Les abstractions informatiques peuvent-elles concrétiser les mathématiques?

Emmanuel Beffara

LES ABSTRACTIONS INFORMATIQUES PEUVENT-ELLES CONCRETISER LES MATHEMATIQUES ?

Emmanuel **BEFFARA**
I2M, Université d'Aix-Marseille
emmanuel.beffara@univ-amu.fr

Résumé

Cette présentation propose une réflexion sur le rôle que peut jouer l'informatique dans le rapport aux objets mathématiques que les élèves développent. Depuis longtemps, l'outil informatique permet d'expérimenter avec les objets et notions mathématiques, que ce soit dans le champ de la statistique, de la géométrie ou de l'étude des fonctions, de façon souvent plus concrète que lors d'activités classiques sur papier. Cela peut permettre de développer des intuitions pour aider à comprendre les objets, à condition que l'outil soit compris pour ce qu'il est et ne soit pas perçu comme magique. La science informatique est alors nécessaire pour parvenir à une compréhension suffisante de tels outils. Ce faisant, elle fait découvrir de nouvelles notions et façons de penser, comme la représentation de l'information et l'algorithmique, qui portent leur lot d'abstractions propres. On peut alors se demander si l'irruption de l'informatique est source de plus de concret ou de plus d'abstrait, et si elle peut donner, plus largement, un rapport nouveau aux objets mathématiques.

Mots clefs

Informatique, Calcul, Abstraction, Intuition, Démonstration

I. INTRODUCTION

Les réflexions qui suivent sont celles d'un informaticien théoricien sans expertise sérieuse en didactique, mais leur but est néanmoins de nourrir la réflexion didactique. Mon domaine de recherche est la théorie de la démonstration et la sémantique des langages de programmation, j'ai donc un pied en mathématiques et un pied en informatique fondamentale. Mon expérience d'enseignement se situe en informatique et en mathématiques à l'université, et également en formation initiale et continue d'enseignants pour l'informatique dans le secondaire. Ma préoccupation ici est orientée par la vision particulière que me donne ce parcours : il s'agit d'étudier ce que peut dire l'interaction entre logique mathématique et informatique dans le contexte de l'enseignement.

II. L'INSTRUMENTATION

L'outil informatique n'est en premier lieu qu'un épisode dans la longue histoire de l'instrumentation en mathématiques. En effet, l'emploi d'instruments est aussi vieux que les mathématiques elles-mêmes : bouliers et autres abaques ont été employés depuis l'Antiquité comme moyens de poser les calculs sur les grands nombres, quand la tête ou les doigts ne suffisent plus ; la géométrie classique s'est construite sur l'emploi de la règle et du compas, et de façon plus marginale sur d'autres instruments permettant la construction de figures avec un certaine précision ; de nombreux instruments de mesure ont été développés au fil des siècles et des savoirs mathématiques se sont développés simultanément, afin de concevoir, utiliser et comprendre de tels outils.
Les instruments ont donc joué des rôles très différents, tantôt comme aides au raisonnement exact, dans le cas des abaques, tantôt comme intermédiaires entre le monde physique et les notions abstraites, dans le cas de la mesure des grandeurs que l'on qualifierait aujourd'hui de *continues*, tantôt comme moyen d'expérimenter les objets par la représentation, comme dans le cas des constructions géométriques.
L'informatique est née de ce besoin d'instrumentation, comme une amélioration technologique sur les précédents dispositifs. Il s'agit à l'origine de faire des calculs de façon plus rapide et plus fiable, en automatisant de plus en plus les manipulations, dans le but de s'attaquer à des problèmes demandant de traiter beaucoup de données : calculs numériques, simulations physiques pour l'ingénierie, visualisation de données, puis le calcul formel, etc.

1. L'outil informatique dans l'enseignement

Ainsi, les instruments font partie de la pratique mathématique, il est donc naturel qu'ils aient leur place dans l'enseignement. Il s'agit autant d'apprendre à utiliser les outils de façon pertinente que de s'en servir pour appréhender les notions abstraites, dont certaines sont précisément nées de l'emploi de ces instruments.
L'emploi de calculatrices et de logiciels dans des champs variés comme le calcul numérique, le tracé de courbes, la géométrie dynamique ou encore le calcul formel est omniprésent depuis plusieurs décennies et il est abondamment documenté. Les usages possibles sont nombreux et les énumérer n'est pas le propos ici.

2. Automatismes et automatisation

Néanmoins, les usages permettent de mettre en évidence les étapes successives de l'instrumentation du calcul :

1. Découverte de techniques de calcul,
2. Systématisation de la technique,
3. Développement d'automatismes et d'intuitions associées,
4. Transfert de la tâche automatique vers la machine.

On peut illustrer ce cheminement avec différents phénomènes calculatoires. Pour les opérations élémentaires sur les nombres entiers, les techniques de calcul se sont développées en même temps que les systèmes de numération, puisque chaque façon de représenter les nombres induit des façons particulières de poser les opérations ; la numération de position s'est imposée parce qu'elle offre une efficacité opératoire plus grande que les systèmes qui l'ont précédée. En fin de compte, c'est celle qui a permis le développement des calculateurs mécaniques puis électroniques. Dans un contexte plus avancé, le développement de

techniques de résolution de systèmes d'équations linéaires a conduit à la mise au point du calcul matriciel, dont l'efficacité calculatoire autant que conceptuelle a permis à la fois l'implémentation dans des machines de calculs géométriques et le développement théorique de l'algèbre linéaire.
Ce processus d'automatisation nécessite donc de comprendre les objets mathématiques et leur manipulation *avant* de s'économiser la peine de faire les calculs à la main par l'emploi d'une machine. Cela s'applique particulièrement bien pour le calcul *exact*, dans le domaine du *discret*. Dans le domaine du *continu*, l'affaire est plus subtile, technologiquement et théoriquement.

III. LE RAPPORT AUX OBJETS

Comprendre un objet mathématique avec assez de précision pour automatiser les calculs qui lui sont associés conduit nécessairement à questionner la nature des objets et le rapport que l'on a à eux : ont-ils une existence physique que l'outil permet de manipuler ? Sont-ils des constructions abstraites pour lesquelles l'outil sert à aider l'intuition ? Ou sont-ils des émanations de l'outil, la part abstraite de son fonctionnement, ce qui reste quand on oublie le détail physique de l'objet ?

1. Calcul numérique

La notion de *nombre* est évidemment fondamentale et elle recouvre une grande variété de constructions, de niveaux d'abstraction variés : les entiers naturels, les entiers relatifs, les décimaux, les rationnels, les réels…
S'ils paraissent nets et bien établis au mathématicien moderne, leur transcription informatique n'est pas anodine. Dans le cas des entiers, la représentation est fidèle grâce à l'existence de systèmes de notation efficaces et précis : tant que la question de la quantité de mémoire disponible ne se pose pas, les entiers sont représentés de façon parfaitement fidèle et les calculs se font de façon efficace. La même chose s'applique aux décimaux (qui ont un nombre fini de chiffres après la virgule) et aux rationnels (qui ont une représentation irréductible unique). Dans le cas des nombres que la tradition mathématique appelle *réels*, la question de la précision est incontournable : les nombres à virgule flottante, qui servent au calcul numérique, utilisent une forme de « notation scientifique » et imposent de fixer un nombre de chiffres significatifs. Dès lors, le calcul se fait à une précision fixée et la moindre opération peut nécessiter des arrondis. Ainsi, la machine ne manipule plus l'objet mathématique, mais une approximation de celui-ci. Comprendre et dompter cette approximation est un sujet mathématique en soi et est au cœur des préoccupations de l'analyse numérique.
Cette nécessité d'approximation n'est pas une insuffisance technologique mais un phénomène bien plus fondamental. En effet, il est parfaitement possible de traiter des nombres réels de façon exacte comme on le fait sur papier, au moyen du calcul formel, mais cette élimination de l'approximation se fait au prix de la décidabilité : si système de notation employé est assez expressif pour représenter tous les nombres que l'on peut rencontrer dans la pratique courante (polynômes, fonctions usuelles, solutions d'équations et fonctions implicites), alors il ne peut pas exister d'algorithme capable de déterminer si deux formules quelconques désignent le même nombre réel réel (théorème établi par Richardson (1969)). Autrement dit, l'égalité entre les nombres est hors de portée de l'outil de calcul. Voilà qui questionne le caractère *réel* de ces nombres…

2. Fonctions

Le mot *fonction* se réfère lui aussi à une notion abstraite qui est difficile à appréhender, difficulté qui est renforcée par le fait que le même mot désigne en mathématiques en informatique des objets subtilement différents.
En mathématiques savantes, une fonction est une relation, ce que la théorie des ensembles identifie à un ensemble de couples. Ce n'est bien sûr pas cette notion qui est la plus abordable dans l'apprentissage de la discipline, mais c'est la forme qui s'est avérée la plus efficace dans la pratique mathématique de la démonstration.
À l'inverse, en informatique, et plus précisément dans le contexte de la programmation, une fonction est un procédé de calcul, permettant d'obtenir un résultat étant donné une valeur donnée en entrée. C'est donc une vision nettement opératoire, qui n'est rien d'autre que la version formalisée d'un algorithme.
Ces deux interprétations de la notion de fonction ne sont pas contradictoires, il s'agit plutôt de deux constructions mentales différentes autour de la même idée, répondant à des besoins différents. En effet, la version opératoire correspond à l'idée de calcul et prolonge en quelque sorte la vision « naïve » qui identifie la fonction mathématique à la formule qui la définit, elle permet la manipulation formelle et l'application, alors que la vision ensembliste permet le raisonnement générique et l'approche de la fonction comme « boîte noire » (au sens où la fonction envoie une valeur d'entrée sur une valeur de sortie sans que l'on ait accès à la façon dont cette sortie est obtenue). Il est naturel de se demander où se situe la notion qui se construit dans l'esprit de l'élève et cela dépend certainement du contexte qui justifie l'introduction de la notion.

3. Géométrie

Dans le cas des constructions géométriques, la situation est un peu différente parce que le statut de la figure tracée n'est pas le même que celui du nombre ou de la fonction définis par une formule. En effet, à partir d'un certain niveau, le dessin obtenu par une construction géométrique sur papier est toujours considéré comme une illustration nécessairement imparfaite d'un objet abstrait (même dans les petites classes où le dessin est le domaine d'expérimentation et de mesure, il vient vite l'idée que si l'on dessinait avec toujours plus de soin, on aurait des mesures toujours plus proches d'un résultat idéal).
Dans ce contexte, les logiciels de géométrie dynamique permettent de s'affranchir des difficultés liées à la construction de figures à la main : grâce à l'outil, on gagne en rapidité (une fois acquise l'habitude du logiciel, ce qui n'est pas forcément anodin) et en précision dans les constructions, on peut mesurer précisément pour conjecturer des propriétés, et le fait de pouvoir très facilement faire bouger les objets peut faciliter grandement le développement des intuitions.
La limite de l'utilisation de l'outil serait éventuellement l'illusion de l'exactitude, si la précision accrue donnée par l'outil fait oublier que l'objet manipulé est toujours une approximation puisque là encore, les nombres réels ne peuvent pas être représentés de façon fidèle. Si la géométrie est *l'art de raisonner juste avec des figures fausses*, l'outil informatique n'y change rien, les figures sont simplement un peu moins fausses…

IV. LES ABSTRACTIONS INFORMATIQUES

Ces considérations sur le rapport aux objets issu de la pratique informatique mettent en évidence que, quel que soit l'objet considéré, le traitement informatique impose ses règles : la finitude de l'information traitée et l'effectivité de son traitement, qui imposent une approximation dès que l'objet représenté est infini.

1. L'informatique est aussi une science

Quand on emploie des outils informatiques, on ne peut donc pas faire l'économie d'une étude de l'informatique elle-même, en tant que façon de penser et d'aborder les problèmes, c'est-à-dire en tant que science. Cette étude doit prendre en compte les quatre piliers de la science informatique, pour reprendre l'analyse de Dowek (2011) :

- La machine, comme dispositif mettant en œuvre les procédures effectives de calcul : si une tâche doit être en fin de compte effectuée par une machine, alors il est nécessaire de prendre en compte ce qu'impose la machine ;
- L'information, comme objet traité par la machine : avec un dispositif technique, on ne manipule pas les concepts abstraits mais leur représentation, c'est-à-dire leur description ;
- L'algorithme, comme méthode pour l'élaboration et l'étude des procédés de calcul : confier une tâche à une machine impose de décrire précisément les façons de procéder, parce que l'outil n'a pas accès à l'intuition qui permet à l'humain de mener ses raisonnements ;
- Le langage, comme moyen de formaliser l'information pour en rendre le traitement possible : représenter l'information, qu'il s'agisse du procédé de calcul sous la forme d'un programme ou de l'objet à traiter sous la forme d'une donnée numérisée, impose de définir les règles de représentation sans ambiguïté.

La compréhension de ces quatre aspects et de leurs interactions est un sujet en soi, c'est le cœur de la science informatique. Il s'agit bien là d'une science, avec son versant expérimental et son versant théorique, le second visant à modéliser et prédire ce qui est observé par le premier.

Comme dans toute science, il y a donc dans l'informatique un dialogue constant entre le concret et l'abstrait. Ici, le concret désigne tout ce qui se rapporte à la machine réelle : les programmes, les données traitées et tout ce qui permet d'obtenir des résultats et d'agir sur le monde réel et dans le monde réel, en tenant compte des contraintes matérielles. L'abstrait désigne alors tout l'attirail théorique, de nature fondamentalement mathématique, qui permet la conception et l'étude des systèmes concrets : la logique, l'algorithmique, la théorie de l'information, la théorie de la calculabilité et de la complexité, etc.

2. Représentation de l'information

C'est la notion de *codage* qui fait le lien entre le concret et l'abstrait en informatique : le code est la représentation concrète d'un objet abstrait.

En effet, la machine informatique ne traite pas les objets mathématiques mais leur représentation symbolique. Mais si la machine ne traite que les symboles, l'intention se situe bien du côté de l'objet abstrait. Ce n'est pas le code en soi qui est important, mais bien la signification qu'on lui attribue. Dès lors, il est nécessaire de bien distinguer l'objet de son

écriture. Pour prendre l'exemple le plus simple, le nombre n'est pas la même chose que son écriture en chiffres.
De plus, en dehors du domaine des objets discrets (nombres entiers, textes, graphes, objets combinatoires), la représentation induit une restriction de l'espace des objets. Il s'agit d'une simple raison de cardinalité : un code, d'une façon ou d'une autre, pourra toujours se voir comme une suite finie de chiffres, et l'ensemble des telles suites est dénombrable. L'ensemble des nombres réels étant indénombrable, il est donc impossible de tous les représenter, quel que soit le choix de représentation. L'argument s'applique aussi bien aux fonctions dès que leur domaine de définition est infini et par extension à tout ce qui est *analogique* : signaux (sons, images), mesures physiques…
Quel est alors le rapport entre l'objet mathématique et l'objet informatique censé le représenter ? Lorsque la représentation impose une approximation, l'objet informatique contient moins d'information que l'objet mathématique (ou réel) : il conserve en principe assez d'information pour permettre le traitement, mais il en oublie une partie qui est inaccessible au calcul (par exemple la valeur exacte d'un nombre réel, au-delà des chiffres significatifs de sa représentation approchée). Alors, c'est l'objet informatique qui est une *abstraction* de l'objet mathématique, peut-être dans un sens un peu différent du mot.

3. Enseigner les abstractions informatiques

L'emploi de l'informatique, que ce soit comme science ou comme outil, impose donc de se confronter à ses abstractions, c'est-à-dire à la fois aux notions abstraites qui la fondent comme science et aux mécanismes d'abstraction que mettent en œuvre ses objets. Par conséquent, l'emploi de l'informatique dans l'enseignement nécessite d'enseigner ces abstractions. Le problème n'est pas nouveau et de nombreux travaux ont été entrepris depuis un demi-siècle pour permettre de le faire.
Avec la machine, on peut distinguer deux grands types d'approches, qui correspondent aux deux types d'abstraction évoqués. Pour comprendre les représentations et leur impact, il s'agit des innombrables activités utilisant l'outil (calculatrice ou logiciels), dès lors qu'on porte une attention sérieuse aux effets de cet outil par rapport à la pratique manuelle.
Quant à l'enseignement des notions abstraites de la science informatique, il passe par l'initiation à l'informatique dans des environnement conçus pour cet apprentissage. L'exemple le plus emblématique est le système Logo, qui crée des *micromondes* volontairement simples où expérimenter la programmation pour en acquérir les intuitions et les principes. Ce système a été développé initialement par Papert et son équipe dans les années 1960 (Papert, 1993), reprenant dans le contexte informatique des idées de la psychologie du développement issue notamment des travaux antérieurs de Piaget (1936). Le système Scratch, aujourd'hui en vogue, en est un descendant assez direct.
Il est important de mentionner aussi que l'acquisition des notions abstraites de l'informatique est nettement favorisée par la pratique d'activités dites *débranchées*. Par ce mot, on désigne des activités sans ordinateur, souvent sous forme de problèmes scénarisés, un peu ouverts, dont l'étude met en œuvre des raisonnements et des notions liés à l'algorithmique, la théorie de l'information, etc (on pourra se reporter à Bell et al. (2012) pour une présentation générale de l'idée). La grande vertu de ces activités est de faire aborder les abstractions propres de l'informatique tout en se libérant de la pesanteur de l'outil technique, dans ce qu'il peut avoir d'encombrant intellectuellement : l'utilisation de l'ordinateur impose de se confronter à toutes les dimensions de l'informatique (y compris à des contingences matérielles non essentielles voire antagonistes à la compréhension du contexte, à commencer par le pouvoir de distraction de l'objet) alors que l'activité débranchée se focalise sur la notion qu'elle vise à aborder.

V. CALCUL ET DEMONSTRATION

1. La programmation comme discipline de pensée

Au-delà de son rôle productif très concret, l'activité de programmation induit une certaine façon d'aborder les problèmes, en quelque sorte une « discipline de pensée » très constructive et méthodique, notamment adaptée à la résolution de problèmes même mathématiques. L'approche constructiviste, poussée à son terme, peut faire considérer l'activité mathématique comme une variation sur l'activité informatique :

- On construit les objets mathématiques comme des logiciels : au moyen de briques de base (les fondements logiques, vus comme le langage de programmation des mathématiques) ou d'objets déjà définis (des théories établies, collections de définitions et de propriétés, vus comme des bibliothèques de sous-programmes).
- Les notions et théories mathématiques sont vues comme des interfaces de programmation : je vous dis comment utiliser les choses (quelles définitions, quels axiomes, quels énoncés de théorèmes, etc.) mais pas comment ces choses ont été construites (détails de démonstration, sans influence sur l'emploi des théorèmes).
- Un objectif toujours présent est la généralisation (trouver des théories mathématiques moins contraintes, ayant donc plus de modèles ; réécrire son code pour qu'il soit utilisable dans plus de contextes), avec comme outil la modularité (possibilité de travailler sur une partie d'un développement sans remettre en cause la construction globale).

Cette analogie n'est pas exagérée : la démonstration assistée par ordinateur, qui s'est beaucoup développée ces dernières années, lui donne corps en présentant l'élaboration de théories et de démonstrations (formalisées) comme une forme particulière d'ingénierie logicielle.

2. Stratégies de calcul et de démonstration

La même analogie se poursuit d'un point de vue plus abstrait, en comparant les stratégies adoptées pour calculer et pour démontrer. En effet, si un algorithme est (presque par définition) une stratégie de calcul, une démonstration est en un certain sens une stratégie d'argumentation permettant de convaincre un interlocuteur en étant capable de répondre à tout objection potentielle. Pour préciser la correspondance, on a :

- énoncé mathématique = type d'information = règles d'un jeu,
- démonstration d'un énoncé = programme produisant une information = stratégie d'un joueur,
- exemple d'application = exécution pour une entrée = partie dans le jeu.

Une démonstration peut alors être vue comme un algorithme pour gagner dans une argumentation visant à justifier une assertion mathématique. Cette analogie prend une forme très précise sous le nom de *correspondance de Curry-Howard*, élément central en théorie de la démonstration et des langages de programmation.

De fait, la logique est un fondement commun des mathématiques et de l'informatique, comme l'illustrent ces analogies. En forçant à peine le trait, on peut d'ailleurs reconnaître les quatre piliers de l'informatique dans les travaux des pères fondateurs de la logique moderne :

- Gödel (1931) : théorèmes d'incomplétude, fondés sur l'idée de *codage* de l'information

- Turing (1936) : formalisation mathématique de la notion d'*algorithme*
- Church (1930–) : description du calcul et de l'information par un *langage* formel
- Von Neumann (1945) : notion effective de *machine* universelle, à l'origine des ordinateurs

VI. POUR CONCLURE

Ces réflexions illustrent, du moins je l'espère, qu'il y a entre les mathématiques et la science informatique à la fois un fondement commun dans la logique et un spectre d'approches diverses sans être contradictoires. En plaçant le concret et l'abstrait sur des axes différents, cela donne différents rapports aux objets, qu'il s'agisse d'idées mathématiques ou de phénomènes du monde réel.

Si dans le domaine du discret l'informatique apparaît comme un moyen efficace de prolonger des automatismes, c'est dans le domaine du continu que la différence d'approche devient manifeste : là où les mathématiques classiques privilégient l'abstraction de l'*infini en acte* (les nombre réels comme structure statique, nécessairement indénombrable), le caractère incontournable de l'effectivité en informatique impose la vision plus dynamique de l'*infini potentiel* (les nombres réels comme phénomène émergent dans la manipulation d'approximations à précisions finies).

Cette démarche opératoire donne une importance fondamentale à l'idée de stratégie, qu'il s'agisse de la stratégie de calcul d'un algorithme ou de la stratégie d'argumentation d'une démonstration.

Finalement, les abstractions informatiques peuvent-elles concrétiser les mathématiques ? La réponse n'est évidemment pas binaire. L'informatique, tant comme outil que comme science, donne un sens opératoire aux raisonnements, ce qui peut constituer une façon d'éprouver les objets et donc de contribuer à en donner une compréhension concrète. Plus généralement, cela permet de proposer des cadres d'expérimentation même pour des notions abstraites. La contrepartie est la nécessité pour cela de se confronter aux questions liées spécifiquement à l'écriture des objets et à leur manipulation symbolique, ce qui apporte son lot d'abstractions, d'une nature différente. Peut-être faut-il voir là une sorte de loi de conservation de l'abstrait ?

RÉFÉRENCES BIBLIOGRAPHIQUES

BELL, T., ROSAMOND, F., ET CASEY, N. (2012). *Computer Science Unplugged and Related Projects in Math and Computer Science Popularization*. In The Multivariate Algorithmic Revolution and Beyond, H.L. Bodlaender, R. Downey, F.V. Fomin, et D. Marx, éd. (Berlin, Heidelberg: Springer Berlin Heidelberg), p. 398-456.

DOWEK, G. (2011). Les quatre concepts de l'informatique. In G.-L. Baron, É. Bruillard, & V. Komis. (Éd.), *Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques.* (p. 21-29). Consulté à l'adresse https://edutice.archives-ouvertes.fr/edutice-00676169

PAPERT, S. (1993). *Mindstorms: children, computers, and powerful ideas* (2nd ed). New York: Basic Books.

PIAGET, J. (1936). *La naissance de l'intelligence chez l'enfant*. Lausanne: Delachaux et Niestlé.

RICHARDSON, D. (1969). *Some Undecidable Problems Involving Elementary Functions of a Real Variable*. Journal of Symbolic Logic 33, 514-520.