessaire : le groupe 45e formalise $b_7^6 = (6-1) \times 2$ sans passer par un telle généralisation. Ils ont sans doute identifié cette relation par une compréhension du script et de ce que chaque boucle trace, grâce à leur méthode exploratoire consistant à diviser le script et l'étudier par partie. Cette compréhension semble davantage nécessaire.

groupe	grpt	mesure	mesure-1	NG-1	SG	SG-1	NSG	NSG-1	b7 GAN	b7=mesure+...
45h	-	✓	✓	4-1					mesure+mesure-2	✓
45b	5	✓	✓	5-1					TS11 ;TS40 ;TS58 etc, mais devient mesure+	✓
45j	4	✓	✓	11-1			0			✓
45l	4	✓	✓					0-1		✓
45m	5	✓	✓							✓
46e	5	✓	✓		✓		0			✓
46m	4	✓	✓	8-1	vrai ; []x[]	[]-1			TS22 ;TS85, implicite (b7=8+ ? devient b7=16-2)	
45e	4	✓		6-1					(6-1)x2 (sans GAN mais diviser pour régner)	
46k	4	✓		5-1 ; 1400-1	[]-[]		0			
45a	2						1+1			
45f	2				✓					
45k	3			10-1 ; 45-1 etc.	✓	[]-1	1x1	0-1		
46b	3			5-1	✓		2x0 ;2x1			
46c	3				[]-[]		0-[]			
46f	3			20-1	[] ; []-[]					
46g	3				[] ; []+[]		1+[]		TS22 ;TS100 oralisé, puis perdu	
46i	3			15-1					TS22, implicite (30-1 pour 15-1, mais fini 31-1 pour 16-1)	
46l	3			5-1	[]-[] ; 12-[]				TS9 mais pas TS27	

Tableau 3.8 – Occurrences de différents types de généralisations suivant les groupes

3.7.5 Généralisation algébrique

En ce qui concerne la généralisation algébrique symbolique, nous avons noté divers points. En premier lieu, on constate :

— une faible utilisation de mesure (8 groupes),

— 6 groupes expriment correctement b_1 et b_2 en fonction de la mesure,

— aucun groupe n'exprime de façon valide b_7. Le groupe 45e en était proche, et aurait sans doute réussi sans le faux-fait auquel ils ont été confrontés.

De façon surprenante, avant même d'envisager une relation avec la mesure, plusieurs groupes sont à la recherche d'un bloc, d'un « truc », d'une expression, d'un opérateur qui permettrait à l'ordinateur de trouver la solution. L'hypothèse des élèves semble être qu'il existe un moyen (une expression, une instruction) de θ pour trouver les b_i. Cependant, cela n'implique pas de suite qu'il existe une expression R_θ dénotant la mesure. Comme nous l'avons déjà signalé, la quasi-absence de jeu de cadres et d'explication rendant visible les relations trans-objectales (donc basées sur la mesure), ainsi que l'absence de liens rendus visibles entre le script et la mesure, semblent expliquer en partie cette non construction : pourquoi chercher une expression d'un objet dont on ne perçoit pas la nécessité ?
D'autre part, le nombre générique (NG) est assez présent, comme attendu (10 groupes). Cependant, son utilisation n'amène pas forcément à construire la nécessité de R_θ (en tout cas pas encore) : 7 groupes font [NG-1] sans aboutir à $mesure - 1$, y compris le groupe 45k qui simule de nombreuses instances. En se référant aux constats faits plus haut, on peut se demander si cette non construction est issue du manque de visibilité du lien entre boucles et mesure, au profit du lien entre boucle et valeur entrée. Une autre hypothèse serait la seule mobilisation de faits partagés en début de séance 5 : les élèves testent « 5-1 », car ils ont compris que l'enseignante proposait implicitement l'utilisation d'opérateurs, et les faits $mesure = 5 \implies b_1 = 5 - 1$ et $mesure = 6 \implies b_1 = 6 - 1$ ont été partagés. Les faits partagés ont été mobilisés, mais ont-ils pour autant été construits, ont-ils fait sens pour les élèves ?
Le PNG, ce nombre ayant perdu sa généricité, était quant à lui moins attendu. Sa présence assez importante peut-être issue de relations intra-objectales construites entre la famille B_2 et la famille B_1. L'importance accordée à la relation plutôt qu'à la propriété caractéristique du TS (la mesure) est sans doute aussi impliquée : les élèves déterminent b_1 en faisant « moins un », en se référant à la valeur entrée, puis ils déterminent b_2 en faisant de nouveau « moins un », cette fois en se référant à la boucle b_1. Nous n'avons pas noté la présence de ces PNG dans les articles consultés concernant la généralisation de motifs « manuels ». Une raison possible de cette absence est envisageable, en lien une fois de plus avec l'absence de jeux de cadres. En effet, les explications des généralisations de motifs classiques s'appuient sur les caractéristiques des motifs et leur schématisation, tandis que dans notre cas les explications sont basées sur la procédure à faire-faire, donc sur les boucles et leurs relations, et plus spécifiquement sur les nombres d'itérations de ces boucles.
Si ces PNG n'apparaissent pas dans d'autres situations, on peut se demander si leur mobilisation est un obstacle gênant, ou un obstacle utile ? En effet, c'est aussi ce qui contribue à créer la nécessité de R_θ, d'une représentation univoque de la mesure. C'est ce qui distingue cette situation d'une généralisation de motifs non informatisée, et est un des intérêts de l'interprétation de la procédure par une machine.
Enfin, les signes génériques ou nombres-signe génériques (NG/NSG) sont présents dans la moitié des groupes qui ont abouti à l'utilisation de la variable *mesure* (4 groupes sur 8, voir tableau 3.8, p. 450). On a ainsi pu observer des enchainements de représentations tels que $0 - 1 \rightarrow mesure - 1$ ou $8 - 1 \rightarrow [] - 1 \rightarrow mesure - 1$. De tels enchainements sans aller jusqu'à remplacer les signes

par la variable *mesure*, sont aussi visibles par exemple pour le groupe 45k, qui a expérimenté les expressions [] − 1 et 0 − 1. On peut imaginer que ce groupe était en cours de construction de la nécessité de R_θ. Ces signes sont certainement des marques-place, des indications dans le texte d'éléments à changer, mais ils ne sont pas forcément univoques. Ainsi, dans un formulaire, on peut trouver une expression telle que « nom : [ ] prénom : [ ] ». Ces deux marques-places seront substitués par des valeurs différentes. Ainsi, si le paramètre implique l'existence d'une désignation univoque, un signe ne l'est pas forcément. L'avantage de la situation informatisée, c'est que cette désignation univoque est rendue nécessaire, ce qui amène les élèves à devoir dépasser ce qui est équivoque : une élève du groupe 46e explique ainsi « ça fait tout le temps la mesure » (46e, transcription S5, 5'26).

Si l'on s'intéresse à l'utilisation de mesure, peut-on considérer que cette utilisation est la manifestation d'une nécessité construite, ou bien est-ce une injonction implicite de la plénière du début de séance 5, lors de laquelle il a été rappelé aux élèves où était stockée la valeur entrée par l'utilisateur ? L'utilisation de mesure ne suffit pas, selon nous, pour pouvoir affirmer que les élèves ont construit la nécessité du paramètre : il peut tout aussi s'agir d'une exploration des blocs possibles, déjà illustrée par des instructions dont le sens que leur donnerai un expert n'est certainement pas le sens donné par les élèves (par exemple répéter 5 est un(e) nombre ?). En revanche, on peut affirmer que l'utilisation de ce mot de θ n'est pas issue d'une injonction, mais plutôt d'une mobilisation, lorsque cela est devenu nécessaire, des faits partagés en classe. En effet, l'utilisation de mesure ne succède pas immédiatement à la fin de plénière. C'est le cas pour le groupe 46e, mais ces élèves avaient déjà envisagé l'utilisation de cette expression en séance 4, et les échanges nous montrent que la nécessité était en cours de construction.

Concernant, pour finir, les expressions des boucles, nous nous demandons si les expressions *mesure*−1 et *mesure*−2 peuvent être considérées comme des expressions algébriques, manipulant le paramètre comme si sa valeur était connue. On peut penser que ces expressions, pour les élèves, peuvent être considérées comme des procédures : « on fait moins un » et « on fait moins et encore moins un », ce qui a été exprimé en classe par « un chiffre de moins » ou « deux chiffres en dessous ». On se situe ici dans le REX du nombre (Peano) plutôt que dans celui des expressions arithmétiques ou algébriques. Ainsi, les relations permettant de déterminer b_1 et b_2 sont potentiellement insuffisantes pour créer la nécessité d'une expression manipulant une représentation de la mesure. La boucle $b7$, ou toute relation s'écrivant sous la forme $an + b$, avec $a > 1$ et $b \neq 0$, parait plus susceptible de créer cette nécessité.

Cette expression du nombre d'itérations de la boucle b_7 peut être déterminée de deux façons : soit en considérant une relation intra-objectale liée aux nombres ; soit en considérant une relation trans-objectale liée au tracé. Si b_7 est vue en intra-objectale ($b_7^p = 2 \times b_6^p$), cela permet de construire une instance directe. Mais est-ce que cela complique, ou non, la généralisation formelle ? Il s'agit dans ce cas de substituer un opérande par une fonction (un opérateur) pour passer de $\delta(b_6) \times 2$ et $b_6 = 7 - 1$ (par exemple, ici avec un NG) à $7 - 1 \times 2$. Cette substitution d'un signe par une expression nous parait nécessiter une pensée algébrique formelle bien installée, sans doute pas immédiatement accessible pour tous les élèves. Construire b_7 en considérant sa signification, ce que trace cette boucle relativement au nombre d'hexagones d'un côté du triangle de base, est sans doute plus accessible : il s'agit ici d'établir que b_7 trace un grand côté ($2 \times mesure$), moins deux hexagones qui seraient en trop ($b_7 = 2 \times mesure - 2$), ou encore de concevoir b_7 comme traçant deux fois un côté du triangle de base sans une de ces extrémités ($mesure - 1$), ce qui aboutirait à la même expression que celle issue d'une relation intra-objectale. Une même expression, mais deux sens différents, et deux constructions différentes. Ainsi, de nouveau, il nous semble nécessaire de favoriser les jeux de cadres.

3.7.6 Faits, TEA et généralisation : une question d'entropie ?

Comme on l'a évoqué plus haut, deux phénomènes semblent traverser une bonne partie des groupes : l'un concerne les TEA et leur mobilisation malgré une invalidation, l'autre les difficultés de généralisation suite à l'utilisation de PNG. Il nous semble que ces deux phénomènes peuvent être décrits de façon similaire.
Ainsi, on a pu constater en suivant ces groupes qu'un TEA pouvait être de nouveau mis en œuvre même après avoir été invalidé. On a ainsi à plusieurs reprises l'enchaînement suivant :

1. application du TEA $+1/+1/+1$ pour une instance ;
2. constat de l'invalidité du script ;
3. correction du script ;
4. application du TEA $+1/+1/+1$ pour une autre instance ;

Parfois, un TEA préalablement invalidé par la rétroaction est remobilisé non pas immédiatement après l'invalidation, mais ultérieurement, notamment, comme on l'a vu, lorsqu'un autre problème est traité. Ainsi, si l'on ne regarde que le TEA $+1/+1/+1$ ou sa variante plus générique $+a/+a/+a$, pour la classe 45 :

— le groupe 45a applique ce TEA deux fois de suite pour une même instance (mais sur deux séances) ;

— le groupe 45b l'applique sur deux instances consécutives en séance 3, puis de nouveau, pour une instance pourtant déjà créée, en séance 5 ;

— le groupe 45d l'applique deux fois de suite sur une même instance (sur deux séances), puis sur l'instance suivante. Ce TEA sera de nouveau visible en séance 5, sur une instance déjà créée ;

— le groupe 45j applique à plusieurs reprises, non consécutives et sur les trois séances, le TEA $+a/+a/+a$;

— le groupe 45k applique le TEA $+1/+1/+1$ une fois en début de séance 4, et une nouvelle fois sur la même instance en début de séance 5 ;

— ce même TEA est mis en œvre par le groupe 45m en séance 3 et en début de séance 4, suivi d'une instanciation de $+a/+a/+a$.

Pour la classe 46 :

— 46a applique le TEA $+1/+1/+1$ sur deux instances consécutives ;

— 46b fait de même sur sept instances, mais sans test, donc sans invalidation. Une fois ces tests faits et des corrections apportées, le TEA sera de nouveau mobilisé en séance 5 ;

— 46e, de même que 46f, l'appliquent sur deux instances consécutives. Après avoir appliqué le TEA valide $+1/+1/+2$, ces deux groupes mettront en œuvre la variante $+a/+a/+a$ (et $*a/*a/*a$ pour le groupe 46f) ;

— 46k mobilisera ces deux TEA, sur plusieurs instances, mais de façon non stabilisée, avec des variations suivant les boucles ;

— enfin, 46m appliquera lui aussi le TEA $+1/+1/+1$ sur deux instances consécutives, et son extension $*a/*a/*a$ en séance 5.

Ainsi, un fait invalidant un script n'invalide pas le TEA qui organise l'activité des élèves pour la création de ce script. Tout se passe comme si les faits modifiaient la probabilité que ce TEA soit vrai. Pour un même fait et un même TEA, cette modification est variable.

Or, un TEA est un énoncé tenu pour vrai dont l'application, dans notre situation, est elle-même constituée d'énoncés : le TEA $+1/+1/+1$ appliqué au script $\{4/3/6\}$ produit l'énoncé « $\{5/4/7\}$ est vrai" ». C'est ce dernier énoncé qui se trouve invalidé par la rétroaction. On peut considérer, dans notre situation, qu'un TEA est un système de production d'énoncés réputés vrais.
Ainsi, un fait, porteur d'informations, viendrait modifier l'incertitude des systèmes de production d'énoncés.
Cela nous incite à faire un parallèle avec la théorie mathématique de la communication de Shannon (1948), plus particulièrement sur ses définitions de l'entropie d'une information.

Théorie mathématique de la communication et entropie de l'information

Pour Shannon et Weaver, une information réduit l'incertitude. Comme ils le signalent, ce rapprochement information et incertitude peut être perturbant :

> The concept of information developed in this theory at first seems disappointing and bizarre - disappointing because it has nothing to do with meaning, and bizarre because it deals not with a single message but rather with the statistical character of a whole ensemble of messages, bizarre also because in these statistical terms the two words information and uncertainty find themselves to be partners. (Shannon et Weaver, 1964, p. 27)

L'information est ainsi vue de façon statistique, au sein d'un ensemble de messages et sans se préoccuper, à ce niveau, des questions de signification. Cela peu paraître surprenant de ne pas parler de signification alors qu'on parle d'apprentissage, nous y reviendrons.
Cette réduction de l'incertitude est quantifiable : ce que nous appelons maintenant « 1 shannon » est « un bit d'information », c'est-à-dire une diminution par deux de l'incertitude. Même si nous n'allons pas envisager de quantifier de façon précise la quantité d'information portée par les faits construits, nous allons prendre un exemple classique permettant d'illustrer cette quantification. Si je demande à, disons, une station météo, « le soleil va-t-il se lever demain ? » — ce qui, pour moi, est certain —, la réponse envoyée ne va pas modifier l'incertitude : la quantité d'information est nulle. Supposons que je sois dans un certain endroit depuis un certain temps, et que chaque jour, soit il pleut, soit il ne pleut pas, et ce un jour sur deux. La réponse à la question « va-t-il pleuvoir demain », quelle qu'elle soit, va donc me permettre de passer de deux possibilités équiprobables à une seule, il s'agit d'une division par deux de l'incertitude, donc 1 shannon. Par contre, s'il pleut un jour sur quatre — la probabilité d'avoir de la pluie est donc de 0,25 —, la quantité d'information portée par le message dépendra de la réponse : si la réponse est « il ne va pas pleuvoir » (ce qui est très probable), cela me donne moins d'information que si la réponse est « il va pleuvoir » (ce qui est moins probable). Puisque 1 bit d'information est une division par deux de l'incertitude, si la réponse est « il va pleuvoir », cela donnera $-log_2(0.25) = 2$ bits d'informations, tandis que si la réponse est « il ne va pas pleuvoir », le message portera $log_2(0.75) \approx 0.41$ bits d'informations. En moyenne, la station m'enverra donc $0.75 \times 0.41 + 0.25 \times 2 \approx 0.81$ bits d'informations : c'est l'entropie d'information de ce système, ou la mesure de l'incertitude des évènements de ce système.
Plus généralement l'entropie de Shannon d'une distribution p d'événements est :

$$H(p) = -\sum_{i=1}^{n} p_i log_2(p_i)$$

Entropie de l'information et problèmes

Prenons l'exemple que Pólya (2008) donne concernant la conjecture de Goldbach « Tout nombre entier pair supérieur à 3 peut s'écrire comme la somme de deux nombres premiers » (Hersant, 2021). Dans un premier temps, Polya, s'adressant au lecteur, imagine que celui-ci rencontre « par hasard » trois relations : $3+7=10$, $3+17=20$, $13+17=30$. Ces faits constituent ce que Polya nomme des « points de contact suggestifs », qui font que le lecteur remarque « une certaine ressemblance entre elles » : en l'occurrence, qu'il s'agit de sommes de nombres premiers. Le lecteur s'interroge alors : s'il est certain que la somme de deux nombres premiers impairs est un nombre pair, un nombre pair est-il toujours la somme de deux nombres premiers impairs ? C'est une étape de généralisation : « nombre pair=nombre premier + nombre premier ». Puis, le lecteur est supposé « éprouver » cette hypothèse, en construisant de nouveaux faits qui sont des « points de contact de confirmation ».
On peut reprendre cet exemple en considérant que les relations sont des faits, des énoncés tenus pour vrais (ibid.) : « $3+7=10$ » ou « $3+7=10$ est vrai », ou encore « 3 est un nombre premier ». Les énoncés correspondant à la conjecture peuvent être produits par un « système qui produit une séquence de symboles » [95] S. En situation d'apprentissage visant à faire émettre cette conjecture par les élèves, ce système producteur d'énoncés est le système que l'on souhaite faire construire par l'élève, un système cible. Cet élève, en débutant l'activité, peut envisager divers systèmes produisant des énoncés.

- S_1 : un nombre est la somme de plusieurs nombres ;
- S_2 : un nombre pair est la somme de deux nombres.
- S_3 : un nombre pair est la somme d'un nombre impair et d'un nombre impair ;
- S_4 : un nombre pair est la somme d'un nombre premier impair et d'un nombre impair ;
- S_5 : un nombre pair est la somme de deux nombres premiers
- ... ;

Ces différents systèmes permettraient de produire les « séquences de symboles » constituées par les énoncés. Mais d'autres systèmes liés aux écritures arithmétiques sont bien entendu probables :

- S_6 : `[0..9]*3 + [0..9]*7=[0..9]*0` (soit une expression régulière reconnaissant par exemple $13+27=40$ [96])
- S_7 : un multiple de dix est la somme de deux nombres impairs ;
- ...

D'autres systèmes sont possibles, produisant par exemple des énoncés tels que « 16=17-1 » ou « baba is you » [97]. En dehors de tout problème — situation totalement fictive —, on pourrait considérer qu'un individu peut produire un ensemble infini d'énoncés, avec une probabilité identique pour chacun de ces énoncés. Cependant, dans un REX donné (ou pour un problème donné), la probabilité de produire tel ou tel énoncé change : si les élèves se posent un problème relevant de l'arithmétique des nombres premiers (en s'appuyant sur le REX adéquat), la probabilité qu'ils produisent un énoncé tel que « baba is you » est plus faible que « 16=17-1 » ou même que « 16=17- ». Ces différents systèmes $S_1, S_2 ... S_p$ n'ont ainsi pas la même entropie, que Shannon note H : $H(S_1) \neq H(S_2)$. En outre les énoncés, ou les messages, produits (par le milieu, par

95. « A system which produces a sequence of symbols (which may, of course, be letters or musical notes, say, rather than words). »
96. Mais aussi $13+27=60$.
97. https://hempuli.com/baba/, jeu fort intéressant de logique formelle.

l'enseignant, par l'élève...) vont modifier l'entropie de ces systèmes : l'énoncé « $3 + 7 = 10$ » diminue l'entropie de S_2 par exemple, mais aussi de S_3. L'entropie de S_1 et du système produisant « baba is you » augmente : cet énoncé, ainsi que « 10=7+2+1 », sont moins probables. De plus, le REX dans lequel se situent les élèves, qui est lié au problème qu'ils se posent, a une influence sur la modification de l'entropie :

— si les élèves sont dans un REX expliquant les propriétés des nombres premiers, l'entropie de S_3 est plus faible que celle de S_2, et de S_6 ;

— si les élèves sont dans un REX numérique basé sur l'écriture des nombres, S_6 a une plus faible entropie que S_2 ou S_3.

Ainsi, les faits supplémentaires « 3+17=20 » et « 13+17=30 » modifient, eux aussi, l'entropie de tous les systèmes, et notamment diminuent S_3 et S_6. Mais cette modification, dépend du REX dans lequel se situent les élèves. Cependant, plus on produira de messages similaires ($23+17 = 40$, $23 + 37 = 50$...), moins la quantité d'information apportée sera importante, puisque la probabilité de ces énoncés a augmenté à chaque exemple. Les faits ont donc une influence plus ou moins importante sur les systèmes, on pourrait dire que certains faits sont plus entropiques que d'autres, ou ont un potentiel entropique plus ou moins important. Ainsi, $5 + 7 = 12$ est peu entropique concernant S_3, mais beaucoup plus entropique concernant S_6 : la probabilité que le prochain message soit un énoncé produit par S_6 devient très faible.
On pourrait ainsi dire que les points de contact suggestifs sont sans doute plus entropiques que les points de contact de confirmation. De même, des faits invalidants, ce que l'on pourrait nommer des « points de contact invalidants », sont plus entropiques qu'un point de contact de confirmation. D'une façon générale, un fait inédit, soit un énoncé non encore produit, est porteur de davantage d'information qu'un fait non-inédit, c'est-à-dire d'une forme déjà identifiée. Ainsi, la première occurrence d'un fait de la forme « $\square$ est un nombre premier », par exemple f_1 =« 7 est un nombre premier » va diminuer fortement l'entropie de S_4 ou S_5. Ce fait est porteur d'une quantité d'information importante pour ces systèmes. Une deuxième occurrence sera moins entropique : f_2 =« 3 est premier » diminue de nouveau l'entropie de S_4, mais la probabilité de cet énoncé étant plus grande, l'information apportée est moindre, l'entropie de S_4 sera moins modifiée. Si l'on note H_0, H_1,H_2 respectivement l'entropie avant f_1, après f_1 et après f_2, on a $H_1(S_4) - H_0(S_4) > H_2(S_4) - H_1(S_4)$.

Entropie, TEA et faits

Si l'on revient à notre situation, nous pouvons faire des parallèles avec cet exemple : il s'agit d'établir des conjectures et de les éprouver, en notant des points communs locaux (Radford, 2008) et en les généralisant à un ensemble d'instances. Les TEA mobilisés ou construits dans cette situation permettent de produire des énoncés tenus pour vrai. Les faits construits par les élèves ne vont ainsi pas invalider ou valider un TEA, mais diminuer ou augmenter, dans des proportions diverses, son entropie. Lorsque les élèves appliquent $T_1 = +1/+1/+1$ sur un script $\{4/3/8\}$, le fait « $\{5/4/9\}$ est invalide » augmente l'entropie de T_1 et diminue l'entropie de $T_2 = +1/+1/+2$ ou $T_3 = +1/+1/+1+1$ [98]. Suite à cette action, si les élèves ajustent $b_7 \leftarrow b_7 + 1$, $H(T_1)$ augmente encore, et $H(T_2)$ ou $H(T_3)$ diminuent. Si l'on considère, dans cette situation, que les TEA sont des systèmes producteurs d'énoncés prétendus vrais, les faits sont des énoncés porteurs d'information, modifiant l'entropie des TEA possibles. Un même fait sera plus ou moins entropique, selon le problème que se pose l'élève, mais aussi selon l'entropie de tous ses

98. Puisque ces systèmes ne produisent pas $\{5/4/9\}$.

autres TEA plus ou moins probables. C'est une façon d'expliquer les TEA non invalidés après un fait invalidant, et la différence entre changement de TEA entre les groupes : ceux qui restent sur deux instances avec le TEA T_1 ont sans doute commencé l'activité avec une entropie pour T_1 plus faible que celle de ce TEA pour les groupes ayant directement enchainés sur T_2 ou T_3.
En outre, si le problème p_1 est « construire une instance à partir de la précédente », et si p_2 est « construire une certaine instance », on a $H_{p_1}(T_1) \neq H_{p_2}(T_1)$: l'entropie de T_1 n'est pas la même selon que l'élève est dans le problème p_1 avec un REX R_1 ou le problème p_2 avec un REX R_2 (y compris si $R_1 = R_2$).
Le potentiel entropique d'un fait f, son influence sur l'entropie d'un système de production d'énoncé, ici sous la forme d'un TEA T, dépend donc :

— du problème,

— de l'entropie initiale (avant la construction de faits concernant le problème) de T,

— des faits préalablement construits (dans le cadre du principalement, mais aussi en dehors)

Entropie et types de généralisation

Si l'entropie de l'information semble pouvoir rendre compte des mouvements de TEA en fonction des faits[99], elle peut aussi illustrer les moments de généralisation. Considérons le cas où, comme cela était attendu, les élèves passent par les étapes suivantes (voir aussi figure 3.99, p. 457) :

— construction d'instances consécutives : par exemple $TS(5) = \{4/3/8\}$ et $TS(6) = \{5/4/10\}$;

— identification des relations permettant de construire une certaine instance distante : par exemple $TS(15) = \{14/13/28\}$

— expression de ces relations avec des nombres génériques : par exemple $TS(6) = \{6 - 1/6 - 2/2 \times 6 - 2\}$.

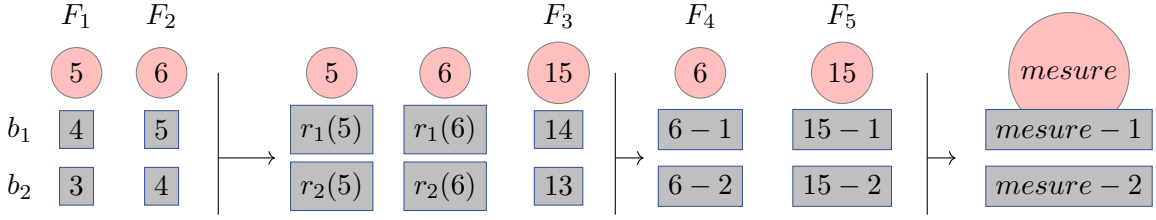


Figure 3.99 – Faits attendus

Les différents faits construits diminuent l'entropie du système produisant l'énoncé, pour les expressions des boucles, $\{\square - 1/\square - 2/2 \times \square - 2\}$ (figure 3.100, p. 458).
En revanche, si les faits construits mobilisent des nombres génériques et des nombres ayant perdu leur généricité (figure 3.101, p. 458), l'entropie du système qui permettra d'aboutir à la production d'expressions formelles du nombre d'itérations des boucles n'est pas diminuée — on peut cependant considérer que le système produisant les énoncés pour b_1 voit néanmoins son entropie diminuer.
Le fait « $\{5 - 1/4 - 1/10 - 2\}$ est valide », si l'on considère le problème de la généralisation, est peu entropique. Si l'expression des boucles ne mobilise pas le NG, comme la figure 3.102 (p. 459) l'illustre pour le groupe 46f (étape 30), ces faits sont *encore moins* entropiques.

99. Nous ne voyons pas l'entropie comme une *raison* de ce qu'il se passe chez les élèves, du point de vue des TEA, mais plutôt comme une *description* de ce qu'il se passe.

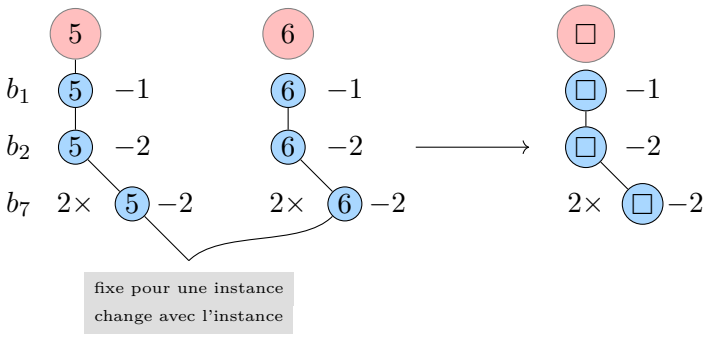


Figure 3.100 – Généralisation NG

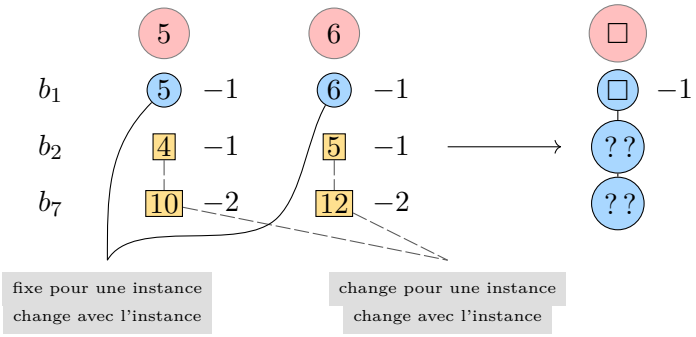


Figure 3.101 – Généralisation PNG

Enfin, un faux-fait peut être fortement entropique. Lorsque le groupe 45e (étape 43) teste un script avec $b_7 =$ (6-1 × 2), voulant représenter l'expression dénotant $b_7 = (6-1) \times 2$ pour un TS6 — expression valide —, ils auraient ainsi pu construire les faits représentés dans la figure 3.103 (p. 459) : si ces faits ne sont pas aussi entropiques que ceux n'impliquant que des NG (figure 3.100, p. 458), ils le sont plus que ceux impliquant un NG seulement pour b_1 (figure 3.101, p. 458). L'invalidation par l'EPGB de l'expression $b_7 =$ (6-1 × 2) est un faux fait porteur de beaucoup d'entropie : la probabilité d'obtenir une expression de la forme $\{\square - 1/??/(\square - 1) \times 2\}$ devient très faible. L'entropie du système « cible », $\{\square - 1/\square - 2/(\square - 1) \times 2\}$ est fortement augmentée, ce qui sera un obstacle important pour les élèves sur la voie de la formalisation des relations entre nombre de répétitions des boucles et valeur de la mesure.

Anticipation des faits entropiques ?

Ainsi, la question des faits plus ou moins entropiques semblent traverser la plupart des groupes observés. Ils posent alors de nouvelles questions, sur lesquelles il serait peut-être intéressant de se pencher :

— comment identifier *a priori*, dans une certaine situation, les faits (et les faux-faits) plus ou moins entropiques ?

— comment utiliser ces faits plus ou moins entropiques d'un point de vue didactique ? Par exemple, la construction de faits entropiques est-elle nécessaire, et si oui, est-ce que ces faits peuvent être partagés par le groupe classe, ou doit-on amener les élèves à les construire ?

— que faire des faux-faits, et de leur potentiel entropique plus ou moins important ?

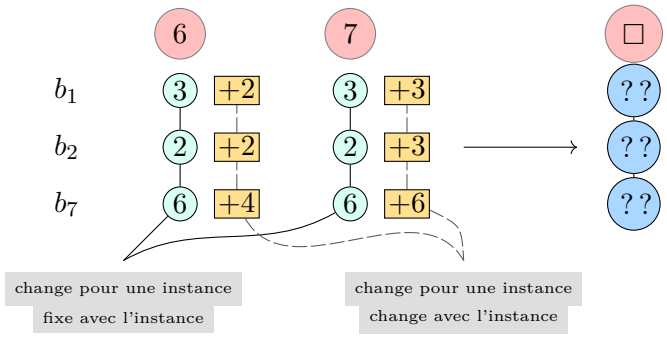


Figure 3.102 – Généralisation b(i)

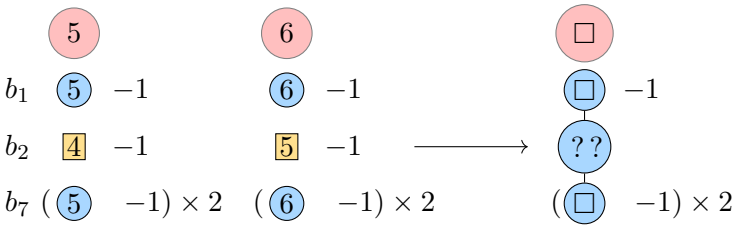


Figure 3.103 – Généralisation 45e, sans faux fait

Cette liste est bien sûr non exhaustive. Concernant notre situation, les faits les plus entropiques sont ceux rendant manifeste la relation avec la valeur de la mesure, que ce soit par un NG, un NSG ou un SG. Lorsque l'enseignante utilise systématiquement le terme « mesure » accompagné de sa valeur pour décrire un TS, elle contribue ainsi à la construction de faits entropiques. De même lorsqu'elle insiste, en plénière de début de séance 5, pour que les élèves verbalisent la relation entre la mesure et le nombre de répétitions des boucles. Ainsi, lorsqu'un élève évoque une relation intra-objectale qui risque de déboucher sur un fait invoquant un PNG (« un chiffre de moins que la boucle une »,3.36, 257), elle n'invalide pas cette relation mais incite les élèves à la reformuler par rapport à la mesure : « et est-ce qu'on peut dire par rapport à la mesure » (3.36, 259-262). Cela devrait contribuer à la construction de faits entropiques. Cependant, on a vu que certains groupes ne mobilisaient pas ces faits partagés, en partie parce qu'au moment de la plénière ils n'étaient pas encore sur le problème de l'expression d'un lien avec la mesure. On peut envisager, comme nous l'avons déjà évoqué, que l'évocation systématique de la mesure pour un tracé, avec un renforcement des contraintes de validité des tracés impliquant sa prise en compte, pourraient favoriser la construction de ces faits entropiques, qu'ils soient points de contact suggestifs ou points de contact de confirmation, comme les nomme Polya. Ainsi, un fait pertinent devrait être un fait entropique, mais un fait entropique n'est pas forcément pertinent (faux-fait).

246	14 :29.570	Prof	l'ordinateur il est capable de le faire tout seul
247	14 :32.130	Prof	de trouver de lui-même | ce qu'il va falloir mettre
248	14 :35.850	Prof	dans les boucles
249	14 :37.810	Prof	alors en fait là euh le principe
250	14 :41.250	Prof	général pour trouver la première boucle il suffit de faire quoi
251	14 :48.310	Eleves	bah c'est un chiffre de moins que
252	14 :50.240	Prof	c'est un chiffre de moins
253	14 :52.820	Eleves	que la mesure
254	14 :53.540	Prof	que la mesure
255	14 :54.520	Prof	dans la deuxième boucle c'est quoi ?
256	14 :58.750	Prof	H ?
257	14 :59.310	Eleves	un chiffre de moins que la boucle une
258	15 :02.050	Prof	c'est un de moins que la boucle une
259	15 :04.280	Prof	et est-ce qu'on peut dire par rapport à la mesure
260	15 :10.720	Eleves	c'est deux chiffres en moins
261	15 :11.860	Prof	c'est deux chiffres en dessous que la mesure
262	15 :13.940	Prof	si c'est huit c'est deux de moins ça fait six

Transcription 3.36 – 46 Prof, S5 : marquer la mesure

3.8 Séance 6 : vers l'algèbre

L'analyse des séances 3 à 5 laisse entrevoir une avancée des élèves vers la généralisation algébrique. Lors de la séance suivante, après avoir partagé en classe le script solution mobilisant la variable mesure, il est demandé aux groupes (formés cette fois d'un mélange des différents binômes) « d'aller plus vite que l'ordinateur ». En effet, un autre fait est partagé en classe : plus la valeur de la *mesure* est grande, plus le temps de tracé et dénombrement est long, et il devient rédhibitoire. Notons qu'il est seulement demandé aux élèves de mettre par écrit une méthode permettant « d'aller plus vite que l'ordinateur », l'utilisation d'un symbolisme n'est pas exigé.
Sur les 14 groupes formés par les deux classes, on observe que 10 aboutissent à une méthode valide. En outre, sur ces 10 groupes, 5 utilisent une formulation générique rhétorique, et 4 utilisent une expression mobilisant le paramètre *mesure*. Le dernier groupe utilise une formulation avec deux PNG soulignés : $(\underline{9} \times 3) + (\underline{3} \times 3) - 3$ pour un TS5, mais indiqué comme étant un TS7.
Concernant les démarches non valides, deux groupes ont cherché une relation sans doute liée à la proportionnalité, consistant à trouver l'un des nombres a, b, c dans l'expression $a \times b = c$. Un groupe cherche d'abord c si a est la mesure et b le nombre d'hexagones d'un TS de mesure a, puis cherche b lorsque, a dénotant toujours la mesure, c'est b qui est le nombre total d'hexagones. Un autre groupe a seulement évoqué « il faut diviser le nombre d'hexagones par la mesure », ce qui revient au cas précédent. Le dernier groupe a quant à lui identifié des régularités entre le nombre d'hexagones « blancs » et le nombre d'hexagones « colorés » lorsqu'on passe d'une instance à la suivante (figure 3.104). On voit ainsi qu'ils ont identifié le schéma suivant, avec u_i dénotant le nombre d'hexagones blancs pour un TS de mesure i, et v_i dénotant le nombre d'hexagones effectivement tracés pour cette même instance :

— $u_4 = 1$; $u_i = u_{i-1} + (i - 3)$;

— $v_4 = 9$; $v_i = v_{i-1} + 3$

Si ce schéma est valide, les élèves de ce niveau de classe ne disposent pas pour autant des outils permettant de déterminer le $n^{\text{ème}}$ terme d'une suite, sans déterminer les $n - 1$ termes précédents.

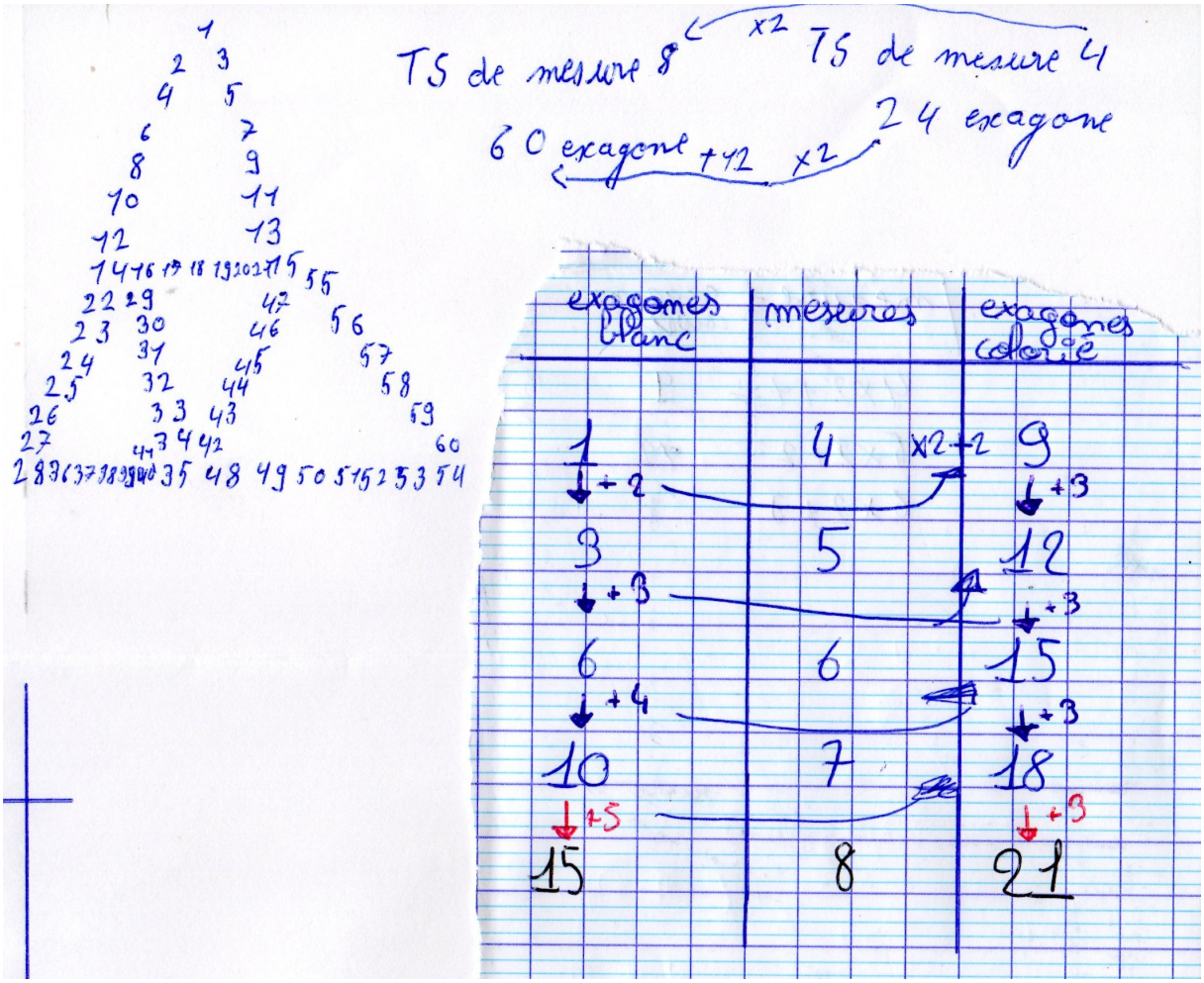


Figure 3.104 – Séance 6, 45, Groupe 2

Une première démarche valide identifiée consiste à considérer chacun des côtés tracés indépendamment et sans leurs sommets (donc formés de $n-2$ hexagones) puis à ajouter les 6 sommets, ce qui se formulerait $(n-2) \times 9 + 6$. Le groupe ayant procédé ainsi a utilisé une formulation syncopée, s'appuyant sur un schéma et des couleurs (figure 3.105) : « la mesure on fait -2 la mesure $\times 9$ $+6$ ». Ici, « la mesure » est non univoque, c'est une expression désignant le dernier nombre obtenu :

1. si on veut un TS de mesure 5, le premier « la mesure » est le nombre 5, ce qui correspond bien à la *mesure* du TS ;
2. on lui enlève deux (les sommets), et on obtient les côtés représentés en rose, de 3 hexagones.
3. « la mesure » dénote alors ces côtés de 3 hexagones, présents 9 fois. Le deuxième « la mesure » est donc le nombre 3 dans cette instance.
4. on ajoute 6 au résultat.

Cette formulation ressemble aux formulations arithmétiques que l'on peut trouver avant l'algèbre : $5 - 2 = 3 \times 9 = 27 + 6 = 33$, chaque signe égal marquant un calcul effectif, ce qui est représenté par ce groupe par une couleur distincte.

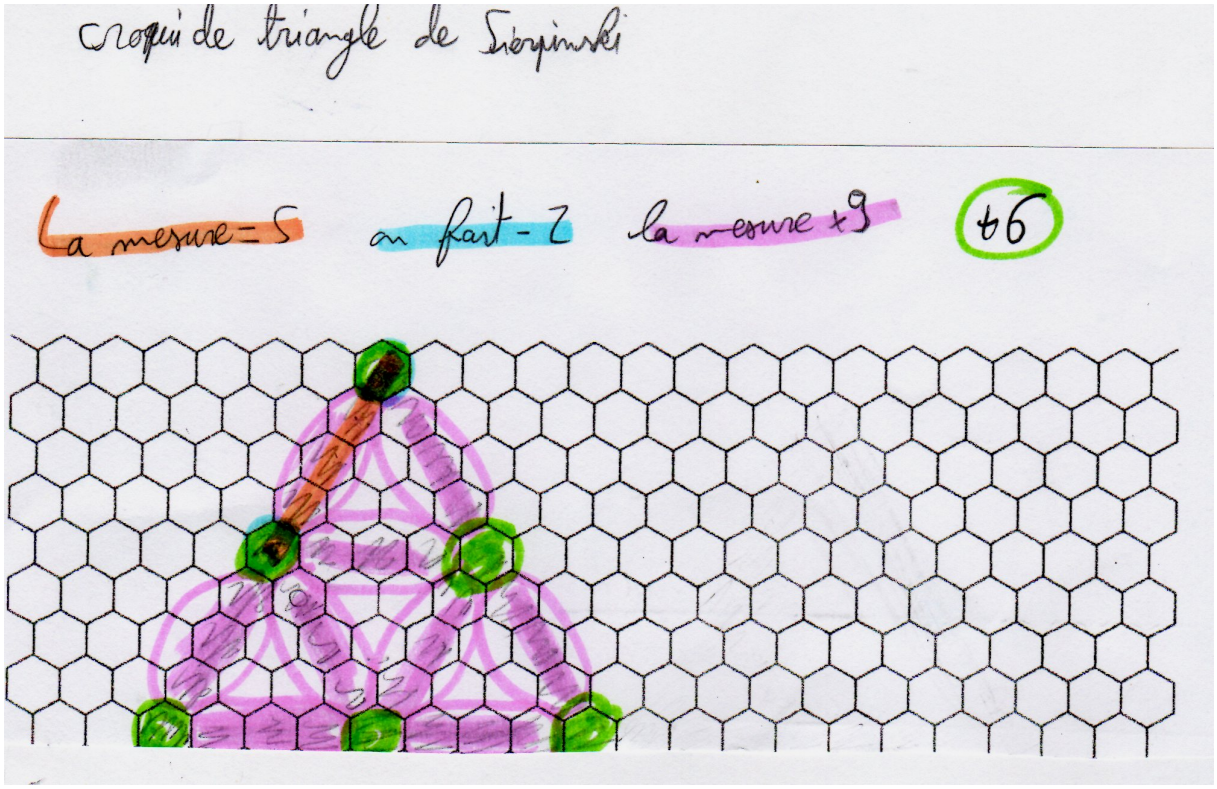


Figure 3.105 – Séance 6, 45, Groupe 4

Une autre démarche utilisée par les élèves consiste en une identification de côtés formés de n hexagones (au nombre de 3), puis au constat que les côtés restants sont constitués de $n-2$ hexagones, et qu'il y en a 6. La formulation dans ce cas est $n \times 3 + (n-2) \times 6$. Un groupe ayant utilisé ce découpage a formulé sa solution sous une forme syncopée mobilisant des PNG (figure 3.106) : « On fait $9 \times 3 = 27$ car dans un triangle on en compte 9 et 3 fois. Après $7 \times 6 = 42$ car il y en reste 7 à chaque fois donc 6 ». Une justification se basant sur la figure a été faite, mais nous avons dû demander aux élèves d'expliciter comment trouver les PNG (les « -2 » ont ainsi été rajoutés après discussion). Les écrits montrent que les élèves n'identifient pas toujours le TS par sa mesure (ici 9), mais aussi par le nombre total d'hexagones (ici 69) : un « TS de mesure 9 » est équivalent pour eux au « TS de mesure 69 ». La mesure n'étant pas identifiée comme propriété définissant les TS, le paramètre associé parait difficilement mobilisable.

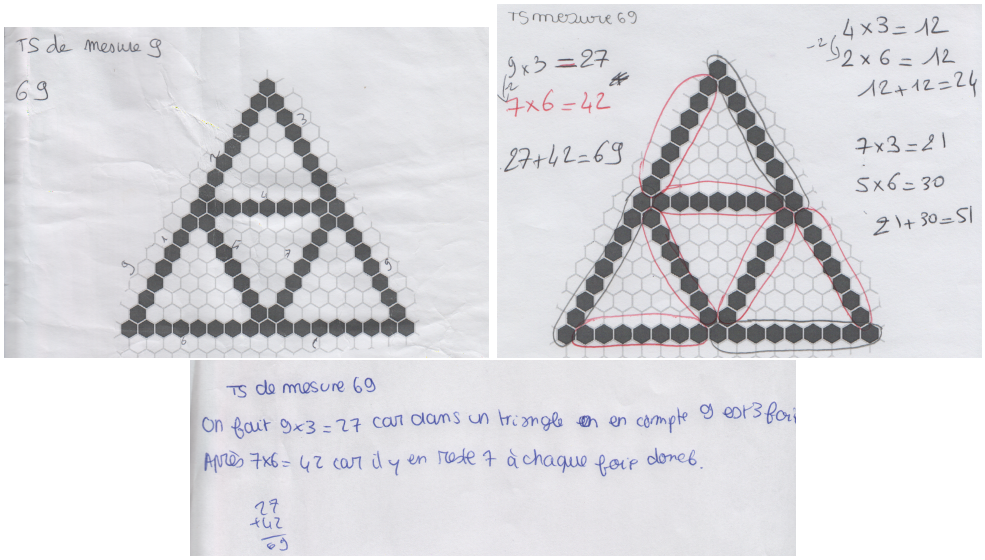


Figure 3.106 – Séance 6, 45, Groupe 12

Une troisième méthode consiste à dénombrer les triangles de base, avec ou sans hexagones surnuméraires :

— Un triangle de base compte $3n - 3$ hexagones, on en prend trois, et trois hexagones seront surnuméraires : $(n \times 3 - 3) \times 3 - 3$. Deux groupes utilisent cette méthode. L'un de façon rhétorique, considérant un « triangle », le nombre d'hexagones d'un triangle de base, et est exprimée ainsi : « on prend un triangle on fait $\times 3$ puis avec le nombre obtenu on fait -3 ». Une explicitation du « triangle » étant donnée : « 1 triangle fait $4 \times 3 = 12$, $12 - 3 = 9$ ». L'autre utilise une formulation symbolique avec le paramètre *mesure* : $(mesure \times 3 - 3) \times 3 - 3$. Cependant, il n'y a pas de parenthèses dans leur écriture, qui en reprend une indiquant verticalement les touches de la calculatrice sur lesquelles il faut appuyer (en terminant par la touche « R »). Dans les deux cas, la solution est envisagée sous son aspect procédural et non structural (Sfard, 1995).

— Un triangle de base compte, sans ses sommets, $(n - 2) \times 3$ hexagones, on en prend trois et on ajoute les six sommets. Le groupe ayant procédé ainsi, d'abord avec un NG, aboutit à la formulation $(mesure - 2) \times 3 \times 3 + 6$. Ce groupe propose alors diverses réécritures en mobilisant associativité et commutativité (mais pas la distributivité), et aboutit entre autres à $(mesure - 2 \times 9) + 6$. Cette fois, ce groupe non seulement met en œuvre l'aspect structural, mais travaille aussi sur les équivalences entre formules.

Deux groupes envisagent en premier lieu le nombre d'hexagones du « grand » triangle (soit $(2n - 1) \times 3$, avec 3 hexagones comptés deux fois) puis le « petit » triangle central. Le triangle central est dénombré comme étant :

— $n \times 3$, avec 3 hexagones surnuméraires,

— $(n - 2) \times 3$, sans hexagone surnuméraire.

Ainsi, un groupe manifeste, d'abord avec des PNG puis de façon rhétorique (avec omission de l'addition de deux termes), une formulation équivalente à $(2n-1)\times 3+(n\times 3)-9$: « on prend la mesure multiplier par deux on soustraie 1 multiplier par trois la mesure et soustraire 9 ». Un autre détaille les calculs intermédiaires en mobilisant des PNG, non explicités, mais soulignés (figure 3.107) :

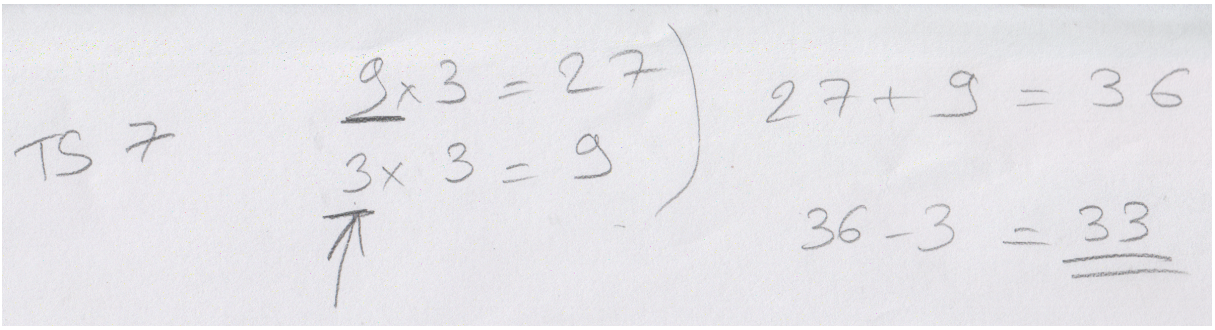


Figure 3.107 – Séance 6, 46, Groupe 14

Enfin, on obtient trois formulations équivalentes à $9n - 12$, où n est la valeur de la mesure. L'une est rhétorique avec une dénomination de la propriété manipulée (figure 3.108) : « il faut faire mesure du triangle × tous les côtés (9) −12 ». Les deux autres sont écrites *mesure* × 9 − 12 (figure 3.109). Il s'agit ici de l'expression réduite, nécessitant éventuellement des réécritures, ce qui pourrait être la manifestation d'un aspect structural et non seulement procédural.

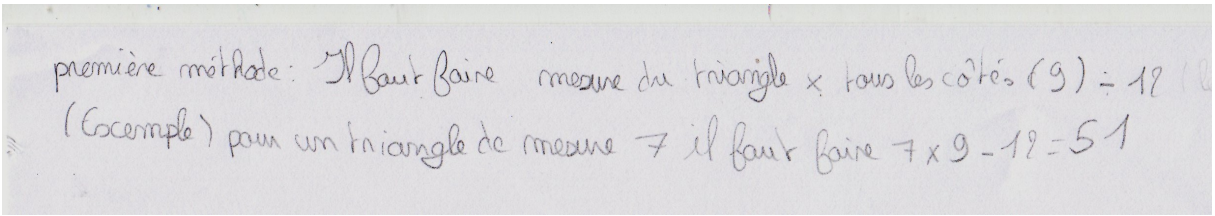


Figure 3.108 – Séance 6, 45, Groupe 6

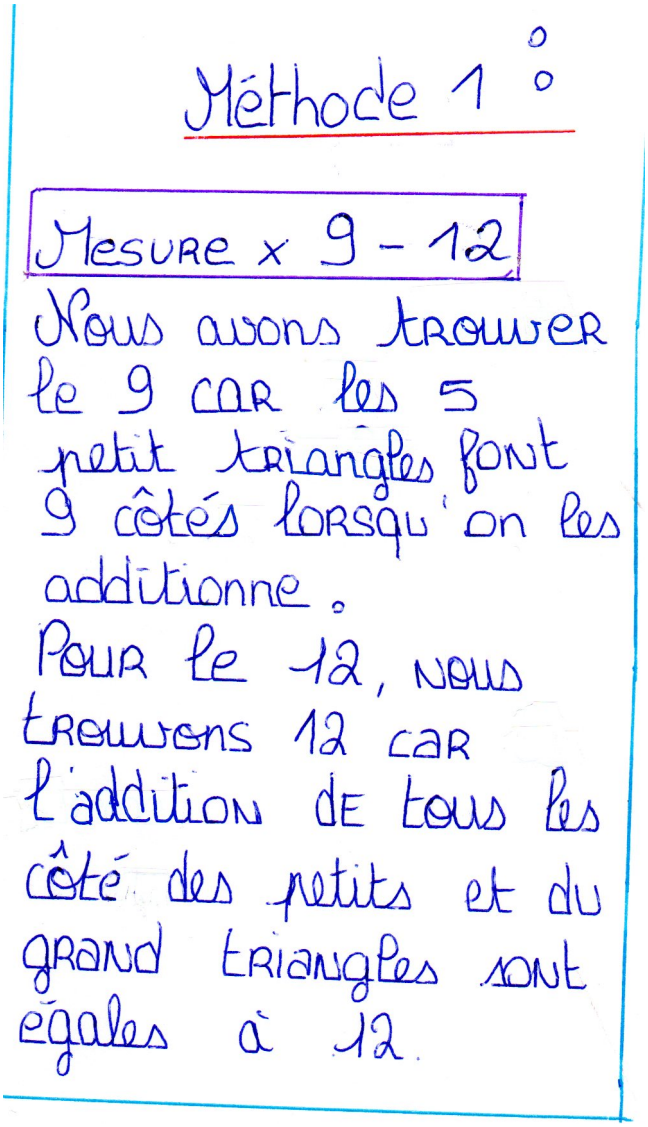


Figure 3.109 – Séance 6, 46, Groupe 7

Ainsi, si neuf groupes aboutissent à une formulation générique, seuls cinq utilisent une expression algébrique (en considérant qu'il s'agit d'une expression dans le langage choisi permettant de manipuler la représentation du paramètre comme s'il était connu). On peut par ailleurs remarquer que les méthodes erronées se basent toutes sur des relations entre nombres, les autres s'appuyant sur un cadre graphique. Il n'était certes pas demandé une formulation symbolique, mais une « méthode ». Cependant, cela laisse penser que si une partie des élèves a construit la nécessité du paramètre et de sa nécessité, le partage de la solution du script générique en groupe classe n'est pas suffisante pour permettre aux autres élèves d'identifier cette nécessité. Il faudrait s'interroger sur la construction de faits réellement partagés par la classe — et là encore des caricatures pourraient être une piste à explorer. De plus, on constate là qu'il ne s'agit que du début de la construction du concept : d'autres situations, dans des contextes plus ou moins proches, en algèbre comme en informatique, seront nécessaires.

groupe	valide	rhétorique	mesure	NG	NG/PNG	formule équivalente
1					✓	$n \times N$ ou $n \times ? = N$
2						suites
3	✓		✓	x		$(n-2) \times 3 \times 3 + 6$ et $(n-2) \times 9 + 6$
4	✓	✓				$(n-2) \times 9 + 6$
5	✓	✓		✓		$n \times 9 - 12$
6	✓				✓	$(2n-1) \times 3 + (n \times 3) - 9$
7	✓		✓			$n \times 9 - 12$
8	✓	✓				$[n \times 3 - 3] \times 3 - 3$
9	✓		✓			$[n \times 3 - 3] \times 3 - 3$
10		✓			✓	$T \times 3 - 3$ (T=triangle de base)
11	✓		✓			$n \times 9 - 12$
12	✓	✓		✓	✓	$(n \times 3) + (n-2) \times 6$
13		✓				N/n
14	✓				✓	$(2n-1) \times 3 + (n-2) \times 3 - 3$

Tableau 3.9 – Séance 6, propositions des groupes (n= valeur de la mesure, N= nombre total d'hexagones)

Conclusion

Informatique et algèbre ont une proximité certaine. Comme le rappelle Berry (2020, p. 71) en reprenant les piliers de l'informatique de Dowek (2011), l'informaticien travaille « sur l'information représentée par des *données numérisées* à l'aide d'*algorithmes*, objets conceptuels qui doivent eux-mêmes être traduits en *programmes* écrits dans des *langages* appropriés pour devenir exécutables par les *machines* que sont les ordinateurs de tous types ». En algèbre aussi, on travaille sur des *données*, afin de produire une procédure, un *algorithme* de résolution du problème, algorithme qui s'écrit sous la forme d'une suite de transformations d'expressions dans un *langage* algébrique, destiné à être exécuté, par un « dispositif d'exécution » (Samurçay et Rouchier, 1985), généralement humain. Dans ces deux cadres, les algorithmes sont constitués de *séquences* ordonnées d'instructions, instructions qui peuvent être des *itérations* (ou des boucles), ou des *conditions* permettant notamment de contrôler des branchements conditionnels. Mais algorithme et algèbre nécessitent aussi la manipulation de *variables*.

Variable et variable dans un rôle de paramètre

En quoi une situation informatisée contraignant l'utilisation d'un langage formel permet-elle de participer à la construction de la nécessité de la variable dans un rôle de paramètre ?

D'un point de vue sémiotique, nous avons explicité pourquoi, selon nous, une variable, dans un langage θ — que ce langage soit un langage informatique, un pseudo-langage algorithmique ou le langage algébrique —, était un symbole du langage θ représentant un objet non forcément déterminé ou connu *a priori*, manipulable dans des expressions de θ comme si cet objet était connu. Or, un des rôles de la variable est celui de *paramètre*, qui porte la généralisation, élément fondamental de l'algèbre comme de l'algorithmique.
Certes, comme l'avait déjà soulevé en son temps Ada Lovelace, les variables informatiques ont une spécificité : elles peuvent, parfois, changer au cours du temps. Ce sont dans ce cas des variables de travail, temporaires, rendues nécessaires pour expliciter ou rendre automatisable une procédure de résolution. Ces rôles particuliers des variables sont nécessaires non du fait de l'informatique, mais parce qu'en informatique on cherche à supprimer la pensée. Il s'agit, comme le disait Pascal, de « te soulager du travail qui t'a souventes fois fatigué l'esprit, lorsque tu as opéré par le jeton ou par la plume » (Pascal, 2016c, p. 1003), ou comme le formule de façon plus moderne le site Pixees :

> Un algorithme, c'est tout simplement une façon de décrire dans ses moindres détails comment procéder pour faire quelque chose. Il se trouve que beaucoup d'actions mécaniques, toutes probablement, se prêtent bien à une telle décortication. Le but est d'évacuer la pensée du calcul, afin de le rendre exécutable par une machine numérique (ordinateur, ...). On ne travaille donc qu'avec un reflet numérique du système réel avec qui l'algorithme interagit. (Pixees, 2014)

Cependant, lorsque Lovelace utilise une variable d'entrée, un paramètre, comme variable de travail, c'est une modification du statut de la variable qui est rendue nécessaire par l'économie de mémoire. La représentation concrète des nombres, la mémoire informatique, est forcément limitée, ce qui peut contraindre à la réutilisation d'espaces mémoires. On perd ainsi de l'information, ce qui par exemple pourrait empêcher toute modification ultérieure du programme qui impliquerait une relation avec le paramètre. Ainsi, le *storehouse* de la machine analytique de Babbage ne pouvait stocker que seize nombres de trente-trois chiffres digitaux (Ricquebourg, 2008, p. 305) : la réutilisation des espaces s'avère rapidement nécessaire. De nos jours, les capacités mémoires peuvent permettre de ne pas réutiliser un espace déjà occupé par la valeur d'une variable. Dans ce cas, la variable qui a un rôle de paramètre peut être non mutable (comme dans les langages

fonctionnels) : variable mathématique et variable informatique se rapprochent encore et cette proximité dépasse alors l'aspect purement sémiotique. Il reste cependant une différence entre variable informatique et mathématique, qui est celle de l'implémentation concrète de la valeur, implémentation qui peut avoir une influence sur le calcul effectif, mais aussi sur l'algorithme utilisé. La situation que nous avons expérimentée utilise, pour le paramètre définissant un motif, une variable qui ne sera pas modifiée. De plus ce paramètre est de type entier, la situation n'est donc pas concernée par les problèmes issus du codage binaire de décimaux.
Ainsi, non seulement le paramètre est fondamental pour l'algèbre comme pour l'informatique, mais c'est un concept à l'interface entre ces deux registres qui peut ne pas impliquer certaines des difficultés inhérentes à la différence entre les aspects mathématiques et informatique de ces variables. D'autre part, nous avons vu que le passage à la variable dans un rôle de paramètre était possiblement un obstacle épistémologique. Concilier l'unique et le multiple, identifier un objet non encore défini à sa représentation, sont des modes de pensée qui ont tardé à émerger. Cette rupture, à l'époque notamment de Descartes, a été rendue nécessaire par une recherche d'efficacité des méthodes utilisées : afin d'être reproductibles, facilement exécutables sans connaissances spécifiques et dénuées d'erreurs, il ne fallait plus seulement généraliser les méthodes, mais aussi les rendre automatisables, exécutables par un dispositif d'exécution automatique. Nous soutenons qu'il est possible de recréer les conditions d'expression de cette nécessité en utilisant une situation informatisée. En effet, « faire faire » par une machine une méthode générique implique la nécessaire existence du paramètre et de sa représentation dans le langage de la machine. Faire faire par un humain en imaginant que cet humain est dénué de pensée est plus problématique, en partie parce qu'on ne connait pas les opérations élémentaires dont est capable cet humain fictif — faut-il détailler comment additionner deux nombres écrits en chiffres ? Une situation informatisée, si elle permet aux élèves d'explorer le champ des possibles, et notamment les possibilités langagières du dispositif, est ainsi susceptible de contraindre la rencontre des élèves avec le savoir ici visé.

Situation

En quoi et à quelles conditions une situation de généralisation informatisée de motif, à l'interface entre algèbre et informatique, permet-elle d'étudier comment et à quelles conditions les élèves peuvent construire le concept de variable dans un rôle de paramètre ?

La situation de généralisation de motif informatisée que nous avons expérimentée nécessite, pour aboutir à une solution valide, que les élèves mettent en œuvre une pensée algébrique au sens de Radford (2018, p. 8) :

- cela concerne des quantités indéterminées, ici la valeur de la *mesure*, le nombre d'hexagones formant un côté du triangle de base d'un TS ;
- ces quantités disposent d'une représentation ou d'une symbolisation, de même que les opérations qui les concernent ;
- ces quantités indéterminées sont manipulées de façon analytique : on opère sur ces quantités comme si elles étaient connues.

Le critère d'analycité, essentiel, est présent si les élèves généralisent le script : ils font bien de l'algèbre et sont amenés à construire la nécessité de la variable dans un rôle de paramètre et de la manipuler. En outre, cette situation met en place un milieu contraignant, proactif et rétroactif qui, malgré certaines limites (notamment lors d'un cumul d'erreurs), permet une interprétation de la rétroaction. Cette interprétation ne se limite pas à la validation, puisqu'elle permet une mise en relation des faits premiers produits par la rétroaction, pouvant aboutir sur la construction de faits

tiers, faits raisonnés expliquant le lien entre les actions de programmation et la rétroaction. Ces faits construits par les élèves, mis en tension avec les nécessités du problème, doivent permettre la construction du concept : comme le disait C.-S. Peirce (2000, p. 4), « le but du raisonnement est de découvrir par l'examen de ce qu'on sait déjà quelque autre chose qu'on ne sait pas encore. » Cependant, nous avons constaté, lors de l'analyse, que les faits construits par les élèves étaient notamment liés au problème ou au sous-problème posé par les élèves. Ainsi les faits construits dans un problème (la construction d'instances) ne sont pas forcément mobilisés dans un autre problème (la généralisation du script), si les élèves considèrent que ces deux problèmes sont indépendants : un fait dans un problème p_1 n'est pas nécessairement un fait dans un autre problème p_2.
De même, les faits partagés en classe, parce qu'ils ne concernent pas forcément le problème posé par les élèves au moment du partage, ne sont pas toujours mobilisés ultérieurement. On peut noter que ces mises en commun, lors de notre expérimentation, n'avaient pas été conçues comme un travail de problématisation. Ainsi, les faits partagés en groupe classe étaient pour l'essentiel des faits seconds. Par exemple, les faits $[mesure = 5 \implies b_1 = 4 \wedge b_2 = 3]$ ou $[mesure = 5 \implies b_1 = 4 \wedge b_2 = 3]$ sont présentés comme des faits (seconds) assertoriques : c'est ainsi et pas autrement. Un travail à base de caricatures (Orange, 2012, p. 102) à partir des programmes des élèves aurait peut-être permis de travailler sur les *raisons* de ces faits, construisant ainsi des faits tiers, apodictiques. Ces raisons, comme nous l'avons évoqué dans l'analyse du groupe 45f (p. 284), ne sont pas des plus simples dans la situation expérimentée. Dans celle-ci, la boucle b_1 trace tous les hexagones du premier côté, sauf son premier sommet qui sera tracé par la boucle b_3 et la boucle b_2 trace les hexagones du deuxième côté excepté ses sommets, dont le premier a été tracé par b_1 et le second le sera par b_5. Pour b_2, tout se passe comme si, dans le tracé, on pivotait sans raison avant son second sommet. La variation du script permettant de pivoter « avec raison » (script 3.3, p. 286) pourrait rendre ces raisons plus accessibles, mais au détriment de l'interprétation des erreurs rendues visibles par la rétroaction (p. 293).
Ce travail de construction de faits raisonnés pourrait venir en appui pour les élèves n'ayant pas construit suffisamment de faits validants, ce qui, comme nous l'avons constaté, empêche la généralisation. Il devrait s'appuyer sur les jeux de cadres, que nous n'avons sans doute pas suffisamment sollicités : cela permettrait d'amener les élèves à raisonner sur l'algorithme et sa logique, en lien avec le tracé, et non seulement sur les nombres, comme de nombreux groupes l'ont fait.
Toujours concernant la question des faits et de leur construction, il nous semble important de traiter de la question des faux-faits, qu'ils soient issus de la situation ou de l'artefact mobilisé. Ainsi, la relation transparente entre la valeur de la mesure et la dimension des hexagones tracés a perturbé l'identification de la valeur entrée à la valeur de la mesure du TS à tracer. La tâche prescrite maladroitement choisie consistant à tracer « un TS17, TS19 ou TS21 », avec une valeur de la mesure fixée à 22, a eu parfois le même effet. Ce sont ici des faux-faits évitables avec une analyse *a priori* plus fine de la situation. D'un autre côté, les faux-faits issus du dispositif d'exécution, la version de Snap ! utilisée, nous incitent à considérer qu'une exploration didactique de l'artefact est nécessaire. Ces faits, dépendant de l'artefact et non de la situation, ne sont évitables que par une utilisation d'un autre artefact (par exemple une version plus récente de Snap !). Cependant, l'identification de ces difficultés didactiques permettrait non seulement d'anticiper l'apparition de ces faux-faits, mais aussi éventuellement de les utiliser dans la recherche de raisons : par exemple, pourquoi, dans la version de Snap ! utilisée, a-t-on 

? Cela pourrait être le début (ou le réinvestissement) d'un travail sur les représentations non numériques des données, sources d'autres difficultés chez les élèves, comme le rappellent Lagrange et Rogalski (2017).

Pour clore la réflexion sur la situation choisie, soulignons que, ayant été mise en œuvre au début de ce travail de recherche, la production du texte de savoir est restée limitée aux énoncés de définition « classique » de la variable. Il serait probablement plus intéressant d'aboutir à un texte évoquant la nécessité des variables manipulables comme des nombres connus. Là encore, un travail collectif sur les caricatures serait à construire, discutant par exemple de la recherche de blocs qui permettraient de résoudre le problème, et qui manifestent la tension fixe-variable (tel 

). Les nécessités du problème sont établies par une partie des élèves, il faudrait les rendre explicites et partagées par la communauté scientifique que constitue le groupe classe.

Généralisation

Comment et à quelles conditions les élèves construisent-ils les faits et les nécessités du problème de généralisation informatisée d'un motif ?

En ce qui concerne la généralisation, nous avons pu voir à quel point celle-ci s'appuyait sur la tension fixe-variable : un même script doit produire plusieurs tracés, alors que ses instructions ne changent pas. Dans notre situation, cela revient à régler la tension entre le fait que les expressions des boucles b_i ne changent pas, et le fait que $\delta(b_i)$ change suivant la mesure entrée. Ici encore, on voit l'apport d'une situation informatisée, qui permet de faire vivre et d'explorer cette tension. Les relations *et* la variable ont été considérées par les élèves, montrant une hésitation entre faire « bouger » la valeur de la mesure, en prenant au départ un signe à caractère générique (SG), et faire « bouger » la relation. Cela amène à des créations de faits plus ou moins entropiques : par exemple, les pertes de généricité induites par les PNG diminuent la probabilité d'identifier la relation entre la propriété caractéristique des TS (la *mesure*), et les valeurs du nombre de répétitions des boucles. La recherche de blocs spécifiques traitant la relation, ou de représentations traitant le paramètre, est une exploration rendue possible par le dispositif. On notera aussi la recherche satisfaisant à la nécessaire existence d'un signe portant la généricité, par l'utilisation de nombres-signe génériques (NSG), ou de signes génériques (SG). Le SG signe vide dans les opérateurs, par exemple dans 

, amène ainsi à une précision essentielle : la variable n'est pas qu'un signe marquant une place dans une expression, c'est un signe *univoque*. Ce peut être le cas lorsque les élèves définissent $b_1 =$ 

et $b_2 =$ 

: le même signe est remplacé — parfois par simulation des élèves — par une même valeur. Mais ce n'est pas le cas pour le groupe ayant abouti à $b_7 =$ 

, puisque dans ce cas les SG des boucles b_1 et b_2 se voient substitués par la valeur de la mesure, mais le SG de b_7 par le double de la mesure. Cette univocité est là encore rendue nécessaire par le besoin de supprimer toute forme d'implicite.
L'identification de la valeur entrée par l'utilisateur à la valeur de la mesure du TS est nécessaire pour généraliser le script, afin d'identifier la relation entre la mesure et le nombre d'itérations des boucles : c'est ce qui doit permettre de faire vivre la tension entre script unique et tracés multiples. Cependant, l'instruction de l'EPGB permettant de demander une entrée, de l'attendre et de la stocker, a été considérée comme problématique, puisque affectant de façon transparente la valeur entrée à la variable prédéfinie 

. Elle a donc été elle-même rendue transparente, utilisée dans les blocs d'initialisation des scripts, non accessibles aux élèves. Cette transparence nous apparait comme problématique : il a ainsi fallu à plusieurs reprises que l'enseignante explicite le fait que la valeur entrée était visible dans l'affichage de la variable 

. Nous craignions avec l'enseignante que cela soit trop incitatif, mais non seulement ce ne fut pas le cas, mais aussi cela a entraîné une autre difficulté. Comme nous l'avons déjà évoqué, l'initialisation consistait entre autres à définir la dimension des hexagones tracés afin que le tracé du TS reste dans les limites de l'espace d'exécution. Le paramètre 

apparait ainsi dans deux types de relations

très différentes : celles entre la mesure et les valeurs du nombre de répétitions des boucles, mais aussi celle entre la mesure et la dimension du tracé. Or, la rétroaction étant visuelle, c'est le lien entre mesure et dimension du tracé qui apparait lorsqu'on modifie la valeur entrée. Il s'agit ici d'un obstacle à l'identification de la valeur entrée et de la valeur de la mesure, que l'on peut sans doute lever en séparant la dimension de la valeur de la mesure — ce qui implique que les élèves devraient définir eux-mêmes la dimension, ou une valeur exprimant cette dimension du tracé. Cela étant, dans le cas où les élèves ont identifié la mesure à la valeur entrée, nous avons pu observer la tension unique-multiple à laquelle étaient soumis les élèves, qui alors ont cherché une expression manipulant une représentation de la mesure. Là encore, faire vivre cette tension nous parait essentiel pour construire le concept de variable dans un rôle de paramètre.

Méthodologie

La méthodologie mise en place permet-elle d'identifier les faits et nécessités construits par les élèves, ainsi que leur mise en tension, à partir des traces d'actions de programmation ?

D'un point de vue méthodologique, notre recherche nous parait montrer l'intérêt d'un dispositif permettant de disposer d'une trace de la dynamique de l'activité, représentable de diverses façons et avec diverses granularités. La captation automatisée d'actions de programmation est ainsi un outil permettant d'analyser de façon fine l'activité des élèves. Cela donne aussi la possibilité de traiter éventuellement un grand nombre de cas : une analyse quantitative est alors possible. Cependant, considérant que l'activité des élèves ne se limite pas à des actions de programmation, mais concerne aussi les interactions langagières, n'utiliser que la captation de ces actions implique aussi une perte d'informations. C'est un complément utile des interactions langagières, car les élèves ne disent pas tout ce qu'ils font, et les interactions langagières sont un complément utile du dispositif, qui ne donne pas à voir ce que disent les élèves de ce qu'ils font. En outre, la possibilité de disposer de l'histoire du programme et de la rendre visible nous parait être intéressante en ce qui concerne la mise en œuvre d'épisodes s'appuyant sur des caricatures, dont nous avons évoqué l'intérêt possible.

Clôture

Une situation de généralisation informatisée de motif nous semble ainsi potentiellement pertinente dans le cadre de la construction de la variable dans un rôle de paramètre. Elle ne suffit bien entendu pas. Il faudrait à présent concevoir un ensemble organisé de situations aboutissant à la construction des différents savoirs algébriques. L'utilisation du motif pour travailler les équivalences entre formules, comme l'a fait l'enseignante, est une possibilité. L'intégration de la situation de généralisation informatisée entre des situations de généralisation de motif non informatisées, non symboliques dans un premier temps, symboliques dans un deuxième temps est aussi une piste à explorer.
Nous n'avons pas abordé ici les spécificités des variables mutables, par exemple la variable dans un rôle de compteur qui a été utilisée dans la séance 2. Cette variable n'a pas semblé, *a priori*, être un obstacle, peut-être du fait de la proximité avec les divers jeux numériques dont nous sommes entourés. Ce rôle peut être travaillé dans des situations sans doute plus pertinentes, comme par exemple « le retour au port », expérimentée dans le cadre du groupe IREM-algo de l'Académie de Nantes [100].

100. https://irem.univ-nantes.fr/groupes-de-recherche/algorithmes-et-programmation/cycle4/retour-au-port

Une prolongation serait aussi à envisager pour préciser le caractère défini ou non défini du paramètre. Celui-ci n'est mathématiquement pas défini avant qu'une valeur lui soit donnée. Cependant, du point de vue informatique, il est défini, dans les EPGB, avant même sa première initialisation. En effet, lors de la création d'une variable, celle-ci est automatiquement initialisée à zéro, de façon transparente.
Enfin, les allers-retours entre variables informatiques, variables algorithmiques et variables mathématiques seront aussi à questionner : commencer par la proximité des concepts de variable dans les deux registres informatique et algébrique, avant de poursuivre par les différences, n'est-ce pas source d'autres difficultés ? La construction de savoirs apodictiques concernant les variables nous parait essentielle pour dépasser ces éventuelles difficultés.

Liste des figures

Liste des tableaux

Liste des Algorithmes

Bibliographie

Abiteboul, S. et G. Dowek (2017). *Le temps des algorithmes*. Paris, France : Éditions le Pommier. 191 p. ISBN : 978-2-7465-1175-0 (cf. p. 53).

Algèbre classique (2019). In : *Wikipédia*. URL : https://fr.wikipedia.org/w/index.php?title=Alg%C3%A8bre_classique&oldid=163621400 (visité le 17/08/2020) (cf. p. 22).

Anawati, G. C. et R. Rashed (2020). *ISLAM (La civilisation islamique) - Les mathématiques et les autres sciences*. In : Encyclopædia Universalis. URL : http://www.universalis-edu.com.budistant.univ-nantes.fr/encyclopedie/islam-la-civilisation-islamique-les-mathematiques-et-les-autres-sciences/ (visité le 15/04/2020) (cf. p. 22).

Bachelard, G. (1938). *La formation de l'esprit scientifique : contribution à une psychanalyse de la connaissance objective*. 16. éd. Bibliothèques des textes philosophiques. Paris : Vrin. 256 p. ISBN : 978-2-7116-1150-8 (cf. p. 187).

Bächtold, M., V. Durand-Guerrier et V. Munier (2017). *Épistémologie & didactique : synthèses et études de cas en mathématiques et en sciences expérimentales*. Pratiques et techniques. Besançon : Presses universitaires de Franche-Comté. ISBN : 978-2-84867-603-6 (cf. p. 21, 120).

Backus, J., H. Herrick et I. Ziller (1954). *Preliminary Report : Specifications for the IBM Mathematical FORmula TRANSlating System, FORTRAN*. Programming Research Group, Applied Science Division, International Business Machines Corporation. URL : http://www.softwarepreservation.org/projects/FORTRAN/BackusEtAl-Preliminary%20Report-1954.pdf (visité le 09/06/2019) (cf. p. 91, 92).

Backus, J. (1978). « Can Programming Be Liberated from the von Neumann Style ? A Functional Style and Its Algebra of Programs ». In : *Communications of the ACM* 21.8, p. 613-641. ISSN : 0001-0782. DOI : 10.1145/359576.359579. URL : https://dl.acm.org/doi/10.1145/359576.359579 (visité le 16/07/2023) (cf. p. 94).

Balacheff, N. (1987). « Processus de preuve et situations de validation ». In : *Educational Studies in Mathematics* 18.2 (2), p. 147-176. ISSN : 0013-1954, 1573-0816. DOI : 10.1007/BF00314724. URL : http://link.springer.com/10.1007/BF00314724 (visité le 18/04/2022) (cf. p. 271, 300).

Bardini, C., L. Radford et C. Sabena (2005). « Struggling with Variables, Parameters, and Indeterminate Objects or How to Go Insane in Mathematics ». In : (cf. p. 121).

Baudé, J. (2019). *Quelques Points de Repère Dans Une Histoire de 40 Ans : L'association EPI*. URL : https://www.epi.asso.fr/revue/histo/h11epi_jb.htm (visité le 10/09/2019) (cf. p. 97).

— (2023). *L'informatique Dans Les Écoles et Les Collèges*. URL : https://www.epi.asso.fr/revue/histo/h85_info-primaire-college_jb18.htm (visité le 14/08/2023) (cf. p. 98).

Bauer, F. L., H. Bottenbruch et al. (1958). « Proposal for a Universal Language for the Description of Computing Processes ». In : *Computer Programming and Artificial Intelligence, University of Michigan Summer School 1958*. Sous la dir. de J. Carr, p. 355-373. URL : https://www.softwarepreservation.org/projects/ALGOL/report/BauerBRS-Proposal_for_a_Universal_Language-1958.pdf/view (cf. p. 91, 92).

Bauer, F. L. et H. Wössner (1972). « The "Plankalkül" of Konrad Zuse : A Forerunner of Today's Programming Languages ». In : *Communications of the ACM* 15.7, p. 678-685. ISSN : 0001-0782. DOI : 10.1145/361454.361515. URL : https://dl.acm.org/doi/10.1145/361454.361515 (visité le 02/08/2023) (cf. p. 88-90).

Berry, G. (2020). « Les langages informatiques : de la pensée à l'exécution automatique ». In : *Langue et science, langage et pensée*. Colloque annuel du Collège de France. Paris : Odile Jacob, p. 71-90. ISBN : 978-2-7381-5016-5. DOI : 10.3917/oj.rober.2020.01.0071. URL : https://www.cairn.info/langue-et-science-langage-et-pensee--9782738150165-p-71.htm (visité le 14/08/2023) (cf. p. 92-94, 468).

Bosc-Miné, C. (2014). « Caractéristiques et fonctions des feed-back dans les apprentissages ». In : *LAnnee psychologique* Vol. 114.2 (2), p. 315-353. ISSN : 0003-5033. URL : https://www.cairn.info/revue-l-annee-psychologique1-2014-2-page-315.htm (visité le 10/06/2021) (cf. p. 177-180, 182).

Briant, N. (2013). « Étude Didactique de La Reprise de l'algèbre Par l'introduction de l'algorithmique Au Niveau de La Classe de Seconde Du Lycée Français ». Theses. Université Montpellier II - Sciences et Techniques du Languedoc. URL : https://tel.archives-ouvertes.fr/tel-00920506 (visité le 26/05/2017) (cf. p. 50, 109, 113, 114, 116).

Bronner, A. et H. Squalli (2021). « La Généralisation Dans La Pensée Algébrique ». In : *Revue québécoise de didactique des mathématiques* 3.0 (0). URL : https://rqdm.recherche.usherbrooke.ca/ojs/ojs-3.1.1-4/index.php/rqdm/article/view/32 (visité le 18/04/2022) (cf. p. 15, 58, 123, 132).

Brousseau, G. (1981). « Problèmes de didactique des décimaux - Revue RDM ». In : *Recherches en didactique des mathématiques* 2.1, p. 37-127. URL : https://revue-rdm.com/1981/problemes-de-didactique-des/ (visité le 07/09/2023) (cf. p. 358).

— (2011). « La théorie des situations didactiques en mathématiques ». In : *Éducation et didactique* 5.vol. 5, no. 1 (vol. 5, no. 1), p. 101-104. ISSN : 1956-3485. DOI : 10.4000/educationdidactique.1005. URL : http://journals.openedition.org/educationdidactique/1005 (visité le 22/04/2020) (cf. p. 23, 140).

Calmet, C., M. Hirtzig et D. Wilgenbus (2016). *1, 2, 3 ... codez ! : enseigner l'informatique à l'école at au collège (cycles 1, 2 et 3)*. Paris : Le Pommier. ISBN : 978-2-7465-1106-4 (cf. p. 25).

Cazalas, G. (1932). « Le Calcul de La Table Mathématique AO 6456 ». In : *Revue d'Assyriologie et d'archéologie orientale* 29.4 (4), p. 183-188. ISSN : 0373-6032. JSTOR : 23284034. URL : https://www.jstor.org/stable/23284034 (visité le 30/12/2020) (cf. p. 31).

Chabert, J.-L., éd. (2010). *Histoire d'algorithmes : du caillou à la puce*. Paris, France : Belin. 607 p. ISBN : 978-2-7011-5518-0 (cf. p. 21, 23, 28-31, 34, 66).

Chaigneau, P. (2019). « Otto Neugebauer, François Thureau-Dangin et l'édition des textes mathématiques cunéiformes dans les années 1930 ». Thèse de doct. Université Paris Cité. URL : https://theses.hal.science/tel-03450379 (visité le 10/07/2023) (cf. p. 30).

Chevallard, Y. (1989). « Le Passage de l'arithmétique à l'algébrique Dans l'enseignement Des Mathématiques Au Collège. Deuxième Partie : Perspectives Curriculaires : La Notion de Modélisation. » In : *Petit x* 19 (19), p. 45-75. URL : http://publimath.irem.univ-mrs.fr/biblio/IGR89002.htm (visité le 14/04/2020) (cf. p. 23, 58, 60, 104-107, 118, 148, 303).

Chiprianov, V., L. Coulange et G. Train (2018). « Enseigner l'informatique à l'École : à La Recherche d'une Raison d'être ». In : *Bulletin de la Société Informatique de France* 12, p. 15-35 (cf. p. 15, 99, 100).

Choquet, C. et al. (2023). « Le Travail Mathématique à l'aune Du Cadre de l'apprentissage Par Problématisation : Travail Mathématique et Processus de Problématisation Chez Les Élèves.Strasbourg, 27 Juin - 2 Juillet 2022 ». In : *Actes Du VIIe Symposium International d'étude Sur Le Travail Mathématique (ETM)*. Actes Du Septième Symposium d'Étude Sur Le Travail Mathématique. Sous la dir. de C. Derouet et al. Strasbourg : IREM de Strasbourg, p. 131-142 (cf. p. 138).

Christianidis, J. (2007). « The Way of Diophantus : Some Clarifications on Diophantus' Method of Solution ». In : *Historia Mathematica* 34.3 (3), p. 289-305. ISSN : 0315-0860. DOI : 10.1016/j.hm.2006.10.003. URL : https://www.sciencedirect.com/science/article/pii/S0315086006001285 (visité le 13/08/2021) (cf. p. 40, 42).

— (2018). « La Démarche de Diophante Démystifiée : L'algèbre Prémoderne Au Service de La Résolution de Problèmes ». In : *Actes du XIIe colloque Maghrébin sur l'histoire des mathématiques arabes (Marrakech 26-28 mai 2016) ed. A. Laabid. Marrakech : Ecole Normale Supérieure, Marrakech*, p. 41-69. URL : https://www.academia.edu/36426142/LA_D%C3%89MARCHE_DE_DIOPHANTE_D%C3%89MYSTIFI%C3%89E_LALG%C3%88BRE_PR%C3%89MODERNE_AU_SERVICE_DE_LA_R%C3%89SOLUTION_DE_PROBL%C3%88MES (visité le 13/08/2021) (cf. p. 40, 41, 56).

CNRTL (2023a). *HISTOIRE : Définition de HISTOIRE*. URL : https://www.cnrtl.fr/definition/histoire (visité le 28/08/2023) (cf. p. 156).

— (2023b). *VARIABLE : Définition de VARIABLE*. URL : https://www.cnrtl.fr/definition/variable (visité le 17/08/2023) (cf. p. 111).

Coppé, S. et B. Grugeon (2009). « Le calcul littéral au collège. Quelle articulation entre sens et technique ? » In : Colloque de la CORFEM. URL : https://halshs.archives-ouvertes.fr/halshs-00959612 (visité le 06/03/2020) (cf. p. 122, 123, 133, 141).

Descartes, R. (.-1. A. du texte (1897-1913). *Oeuvres de Descartes. Tome 6 / publiées par Charles Adam et Paul Tannery*. URL : https://gallica.bnf.fr/ark:/12148/bpt6k3411414j (visité le 10/09/2020) (cf. p. 61-63, 65, 105).

Descartes, R. (.-1. A. du texte et J.-B. (.-1. A. du texte Salgues (1824-1826). *Oeuvres de Descartes. [Volume 11] / publ. par Victor Cousin... ; [et précédées de l'éloge de René Descartes par Thomas]*. URL : https://gallica.bnf.fr/ark:/12148/bpt6k942726 (visité le 16/07/2023) (cf. p. 63, 105).

Diderot (1751-1765). *Encyclopédie ou Dictionnaire raisonné des sciences, des arts et des métiers. Tome premier, A-Azyme / par une société de gens de lettres ; mis en ordre et publié par M. [Denis] Diderot,... et quant à la partie mathématique, par M. [Jean Le Rond] d'Alembert,..* URL : https://gallica.bnf.fr/ark:/12148/bpt6k50533b (visité le 05/08/2023) (cf. p. 67).

Disjkstra, E. W. (1968). « Go-to Statement Considered Harmful ». In : *Communication of the ACM* 11.3, p. 147-148. URL : http://www.cs.utexas.edu/users/EWD/transcriptions/EWD02xx/EWD215.html (visité le 21/03/2018) (cf. p. 89).

Douady, R. (1984). « Jeux de cadres et dialectiques outil-objet dans l'enseignement des Mathématiques. Une réalisation dans tout le cursus primaire. » Thèse de doct. Université paris VII. URL : https://tel.archives-ouvertes.fr/tel-01250665/document (visité le 05/11/2017) (cf. p. 15, 97, 140, 403).

Doussot, S. et al. (2022). *Le Cadre de l'apprentissage Par Problématisation*. Presses Universitaires de Rennes. URL : https://hal.archives-ouvertes.fr/hal-03787216 (visité le 09/12/2022) (cf. p. 125, 137-139, 186, 196).

Dowek, G. (2011). « Les quatre concepts de l'informatique ». In : URL : https://edutice.archives-ouvertes.fr/edutice-00676169 (visité le 16/10/2019) (cf. p. 32, 33, 104, 468).

Dowek, G. et al. (2010). *Les ingrédients des algorithmes*. Interstices. URL : https://interstices.info/les-ingredients-des-algorithmes/ (visité le 11/07/2023) (cf. p. 34).

Drouhard, J.-P. (1992). « Les Écritures Symboliques de l'Algèbre Élémentaire ». In : DOI : 10.13140/2.1.2823.5043. URL : http://rgdoi.net/10.13140/2.1.2823.5043 (visité le 30/07/2022) (cf. p. 27).

Drouhard, J.-P. et M. Panizza (2012). « Hansel et Gretel et l'Implicite Sémio-Linguistique En Algèbre Élémentaire ». In : *Recherches en Didactique des Mathématiques* Numéro Spécial, p. 209-235 (cf. p. 150, 364, 435).

Duval, R. (1993). « Registres de Représentation Sémiotique et Fonctionnement Cognitif de La Pensée ». In : *Annales de Didactique et de Sciences Cognitives* 5 (5), p. 37-65. URL : https://numerisation.irem.univ-mrs.fr/ST/IST93004/IST93004.pdf (cf. p. 56, 133).

— (2002). « Comment décrire et analyser l'activité mathématique ? Cadres et registres ». In : Séminaire TECFA. Genève, p. 24. URL : http://tecfa.unige.ch/tecfa/teaching/staf26/Doua.pdf (cf. p. 45, 46, 104).

— (2006a). « A Cognitive Analysis of Problems of Comprehension in a Learning of Mathematics ». In : *Educational Studies in Mathematics* 61.1-2 (1-2), p. 103-131. ISSN : 0013-1954, 1573-0816. DOI : 10.1007/s10649-006-0400-z. URL : http://link.springer.com/10.1007/s10649-006-0400-z (visité le 14/07/2022) (cf. p. 365).

— (2006b). « Quelle Sémiotique Pour l'analyse de l'activité et Des Productions Mathématiques ? » In : *RELIME. Revista latinoamericana de investigación en matemática educativa, ISSN 1665-2436, Vol. 9, Nº. 1, 2006, pags. 45-82* 9 (cf. p. 47, 50).

ELAN (2020). Version 6.0. Nijmegen : Max Planck Institute for Psycholinguistics, The Language Archive. URL : https://archive.mpi.nl/tla/elan (cf. p. 275).

Ely, R. et A. Adams (2012). « Unknown, Placeholder, or Variable : What Is x ? » In : *Mathematics Education Research Journal* 24. DOI : 10.1007/s13394-011-0029-9 (cf. p. 106, 107, 109, 110, 120, 121).

Fabre, M. (2017). *Le sens du problème : Problématiser à l'école ?* Le point sur... Pédagogie. De Boeck. ISBN : 978-2-8041-9525-0 (cf. p. 138, 139, 151, 182).

— (2019). *Education et (Post) Vérité. L'épreuve Des Faits*. Hermann. URL : https://hal.archives-ouvertes.fr/hal-02345478 (visité le 15/01/2021) (cf. p. 183, 186, 189, 191).

Farès, N. (2015). « Al-Khwarizmi et Le Fondement Axiomatique de l'algèbre ». In : *Lebanese Science Journal* 16, p. 107-126. URL : https://hal.archives-ouvertes.fr/hal-01683590 (visité le 16/04/2020) (cf. p. 21, 51, 55).

— (2017a). « AL-KHWĀRIZMĪ Vie, Oeuvre et Livre Algébrique ». In : *Naissance et Développement de l'algèbre Dans La Tradition Mathématique Arabe, Dār al-Fārābī, 20017, Beyrouth*. URL : https://hal.archives-ouvertes.fr/hal-01722173 (visité le 05/06/2020) (cf. p. 48-50).

— (2017b). « Diophante et l'algèbre ». In : *Naissance et Développement de l'algèbre Dans La Tradition Mathématique Arabe, Dār al-Fārābī, 2017, Beyrouth*. URL : https://hal.archives-ouvertes.fr/hal-01741649 (visité le 22/05/2018) (cf. p. 21, 22, 39, 40).

Frege, G. (1994). *Ecrits logiques et philosophiques*. Collection points série essais 296. Paris : Seuil, 1994. 233 p. ISBN : 978-2-02-022966-1 (cf. p. 27).

Freudenthal, H. (2020). *Notation Mathématique*. In : Encyclopædia Universalis. URL : http://www.universalis-edu.com.budistant.univ-nantes.fr/encyclopedie/notation-mathematique/ (visité le 15/04/2020) (cf. p. 112).

Furinghetti, F. et D. Paola (1994). *1994 - Parameters, Unknowns and Variables : A Little Difference ?* T. 2, p. 375. 368 p. (cf. p. 120).

Giloi, W. (1997). « Konrad Zuse's Plankalku/Spl Uml/l : The First High-Level, "Non von Neumann" Programming Language ». In : *IEEE Annals of the History of Computing* 19.2, p. 17-24. ISSN : 1934-1547. DOI : 10.1109/85.586068 (cf. p. 88).

Grugeon-Allys, B. et J. Pilet (2017). « Quelles connaissances et quels raisonnements en arithmétique favorisent l'entrée dans l'algèbre ? » In : *Nouveaux cahiers de la recherche en éducation* 20.3 (3), p. 106-130. ISSN : 1911-8805. DOI : 10.7202/1055730ar. URL : https://www.erudit.org/en/journals/ncre/2014-v17-n1-ncre04255/1055730ar/ (visité le 14/04/2020) (cf. p. 123).

Guttag, J. V., E. Horowitz et D. R. Musser (1978). « Abstract Data Types and Software Validation ». In : 21.12 (cf. p. 16).

Hartley, R. V. L. (1928). « Transmission of Information ». In : *Bell System Technical Journal* 7.3, p. 535-563. ISSN : 00058580. DOI : 10.1002/j.1538-7305.1928.tb01236.x. URL : https://ieeexplore.ieee.org/document/6769394 (visité le 29/12/2022) (cf. p. 183, 185).

Harvey, B. et J. Mönig (2020). « Snap Reference Manual 8.0 ». In : URL : https://snap.berkeley.edu/snap/help/SnapManual.pdf (visité le 14/08/2023) (cf. p. 103).

Hattie, J. et H. Timperley (2007). « The Power of Feedback ». In : *Review of Educational Research* 77.1 (1), p. 81-112. ISSN : 0034-6543, 1935-1046. DOI : 10.3102/003465430298487. URL : http://journals.sagepub.com/doi/10.3102/003465430298487 (visité le 10/07/2022) (cf. p. 177, 179, 180).

Heeffer, A. et M. Van Dyck, éd. (2010). *Philosophical Aspects of Symbolic Reasoning in Early Modern Mathematics*. Studies in Logic 26. London : College Publications. 302 p. ISBN : 978-1-84890-017-2 (cf. p. 56).

Hersant, M. (2010). *Le Couple (Contrat Didactique, Milieu) et Les Conditions de La Rencontre Avec Le Savoir En Mathématiques : De l'analyse de Situations Ordinaires Au Développement de Situations Pour Les Classes Ordinaires*. ote de synthèse pour l'Habilitation à diriger des recherches. URL : http://magali.hersant.free.fr/articles/NS_HDR_Hersant.pdf (visité le 06/03/2020) (cf. p. 140).

— (2020). « L'usage Du Cadre de l'Apprentissage Par Problématisation En Didactique Des Mathématiques : Quels Apports Possibles ? » In : *Actes Du Séminaire National de Didactique Des Mathématiques*. Sous la dir. d'A. Chesnais et H. Sabra. ARDM, p. 204-220 (cf. p. 138).

— (2021). « Factualisation En Mathématiques. Une Étude à Partir de Polya « Les Mathématiques et Le Raisonnement Plausible » ». 18è Colloque Du Réseau Probléma18è Colloque Du Réseau Probléma (Distanciel) (cf. p. 455).

Hersant, M. et D. Orange Ravachol (2015). « Démarche d'investigation et Problématisation En Mathématiques et En SVT : Des Problèmes de Démarcation Aux Raisons d'une unionInquiry Based Learning in Mathematics and Earth Science Education : From Issues Dividing to Reasons for Union ». In : *Recherches en éducation*. DOI : 10.4000/ree.7533 (cf. p. 152).

Hoc, J.-M. (1981). « La Planification Dans La Résolution de Problème : L'apprentissage de La Programmation Informatique ». In : *Le Travail Humain* 44.2 (2), p. 261-267. ISSN : 0041-1868. JSTOR : 40657739. URL : https://www.jstor.org/stable/40657739 (visité le 24/12/2020) (cf. p. 32).

Houzel, C. (2015). *Introduction à l'histoire de l'algèbre, d' al-Khwârizmî à Descartes*. DOI : 10.13140/RG.2.1.3466.4400 (cf. p. 50).

Høyrup, J. (2001). « The Old Babylonian Square Texts BM 13901 and YBC 4714. Retranslation and Analysis ». In : *Changing Views on Ancient Near Eastern Mathematics* 19, p. 155-218 (cf. p. 33).

Høyrup, J. (2010). « Old Babylonian "Algebra", and What It Teaches Us about Possible Kinds of Mathematics ». In : *Ganita Bharati. Bulletin of the Indian Society for History of Mathematics* 32.1-2 (1-2), p. 87-110. ISSN : 0970-0307 (cf. p. 30-37).

— (2017). *Algebra in Cuneiform : Introduction to an Old Babylonian Geometrical Technique*. MPRL – Textbooks. Berlin : Max-Planck-Gesellschaft zur Förderung der Wissenschaften. ISBN : 978-3-945561-15-7. DOI : 10.34663/9783945561157-00. URL : https://mprl-series.mpg.de/textbooks/2/index.html (cf. p. 34).

— (2019). « On Old Babylonian Mathematical Terminology and Its Transformations in the Mathematics of Later Periods ». In : *GANITA BHARATI* 40, p. 53-99. DOI : 10.32381/GB.2018.40.01.3 (cf. p. 33, 34, 38).

— (2020). « The Babylonian Cellar Text BM 85200 + VAT 6599 Retranslation and Analysis ». In : p. 315-358. URL : https://www.academia.edu/31716181/The_Babylonian_Cellar_Text_BM_85200_VAT_6599_Retranslation_and_Analysis (visité le 27/12/2020) (cf. p. 28).

Juignet, P. (2015). *Les Paradigmes Scientifiques Selon Thomas Kuhn*. Philosophie, science et société. URL : https://philosciences.com/113 (visité le 01/05/2023) (cf. p. 252).

Jullien, V. (1996). *Descartes, La "Géométrie" de 1637*. Philosophies 76. Paris : Presses universitaires de France. 128 p. ISBN : 978-2-13-047829-4 (cf. p. 61).

Kieran, C. (2004). « Algebraic Thinking in the Early Grades : What Is It ». In : *The Mathematics Educator* 8, p. 139-151 (cf. p. 14).

Kieran, C. et al. (2016). *Early Algebra : Research into Its Nature, Its Learning, Its Teaching*. ICME-13 Topical Surveys. Cham : Springer International Publishing. ISBN : 978-3-319-32257-5 978-3-319-32258-2. DOI : 10.1007/978-3-319-32258-2. URL : http://link.springer.com/10.1007/978-3-319-32258-2 (visité le 21/11/2021) (cf. p. 14, 123).

Kleiner, I. (2007). *A History of Abstract Algebra*. Boston : Birkhäuser. 168 p. ISBN : 978-0-8176-4684-4 978-0-8176-4685-1 978-0-387-34050-0 (cf. p. 59, 60).

Knuth, D. E. et L. T. Pardo (1976). *The Early Development of Programming Languages*. Stanford University, Computer Science Department. URL : https://web.archive.org/web/20170912102014/http://bitsavers.org/pdf/stanford/cs_techReports/STAN-CS-76-562_EarlyDevelPgmgLang_Aug76.pdf (visité le 26/10/2020) (cf. p. 84, 87, 88, 90).

Komis, V., S. Touloupaki et G.-L. Baron (2017). « Analyse Cognitive et Didactique Du Langage de Programmation ScratchJr ». In : *L'informatique et Le Numérique Dans La Classe : Qui, Quoi, Comment ?* Sous la dir. de J. ( ) Henry, A. Nguyen et E. Vandeput. Informatique. Namur : Presses Universitaires de Namur, p. 109-121. ISBN : 978-2-87037-976-9 (cf. p. 100, 101, 424).

Küchemann, D. E. (1981). « In Hart K (Ed) 1981, Children's Understanding of Mathematics : 11-16. John Murray, London. » In : (cf. p. 109).

Kuhn, T. S., L. Meyer et J.-P. Luminet (2018). *La structure des révolutions scientifiques*. Champs. Paris : Flammarion. ISBN : 978-2-08-139601-2 (cf. p. 252).

Lagrange, J.-B. et J. Rogalski (2017). « Savoirs, Concepts et Situations Dans Les Premiers Apprentissages En Programmation et En Algorithmique. » In : *Annales de Didactiques et de Sciences Cognitives*. URL : https://hal.archives-ouvertes.fr/hal-01740442 (visité le 21/05/2018) (cf. p. 15, 16, 470).

Laguës, M., D. Beaudouin et G. Chapouthier (2017). *L'invention de La Mémoire : Écrire, Enregistrer, Numériser*. Paris : CNRS éditions. 383 p. ISBN : 978-2-271-08933-5 (cf. p. 24, 26, 66, 73).

Larousse, É. (2020a). *Définitions : variable - Dictionnaire de français Larousse*. URL : https://www.larousse.fr/dictionnaires/francais/variable/81109 (visité le 15/04/2020) (cf. p. 108).

— (2020b). *Encyclopédie Larousse en ligne - algèbre latin médiéval algebra de l'arabe al-djabr réduction*. URL : https://www.larousse.fr/encyclopedie/divers/alg%C3%A8bre/19868 (visité le 14/08/2020) (cf. p. 21).
— (2023). *Définitions : paramètre - Dictionnaire de français Larousse*. URL : https://www.larousse.fr/dictionnaires/francais/param%C3%A8tre/57952 (visité le 17/08/2023) (cf. p. 107).
Lazard, E. et P. Mounier-Kuhn (2022). *Histoire illustrée de l'informatique*. 3e éd. Les Ulis : EDP sciences. ISBN : 978-2-7598-2704-6 (cf. p. 66, 69, 72, 73, 75, 87).
Le Site de l'équipe Compréhension, Raisonnement et Acquisition de Connaissances - Christelle Bosc-Miné (2023). Laboratoire Paragraphe. URL : http://paragraphe.crac.free.fr/articles.php?lng=fr&pg=112 (visité le 04/04/2023) (cf. p. 177).
Legrand, J.-M. (2019). « Captation Automatisée et Visualisation de Traces de Programmation Dans Un Environnement de Programmation Graphique Par Blocs ». RJC EIAH (Paris, France). URL : https://hal.archives-ouvertes.fr/hal-02615243 (visité le 05/04/2021) (cf. p. 100, 103, 155, 159).
— (2023). « Algorithmique et Entrée Dans l'Algèbre Élémentaire : La (Difficile) Construction Du Concept de « Paramètre » Strasbourg, 27 Juin - 2 Juillet 2022 ». In : *Actes Du VIIe Symposium International d'étude Sur Le Travail Mathématique (ETM)*. Septième Symposium d'Étude Sur Le Travail Mathématique. Sous la dir. de C. Derouet et al. Strasbourg : IREM de Strasbourg, p. 203-214 (cf. p. 119).
León, N. (2019). « An Epistemological Study of Recursion and Mathematical Induction in Mathematics and Computer Science ». In : *Eleventh Congress of the European Society for Research in Mathematics Education*. Sous la dir. d'U. T. Jankvist, M. van den Heuvel-Panhuizen et M. Veldhuis. T. TWG01. Proceedings of the Eleventh Congress of the European Society for Research in Mathematics Education (CERME11) 26. Utrecht, Netherlands : Freudenthal Group. URL : https://hal.science/hal-02398465 (visité le 16/08/2023) (cf. p. 103).
León, N. et S. Modeste (2020). « Récurrence et Récursivité à l'interface Des Mathématiques et de l'informatique ». In : *Repères IREM* 119, p. 45-63. URL : https://hal.science/hal-04161059 (visité le 16/08/2023) (cf. p. 103).
Lescanne, P. (2009). *La pensée informatique*. Interstices. URL : https://interstices.info/la-pensee-informatique/ (visité le 14/07/2023) (cf. p. 53).
Lévénez, É. (2023). *Computer Languages History*. URL : https://www.levenez.com/lang/ (visité le 02/08/2023) (cf. p. 93, 96).
Longo, G. et J. Lassègue (2020). « Actualité de Turing : Entre Captation d'héritage et Ressource Pour l'avenir ». In : *Intellectica - La revue de l'Association pour la Recherche sur les sciences de la Cognition (ARCo)*. URL : https://hal.science/hal-03065372 (visité le 14/08/2023) (cf. p. 94).
Macbeth, D. (2004). « Viète, Descartes, and the Emergence of Modern Mathematics ». In : *Graduate Faculty Philosophy Journal* 25.2 (2), p. 87-117. ISSN : 0093-4240. URL : https://www.academia.edu/790922/Vi%C3%A8te_Descartes_and_the_emergence_of_modern_mathematics (visité le 06/07/2020) (cf. p. 60-64).
Malisani, E. et F. Spagnolo (2009). « From Arithmetical Thought to Algebraic Thought : The Role of the "Variable" ». In : *Educational Studies in Mathematics* 71.1, p. 19-41. ISSN : 0013-1954. JSTOR : 40284583. URL : https://www.jstor.org/stable/40284583 (visité le 15/07/2023) (cf. p. 65, 108, 110).

Menabrea, L. F. (1842). *Sketch of the Analytical Engine Invented by Charles Babbage ... with Notes by the Translator*. Trad. par A. A. K. Lovelace. Bibliothèque Universelle de Genève 82. R. & J. E. Taylor. 124 p. Google Books : EIVqHUm9WlkC (cf. p. 73, 76, 77, 79-81, 83-86, 104).

Merri, M. (2007). « Réponse de Gérard Vergnaud ». In : *Activité Humaine et Conceptualisation : Questions à Gérard Vergnaud*. Questions d'éducation. Toulouse : Presses universitaires du Midi, p. 341-357. ISBN : 978-2-8107-1019-5. URL : http://books.openedition.org/pumi/6086 (visité le 23/07/2022) (cf. p. 174).

Ministère de l'éducation nationale et Commission de réflexion sur l'enseignement des mathématiques (2002). *L'enseignement des sciences mathématiques : rapport au Ministre de l'Éducation nationale*. Sous la dir. de J.-P. Kahane. Paris, France : Centre national de documentation pédagogique : O. Jacob, DL 2002. 284 p. ISBN : 978-2-240-00832-9 978-2-7381-1138-8 (cf. p. 14, 15).

Modeste, S. (2012). « Enseigner l'algorithme pour quoi ? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ? » Thèse de doct. Université de Grenoble. URL : https://tel.archives-ouvertes.fr/tel-00783294/document (visité le 26/05/2017) (cf. p. 23, 25, 26, 52, 66).

Moss, D. L., J. A. Czocher et T. Lamberg (2018). « Frustration with Understanding Variables Is Natural ». In : *Mathematics Teaching in the Middle School* 24.1, p. 10-17. ISSN : 1072-0839. DOI : 10.5951/mathteacmiddscho.24.1.0010. JSTOR : 10.5951/mathteacmiddscho.24.1.0010. URL : https://www.jstor.org/stable/10.5951/mathteacmiddscho.24.1.0010 (visité le 15/07/2023) (cf. p. 110, 111).

Oaks, J. A. (2018). « François Viète's Revolution in Algebra ». In : *Archive for History of Exact Sciences* 72.3 (3), p. 245-302. ISSN : 0003-9519, 1432-0657. DOI : 10.1007/s00407-018-0208-0. URL : http://link.springer.com/10.1007/s00407-018-0208-0 (visité le 14/06/2020) (cf. p. 57, 59, 60).

Orange, C. (2005). « Problématisation et conceptualisation en sciences et dans les apprentissages scientifiques ». In : *Les Sciences de l'education - Pour l'Ere nouvelle* Vol. 38.3 (3), p. 69-94. ISSN : 0755-9593. URL : https://www.cairn.info/revue-les-sciences-de-l-education-pour-l-ere-nouvelle-2005-3-page-69.htm (visité le 27/05/2019) (cf. p. 193, 203).

— (2010). « Situations Forcées, Recherches Didactiques et Développement Du Métier Enseignant ». In : *Recherches en éducation* (HS2). ISSN : 1954-3077. DOI : 10.4000/ree.8864. URL : http://journals.openedition.org/ree/8864 (visité le 26/08/2023) (cf. p. 137).

— (2012). *Enseigner Les Sciences ; Problèmes, Débats et Savoirs Scientifiques En Classe*. Le Point Sur... Pédagogie. de Boeck (cf. p. 138, 139, 224, 244, 268, 327, 441, 447, 470).

Pascal, B. (1665). « Des caractères de divisibilité des nombres déduits de la somme de leurs chiffres ». In : Pascal, B., H. Gouhier et L. Lafuma. *Œuvres complètes*. Nachdr. L'intégrale. Paris : Éd. du Seuil, p. 84-86. ISBN : 978-2-02-000713-9 (cf. p. 66).

— (2016a). « A Monseigneur Le Chancelier ». In : *Blaise Pascal : OEuvres complètes*. Arvensa Edition, p. 1000-1002. ISBN : 978-2-36841-917-5 (cf. p. 68).

— (2016b). « Avis nécessaire à tous ceux qui auront la cuiosité de voir la machine arithmétique, et de s'en servir ». In : *Blaise Pascal : OEuvres complètes*. Arvensa Edition, p. 1003-1009. ISBN : 978-2-36841-917-5 (cf. p. 68, 69).

— (2016c). *Blaise Pascal : OEuvres complètes*. Arvensa Edition. ISBN : 978-2-36841-917-5 (cf. p. 468).

Peirce, C. S., N. Houser et C. J. W. Kloesel (1992). *The Essential Peirce : Selected Philosophical Writings*. Avec la coll. de Peirce Edition Project. T. 1. Bloomington : Indiana University Press. 2 p. ISBN : 978-0-253-32849-6 978-0-253-20721-0 978-0-253-33397-1 978-0-253-21190-3 (cf. p. 131, 171, 183, 189).

Peirce, C.-S. (2000). « Comment se fixe la croyance ». In : *Agone* 23 (23), p. 89-107. ISSN : 1157-6790, 1953-7999. DOI : 10.4000/revueagone.803. URL : http://revueagone.revues.org/803 (visité le 11/07/2022) (cf. p. 171, 172, 175, 470).

Perrin-Glorian, M.-J. et P. M. B. Bellemain (2019). « L'ingénierie didactique entre recherche et ressource pour l'enseignement et la formation des maitres ». In : 9.1 (1), p. 38 (cf. p. 137).

Philippe Roi, T. G. (2013). « Analogie Entre La Comptabilité et Le Système Olfactif (Analogy between Accounting and the Olfactory System) ». In : 69-82 et sa bibliographie (cf. p. 26).

Piaget, J. et R. García (1983). *Psychogenèse et Histoire Des Sciences*. Nouvelle Bibliothèque Scientifique. Paris : Flammarion. 310 p. ISBN : 978-2-08-211137-9 (cf. p. 146, 176, 203, 235, 269, 307).

Piguet, C. et H. Hügli (2004). *Du zéro à l'ordinateur : Une brève histoire du calcul*. ISBN : 978-2-88074-469-4. URL : http://sbiproxy.uqac.ca/login?url=https://international.scholarvox.com/book/88838178 (visité le 27/10/2020) (cf. p. 66, 67, 73, 75).

Pixees (2014). *Algorithme*. Class´Code et ses Pixees. URL : https://pixees.fr/algorithme/ (visité le 21/09/2023) (cf. p. 468).

Pólya, G. (2008). *Les mathématiques et le raisonnement plausible*. Reproduction en fac-similé. Paris : J. Gabay. ISBN : 978-2-87647-294-5 (cf. p. 455).

Proust, C. (2006). *Brève chronologie de l'histoire des mathématiques en Mésopotamie*. URL : http://cm2.ens.fr/content/breve-chronologie-de-lhistoire-des-mathematiques-en-mesopotamie-2095 (visité le 11/07/2023) (cf. p. 31).

Rabardel, P. (1995). « Les hommes et les technologies ; approche cognitive des instruments contemporains ». In : Armand Collin, p. 195 (cf. p. 158).

Radford, L. (1991). « Diophante et l'algèbre Pré-Symbolique ». In : *Bulletin AMQ* (cf. p. 37, 38, 41, 44, 45, 47, 106).

— (2003). « Gestures, Speech, and the Sprouting of Signs : A Semiotic-Cultural Approach to Students' Types of Generalization ». In : *Mathematical Thinking and Learning* 5, p. 37-70. DOI : 10.1207/S15327833MTL0501_02 (cf. p. 301, 349).

— (2006a). « Algebraic Thinking and the Generalization of Patterns : A Semiotic Perspective ». In : 1 (cf. p. 123, 132, 224, 253).

— (2006b). « The Cultural-Epistomological Conditions of the Emergence of Algebraic Symbolism ». In : *In F. Furringhetti, S. Kaijser &amp ; C. Tzanakis, Proceedings of the 2004 History and Pedagogy of Mathematics Conference &amp ; ESU4, Uppsala, Sweden, pp. 509-524 (Plenary Lecture)*. URL : https://www.academia.edu/34611951/The_Cultural_Epistomological_Conditions_of_the_Emergence_of_Algebraic_Symbolism (visité le 08/04/2022) (cf. p. 121, 122).

— (2008). « Iconicity and Contraction : A Semiotic Investigation of Forms of Algebraic Generalizations of Patterns in Different Contexts ». In : *ZDM* 40.1 (1), p. 83-96. ISSN : 1863-9690, 1863-9704. DOI : 10.1007/s11858-007-0061-0. URL : http://link.springer.com/10.1007/s11858-007-0061-0 (visité le 04/01/2022) (cf. p. 131, 132, 204, 205, 223, 224, 254, 299, 389, 456).

— (2014). « The Progressive Development of Early Embodied Algebraic Thinking ». In : *Mathematics Education Research Journal* 26.2 (2), p. 257-277. ISSN : 1033-2170, 2211-050X. DOI : 10.1007/s13394-013-0087-2. URL : http://link.springer.com/10.1007/s13394-013-0087-2 (visité le 14/08/2021) (cf. p. 47, 123).

Radford, L. (2018). « The Emergence of Symbolic Algebraic Thinking in Primary School ». In : *Teaching and Learning Algebraic Thinking with 5- to 12-Year-Olds*. Sous la dir. de C. Kieran. Cham : Springer International Publishing, p. 3-25. ISBN : 978-3-319-68350-8 978-3-319-68351-5. DOI : 10.1007/978-3-319-68351-5_1. URL : http://link.springer.com/10.1007/978-3-319-68351-5_1 (visité le 29/08/2022) (cf. p. 469).

Randell, B. (1976). « The History of Digital Computers ». In : *Bulletin of the Institute of Mathematics and its Applications* 12.11-12, p. 335-346. ISSN : 0950-5628 (cf. p. 87).

Rashed, R. (2007). *Le commencement de l'algèbre*. Collection sciences dans l'histoire. Paris : Blanchard. 386 p. ISBN : 978-2-85367-241-2 (cf. p. 22, 48-55, 106).

Rashed, R. et C. Houzel (2013). *Les "Arithmétiques" de Diophante : Lecture historique et mathématique*. Walter de Gruyter. 640 p. ISBN : 978-3-11-033648-1. Google Books : 0d3mBQAAQBAJ (cf. p. 39, 40).

Recherche (MENSR), M. de l'Éducation nationale de l'Enseignement supérieur et de la (2015). *Programmes d'enseignement du cycle des apprentissages fondamentaux (cycle 2), du cycle de consolidation (cycle 3) et du cycle des approfondissements (cycle 4)*. URL : https://www.education.gouv.fr/bo/15/Special11/MENE1526483A.htm (visité le 14/08/2023) (cf. p. 109).

— (2016). *Document d'accompagnement : Mathématiques Cycle 4, Algorithmique et Programmation*. URL : https://eduscol.education.fr/document/17311/download (cf. p. 100).

Resta-Schweitzer, M. (2011). « Initiation Scientifique et Développement Intellectuel Du Jeune Enfant ». Université d'Angers (cf. p. 146, 147).

Ricœur, P. (2000). *La Mémoire, l'histoire, l'oubli*. L'ordre Philosophique. Paris : Seuil. 675 p. ISBN : 978-2-02-034917-8 (cf. p. 186).

— (2020). « Événement et sens ». In : *L'événement en perspective*. Sous la dir. de J.-L. Petit. Raisons pratiques. Paris : Éditions de l'École des hautes études en sciences sociales, p. 41-56. ISBN : 978-2-7132-3080-6. DOI : 10.4000/books.editionsehess.9600. URL : http://books.openedition.org/editionsehess/9600 (visité le 13/04/2023) (cf. p. 186).

Ricquebourg, F. (2008). « Archéologie de l'informatique ». thesis. Lyon 3. URL : http://www.theses.fr/2008LYO31002 (visité le 15/04/2019) (cf. p. 66, 73-76, 79, 88, 90, 93, 468).

Robert, L. (2023a). *histoire - Définitions, synonymes, conjugaison, exemples | Dico en ligne Le Robert*. URL : https://dictionnaire.lerobert.com/definition/histoire (visité le 28/08/2023) (cf. p. 156).

— (2023b). *paramètre - Définitions, synonymes, conjugaison, exemples | Dico en ligne Le Robert*. In : URL : https://dictionnaire.lerobert.com/definition/parametre (visité le 17/08/2023) (cf. p. 107).

— (2023c). *variable - Définitions, synonymes, conjugaison, exemples | Dico en ligne Le Robert*. URL : https://dictionnaire.lerobert.com/definition/variable (visité le 17/08/2023) (cf. p. 108).

Rojas, R. et al. (2000). *Plankalkül : The First High-Level Programming Language and Its Implementation*. Technical report B-3/2000. Freie Universität Belin : Department of Mathematics and Computer Science (cf. p. 88-90).

Rutishauser, H. (1951). « Über Automatische Rechenplanfertigung Bei Programmgesteuerten Rechenmaschinen ». In : *Zeitschrift für Angewandte Mathematik und Mechanik (Journal of Applied Mathematics and Mechanics)* 31. URL : http://www.softwarepreservation.org/projects/ALGOL/standards/#ALGOL_58_report (visité le 15/04/2019) (cf. p. 25-27, 90, 91).

Sáez-López, J.-M., M. Román-González et E. Vázquez-Cano (2016). « Visual Programming Languages Integrated across the Curriculum in Elementary School : A Two Year Case Study Using “Scratch” in Five Schools ». In : *Computers & Education* 97, p. 129-141. ISSN : 03601315. DOI : 10.1016/j.compedu.2016.03.003. URL : https://linkinghub.elsevier.com/retrieve/pii/S0360131516300549 (visité le 19/04/2022) (cf. p. 101, 102).

Sajaniemi, J. (2002). « An Empirical Analysis of Roles of Variables in Novice-Level Procedural Programs ». In : *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments*. IEEE 2002 Symposia on Human Centric Computing Languages and Environments. Arlington, VA, USA : IEEE Comput. Soc, p. 37-39. ISBN : 978-0-7695-1644-8. DOI : 10.1109/HCC.2002.1046340. URL : http://ieeexplore.ieee.org/document/1046340/ (visité le 28/03/2019) (cf. p. 81, 117).

Sajaniemi, J. (2005). « Roles of Variables and Learning to Program ». In : *Proceedings of the 3rd Panhellenic Conference "Didactics of Informatics"*. Korinthos, Greece (cf. p. 28, 35, 46, 70, 71, 117, 118).

Samurçay, R. (1985). « Signification et Fonctionnement Du Concept de Variable Informatique Chez Des Élèves Debutants (The Concept of Variable in Programming : Its Meaning and Use in Problem-Solving) ». In : *Educational Studies in Mathematics* 16.2, p. 143-161. ISSN : 0013-1954. JSTOR : 3482343. URL : https://www.jstor.org/stable/3482343 (visité le 18/08/2023) (cf. p. 114, 115, 117).

Samurçay, R. et A. Rouchier (1985). « De «faire» à «faire faire» : planification d’actions dans la situation de programmation ». In : *Enfance* 38.2 (2), p. 241-254. ISSN : 0013-7545. DOI : 10.3406/enfan.1985.2883. URL : https://www.persee.fr/doc/enfan_0013-7545_1985_num_38_2_2883 (visité le 23/04/2020) (cf. p. 30-33, 57, 87, 104, 124, 153, 271, 384, 468).

Schoenfeld, A. H. et A. Arcavi (1988). « On the Meaning of Variable ». In : *The Mathematics Teacher* 81.6 (6), p. 420-427. ISSN : 0025-5769, 2330-0582. DOI : 10.5951/MT.81.6.0420. URL : https://pubs.nctm.org/view/journals/mt/81/6/article-p420.xml (visité le 19/05/2021) (cf. p. 108, 122).

Serfati, M. (1997). « La constitution de l’écriture symbolique mathématique. » Thèse de doct. Université Paris I. URL : https://tel.archives-ouvertes.fr/tel-01252590 (visité le 15/06/2020) (cf. p. 37, 38, 44, 59, 60, 65, 107, 121, 122, 386).

— (1998). « Descartes et la constitution de l’écriture symbolique mathématique/Descartes and the establishment of symbolic mathematical writing ». In : *Revue d’histoire des sciences* 51.2, p. 237-290. DOI : 10.3406/rhs.1998.1323. URL : https://www.persee.fr/doc/rhs_0151-4105_1998_num_51_2_1323 (visité le 12/07/2023) (cf. p. 56, 57, 59, 61, 63-65).

Sfard, A. (1995). « The Development of Algebra : Confronting Historical and Psychological Perspectives ». In : *The Journal of Mathematical Behavior* 14.1 (1), p. 15-39. ISSN : 07323123. DOI : 10.1016/0732-3123(95)90022-5. URL : https://linkinghub.elsevier.com/retrieve/pii/0732312395900225 (visité le 06/03/2022) (cf. p. 463).

Shannon, C. (1948). « A Mathematical Theory of Communication ». In : *Bell System Technical Journal* 27.3, p. 379-423. ISSN : 00058580. DOI : 10.1002/j.1538-7305.1948.tb01338.x. URL : https://ieeexplore.ieee.org/document/6773024 (visité le 29/12/2022) (cf. p. 183, 185, 454).

Shannon, C. et W. Weaver (1964). « The Mathematical Theory of Communication ». In : p. 131 (cf. p. 183-185, 444, 454).

Squalli, H. et A. Bronner (2017). « Le développement de la pensée algébrique avant l’introduction du langage algébrique conventionnel ». In : *Nouveaux cahiers de la recherche en éducation* 20.3, p. 1-8. ISSN : 1911-8805. DOI : 10.7202/1055725ar. URL : https://www.erudit.org/fr/revues/ncre/2017-v20-n3-ncre04255/1055725ar/ (visité le 19/09/2023) (cf. p. 14).

Stacey, K. (1989). « Finding and Using Patterns in Linear Generalising Problems ». In : *Educational Studies in Mathematics* 20.2, p. 147-164. ISSN : 1573-0816. DOI : 10.1007/BF00579460. URL : https://doi.org/10.1007/BF00579460 (visité le 07/07/2023) (cf. p. 358).

Variable (2019). In : *Wikipédia*. URL : https://fr.wikipedia.org/w/index.php?title=Variable&oldid=162222557 (visité le 15/04/2020) (cf. p. 25, 108).

Variable (informatique) (2020). In : *Wikipédia*. URL : https://fr.wikipedia.org/w/index.php?title=Variable_(informatique)&oldid=168856866 (visité le 15/04/2020) (cf. p. 25).

Variable (mathématiques) (2020). In : *Wikipédia*. URL : https://fr.wikipedia.org/w/index.php?title=Variable_(math%C3%A9matiques)&oldid=166032092 (visité le 15/04/2020) (cf. p. 112).

Vergnaud, G. (1991). « Langage et pensée dans l'apprentissage des mathématiques ». In : *Revue française de pédagogie* 96.1 (1), p. 79-86. DOI : 10.3406/rfp.1991.1350. URL : https://www.persee.fr/doc/rfp_0556-7807_1991_num_96_1_1350 (visité le 22/07/2022) (cf. p. 174).

— (2007). « Représentation et activité : deux concepts étroitement associés ». In : *Recherches en éducation* 4 (4). ISSN : 1954-3077. DOI : 10.4000/ree.3889. URL : https://journals.openedition.org/ree/3889 (visité le 10/11/2021) (cf. p. 172, 174, 175).

— (2011). « La pensée est un geste Comment analyser la forme opératoire de la connaissance : » in : *Enfance* N° 1.1 (1), p. 37-48. ISSN : 0013-7545. DOI : 10.3917/enf1.111.0037. URL : https://www.cairn.info/revue-enfance2-2011-1-page-37.htm?ref=doi (visité le 21/02/2022) (cf. p. 131, 171, 174-176, 182, 196).

Vergnaud, G. et D. Chartier (1994). *Apprentissages et didactiques, où en est-on ?* Paris : Hachette Éducation. ISBN : 978-2-01-170348-4 (cf. p. 171, 172).

Verley, J.-L. (2020). *Algèbre*. In : Encyclopædia Universalis. URL : http://www.universalis-edu.com.budistant.univ-nantes.fr/encyclopedie/algebre/ (visité le 15/08/2020) (cf. p. 21).

Vitrac, B. (2005). « Peut-on Parler d'algèbre Dans Les Mathématiques Grecques Anciennes ? » In : *Mirror of Heritage (Ayene-ne Miras)* 3, pp. 1-44. URL : https://hal.archives-ouvertes.fr/hal-00174933 (visité le 08/06/2020) (cf. p. 22, 40, 42-45, 271, 385).

Wagner, S. (1983). « What Are These Things Called Variables ? » In : *The Mathematics Teacher* 76.7, p. 474-479. ISSN : 0025-5769. JSTOR : 27963648. URL : https://www.jstor.org/stable/27963648 (visité le 15/07/2023) (cf. p. 110).

Wilson, A. et D. C. Moffat (2010). « Evaluating Scratch to Introduce Younger Schoolchildren to Programming ». In : *PPIG* 1.1, p. 1-12. URL : https://scholar.google.com/scholar_lookup?title=Evaluating%20Scratch%20to%20introduce%20younger%20school%20children%20to%20programming&author=A.%20Wilson&publication_year=2010 (visité le 11/08/2023) (cf. p. 103).

Wing, J. (2011). « Research Notebook : Computational Thinking - What and Why ? » In : *The Link* 3, p. 20-23. URL : https://www.cs.cmu.edu/sites/default/files/11-399_The_Link_Newsletter-3.pdf (visité le 20/09/2021) (cf. p. 76).

Wing, J. (2006). « Computational Thinking ». In : *Communications of the ACM* 49.3. URL : http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf (visité le 14/07/2023) (cf. p. 53, 66, 75, 76).

Zazkis, R. et P. Liljedahl (2002a). « Arithmetic Sequence as a Bridge between Conceptual Fields ». In : *Canadian Journal of Science, Mathematics and Technology Education* 2, p. 91-118. DOI : 10.1080/14926150209556501 (cf. p. 358).

— (2002b). « Generalization of Patterns : The Tension between Algebraic Thinking and Algebraic Notation ». In : *Springer* 49.3, p. 379-402. JSTOR : 3483038. URL : https://www.jstor.org/stable/3483038 (cf. p. 204, 358).

COLLEGE DOCTORAL PAYS DE LA LOIRE
ED EDUCATION, COGNITION, LANGAGES, INTERACTIONS, SANTE
Nantes Université

Titre : Construction d'un concept à l'interface entre informatique et algèbre élémentaire au collège : la variable dans un rôle de paramètre

Mots clés : didactique, informatique, algèbre, problématisation, variable, algorithme

Résumé : L'algorithmique et l'algèbre, et plus généralement l'informatique et l'algèbre, sont réputés être des cadres affichant une certaine proximité. Une étude épistémologique des concepts à l'interface entre ces cadres nous permet de souligner l'importance de la variable dans un rôle de paramètre, nécessaire pour la généralisation, mais aussi obstacle épistémologique potentiel. Dans ce rôle, la traditionnelle opposition entre variable informatique et variable mathématique est dépassée au profit d'une équivalence sémiotique. Afin d'identifier les conditions de la construction par les élèves de la nécessaire existence du paramètre et de sa représentation dans un certain registre sémiotique (algébrique ou informatique), nous étudions une situation de généralisation de motif informatisée, censée permettre la construction de ce concept.

Pour analyser la dynamique de l'activité des élèves avec une granularité fine, nous avons conçu un dispositif de captation automatisée des actions de programmation des élèves dans l'environnement de programmation graphique par blocs mobilisé pour la situation étudiée (Snap!). L'analyse dans le Cadre de l'Apprentissage par Problématisation, qui s'appuie sur les traces de programmation de deux classes de 4ème, montre la nécessité de faire vivre aux élèves la tension fixe-variable du paramètre, entre l'unicité du symbole et la multiplicité de ses dénotés possibles. En outre, les faits construits par les élèves pour poser et construire le problème de la généralisation, sont éminemment liés aux sous-problèmes qu'ils se posent, et cela amène à identifier l'intérêt potentiel des caricatures dans les situations d'apprentissage de concepts informatiques.

Title : Construction of a concept at the interface between computer science and elementary algebra in junior high school: the variable in a parameter role

Keywords : didactics, computer science, algebra, problematisation, variable, algorithm

Abstract : Algorithms and algebra, and more generally computer science and algebra, are reputed to be frameworks that display a certain proximity. An epistemological study of the concepts at the interface between these frameworks allows us to emphasize the importance of the variable in the role of parameter, necessary for generalization, but also a potential epistemological obstacle. In this role, the traditional opposition between computer variable and mathematical variable is overcome in favour of a semiotic equivalence. In order to identify the conditions under which students construct the necessary existence of the parameter and its representation in a certain semiotic register (algebraic or computerised), we study a situation involving the computerised generalisation of a pattern, which is supposed to enable the construction of this concept.

In order to analyse the dynamics of the students' activity at a fine level of granularity, we designed a computerised device to capture the students' programming actions in the block-based graphical programming environment used for the situation under study (Snap!). Analysis within the Problem-Based Learning framework, based on the programming traces of two 4th year classes, shows the need for pupils to experience the fixed-variable tension of the parameter, between the uniqueness of the symbol and the multiplicity of its possible denotations. Furthermore, the facts constructed by the pupils to pose and construct the problem of generalisation are eminently linked to the sub-problems they pose themselves, and this leads us to identify the potential interest of caricatures in situations for learning computer science concepts.