Construire et calculer dans un monde 2D

Florent Becker, Jérôme Durand-Lose

Chapitre 4

Construire et calculer dans un monde 2D

Florent Becker
Jérôme Durand-Lose

Ce chapitre présente ce qu'il est possible d'accomplir dans un espace euclidien en considérant justement cet espace (comme entrée, support, mémoire…). Accomplir à la fois comme calculs (au sens discret, mais aussi continu) et comme construction d'objet.

Trois modèles de calcul basés sur des primitives géométriques — l'utilisation de la règle et du compas, l'émergence de polyèdres ou l'utilisation de dérivées constantes — sont présentés.

Les machines à signaux sont ensuite définies : des segments de droite sont prolongés et, dès qu'ils se rencontrent, ils sont remplacés. Ces machines sont capables de calculer au sens classique. De part la nature continue de l'espace-temps, hyper-calcul comme calcul analogique sont également possibles. De plus, en contrôlant la génération de fractales, le calcul fractal permet de résoudre SAT quantifié en « temps et espace constant ».

Finalement, les assemblages autonomes d'éléments discrets dans le plan forment un modèle d'une grande richesse. Celui-ci relie des objets théoriques et combinatoires que sont les pavages avec des réalisations possibles dans un modèle bio-informatique (calcul à ADN). Le calcul obtenu est asynchrone, et la synchronisation se fait par la géométrie. Temps et espace sont deux dimensions fortement intriquées.

Mots-clés Automates de Mondrian, Bio-informatique, Asynchronisme, Auto-assemblage, Calcul analogique, Calcul au sens de Turing, Calcul

fractal, Fractales, Hyper-calcul, Machines à signaux, Pavages, Piecewise constant derivative, Règle et compas, Tuiles de Wang.

Pré-requis

— Calculabilité : Machines de Turing, indécidabilité de la Halte,
— Complexité : classes P et NP et
— Géométrie euclidienne de collège.

NB. Les automates cellulaires ayant fait l'objet d'un cours dans l'école de l'année précédente à Caen [44], ils ne sont pas abordés bien qu'ils aient toute leur place ici.

4.1 Introduction

Ce chapitre est consacré à ce qu'il est possible de faire comme calcul et comme construction dans un espace euclidien. L'espace n'est pas l'endroit où se situeraient des fils et des portes logiques d'un calcul séquentiel, parallèle ou distribué, mais un substrat où se passent les choses. Le cadre n'est pas celui de la programmation ou des machines et automates, mais d'un espace euclidien où sont localisées des *informations* que font évoluer une dynamique.

Plusieurs types de choses peuvent se produire : construction à la règle et au compas, émergence d'une structure en fonction de règles locales, trajectoire d'un point en fonction du lieu ou de plusieurs points qui interagissent, morceaux d'ADN qui s'assemblent… À chaque fois, le lieu, la distance, la transmission d'information, la place présente, les rencontres… sont autant d'éléments pour lesquels *space matters*.

L'espace est euclidien : d'une part il est continu et d'autre part, la géométrie prévalente est celle des droites, des points et des cercles. La vision géométrique est prépondérante comme le montre l'iconographie de ce chapitre.

Ce cadre euclidien vient avec des limites : pas d'équation aux dérivées partielles qui ne soit triviale, pas de géométrie algébrique… l'évolution est simple en dehors de changement instantanés. Il s'agit donc de systèmes hybrides : ils ont des aspects continus (liés à l'espace et au temps) et discrets (changement de phase, transition, collision, appareillement…).

Le cadre continu ouvre la voie aux *effets Zénon* [1] : une infinité de transitions discrètes en un temps fini. Plusieurs modèles sont ainsi capables

1. en référence au paradoxe de la flèche et de la cible qui doit encore et toujours parcourir la moitié de la distance restant à parcourir.

d'hyper-calcul, c'est-à-dire de résoudre le problème de la Halte (et des problèmes encore « moins » calculables).

Ce cadre est aussi idéal : lignes sans épaisseur, position exacte… Les modèles sont théoriques et physiquement peu acceptables : densité d'information non bornée, hypothèse d'un espace euclidien à toute échelle…

Seul le dernier modèle considéré dans ce chapitre, l'auto-assemblage de tuiles est un modèle sujet à expérimentations. Il est issu d'une remarquable convergence entre un modèle théorique classique, les pavages de Wang, et des résultats expérimentaux sur l'assemblage de polymères d'ADN. Il répond ainsi à la fois à une préoccupation théorique évoquée dans ce cours : quelles formes peut-on obtenir par un assemblage local et irréversible ? et à sa mise en pratique comme solution pour créer des nano-artefacts à coût raisonnable.

Le cours s'articule en trois grandes parties. Dans chaque partie, les résultats principaux ainsi que des références et des exercices sont présentés. Il n'est pas possible de rentrer dans les détails techniques ou les résultats complexes, mais les références idoines sont indiquées pour approfondir les sujets abordés.

La première partie présente trois modèles de calculs ancrés dans un monde euclidien où les primitives font sens. Le premier, les *Geometric Computation Machines* de Huckenbeck [34, 35], utilise un automate pour manipuler la règle et le compas et construire des points, des droites et des cercles. Le second, les *Automates de Mondrian* de Jacopini et Sontacchi [36], impose des contraintes locales (au sens des ouverts de $\mathbb{R}^n$) correspondant à une uniformité et une causalité dans l'espace-temps ; il en émerge des polyèdres réguliers reflétant une dynamique à une autre échelle. Le troisième, le *Piecewise Constant Derivative* de Asarin et Maler [3, 4], découpe l'espace en régions polyédrales où la vitesse est constante ; en partant d'un point, la trajectoire peut être très complexe changeant de région un nombre infini de fois en un temps fini.

La seconde partie se concentre sur un seul modèle : les machines à signaux de Durand-Lose [22]. Après une définition du modèle, une simulation de machines de Turing (et donc de capacité à calculer) est présentée. Tirant partie de la continuité de l'espace-temps, il est possible d'enchaîner dynamiquement des changements d'échelle et d'accélérer le calcul et d'implanter une forme du modèle du trou noir. La construction de fractales permet une répartition du calcul « abyssale », le *calcul fractal* permet de résoudre QSAT (SAT quantifié) en temps constant. Cette partie se termine par la mise en évidence de la capacité à faire des calculs analogiques, c'est-à-dire sur l'ensemble des réels.

La troisième partie développe le modèle d'*auto-assemblage* introduit notamment par Adleman [1]. Après une définition du modèle au travers de ses liens avec les pavages de Wang, il est montré comment y faire des calculs au sens Turing puis comment y assembler des formes efficacement. Cette notion d'efficacité est présentée à la lumière d'intuitions sur une implantation physique de l'auto-assemblage, ce qui ancre ce modèle dans une interprétation moins idéale de la physique.

4.2 Divers modèles en géométrie euclidienne

Cette partie présente trois modèles de calcul géométriques. Le premier utilise la règle et le compas. Le second se base sur des contraintes locales. Le troisième s'intéresse aux trajectoires ayant des dérivées constantes sur des régions polyédrales.

4.2.1 Avec la règle et le compas

Cette section reprend les travaux sur les *Geometric Computation Machines* de U. Huckenbeck [34, 35] qui définissent une machinerie dont les primitives sont les opérations géométriques usuelles que l'on peut effectuer avec une règle et un compas. Le propos n'est pas de faire de l'algorithmique géométrique (discrète, symbolique ou algébrique), mais d'utiliser ces primitives dans un espace Euclidien de dimension deux.

La machine est un automate équipé de registres. Ceux-ci sont de trois types : points, droites et cercles. Les états de l'automate servent à représenter à la fois la ligne du programme exécuté et l'état du calcul (c'est-à-dire Non-terminé, Fini et Erreur, les deux derniers étant finaux).

Les opérations disponibles sont les suivantes :

— copier une valeur (point, droite ou cercle) en sortie,
— mettre dans un registre l'intersection de deux droites,
— mettre dans un registre l'une des intersections d'une droite et d'un cercle (optionnel : différente du point indiqué),
— mettre dans un registre l'une des intersections de deux cercles (optionnel : différente du point indiqué),
— mettre dans un registre la droite passant par deux points,
— mettre dans un registre le cercle centré sur un point dont le rayon est la distance entre deux points,
— copier un registre dans un autre et
— Fini.

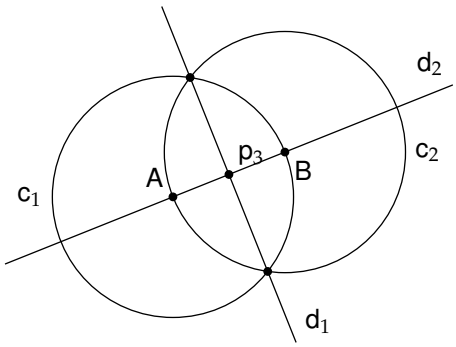
Les intersections n'existent pas forcément et ne sont pas forcément uniques. L'exécution n'est donc pas déterministe. Si une instruction n'est

pas possible, alors la branche (de l'arbre de toutes les exécutions possibles) se clôt par Erreur.

S'il n'y a aucune feuille Erreur et que toutes les branches se terminent (par Fini) avec exactement la même sortie, alors le calcul aboutit et la sortie est la sortie commune (donnée par n'importe quelle branche). Par exemple, le programme de la Fig. 4.1 calcule le milieu d'un segment (les extrémités sont notées A et B). Attention, il y a deux exécutions possibles (où sont p_1 et p_2 ?), mais leurs sorties sont identiques.

1: $c_1 \leftarrow$ Cercle (centre A, rayon d(A,B))
2: $c_2 \leftarrow$ Cercle (centre B, rayon d(A,B))
3: $p_1 \leftarrow$ Intersection (c_1, c_2)
4: $p_2 \leftarrow$ Intersection (c_1, c_2) différente p_1
5: $d_1 \leftarrow$ Droite (p_1, p_2)
6: $d_2 \leftarrow$ Droite (A, B)
7: $p_3 \leftarrow$ Intersection (d_1, d_2)
8: Écrire p_3
9: Fini

(a) Programme



(b) Construction géométrique

FIGURE 4.1 – *Calcul du milieu du segment* AB.

Les instructions de saut sont de la forme « Si $p_k \in$E aller à i: sinon à j: » où p_k est un registre contenant un point et E est un ensemble prédéterminé servant d'oracle.

Un cas simple est celui où l'ensemble E est le singleton origine et où les points $(0,0)$, $(1,0)$ et $(0,1)$ sont donnés comme constantes. Les fonctions (calculables en temps constant) d'un n-uplet de points vers des points sont alors exactement celles où les coordonnées des points en sortie s'expriment à partir de celles en entrée avec des fonctions, par morceau, rationnelles à coefficients entiers.

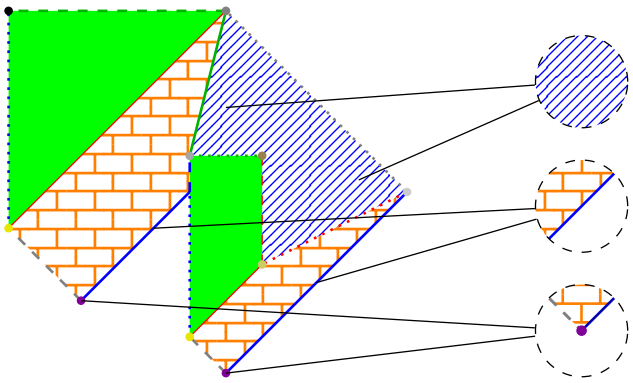
Ce cadre algébrique se comprend par la possibilité d'implanter les primitives suivantes : d'une part $(x,y) \to (x,0)$, $(x,y) \to (y,0)$ et $(x,0)\,(y,0) \to (x,y)$ et d'autre part, sur l'axe des x, les additions, multiplications et divisions.

Cela correspond aux constructions classiques des nombres calculables à la règle et au compas [12]. Mais les coordonnées constructibles y sont également closes par racine carrée. Ici, la condition « quelque soit la branche d'exécution, elle se termine normalement et avec la même sortie » rend *in-fine* impossible l'apparition du radical [35].

4.2.2 Automates de Mondrian

Les travaux de Jacopini et Sontacchi [36] partent de la modélisation de l'espace-temps physique réel. Des hypothèses sont faites dont découlent des contraintes locales qui font émerger des structures polyédrales.

Dans un espace euclidien de dimension quelconque, à chaque point est associé un état, une couleur. L'hypothèse faite est que couleur et voisinage sont liés : si deux points ont la même couleur, alors il existe un rayon suffisamment petit où les voisinages sont identiques. Cela est représenté sur la Fig. 4.2.



(a) Espace coloré (b) Contraintes locales

FIGURE 4.2 – *Espace de Mondrian.*

Par conséquence, s'il existe une petite boule de couleur uniforme alors tous les points de cette couleur n'ont que cette couleur à proximité. Topologiquement, les points de cette couleur forment un ouvert.

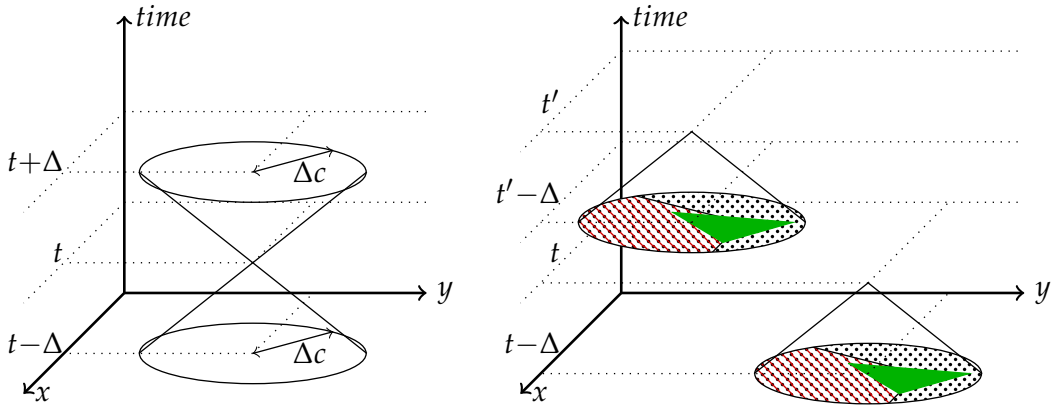
De même, s'il y a une courbe d'une couleur (d'épaisseur nulle) alors la courbe doit être un segment de droite (si les voisinages sont identiques, alors les dérivées également). Tous les points de cette couleur sont sur des segments parallèles. Ce sont toujours les mêmes couleurs d'un coté et de l'autre du segment. Les extrémités des segments sont d'autres couleurs (mais pas forcément toujours les mêmes).

Chaque couleur correspond à des régions polyédrales de mêmes dimensions et parallèles. Elles seront ouvertes dans ces directions. Les couleurs limitrophes dans les directions orthogonales seront toujours les mêmes.

Il y a un nombre fini de couleurs. Il suffit donc d'avoir un voisinage pour chaque couleur pour définir toutes les contraintes sur l'espace coloré, à savoir, pour chaque couleur, les directions des sous-espaces linéaires et les voisinages imposés.

L'étape suivante consiste à ajouter une dimension pour le temps et une règle de causalité. Celle-ci est définie par une vitesse de la lumière, c, et la

condition que la couleur d'un point est totalement déterminée par ce qui se trouve à n'importe quelle date antérieure dans le cône passé (limité par la vitesse de la lumière). Un cône correspondant à un point est représenté sur la Fig. 4.3a, avec en bas le passé, et en haut le futur. La Figure 4.3b montre deux portions d'espace (à des temps différents) où l'on trouve la même coloration. Les deux cônes basés sur ces portions et limités par la vitesse de la lumière sont donc identiques.



(a) Cônes passé et futur (b) Causalité

FIGURE 4.3 – *Cônes et causalité.*

Pour que le système soit réversible, la même contrainte est imposée mais en renversant le temps. Cela correspond à renverser les cônes (c'est-à-dire pointant vers le passé et non le futur) sur la Fig. 4.3b.

Les contraintes temporelles s'expriment aussi à l'échelle des polyèdres. Il est possible de raisonner en terme d'intersections et de collisions (ce qui est développé dans la partie sur les machines à signaux). À ce niveau, une machine de Turing se simule sans difficultés avec des positions rationnelles.

4.2.3 Dérivée constante par région

Dans ce modèle, *Piecewise Constant Derivative*, introduit par Asarin et Maler [3], l'espace est partitionné en un nombre fini de régions polyèdrales. Sur chaque région, une vitesse constante est définie. Sur la Fig. 4.4, les partitions sont en traits épais et les vitesses sont représentées par des flèches.

Partant d'un point, une trajectoire respecte la vitesse. Si un bord est atteint, la trajectoire se prolonge sur la région adjacente en suivant la nouvelle vitesse. Sur la Fig. 4.4, deux trajectoires sont indiquées. Elles partent de la gauche. La trajectoire représentée en tirets change deux fois de région

et part ensuite à l'infini. Celle en pointillés s'enroule à l'infini autour d'un point d'intersection à trois régions.

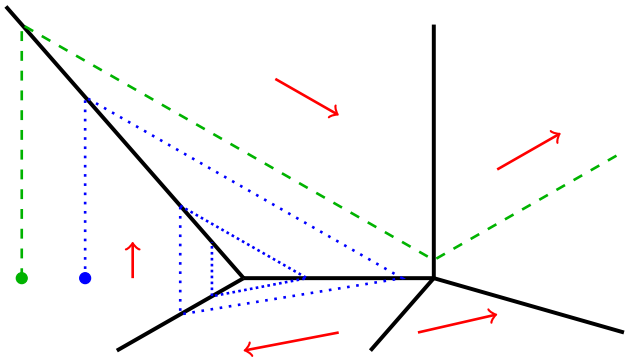


FIGURE 4.4 – *Trajectoires sur un Piecewise Constant Derivative.*

Cette seconde trajectoire est singulière : elle change infiniment souvent de région mais pourtant atteint sa limite en un temps fini (c'est une somme géométrique) et s'y arrête. Il y a deux échelles de temps qui cohabitent : le temps continu fini où la trajectoire atteint sa limite et le temps discret infini des changements de région. C'est un *effet Zénon*.

Les positions dans l'espace et le temps sont continues. Un nombre réel peut contenir beaucoup d'information (par exemple l'arrêt de toutes les machines de Turing pour le nombre Oméga de Chaitin [15]). Un nombre rationnel permet déjà de coder une information discrète. Tout ce qui suit est restreint à des positions rationnelles pour les points de départ et les sommets des polyèdres.

Ce système calcule en considérant qu'il y a une zone de départ (l'endroit exact est la valeur de l'entrée) et que l'arrêt correspond à entrer dans une certaine zone. Le problème d'accessibilité est de décider si l'on atteint ou non une zone à partir d'un point (si le calcul s'arrête).

Il est possible de coder une configuration d'une machine de Turing dans la trajectoire et de faire en sorte que chaque changement de région corresponde à une mise à jour de cette valeur. Avec un espace à trois dimensions, il est ainsi possible de calculer.

Ajouter des dimensions à l'espace permet d'ajouter des niveaux d'effet Zénon. Avec d dimensions, le niveau $d-2$ de la hiérarchie arithmétique (c'est-à-dire Σ_{d-2})[2] est décidable [3, 4]. Cela va même plus loin : le problème de l'accessibilité est complet sur des niveaux de la hiérarchie hyper-arithmétique[3] [9, 11, 10].

2. Σ_0 est décidable, Σ_1 est le niveau du problème de la Halte.
3. Extension de la hiérarchie arithmétique aux indices ordinaux (par-delà les entiers).

4.3 Machines à signaux

Cette partie est consacrée à un seul modèle développé depuis une décennie : les *machines à signaux* [20]. Ce modèle trouve son origine dans une réflexion sur les automates cellulaires (non présentés, voir les actes de l'EJC 2014 à Caen [44]). En effet, une des notions-clés pour la compréhension comme pour la synthèse d'automates cellulaires est le signal : pour véhiculer de l'information, déclencher, synchroniser... Souvent la dynamique se décompose en rencontres de signaux engendrant de nouveaux signaux.

Dans le contexte des automates cellulaires, il s'agit de signaux discrets s'étalant souvent sur plusieurs cellules. Afin de présenter clairement les choses, une fois définis, ces signaux sont souvent représentés par des segments de droites euclidiennes afin de présenter clairement leur dynamique de collision sans alourdir le propos avec la discrétisation. L'idée est de s'affranchir de la discrétisation et de regarder ces signaux comme des segments de droites idéaux [24]. La discrétisation n'est plus un « détail technique » : elle ne fait plus partie du modèle.

Les signaux sont des points (sans dimension) localisés dans un espace euclidien de dimension 1. Ils gardent les caractéristiques propres aux automates cellulaires :

Synchronie ils avancent tous en permanence, à la même cadence, et

Uniformité ils ont toujours la même dynamique ; en particulier, un même motif (signal discret) se déplace toujours de la même façon dans un automate cellulaire, la vitesse d'un signal est une constante qui dépend uniquement de sa nature. De même l'interaction entre signaux ne dépend que de leur nature.

Les signaux sont animés d'un mouvement rectiligne uniforme et engendrent des segments de droite dans l'espace-temps. La nature d'un signal est appelé *méta-signal*.

Après une définition des machines à signaux, il est rapidement montré leur capacité à calculer au sens de Turing. La continuité de l'espace-temps ainsi que la *malléabilité* des calculs sont mis en évidence afin d'implanter le modèle du trou noir et d'hyper-calculer. La possibilité de construire partiellement des fractales est utilisée afin d'obtenir un parallélisme sans limite (et faire du calcul fractal). Enfin, les capacités du modèle à faire du calcul analogique (sur des réels) sont considérées.

4.3.1 Définitions

Dans cette sous-section, définitions informelle et mathématique sont associées. La définition informelle suffit à la compréhension de la section.

Définition 4.3.1 (Machine à signaux)**.** *Une* Machine à signaux *se définit par un triplet* (M, S, R) *où*

- *M est un ensemble fini de* méta-signaux,
- *S est une fonction associant à chaque méta-signal sa* vitesse *et*
- *R est un ensemble de règles de collision. Une* règle de collision *associe à un ensemble d'au moins deux méta-signaux de vitesses distinctes (*entrée*) un ensemble de méta-signaux de vitesses distinctes (*sortie*).*

De plus R doit être déterministe : un ensemble de méta-signaux apparaît en entrée d'au plus une règle de collision.

À un instant donné, il y a un nombre fini de signaux ou de collisions. Ils sont positionnés dans l'espace. Comme chaque signal est complètement défini par le méta-signal associé et une collision par la règle, une configuration est entièrement définie en associant à chaque point de la droite réelle un méta-signal, une règle ou rien.

Définition 4.3.2 (Configuration)**.** *Une* configuration *d'une machine à signaux* (M, S, R) *est entièrement définie par une fonction de l'espace dans les méta-signaux et les règles de collisions auxquels est ajoutée une valeur vide (de méta-signaux et de collision) et telle que le nombre de positions non vides soit fini.*

Soit V l'ensemble $M \cup R \cup \{\oslash\}$ *où* $\oslash \notin M \cup R$*. Une* configuration*, c, est une fonction de* $\mathbb{R}$ *dans V telle que* $|c^{-1}(M \cup R)| < \infty$*.*

Dans la suite, *localisation* désigne un repérage dans l'espace-temps, *position* la localisation spatiale et la *date* la localisation temporelle. Les localisations seront notées (x, t), espace puis temps.

La dynamique d'une machine à signaux se définit de proche en proche : tant que les signaux ne se rencontrent pas, ils se déplacent de manière rectiligne uniforme. Par contre, dès que deux signaux (ou plus) se rencontrent, ils sont immédiatement remplacés.

Le temps est continu, mais il y a une échelle de temps discrète : celle des collisions. La dynamique se définit donc à partir de cette échelle en considérant les temps avec collision(s) et les temps intermédiaires (simples propagations).

Définition 4.3.3 (Dynamique et diagramme espace-temps)**.** *Soit c une configuration, soit t la borne supérieure des dates sans rencontre de signaux (ou la date de la prochaine collision).*

$$t = \sup_{d} \left\{ \begin{array}{c} \forall x_1, x_2 \in \mathbb{R},\ \forall \mu_1, \mu_2 \in M,\ \forall d',\ x_1 \neq x_2, \\ \mu_1 \ltimes c(x_1) \wedge \mu_2 \ltimes c(x_2) \wedge 0 < d' < d \\ \wedge\ (\ x_1 + S(\mu_1)d' \neq x_2 + S(\mu_2)d'\ ) \end{array} \right\}$$

où $\propto$ signifie « est égal (au méta-signal) ou appartient à la sortie (de la règle de collision) ».

Les configurations entre 0 et t ne contiennent pas de collisions et sont définies par la simple propagation des signaux :

$$\forall d, 0 < d < t, c_d(x) = \left\{ \begin{array}{l} \mu \text{ si } \mu \propto c(x - S(\mu)d) \\ \oslash \text{ sinon} \end{array} \right\}$$

où, par définition de t, μ est unique s'il existe.

Si t n'est pas ∞, la configuration à t est définie de manière similaire en ajoutant un cas : s'il y a plusieurs signaux qui arrivent en x alors c'est la règle de collision correspondante.

Les configurations se calculent ainsi de collision en collision. Un diagramme espace-temps *rassemble toutes les configurations entre deux dates.*

Cette définition met bien en avant l'aspect hybride : étapes continues séparées par des événements discrets.

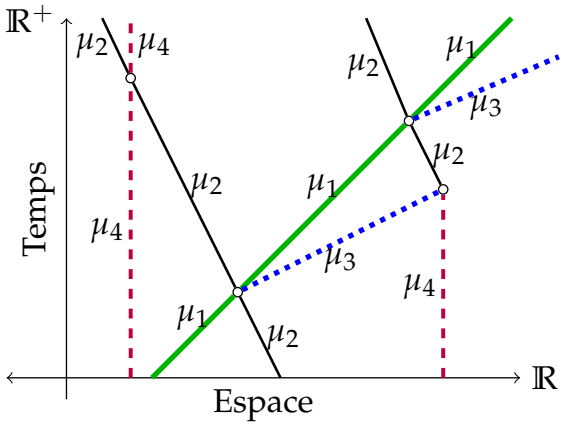
Exemple 4.3.1. La Figure 4.5 montre sur la gauche la définition de 4 méta-signaux et de 2 règles de collisions. Sur la droite, elle montre le diagramme espace-temps engendré à partir d'une configuration où se trouvent (de gauche à droite) des signaux (de méta-signaux) μ_4, μ_1, μ_2 et μ_4.

Nom	Vitesse
μ_1	1
μ_2	$-1/2$
μ_3	3
μ_4	0

(a) Méta-signaux

$\{ \mu_1, \mu_2 \} \to \{ \mu_2, \mu_1, \mu_3 \}$
$\{ \mu_3, \mu_4 \} \to \{ \mu_2 \}$
$\{ \mu_4, \mu_2 \} \to \{ \mu_2, \mu_4 \}$

(b) Règles de collision



(c) Diagramme

FIGURE 4.5 – *Exemple de machine à signaux et de diagramme espace-temps.*

Les règles de collision se lisent sur le dessin. Une règle est dite *blanche* si les méta-signaux sortant sont les mêmes qu'en entrée (comme celle de $\{ \mu_4, \mu_2 \}$ sur l'exemple). Par la suite, elles ne sont plus indiquées.

Pour trouver la localisation d'une collision de deux signaux, il suffit de résoudre un système linéaire de deux équations (une par signal) à deux

inconnues (position et date). La localisation d'une collision de signaux dont les positions et les vitesses sont des nombres rationnels est donc forcément à coordonnées rationnelles. Plus généralement, avec des vitesses rationnelles et partant des positions rationnelles, toutes les collisions sont à coordonnées rationnelles.

Définition 4.3.4 (Machine à signaux rationnelle)**.** *Il s'agit d'une machine à signaux dont les vitesses sont rationnelles et pour lesquelles les positions des signaux dans une configuration initiale sont obligatoirement rationnelles. Dans le diagramme espace-temps, toutes les collisions ont des localisations rationnelles et les signaux sont à des positions rationnelles à chaque date de collision* [4]*.*

Ceci a une importance particulière pour la simulation de machines à signaux rationnelles : elles peuvent être simulées exactement sur ordinateur puisque les nombres rationnels y sont représentables exactement et les opérations y sont faites avec exactitude. Une telle simulation a été implantée en Java [5] et utilisée pour engendrer toutes les figures.

Sauf mention contraire, les constructions et résultats de cette section sont valides sur les machines rationnelles comme sur les machines quelconques.

Exemple 4.3.2 (Calcul du milieu)**.** Il est possible de « calculer » le milieu (exactement), c'est-à-dire de positionner un signal à mi-distance entre deux signaux. Cela est illustré par la Fig. 4.6 où un signal U est positionné exactement au milieu des deux signaux M. Cet ajout est déclenché par l'arrivée d'un signal Aj par la gauche. Quand il rencontre le M de gauche, il se transforme en A et $\overrightarrow{\mathsf{R}}$. Le dernier est trois fois plus rapide que l'autre et rebondit sur le M de droite ; il devient alors $\overleftarrow{\mathsf{R}}$, toujours trois fois plus rapide. Il rencontre A exactement à mi-parcours entre les deux M.

Ceci se démontre par le calcul des positions des collisions (à chaque fois un système de deux équations à deux inconnues). La plupart des calculs pour *vérifier la correction* des diagrammes espace-temps (et donc des dynamiques) est de ce niveau.

Si l'on reprend les règles de la Fig. 4.6, le calcul du milieu ne fait intervenir que les trois premières. La quatrième permet le calcul du milieu entre le M de gauche et le U juste après. Cela est déclenché par le second Aj sur la gauche.

Enfin il est également possible d'enlever le U juste après le M de gauche. Pour cela, un ordre Enl est envoyé, il est transformé en E en passant le M

4. Toutefois, les positions intermédiaires ne sont pas forcément rationnelles.

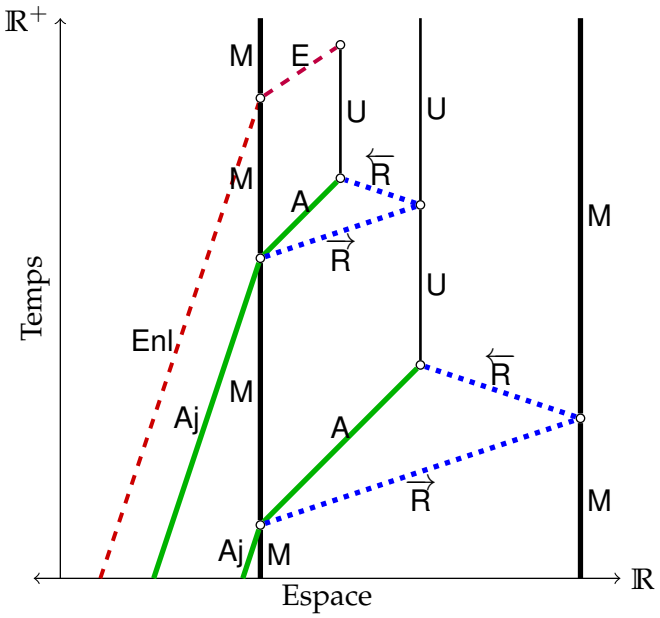
5. L'auteur n'est pas assez content de son code pour le mettre sur internet mais il le fournit sur demande.

Nom	Vitesse
Aj, Enl	1/3
A, E	1
U, M	0
$\overrightarrow{R}$	3
$\overleftarrow{R}$	−3

(a) Méta-signaux

$\{ Aj, M \} \to \{ M, A, \overrightarrow{R} \}$
$\{ \overrightarrow{R}, M \} \to \{ \overleftarrow{R}, M \}$
$\{ A, \overleftarrow{R} \} \to \{ U \}$
$\{ \overrightarrow{R}, U \} \to \{ \overleftarrow{R}, U \}$
$\{ Enl, M \} \to \{ M, E \}$
$\{ E, U \} \to \{ \}$

(b) Règles de collision



(c) Diagramme espace-temps

FIGURE 4.6 – *Calculer le milieu.*

de gauche. Le signal E fait disparaître le premier signal U qu'il rencontre. Ceci correspond aux deux dernières règles.

Exercice 4.3.3. Comment modifier la configuration initiale pour enlever l'autre signal U ? Quelle règle ajouter pour qu'un signal Enl ne continue jamais à droite en l'absence de U ?

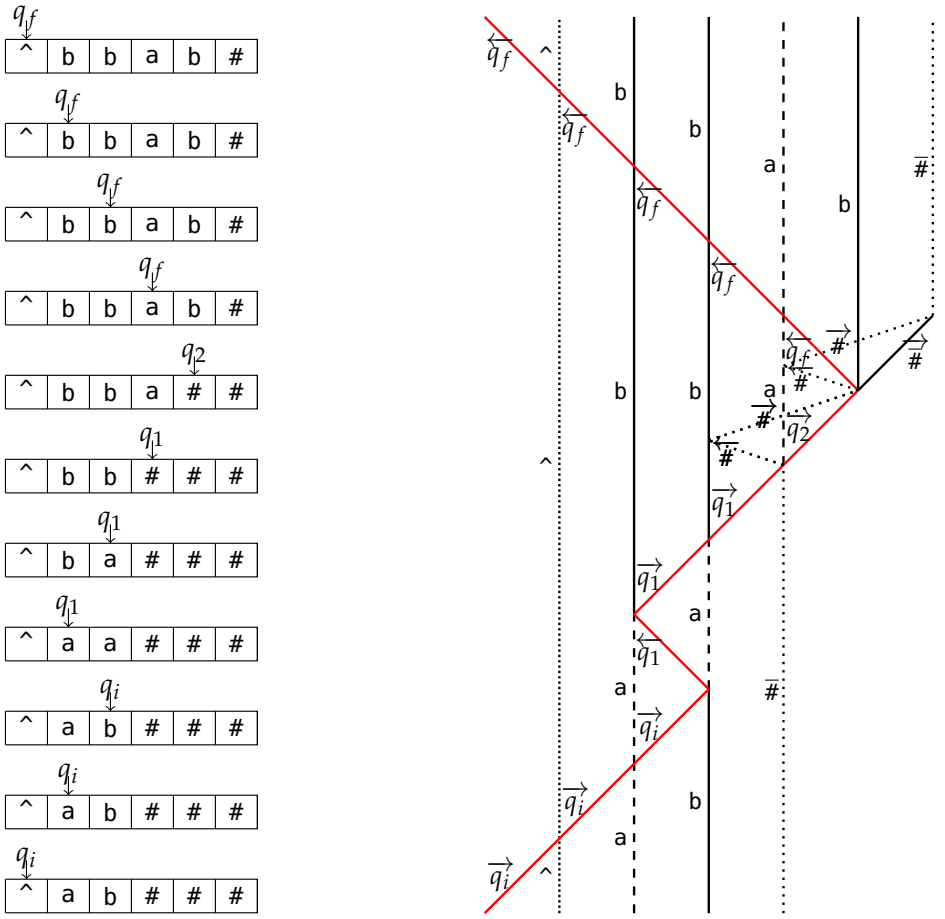
Le calcul du milieu est une primitive très importante. Par exemple, son utilisation répétée permet d'enregistrer n'importe quel nombre (entier) en unaire dans un espace borné (les U de la Fig. 4.6c).

Il y a là une limite du *réalisme du modèle* : le stockage non borné d'information dans un espace borné. Les positions et les vitesses des signaux sont exactes, il n'y a pas de principe d'incertitude d'Heisenberg. Comme expliqué plus bas, le modèle transcende également la thèse de Church-Turing.

4.3.2 Calculer au sens de Turing

Les machines à signaux sont capables de calculer (au sens de Church-Turing) comme le démontre la simulation générique d'une machine de Turing. Une machine de Turing se compose d'un automate fini, d'un ruban non borné composé de cases et d'une tête de lecture/écriture qui permet à l'automate d'agir sur le ruban et de s'y déplacer. Le ruban est toujours fini, mais il est agrandi dès que nécessaire.

La simulation est directe comme le montre la Fig. 4.7 (le temps va toujours vers le haut). L'évolution de la machine de Turing représentée sur la Fig. 4.7a se retrouve Fig. 4.7b. Les signaux verticaux, c'est-à-dire de vitesse nulle, correspondent aux cases du ruban. Les signaux formant un zig-zag indiquent la position de la tête et enregistrent l'état de l'automate. La seule partie technique est l'élargissement du ruban.



(a) Exécution d'une machine de Turing (b) Simulation par une machine à signaux

FIGURE 4.7 – *Simulation d'une machine de Turing par une machine à signaux.*

Le nombre de collisions est linéaire en le nombre de déplacements de la tête sur le ruban. Cette notion de complexité est développée à la section 4.3.4.

Exercice 4.3.4. À chaque fois que la tête va plus à droite, une nouvelle cellule est ajoutée au ruban en assurant toujours la même distance entre les cellules. Les signaux correspondant à la tête sont de vitesses 1 et -1. Sachant que $\overrightarrow{\#}$ est de vitesse 1 et que les vitesses des autres signaux sont de même valeur absolue, que valent les vitesses de $\overleftarrow{\#}$ et $\overrightarrow{\#}$? Reconnaître une construction à rebrousse-temps pour vérifier le résultat.

Comment changer les vitesses de manière à ce que la distance soit à chaque fois la moitié de la distance précédente ? Le ruban n'occupe alors qu'une partie bornée de l'espace. Quelle est cette borne si, dès le départ, il y a ce ratio entre les distances ?

Cette construction est faite sur les machines rationnelles, simulable exactement sur ordinateur. En laissant de coté la définition d'une entrée, de la fin et du résultat d'un calcul pour une machine à signaux, *les machines à signaux rationnelles ont le même pouvoir de calcul que les machines de Turing*. La section suivante sort de ce cadre en considérant les accumulations puis, la section 4.3.5 s'affranchira de la rationalité pour aborder le calcul analogique.

La simulation des machines de Turing entraîne l'indécidabilité des problèmes suivants [6] :

— décider si le nombre de collisions sera fini,
— décider si un méta-signal apparaîtra ou non,
— décider si un signal participera ou non à une collision,
— décider si la configuration va s'étendre sur un coté…

Le diagramme espace-temps fournit la trace de la machine de Turing. L'intérêt des machines à signaux est aussi de fournir une trace graphique compréhensible, par exemple pour un automate à deux compteurs (avec un codage unaire). Cela peut même être fait avec un nombre borné de signaux en utilisant la position pour coder [22].

En ayant plusieurs méta-signaux stockables comme U, il est possible de coder une suite de caractères dans une structure de type pile et le mot à droite (et la mot à gauche) de la tête de lecture d'une machine de Turing. De plus en codant l'état de la pile par une position, il est possible de garder constant le nombre de signaux et de n'utiliser que des règles de collisions qu'entre deux signaux et engendre exactement deux signaux (conservation). Il est possible d'aller encore plus loin en imposant que les règles soient injectives : la collision est entièrement déterminée par les signaux sortant (réversibilité). Les machines de Turing réversibles — universelles au sens du calcul [6, 38, 41] — sont simulables ainsi [29].

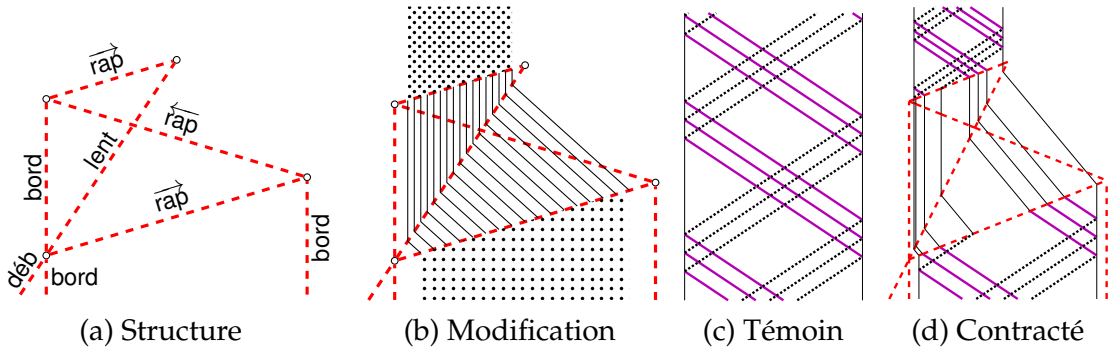
Les machines à signaux simulent également les *Cyclic Tag System* introduits par Cook [14]. Ceux-ci ont relancé la course aux machines universelles de petites tailles, par exemple sur les machines de Turing [54]. La plus petite machine à signaux Turing-universelle connue a été réalisée en simulant les CTS et comporte 13 méta-signaux et 21 règles de collision [26].

6. Tant que les machines sont rationnelles, les problèmes et les instances sont exprimables dans le cadre classique.

4.3.3 Malléabilité de l'espace-temps et trous noir

Il n'y a pas d'échelle ni d'origine dans les définitions ou les figures. Le contexte est celui du continu : un changement d'échelle ou une translation produit la même chose, toutes proportions gardées. En particulier, la division de toutes les distances par deux divise toutes les durées par deux.

Il est possible de changer d'échelle de manière dynamique : le système est arrêté puis contracté et enfin redémarré comme représenté sur la Fig. 4.8. À gauche se trouve la structure avec les noms des méta-signaux ; au centre les pointillés et lignes représentent le diagramme espace-temps actif, gelé et dévié puis actif de nouveau. À droite, se trouve un exemple : un diagramme témoin (Fig. 4.8c) et avec une contraction (Fig. 4.8d).



(a) Structure (b) Modification (c) Témoin (d) Contracté

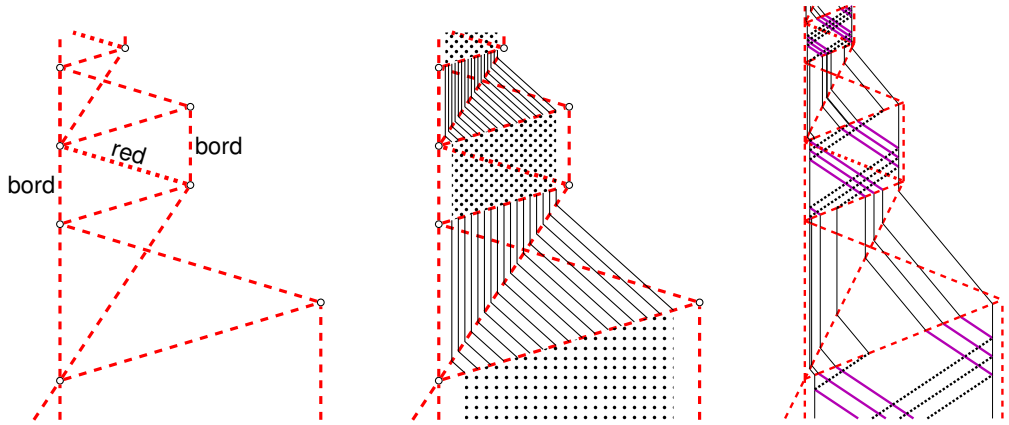
FIGURE 4.8 – *Étape de contraction.*

Pour arrêter, un signal, $\overrightarrow{\mathsf{rap}}$, traverse la configuration et remplace chaque signal qu'il rencontre par un signal de vitesse nulle mémorisant le méta-signal associé. Le système est donc *gelé* suivant une oblique de l'espace-temps sous forme de signaux parallèles. Comme ils sont parallèles, il n'y a pas de collisions et les distances (relatives) entre les signaux sont préservées. Le dégel se fait en utilisant un signal ($\overrightarrow{\mathsf{rap}}$ ici) de même vitesse que le gelant.

Le changement d'échelle se fait en changeant la direction des signaux (parallèles). Le théorème de Thalès montre que les proportions sont conservées. Le signal dégelant est deux fois plus court ; toutes les distances ont été divisées par deux.

Sur du continu, il n'y a pas de limite au changement d'échelle. Pour que le processus se répète de lui-même, il suffit de renvoyer un signal sur la gauche qui redémarre le processus (red sur la Fig. 4.9a).

À chaque réveil, le calcul originel *enchâssé* avance de la même durée relative : à chaque fois, la durée est divisée par deux mais l'échelle l'est également, les collisions prennent deux fois moins de temps. Donc dans cet



(a) Structure (b) Modification (c) Exemple

FIGURE 4.9 – *Contraction itérée.*

espace-temps fini, tout le diagramme espace-temps originel est enchâssé.

Avec la simulation d'une machine de Turing (en espace borné, voir exercice 4.3.4) dans l'enchâssement, si le calcul de la machine s'arrête, alors c'est en temps fini. Avec quelques modifications, il est possible, en cas d'arrêt de faire sortir un signal sur le coté de la contraction itérée.

En dehors de la structure, un calcul peut recevoir ce signal témoin de l'arrêt de la machine. S'il le reçoit, cela est forcément à une date bornée. Cette date limite peut être indiquée par une collision avec un signal de fin d'attente convenablement situé. Donc s'il n'y a pas d'arrêt, la fin d'attente est reçue. Mais s'il y a arrêt, le témoin est reçu avant. À l'extérieur, l'arrêt est décidé. Le modèle est capable d'hyper-calcul en utilisant un effet Zénon. Les premières constructions de ce type datent des premiers travaux dans le domaine [21, 22].

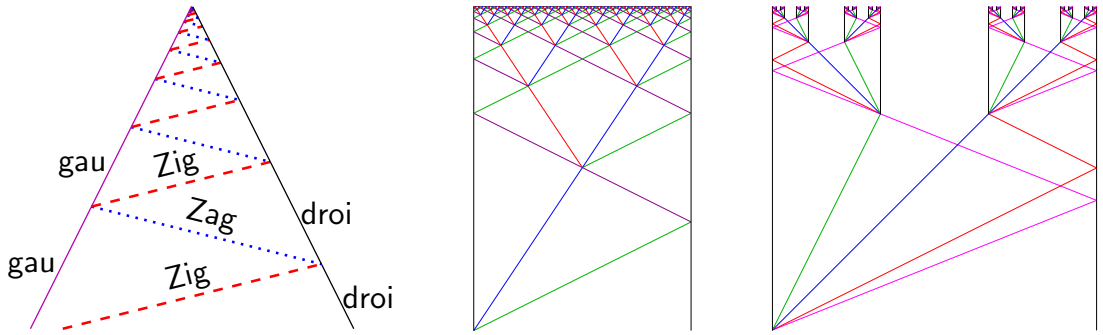
Le principe général implanté ici est d'avoir deux échelles de temps différentes : une infiniment accélérée qui fait le calcul dont l'arrêt est à décider et une autre qui attend un témoin de cet arrêt. Le second possède en plus un moyen de borner le temps de réponse. Ceci correspond au *modèle du trou noir* (voir les travaux de Hogarth [33] et de Németi et ses collègues [2, 31]). L'accumulation en haut de la Fig. 4.9c sert de trou noir.

Il est également possible de déformer un diagramme espace-temps en lui faisant subir des déviations plus complexes (par exemple : augmentation des vitesses par une valeur unique). Il est toujours possible de passer d'une déformation à une autre de manière dynamique tout en *préservant* le calcul : les collisions avec leurs liens de causalité se retrouvent. Il est ainsi possible de rétrécir un calcul sans pour autant le geler [29] (tout en restant réversible et conservatif).

4.3.4 Construction et utilisation de fractales

Beaucoup de fractales se construisent naturellement avec des machines à signaux. Par exemple, quatre méta-signaux suffisent pour construire l'accumulation en haut de la Fig. 4.10a. Le diagramme espace-temps reste indéfini à cette *singularité*.

Construire le milieu du milieu du milieu… permet de construire une fractale comme celle de la Fig. 4.10b. En découpant en trois et en ne recommençant que sur les premiers et derniers morceaux il apparaît une construction classique de l'ensemble diadique de Cantor (Fig. 4.10c). En faisant varier les vitesses, il est possible d'engendrer des ensembles de toute dimension fractale entre 0 et 1 [46, Chap. 5].



(a) Accumulation simple (b) Découpage (c) Cantor

FIGURE *4.10 – Fractales.*

La structure d'enchâssement de la Fig. 4.9b est une structure fractale, elle se répète avec l'échelle un demi en faisant à chaque fois une étape du calcul. Elle travaille en profondeur pour accélérer une séquence de calculs. La Figure 4.10b découpe l'espace à chaque fois en deux. Elle sert de base pour une exécution en largeur afin de mettre en place des calculs parallèles. Il est possible de mener des études de cas, par exemple sur une variable booléenne : à vrai d'un coté, à faux de l'autre.

S'il reste encore des variables, les sous-cas se traitent par d'autres découpages. Il est possible de sous-découper jusqu'à ne plus avoir de variables booléennes non affectées. Tous les cas sont alors traités.

Si les variables apparaissent dans une formule booléenne dont on cherche à savoir si elle satisfiable par une affectation des variables, alors on a un schéma pour résoudre le problème SAT : la satisfaction de formules booléennes (avec des $\neg$, des $\vee$ et des $\wedge$) en $x_1, x_2, \ldots, x_n$ (variables booléennes).

Ce schéma peut être raffiné pour tester si cette satisfaction est indépendante de la valeurs de certaines variables. C'est alors QSAT qui est

résolu. QSAT s'intéresse à la valeur des formules booléennes quantifiées, c'est-à-dire les formules SAT précédées de $Q_1 x_1\, Q_2 x_2 \ldots Q_n x_n$ où les Q_i appartiennent à $\{\forall, \exists\}$. QSAT est un problème PSPACE-complet [47, Sec. 8.3], c'est-à-dire complet pour la réduction polynomiale des problèmes décidables en espace polynomial (déterministe ou non par le théorème de Savitch).

Si une formule booléenne contient dix variables booléennes, dix niveaux de profondeur suffisent à savoir si elle est satisfiable. Le reste de la construction de la fractale ne sert donc à rien. De plus, le diagramme espace-temps n'étant plus défini quand la fractale est construite, il faut empêcher l'achèvement de la construction.

Exercice 4.3.5. Proposer deux façons de limiter une fractale. La première se base sur une série de méta-signaux qui servent de compteur unaire (machine *ad hoc*). La seconde est basée sur la propagation d'un ordre disant de produire un niveau de plus. Pour avoir dix niveaux, il faudra donner dix fois l'ordre (machine générique utilisable pour toute profondeur).

Une formule de QSAT, par exemple $\exists x_1 \forall x_2 \forall x_3\ x_1 \wedge (\neg x_2 \vee x_3)$, est codée par un faisceau de signaux codant chacun de ses éléments. Le calcul est organisé suivant un arbre binaire complet de profondeur 3 (visible sur la Fig. 4.11). À chaque embranchement, une variable booléenne est instanciée, à vrai à droite et à faux à gauche (Fig. 4.12a).

Arrivé aux feuilles de l'arbre, il n'y a plus de variables, la formule est alors évaluée (Fig. 4.12b). Les résultats remontent alors l'arbre (qui se referme) et à chaque nœud il est fait la conjonction (resp. disjonction) s'il s'agissait d'une variable universelle (resp. existentielle) (Fig. 4.13). Les constructions présentent beaucoup d'aspects techniques et sont présentées en détail dans [18, 46, Chap. 7].

Une machine spécifique est engendrée pour chaque formule. En compliquant le codage, il est possible de n'utiliser qu'une seule machine pour toutes les formules. La formule est alors complètement codée dans la configuration initiale [19, 46, Chap. 8].

Il est aussi possible de poser d'autres questions sur la formule non quantifiée : combien de valuations la satisfont (#SAT qui est #P-complet), quelle est la plus « petite » solution… Il « suffit » de changer la façon dont sont agrégés les résultats : au lieu d'une réponse booléenne, il peut s'agir d'un entier, d'une valuation…

Si l'on a besoin de cent niveaux de plus, il suffit d'étendre la construction de la fractale ; cela ne prend pas plus de place ni de temps. Il existe une machine à signaux capable de décider n'importe quelle instance de QSAT en temps et espace constants.

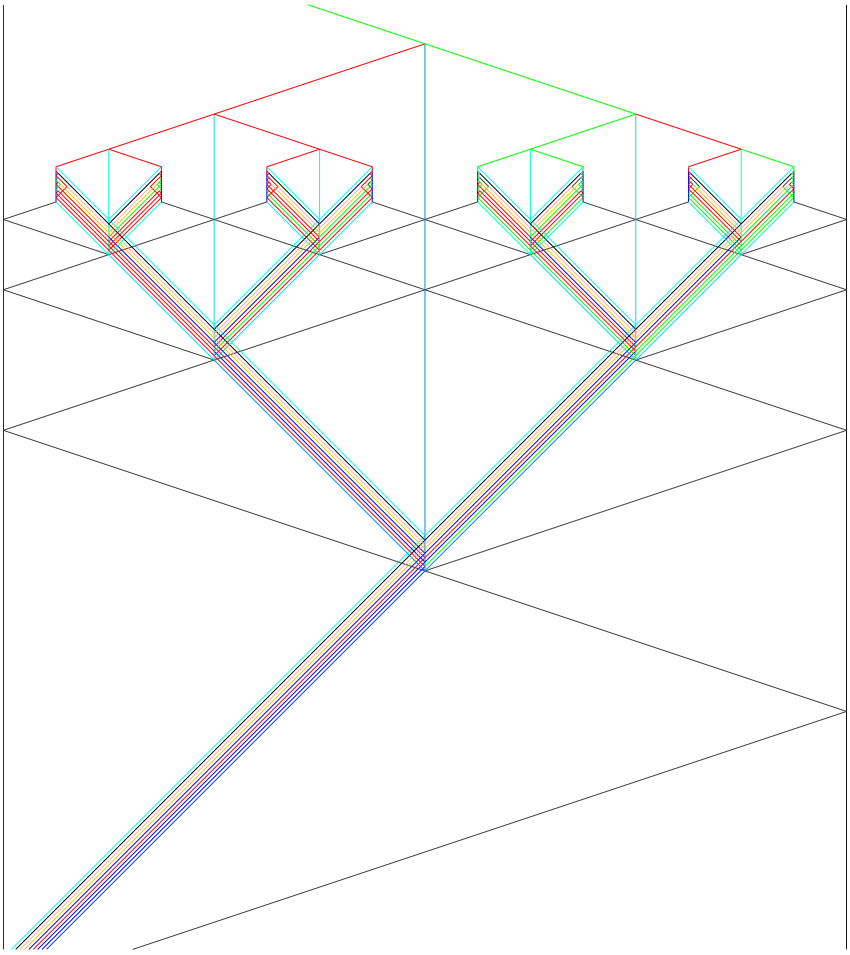
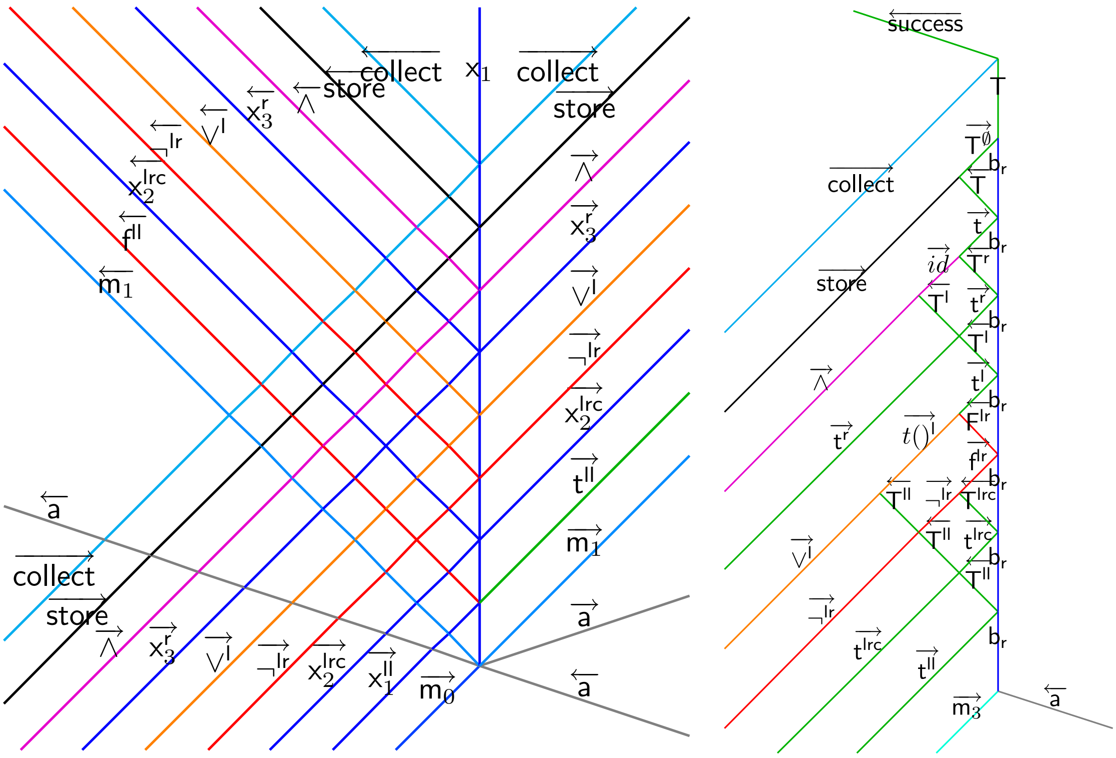


FIGURE 4.11 – *Résolution de QSAT par calcul fractal complet.*

Le modèle du trou noir permet de décider le problème de la Halte en temps et espace constant ; en comparaison, QSAT est trivial. Cela est dû à la continuité du temps et à la possibilité d'accélérer sans limite. Mais avec la résolution de QSAT, il n'y a jamais d'accumulation ; la clé est la continuité de l'espace et la possibilité de toujours couper en deux.

Cette technique de démarrage contrôlé de la construction d'une fractale, de la répartition du calcul dessus et de l'agrégation des résultats s'appelle le *calcul fractal*.

Complexités. Le temps et l'espace sont des mesures continues de complexité. Or, l'évolution est formée de collisions, d'événements discrets. Des mesures de complexité discrètes se construisent en considérant les diagrammes espace-temps comme des graphes dirigés sans cycles. Un *lien de causalité directe* existe entre deux signaux si le premier se termine dans une collision où naît le second. La *complexité en temps* est la taille de la plus longue suite de signaux tels que chacun ait un lien de causalité directe avec



(a) Instanciation de x_1 (b) Évaluation de la formule

FIGURE 4.12 – *Résolution de QSAT par calcul fractal, détails.*

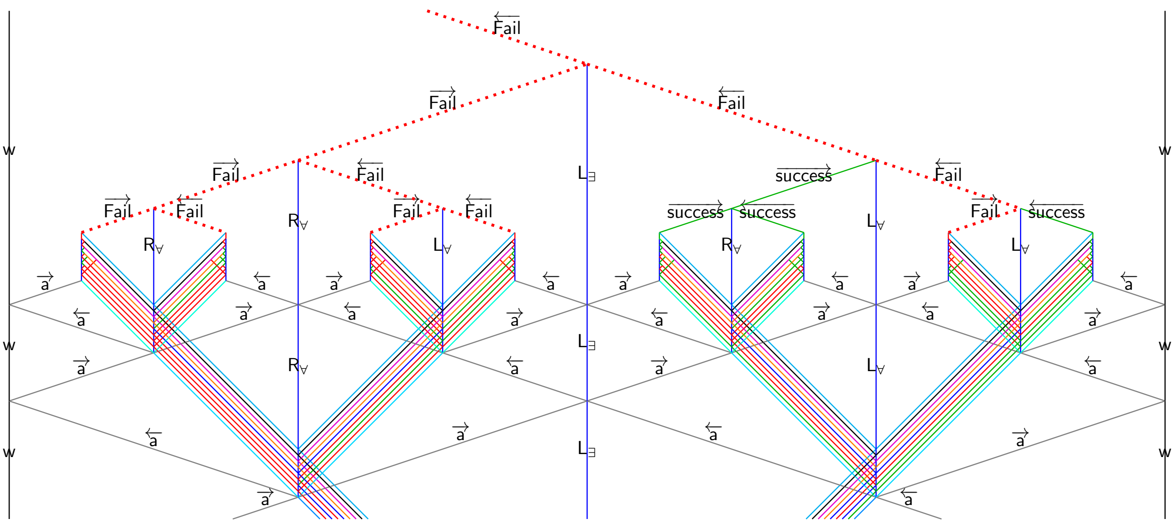


FIGURE 4.13 – *Agrégation du résultat.*

le suivant. Un *lien de causalité* est la fermeture transitive du lien de causalité directe. La *complexité en espace* est le plus grand nombre de signaux sans liens de causalité.

Avec ces définitions, la complexité est quadratique en temps (cubique pour le cas générique) mais exponentielle en espace.

Pour aller plus loin. La Figure 4.10a montre qu'il est possible de faire une fractale avec quatre vitesses. Avec deux vitesses ou moins, le nombre de collisions est bornée. Avec trois vitesses, la situation est duale : c'est impossible avec une machine rationnelle, les signaux sont prisonniers d'un maillage sans accumulations. Par contre cela devient possible s'il y a une vitesse ou une postions irrationnelle. Cette dichotomie se comprend en mettant en lumière un calcul de *pgcd*, celui-ci convergeant sur les rationnels [5].

Qu'en est-il de la capacité à calculer ? Dans le cas rationnel, le maillage montre que la mémoire est forcément bornée. Par contre, dans le cas irrationnel, c'est possible : une étape de la construction de la fractale est faite chaque fois qu'il faut allonger le ruban [30].

4.3.5 Calcul analogique

Il s'agit de calculer avec des nombres réels. Les seules valeurs réelles disponibles sont les distances entre les signaux (si l'on dispose d'une unité) ou les positions des signaux (si l'on dispose d'une origine) ou les rapports entre les longueurs (l'orientation amenant le signe).

Avec deux signaux dont la distance est considérée comme l'unité, il suffit d'avoir un couple de signaux différents — base et val — ou un troisième seul — zéro — pour représenter n'importe quel nombre réel (dans un ordre pour positif, dans l'autre pour négatif et le troisième pour zéro). Diviser par deux revient à trouver le milieu. De la sorte, n'importe quelle division par une constante peut être réalisée. Il faut juste faire attention à la gestion du zéro et du signe.

Si l'on a deux nombres, il est possible d'ajouter le second au premier comme sur la Fig. 4.14. Le parallélogramme prouve l'égalité des distances.

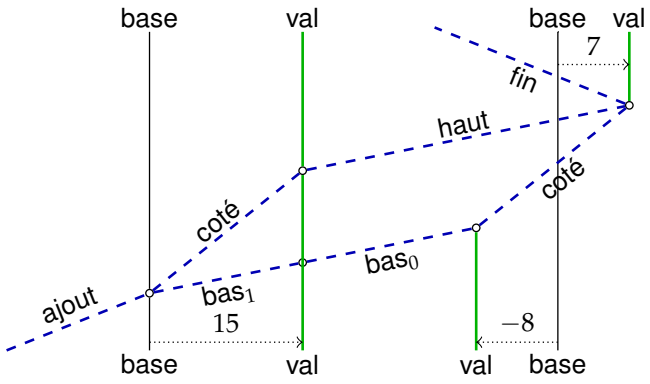


FIGURE 4.14 – *Ajout du réel* 15 *au réel* −8.

Ajouter une valeur négative se gère de façon similaire. Le signe d'une valeur est déduit de l'ordre des signaux base et val. De même, ajouter à

une valeur à gauche et non à droite se fait de manière symétrique.

Si l'on veut ajouter à une valeur qui se trouve un peu plus loin, il suffit de compter les décalages avec une série de signaux que l'on envoie en bas du parallélogramme (par exemple $\mathsf{bas}_k \ldots \mathsf{bas}_1, \mathsf{bas}_0$) comme sur la Fig. 4.14 pour sauter le premier val.

Partant d'une suite de valeurs (comme une suite de cellules pour le ruban), il est possible de multiplier par des constantes et d'ajouter la valeur d'une cellule à un autre. Les opérations se font à distance bornée, dans une fenêtre sur le ruban.

À force d'augmenter, des valeurs pourraient se chevaucher : l'échelle de toutes les valeurs ainsi que la paire servant de valeur unitaire est alors changée (tout en gardant l'espacement d'un base au suivant). Cela est testé et réalisé dynamiquement.

Le déclenchement de différentes opérations peut être géré par un automate fini (ou un programme séquentiel). L'état de l'automate est retenu en l'ajoutant en indice aux signaux réalisant ces opérations (le nombre de méta-signaux reste fini). De même, il est possible de toujours revenir au même endroit pour démarrer l'opération suivante. Il est aussi possible de décaler la position de référence (en élargissant la configuration au besoin comme pour la simulation des machines de Turing).

Tester le signe d'une valeur est simple, cela permet de faire un test pour faire un branchement. Pour achever la construction, il faut un état de fin et la possibilité de mettre des valeurs qui vont servir de constantes.

En résumé, en partant d'un tableau fini de nombre réels (infiniment extensible de chaque coté), il est possible de réaliser l'affectation d'une case en fonction d'une combinaison linéaire des cases autour, de brancher selon le signe et de se déplacer dans le tableau. Cela correspond au modèle de Blum, Shub et Smale (BSS [7, 8]) privé de la multiplication interne. Il s'agit de la version linéaire : lin-BSS [11, 39] avec un nombre non borné de registres.

Les machines à signaux sont donc capables de simuler lin-BSS. La réciproque est vraie. Une configuration d'une machine à signaux est représentée par une suite de blocs représentant le méta-signal, la distance au signal suivant et différents registres annexes. L'automate lin-BSS parcourt toute la configuration et calcule de proche en proche le temps pour que deux signaux voisins se rencontrent. Les vitesses des méta-signaux étant des constantes du programme, il s'agit d'opérations linéaires.

Une fois toute la configuration parcourue, le temps avant la prochaine collision est connu : il s'agit du plus petit temps calculé. La configuration est parcourue de nouveau et les distances sont mises à jour. Quand la distance est nulle, il y a collision de deux ou plus signaux. Ils sont remplacés suivant

les règles de collision (écrites en dur dans le programme). Si le nombre de signaux change, ils sont alors tous décalés de manière à faire de la place ou à faire disparaître le trou.

L'automate BSS est comme celui d'une machine de Turing : il opère sur une fenêtre (et non une case) et peut faire bouger cette fenêtre (comme une tête). Décaler les blocs se fait donc comme décaler les cases d'une machine de Turing.

Pour aller plus loin. Ceci ne correspond qu'au fonctionnement *normal* d'une machine à signaux. Prendre en compte les accumulations, par exemple en marquant leur lieu avec un signal donné, permet d'aller beaucoup plus loin.

Par exemple, il est possible d'extraire d'une distance une suite infinie de 0 et de 1 qui représente son écriture binaire. Ce flux infini peut alors être utilisé pour faire une multiplication : à chaque étape, diviser par deux et ajouter ou non. La multiplication (interne) devient possible et tout le modèle BSS classique est simulé [23].

L'accumulation peut également être vue comme un processus d'approximation convergent. C'est le paradigme des machines de Turing de type 2 : elles prennent en entrée un mot infini représentant des approximations convergentes d'un réel et écrivent en sortie un autre tel mot (avec l'impossibilité de revenir sur le moindre caractère écrit) [51]. Il est effectivement possible de calculer sur des réels en ce sens [25, 27].

Un autre résultat surprenant concerne les accumulations isolées des machines rationnelles. Avec une construction à deux échelles, il a été prouvé que les dates des telles accumulations étaient exactement les réels *computably enumerable* (*c.e.*), c'est-à-dire ceux dont une machine de Turing fournit une suite croissante d'approximations convergentes (mais sans hypothèse sur la qualité de l'approximation). Les positions sont exactement les différences de tels réels (*d.-c.e.*) [28].

4.4 Auto-assemblage

Introduction

Le modèle d'auto-assemblage a été introduit par Adelman[7], Seeman et autres dans [1], puis il a été étudié de façon plus formelle, notamment par Erik Winfree [52]. Au départ, le modèle a été conçu comme une représentation formelle pour une proposition de modèle de calcul à base d'ADN.

7. le A de RSA

L'idée de ce modèle est d'utiliser les forces attractives *sélectives* entre brins d'ADN pour construire des motifs intéressants à une échelle nanoscopique. Le modèle physique qui a été développé à cette occasion se trouve être très proche d'un système connu d'informatique théorique : les pavages de Wang [50].

À partir de ce rapprochement, une théorie riche a été construite autour de la question théorique suivante : comment faire croître intelligemment un motif fini par des interactions *locales* et *définitives*. Cette section s'ouvre par une présentation du modèle, suivie des principaux résultats généraux sur les formes qu'il est possible d'assembler, notamment en lien avec le calcul « classique » ; elle se clot, à partir d'une vision physique du modèle, sur la notion d'efficacité propre à l'auto-assemblage.

Des ressources sont disponibles en ligne (et en anglais) sur `http://self-assembly.net/`. *On y trouve un wiki, ainsi que le simulateur ISU - TAS :* `http://self-assembly.net/wiki/index.php?title=ISU_TAS`

4.4.1 Présentation du modèle

Le contenu de cette section se situe dans la grille carrée $\mathbb{Z}^2$. Il est possible de définir des systèmes d'auto-assemblage sur d'autres grilles, mais nous ne nous intéresserons pas à ce cas dans la suite.

Une *arête* (orientée) de $\mathbb{Z}^2$ est un couple de positions adjacentes de $\mathbb{Z}^2$. La *direction* de l'arête (v_1, v_2) est le vecteur $v_2 - v_1$. Par convention, on note n le vecteur $(0, 1)$, $e = (1, 0)$, $s = -n$ et $w = -e$, ce qui nous donne les directions habituelles de la boussole. On note $\mathrm{dir}(e)$ la direction de l'arête e.

Définition 4.4.1 (Jeu de tuiles de Wang)**.** *Un* jeu de tuiles de Wang *est défini par la donnée de :*

— *un ensemble C de* couleurs,
— *un sous-ensemble de C^4, les* méta-tuiles.

L'ensemble C de couleurs est en général implicite, on identifie donc souvent un jeu de tuiles avec un ensemble de quadruplets quand il n'y a pas d'ambiguïté sur l'ensemble C. On interprète chaque méta-tuile (n, e, s, w) comme un modèle de carré unité dont les côtés portent les couleurs indiquées, en partant du haut et dans le sens horaire. Étant donné une méta-tuile t, on pose $t = (n(t), e(t), s(t), w(t))$: les fonctions n, e, s, w permettent de récupérer la couleur de chaque côté d'une méta-tuile. La notation $d(t)$, avec d une variable représentant une direction permet de désigner la couleur sur la face d d'une tuile.

Définition 4.4.2 (Tuile de Wang)**.** *Étant donné un jeu de tuiles de Wang W, une* tuile $t@(x,y)$ *est donnée par :*

— *une méta-tuile $t \in W$ et*
— *une position $(x,y) \in \mathbb{Z}^2$.*

Une tuile est donc une instance d'une méta-tuile, quelque-part sur le plan $\mathbb{Z}^2$. Si on se donne un ensemble de tuiles dont les positions sont deux à deux distinctes, on obtient un *motif*[8] défini sur une partie du plan. Ce motif est *correct* si, à chaque fois que deux tuiles sont adjacentes, leur côté commun porte la même couleur. Formellement, les définitions suivantes capturent ces notions.

Définition 4.4.3 (Motif)**.** *Étant donné un jeu de tuiles de Wang W, un ensemble de tuiles M est un* motif *si $\forall t_1@(x_1,y_1), t_2@(x_2,y_2) \in M, (x_1,y_1) \neq (x_2,y_2)$. On dit que M* couvre *$\{(x,y) | t@(x,y) \in M\}$*

Le *domaine* $dom(M)$ d'un motif M est l'ensemble des positions de ses tuiles. On peut aussi voir M comme une fonction de $dom(M)$ dans W.

Définition 4.4.4 (Motif correct)**.** *Étant donné un jeu de tuiles de Wang W, un motif M est correct si :*

$$\forall t_1@(x_1,y_1), t_2@(x_1+1,y_1) \in M, \quad e(t_1) = w(t_2)$$
$$\forall t_1@(x_1,y_1), t_2@(x_1,y_1+1) \in M, \quad n(t_1) = s(t_2).$$

Définition 4.4.5 (Pavage de Wang)**.** *Un* pavage du plan *par un jeu de tuiles de Wang W est un motif correct qui couvre tout le plan $\mathbb{Z}^2$.*

Les pavages de Wang constituent un des modèles les plus simples de pavages. En se plaçant sur la grille $\mathbb{Z}^2$, on élimine tous les problèmes liés à la *forme* des tuiles, et ne restent que les questions liées à la combinatoire de leurs couleurs. Malgré cette simplicité, Robinson [45] a montré comment utiliser ce modèle de pavages pour « cacher » du calcul.

Théorème 4.4.6 ([45], Indécidabilité du problème de pavages par tuiles de Wang)**.** *Le problème suivant est indécidable : étant donné un jeu de tuiles de Wang, existe-t-il un pavage par ce jeu de tuiles ?*

Autrement dit, il n'y a pas d'algorithme général qui, si on lui donne des pièces de puzzle carrées[9], décide s'il est possible de recouvrir tout

8. On utilise souvent le terme *patch* en anglais.
9. une infinité d'exemplaires d'un nombre fini de pièces différentes

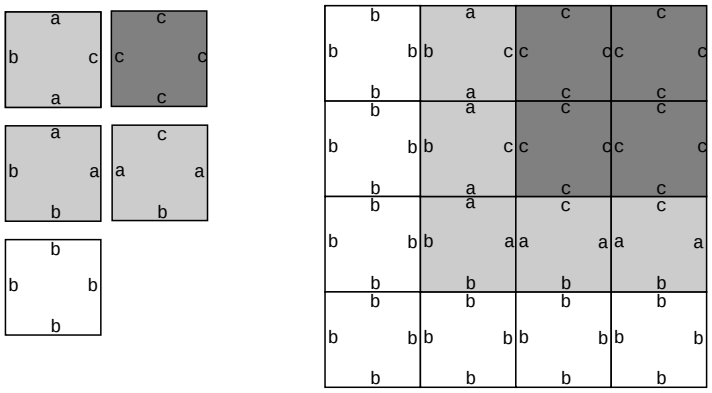


FIGURE 4.15 – *Un exemple de jeu de tuiles de Wang et un morceau d'un pavage possible par ce jeu de tuiles.*

le plan ou si c'est impossible. Il n'existe encore moins d'algorithme qui indique *comment* recouvrir le plan avec un jeu de tuiles arbitraire pour lequel c'est possible.

Puisque ce problème est indécidable, on peut chercher, étant donné un jeu de tuiles de Wang, à obtenir un algorithme qui nous indique comment recouvrir le plan —ou une partie donnée du plan— avec ses tuiles. Cette question est à la fois une question de mathématiques discrètes, et une question de physique théorique : comment des systèmes physiques parviennent-ils à trouver des configurations respectant certaines règles locales ? Dans cette perspective physique, les pavages de Wang sont une forme de discrétisation aussi bien de l'espace que des données que traitent ces règles locales.

Auto-assemblage comme algorithme de pavage *local*. Il paraît alors naturel de chercher des algorithmes de pavages *locaux*, qui n'utilisent pas d'information à distance. Autrement dit, une règle qui, à partir d'un petit morceau d'un pavage, indique comment l'étendre.

Définition 4.4.7 (Jeu de tuiles d'auto-assemblage)**.** *Un* jeu de tuiles d'auto-assemblage $\mathcal{S}$ *est composé de :*

— *un alphabet fini* G
— *une fonction de* force, $G \to \mathbb{N}$
— *un jeu de tuiles de Wang sur l'alphabet* G.

Par la suite, les *méta-tuiles* de $\mathcal{S}$ sont celles du jeu de tuiles de Wang qui définit $\mathcal{S}$.

À la différence d'un jeu de tuiles de Wang, dans un jeu de tuiles d'auto-assemblage, on n'interprète pas les couleurs comme des informa-

tions combinatoires inertes, mais comme des *colles*, actives et capables de créer des *liens*.

Définition 4.4.8. *Soient t_1, t_2 deux méta-tuiles, et d une direction. Le* lien *entre t_1 et t_2 pour la direction d est :*

— $\text{lien}(t_1, d, t_2) = 0$ *si* $d(t_1) \neq (-d)(t_2)$

— $\text{lien}(t_1, d, t_2) = f$ *si* $d(t_1) = d(t_2)$ *et* $\text{force}(d(t_1)) = f$.

Quand t_2 est placée à côté de t_1, les deux tuiles s'attirent si elles ont la même couleur sur leur côté commun, avec une force définie par la fonction force de $\mathcal{S}$. Si les couleurs diffèrent, la force d'attraction entre les deux tuiles est nulle.

Étant donné un motif M de domaine $R \subset \mathbb{Z}^2$ d'un jeu de tuiles de Wang, on appelle *coupe* de M un ensemble C d'arêtes de la grille $\mathbb{Z}^2$ bordées des deux côtés par R, et qui séparent R en deux parties, R^+ et R^-. Pour chaque arête $e \in C$, on appelle e^+ la position adjacente à e qui est dans R^+, et e^- celle qui est dans R^-.

Si l'on tente de couper un motif M le long d'une telle coupe, la force attractive entre les deux moitiés est la somme des liens le long de chaque arête de la coupe.

Définition 4.4.9. *Soit S un jeu de tuiles d'auto-assemblage, et soit M un motif (au sens du jeu de tuile de Wang sous-jacent). Soit C une* coupe *de $dom(M)$; le lien le long de C est défini par :*

$$\text{lien}(C) = \sum_{e \in C} \text{lien}(M(e^-), \text{dir}(e), M(e^+))$$

Définition 4.4.10 (Stabilité)**.** *On dit qu'un motif M de $\mathcal{S}$ est* stable à température τ *si, pour toute coupe C de M, on a :*

$$\tau \leq \text{lien}(C).$$

À la lumière de cette formule, la température τ s'interprète comme une agitation thermique qui a tendance à casser les motifs les plus fragiles en deux. Par exemple, si $\tau > 0$, un motif qui n'est correct nulle part ne sera pas stable : le lien sera nul pour chaque arête, et donc pour toute coupe. Un tel motif aurait tendance à se désagréger.

Exercice 4.4.1. À quelles températures les motifs de la Fig. 4.17 sont-ils stables ?

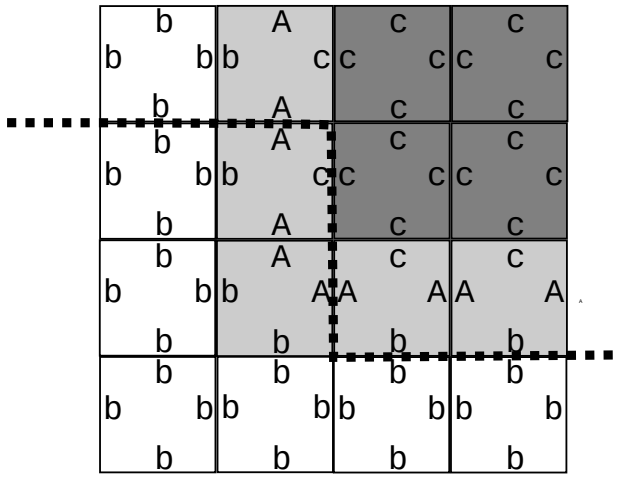


FIGURE 4.16 – *Un exemple de* lien *le long d'une coupe d'un motif ; les colles représentées par une minuscule ont une force* 1*, et les majucscules une force* 2 *; le lien le long de la coupe représentée est de* 8

Exercice 4.4.2. Donner un exemple de jeu de tuile d'auto-assemblage ayant un motif qui est stable à température 2 mais qui n'est pas correct au sens des jeux de tuiles de Wang.

Nous avons défini à quelle condition les motifs pouvaient être stables et ne pas se désagréger, mais comment croissent-ils ? Un motif croît par ajouts successifs de tuiles. Cela revient à considérer une soupe dans laquelle il y a beaucoup de tuiles, celles-ci s'agrègeant lentement en motifs. Chacun de ses motifs tend donc à rencontrer de nombreuses tuiles le long de sa frontière, mais les motifs sont trop rares pour se rencontrer entre eux.

On obtient ainsi la définition suivante, qui permet de donner une *dynamique* à température τ à $\mathcal{S}$.

Définition 4.4.11. *$\mathcal{S}$ a une transition entre deux motifs M et M' à température τ si :*

- *M et M' sont stables à température τ,*
- $\mathrm{dom}(M') = \mathrm{dom}(M) \cup \{(x,y)\}$, $(x,y) \notin \mathrm{dom}(M)$ *et M et M' coïncident sur* $\mathrm{dom}(M)$.

En d'autres termes, une transition à partir d'un motif M consiste à ajouter une nouvelle tuile à M à une position où la somme des liens entre cette nouvelle tuile et M sera supérieure à τ. La figure 4.18 donne des exemples de transitions possibles à température 2 à partir d'un motif (en blanc). Ces transitions correspondant aux ajouts d'une tuile (en gris) à diverses positions.

Il est important de noter qu'une transition peut produire un motif qui n'est pas correct du point de vue du jeu de tuiles de Wang sous-jacent à $\mathcal{S}$

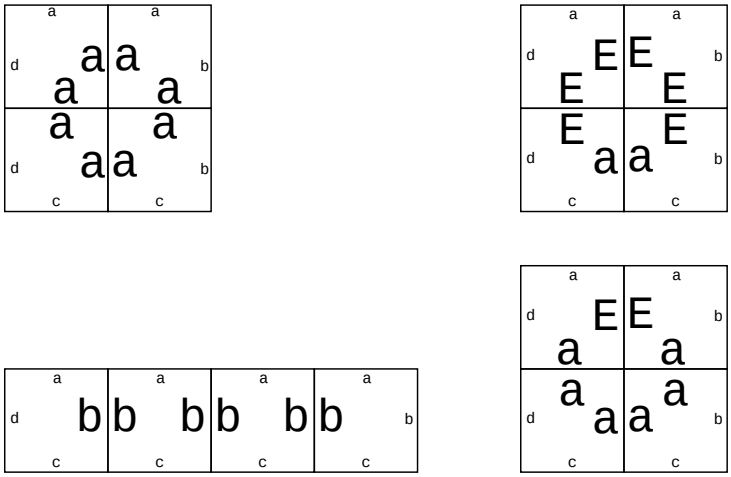


FIGURE 4.17 – *Des exemples de motifs auto-assemblés ; les colles de force* 2 *sont représentés par des majuscules, les colles de force* 1 *par des minuscules. À quelle température chaque motif est-il stable ?*

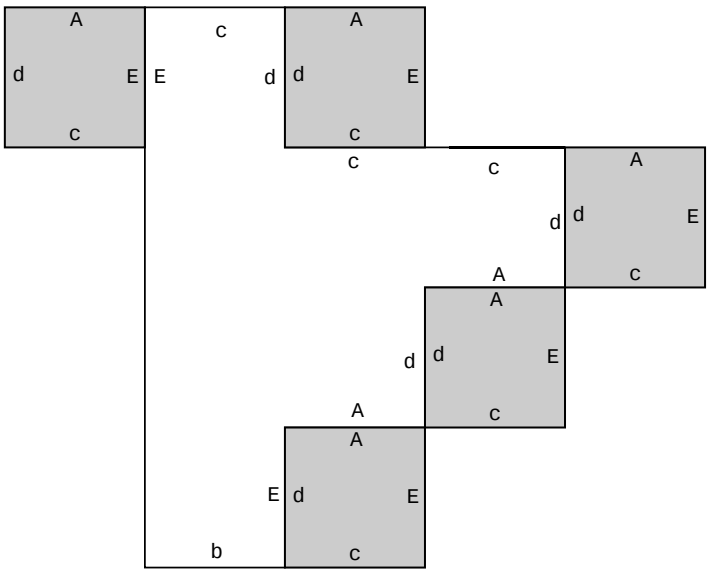


FIGURE 4.18 – *Un exemple de motif avec des transitions possibles à température* 2*.*

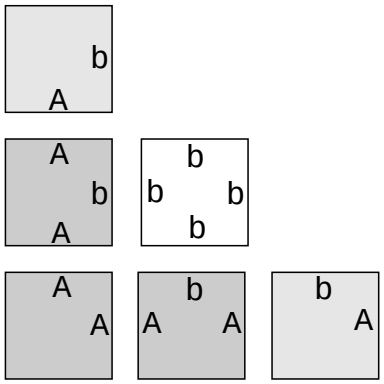


FIGURE 4.19 – *Un système d'auto-assemblage pour les rectangles à température* 2.

si la force des liens « corrects » pour la tuile qu'on ajoute compense le lien de valeur 0 le long de l'arête qui rend le motif incorrect.

Spécifier une « entrée » pour un jeu de tuiles d'auto-assemblage consiste à donner une *température* τ et un motif de départ, la *graine*. La donnée d'un jeu de tuiles, d'une température et d'une graine est appelée un *système d'auto-assemblage*. Une *production* d'un système d'auto-assemblage est un motif que l'on peut atteindre par une suite de transitions à température τ à partir de la graine. Une *production finale* est une production à partir de laquelle plus aucune transition n'est possible.

Un *système d'auto-assemblage* produit l'ensemble de ses productions finales. Par extension, il produit aussi l'ensemble de *formes* suivant :

$$\{dom(p) | p \text{ production finale de } \mathcal{S}\}.$$

Le contexte indique si l'on considère des ensembles de *formes* ou de *motifs*.

On note $\mathcal{S}_\square$ l'ensemble des productions finales de $\mathcal{S}$ –et parfois, par abus, l'ensemble de leurs formes.

4.4.2 Assemblage de formes simples

Le système d'auto-assemblage de la Fig. 4.19 produit l'ensemble des rectangles de toutes dimensions supérieures à 2×2 ; la colle sans nom est de force 0, et la source est le motif réduit à la tuile en bas à gauche. De même, le système de la Fig. 4.21 produit l'ensemble des carrés de toutes tailles supérieures à 2.

Dans ces deux exemples, dont la progression est indiquée respectivement sur les figures 4.20 et 4.22 on voit que le contrôle de l'assemblage se fait par le fait que la température est 2 et que la plupart des colles sont de force 1 : la première tuile dans une rangée ou dans une colonne donnée est nécessairement ajoutée avec une colle de force 2. C'est ce qui fait que, dans

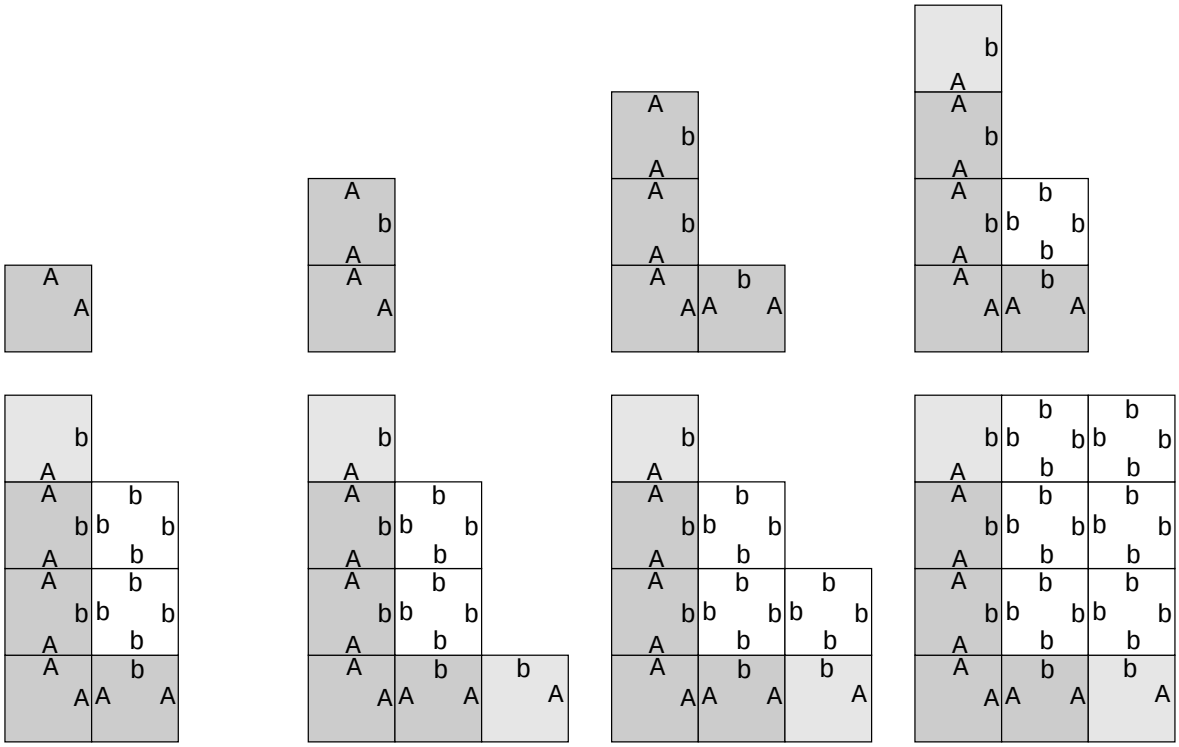


FIGURE 4.20 – *Une progression possible de l'assemblage d'un rectangle.*

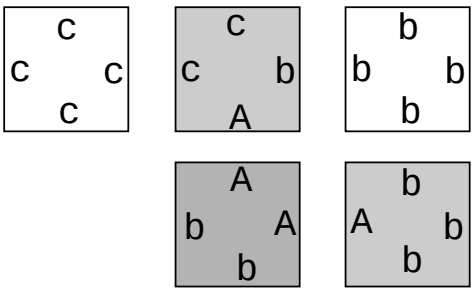


FIGURE 4.21 – *Un système d'auto-assemblage pour les carrés à température* 2.

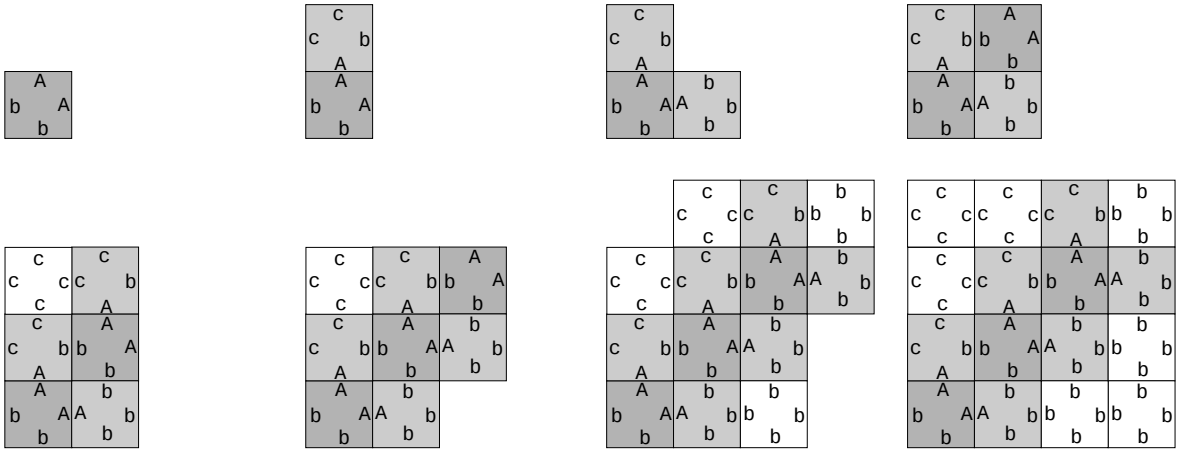


FIGURE 4.22 – *Progression de l'assemblage d'un carré.*

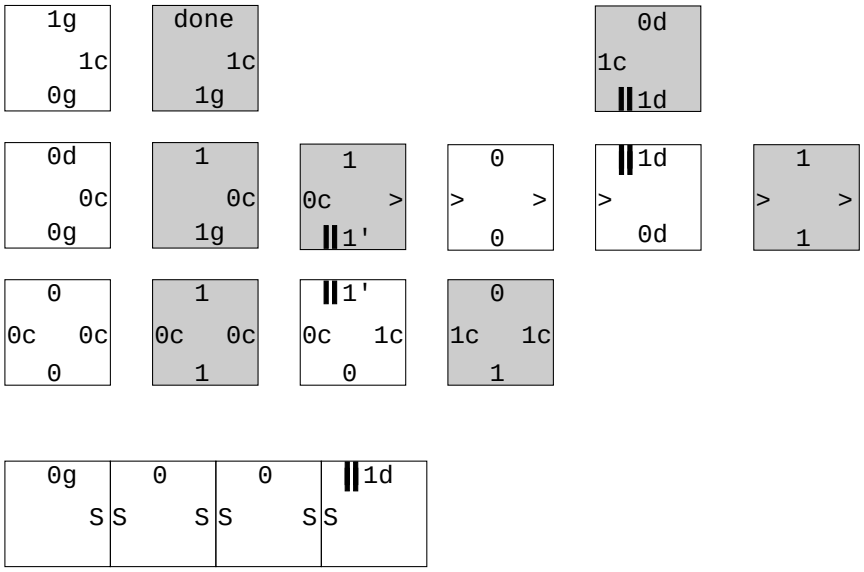


FIGURE 4.23 – *Un compteur à température* 2 *; la couleur de chaque tuile indique le bit qu'elle représente.*

le cas des rectangles, ce sont les côtés bas et gauche qui avancent en premier, et dans le cas des carrés, la diagonale. On remplit ensuite le rectangle qui les englobe avec deux colles de force 1 pour chaque ajout de tuile. Cette façon de procéder, avec des *attachements coopératifs*, est centrale à température 2, et elle permet d'implémenter toutes sortes de comportements complexes.

Ainsi, dans [48], Winfree et al. donnent une méthode pour assembler un carré ou un rectangle de dimension donnée n. Contrairement aux exemples précédents, il s'agit d'un système *déterministe*, c'est à dire ayant une unique production finale. Pour cela, ils utilisent un *compteur*, détaillé ci-dessous.

Le système de la Fig. 4.23 assemble une unique production, qui est un rectangle $n \times 2^n$, dont la rangée i encode l'entier i en binaire. Les colles $.g$ et $.d$ marquent les bords du compteur. Les colles sont de force 1, sauf celles qui sont marquées par une double barre. La source est une barre de longueur n, telle que représentée en bas de la figure. Sur la Fig. 4.24, on voit à la fin la nature asynchrone du calcul : les bits de poids faible peuvent prendre de l'avance sur les bits de poids fort.

Chaque rangée initie la construction de la rangée suivante à partir de son bit `0` le plus à droite (de poids le plus faible). À partir de cette position, la rangée se construit vers la droite en passant tous les bits à `0`, et vers la gauche, en passant tous les bits à `1` jusqu'à trouver un `0`. La position du nouveau `0` se détermine également ainsi (alternativement tout à droite, ou vers la gauche). Quand une retenue arrive au-delà de la colonne n, c'est qu'on a atteint la ligne 2^{n-1}, il est temps d'arrêter la construction : il n'y a

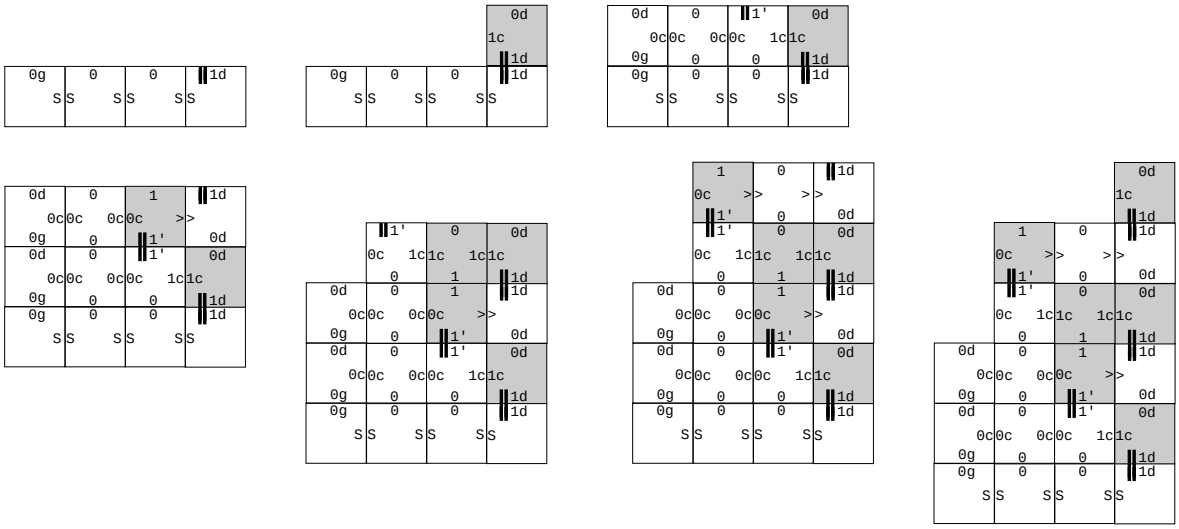


FIGURE 4.24 – *Assemblage du compteur à température* 2.

pas de colle de force 2 vers le haut sur la tuile correspondante.

Un tel compteur peut s'utiliser pour assembler un rectangle ou un carré de dimensions données.

4.4.3 Calcul dans l'auto-assemblage

Il est possible de généraliser la construction du compteur pour simuler une machine de Turing arbitraire. On obtient le résultat suivant [48] :

Théorème 4.4.12. *Le problème de décider si un système d'auto-assemblage $\mathcal{S}$ a une production finale (de taille finie) est indécidable.*

Démonstration. Pour montrer ce théorème, nous allons simuler dans un système d'auto-assemblage le fonctionnement d'une machine de Turing avec une entrée vide. Soit $M = (Q, q_0, q_f, \delta)$ une machine de Turing sur l'alphabet $\Sigma = \{0, 1\}$; le ruban est bi-infini et initialement rempli de symboles `0`. Soit $\mathcal{S}$ le jeu de tuile auto-assemblant de la Fig. 4.25. Les tuiles grises correspondent aux transitions de la machine. Les symboles q, q', r, s représentent des états, x et y des symboles. La tuile L est présente pour chaque transition où la tête se déplace à gauche, C pour celles où la tête reste sur place, et R si la tête va à droite. La source est formée des trois tuiles en bas de la figure ; qi est l'état initial.

Pour chaque transition de la machine de Turing définie par « dans l'état q, sur l'entrée i, écrire o, passer à l'état q' et déplacer la tête de lecture dans la direction d », on introduit les tuiles correspondantes. On obtient alors comme production un diagramme espace-temps de la machine de Turing, le temps allant vers le haut.

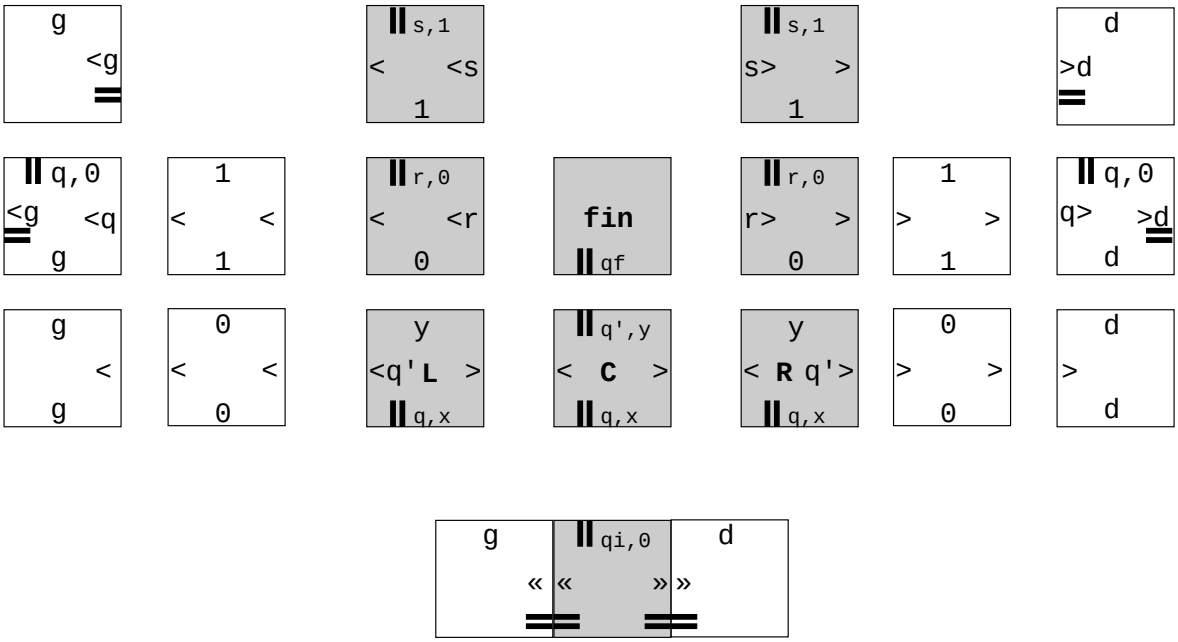


FIGURE *4.25 – Le jeu de tuile pour simuler une machine de Turing.*

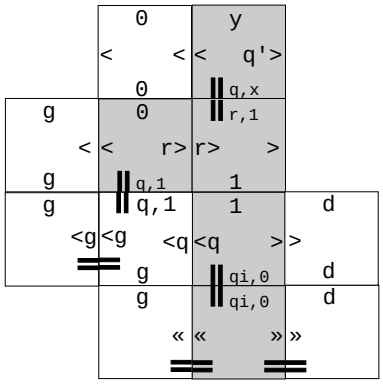


FIGURE *4.26 – Un exemple de calcul par le jeu de tuiles simulant une machine de Turing.*

La simulation est représentée sur la Fig. 4.26. Elle continue tant que l'état final q_f n'apparaît pas. Ainsi, le seul moyen d'obtenir une production finale est si la machine passe par l'état q_f. Ici encore, le calcul est asynchrone : la tête (en gris) doit attendre que le symbole «fin de ruban» (d) lui parvienne avant de poursuivre le calcul.

□

Il est également possible d'utiliser une machine de Turing pour diriger le processus d'auto-assemblage. Comme on l'a vu ci-dessus, la simulation de la machine de Turing consiste à assembler un *diagramme espace-temps* de son calcul, c'est-à-dire un motif où chaque ligne représente une étape du calcul de la machine. *A priori*, si l'on assembler une forme arbitraire, on

n'a pas la place pour mettre ce calcul dans la forme —il faut un «rectangle plein» dans lequel dessiner. Pour créer cette place, il faut assembler non pas une forme, mais un de ses homothétiques. Ainsi, dans chacun des carrés correspondant à une tuile de la forme originelle, il y a la place de simuler la machine de Turing idoine.

On peut ainsi transférer vers l'auto-assemblage deux mesures de complexité classique, la complexité en temps et la complexité de Kolmogorov. Intuitivement, la complexité en temps correspond à un facteur d'homothétie.

La *complexité de Kolmogorov* capture la difficulté à décrire un objet (ici une forme du plan). $\mathfrak{K}(x)$ est la taille (en nombre d'états) de la plus petite machine de Turing sur l'alphabet $\{0,1\}$ qui donne x sur l'entrée vide. En prenant une machine minimale, on obtient le résultat suivant.

Théorème 4.4.13. *[48] Il existe 4 constantes a_0, a_1, b_0, b_1 telles que pour toute forme 4-connexe finie du plan discret $F \subset \mathbb{Z}^2$, si $K_a(F)$ est la taille (en nombre de tuiles) du plus petit système d'auto-assemblage qui assemble un homothétique de F, alors*

$$a_0\mathfrak{K}(F) + b_0 \leq K_a(F)\log K_a(F) \leq a_1\mathfrak{K}(F) + b_1$$

.

Démonstration. Pour assembler un homothétique de F, nous allons simuler une machine de Turing minimale qui décrit F dans chaque rectangle $s(F) \times s(F)$ représentant un point de F. Il y a bien sûr plusieurs façons de décrire une forme par une machine de Turing, nous allons en choisir une qui est adaptée à cette situation. Puisque F est 4-connexe, on peut se donner un *arbre couvrant* A_F de F dont les arêtes sont des arêtes de la grille $\mathbb{Z}^2$. La machine M_F prend en entrée deux entiers (x,y) représentant une position sur $\mathbb{Z}^2$, et renvoie le sous-ensemble de $\{n,s,e,w\}$ indiquant lesquels des voisins de (x,y) sont ses descendants dans A_F (par convention, cet ensemble sera vide si $(x,y) \notin F$). Comme F est une forme finie, cette fonction est calculable. Soit $t(F)$ le temps de calcul maximal de M_F sur les entrées appartenant effectivement à F.

Le système d'auto-assemblage recherché $\mathcal{S}$ est un système qui, dans chaque carré $2t(F) \times 2t(F)$, va simuler M_F sur l'entrée (x,y), puis va lancer la construction des carrés successeurs suivant la sortie de $M_F(x,y)$.

La borne inférieure vient du fait qu'un système à n tuiles peut se décrire avec $n \log n$ bits. □

Les questions de possibilité ou non d'assembler un motif deviennent plus complexes pour les ensembles de formes (ou de motifs) [42], ou des motifs infinis, par exemple fractals [43].

L'importance de la température

La simulation de la machine de Turing utilise fortement le fait d'être à température 2 ; en effet, pour que chaque tuile attende d'avoir à la fois la valeur de la case correspondante au temps précédent et la présence –ou non— de la tête, il faut mettre en place une synchronisation par coopération. Chacune de ces informations est portée par une colle de force 1, et il faut ces deux colles pour ajouter la nouvelle tuile.

Le cas de la température 1. Est-il possible de simuler une machine de Turing à température 1 ? Dans $\mathbb{Z}^3$, une telle simulation est possible. [10] En revanche, dans $\mathbb{Z}^2$, la question est ouverte.

Conjecture 4.4.14. *Le problème suivant est décidable : étant donné un système d'auto-assemblage à température* 1*, a-t-il une production finale de taille finie ?*

Les températures au-delà de 2. Les températures au-delà de 2 ne diffèrent pas sensiblement de la température 2. En effet, il est possible [17] de simuler tout assemblage par un système à température quelconque par un système à température 2. Nous ne donnerons pas ici la définition de simulation, mais le principe est de simuler l'ajout d'une tuile dans le système simulé par l'assemblage d'un carré dans le simulateur. Il existe également [17] un jeu de tuile *universel* $\mathcal{U}$: si on donne à $\mathcal{U}$ comme graine la description d'un système d'auto-assemblage $\mathcal{S}$, $\mathcal{U}$ simule $\mathcal{S}$.

4.4.4 Modèle plus physique : kTAM

Le modèle kTAM

Le modèle de base que nous avons défini ci-dessus permet de répondre à la question « qu'est-il possible d'assembler avec une règle locale ? ». En revanche, il ne permet pas de donner une mesure d'efficacité pour cet assemblage. En effet, le nombre d'ajouts de tuiles sera toujours égal à la surface de la forme à assembler.

Pour répondre à cette préoccupation, il existe un modèle plus proche de la physique, le modèle d'auto-assemblage *cinétique*, ou kTAM. Dans ce modèle, un motif est plongé dans une solution contenant toutes les tuiles du système d'auto-assemblage *à des concentrations données*.

10. La définition d'auto-assemblage donnée dans le plan s'étend de façon naturelle à $\mathbb{Z}^3$.

Définition 4.4.15. *Un système d'assemblage* cinétique *est un système d'auto-assemblage avec la donnée, pour chaque méta-tuile, d'un réel positif, sa* concentration.

On considèrera des systèmes normés, c'est-à-dire des systèmes dans lesquels la somme des concentrations est 1.

La dynamique —c'est à dire les transitions— d'un système cinétique sont décrites par un processus de Markov en temps continu. Les *états* de ce protocole sont les productions du système d'auto-assemblage sous-jacent, et ses transitions sont aussi celles du système d'auto-assemblage. Le *taux* (c'est à dire la vitesse) de la transition $p \to p'$ est donné par la concentration de la tuile ajoutée pour passer de p à p'.

Exemple. Considérons un système cinétique $\mathcal{S}$. Supposons qu'à partir d'une production de droite, il y ait 3 transitions possibles 1, 2 et 3, correspondant à trois ajouts de tuiles possibles, 1 et 2 étant à la même position. Pour chacun de ces ajouts de tuiles, on tire, suivant une loi de Poisson le temps avant que cet ajout ne se fasse. On a donc trois réels aléatoires, t_1 le temps nécessaire pour faire la transition 1, t_2 pour faire la transition 2 et t_3 pour faire la transition 3. Ils sont donnés par :

$$\begin{aligned} P(t_1 > x) &= e^{-k_1 x} \\ P(t_2 > x) &= e^{-k_2 x} \\ P(t_3 > x) &= e^{-k_1 x} \end{aligned}$$

Ainsi, si t_1 a la plus petite valeur, alors on réalisera la transition 1 au bout de ce temps t_1. De même, si t_2 est minimal, c'est la transition 2 qui sera choisie, et la transition 3 pour t_3. Notons que les transitions 1 et 2 sont mutuellement exclusives, puisqu'elles concernent l'ajout de deux tuiles différentes au même endroit, tandis que 3 est indépendante de celles-ci et peut se faire en parallèle. Prenons le cas où on réalise la transition 1 ; on arrive dans un état où seule la transition 3 est possible. Celle-ci sera alors réalisée au bout d'un temps t' avec $P(t' > x) = e^{-k_1(x-t_1)}$ (pour $x > t_1$). Dans le cas où la transition 3 est déclenchée en première, on a : $P(t_3 > x) = e^{-k_1 x}$, ce qui nous donne pour toute valeur de t_1 : $P(t_3 > x | t_3 > t_1) = e^{-k_1(x-t_1)}$ (pour $x > t_1$). On retrouve bien le *parallélisme* entre ces deux transitions : t_3 mettra le même temps à se déclencher que t_1 ait lieu entre temps ou non.

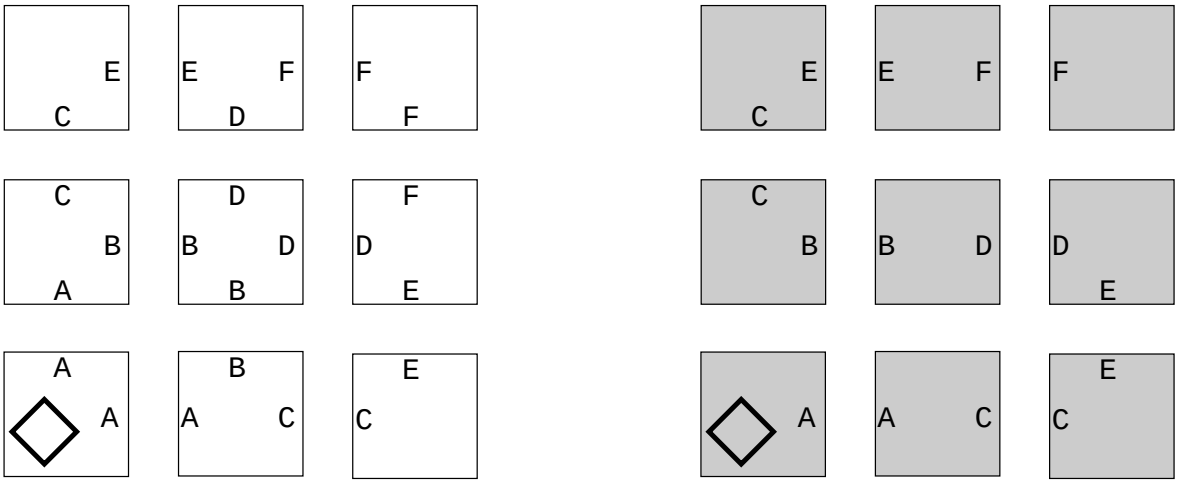


FIGURE 4.27 – *Deux systèmes cinétiques à température* 1 *pour assembler un carré* 3×3.

Temps d'assemblage

Dans un système cinétique, on peut parler du *temps d'assemblage d'une production* : c'est le temps moyen pour passer de la source à cette production, *sachant que le système va donner cette production*.

Exercice 4.4.3. Pour chacun des deux systèmes de la Fig. 4.27 (à température 1), donner le temps d'assemblage pour l'unique production qui recouvre un carré 3×3. Toutes les concentrations sont égales à $1/9$. Dans chacun des cas, la source est la tuile en bas à gauche.

Temps parallèle et temps Markov. Calculer le temps d'assemblage des productions à l'aide de la chaîne de Markov est un peu fastidieux. On peut donner une autre définition du temps d'assemblage qui tienne compte du parallélisme. La première idée est la suivante : considérer que les ajouts de tuiles indépendants se font en parallèle revient à considérer qu'ils se font tous en même temps. On pourrait donc prendre une dynamique ou chaque transition représente non pas l'ajout d'une tuile possible, mais l'ajout d'une tuile *à chaque endroit possible*. Malheureusement, pour certains systèmes, cette dynamique, la *dynamique parallèle* ne donne pas les mêmes productions que la dynamique asynchrone considérée jusque-là.

Exercice 4.4.4. Montrer que toute production d'un système $\mathcal{S}$ suivant la dynamique parallèle est aussi une production suivant la dynamique asynchrone.

Donner un exemple de système d'auto-assemblage avec une production qui n'est pas possible suivant la dynamique parallèle.

Donner un exemple où la production en question est *finale*.

En observant les productions de la dynamique parallèle, on constate que dans chaque production, toute tuile ajoutée au temps t est adjacente à au moins une tuile ajoutée au temps $t-1$. Dans la dynamique asynchrone en revanche, une tuile peut être ajoutée longtemps après toutes ses voisines. Dans certains cas, il est possible de reconstituer un «historique parallèle plausible», l'*ordre d'assemblage* qui nous permet notamment de calculer le temps d'assemblage d'une production.

Définition 4.4.16. *Soit $\mathcal{S}$ un système d'auto-assemblage, P une production de $\mathcal{S}$, et $<_P$ une relation d'ordre partiel sur les positions de* $\mathrm{dom}(P)$. *$<_P$ est un* ordre d'assemblage *pour P si, pour toute suite de transitions $(t_i)_{i\leq k}$ de $\mathcal{S}$ arrivant à P, on a que pour tout i, si t_i consiste à ajouter une tuile à la position z à une production p, alors $\forall d(z) in$* $\mathrm{dom}(p)$, *$\not< >_P d(z)$.*

Un ordre d'assemblage est donc un ordre partiel compatible avec les ajouts de tuiles : chaque tuile est supérieure pour cet ordre aux tuiles *adjacentes* qui la précèdent. Un tel ordre, quand il existe, permet de donner une estimation du temps d'assemblage.

Théorème 4.4.17. *Soit $\mathcal{S}$ un système cinétique d'auto-assemblage où toutes les concentrations sont égales à 1, et dont la source est réduite à une seule tuile. Soit P une production ayant un* ordre d'assemblage *$<_P$. Soit $p(<_P)$ la profondeur de $<_P$ (la taille de la plus longue suite croissnte de P), et $t(P)$ le temps d'assemblage de P. Alors $t(P) = \Omega(p(<_P)$.*

Programmation par les concentrations

L'utilisation des concentrations permet de diriger l'assemblage. Par exemple, les productions finales du jeu de tuiles de la Fig. 4.21 sont les carrés de toutes tailles supérieures à 2, mais en jouant sur la concentration de la tuile qui arrête la croissance de la diagonale, on peut contrôler la taille moyenne du carré obtenu.

Le modèle cinétique permet également de « tricher » avec le théorème 4.4.13. En effet, il est possible de cacher de l'information dans les concentrations. Celles-ci sont des réels, et peuvent donc chacune contenir autant de bits que souhaité d'information. Il existe un système d'auto-assemblage qui échantillone ces concentrations pour récupérer l'information, et assemble ainsi une forme arbitraire.

Théorème 4.4.18. *[13] Il existe un unique système d'auto-assemblage $\mathcal{S}_u$, et une fonction $\kappa : \mathcal{P}(\mathbb{Z}^2) \to \mathbb{R}^{\mathcal{S}_u}$ tels que, pour tout $\epsilon > 0$, pour toute forme F*

4-connexe de $\mathbb{Z}^2$, le système cinétique $\mathcal{S}_u$ avec les concentrations $\kappa(P)$ assemble une production finale dont la forme est un homothèthique de F avec une probabilité $p > 1 - \epsilon$.

Assemblage avec erreurs

Ces systèmes plus physiques permettent aussi de représenter le risque d'erreur au cours de l'assemblage. Dans une implémentation physique —ou chimique— de l'auto-assemblage, il est possible que des transitions «illégales» aient lieu. Ces transitions peuvent amener à la formation de motifs stables qui ne sont pas des productions. Des systèmes d'auto-assemblages avec un mécanisme de correction d'erreur [53] permettant de s'assurer que les productions obtenues seront celles du modèle classique avec forte probabilité.

4.4.5 Autres développements

Un nombre important de variantes de l'auto-assemblage ont été introduites, à la fois pour représenter plus finement la plausibilité physique de l'assemblage et pour introduire d'autres moyens de contrôler le processus d'auto-assemblage. On a ainsi notamment des modèles où des motifs déjà assemblés peuvent interagir entre eux [16], des modèles où la température peut changer au cours de l'assemblage [37].

On trouve aussi une étude plus fine de la dynamique de l'assemblage, avec une notion de simulation et d'universalité intrinsèque [17] et un résultat de non-universalité de la température 1 [40] ; on a là le début d'une théorie de la compléxité *propre à l'auto-assemblage*.

4.5 Conclusion

Les modèles présentés sont ancrés dans l'espace et montrent la richesse du domaine. Les possibilités de calcul sont immenses. Ces modèles apportent un autre type d'algorithmique où lieu, distance, agencement… sont autant de contraintes et de possibilités.

Sans surprise, il est possible de calculer au sens de Turing. Dès que l'on peut profiter de la continuité du temps et de l'espace, calculs analogique et hyper-calcul arrivent également.

Cela reste vrai tant que le contexte est idéal : ni erreur ni approximation, pas de limite au découpage de l'espace ni à la densité d'information.

Le grand absent des modèles où la spatialisation est le paradigme est l'automate cellulaire. Comme il a été le sujet d'un cours de l'école

précédente [44], il n'est pas question de l'aborder. Des liens avec les modèles présentés existent néanmoins, par exemple la notion de cône espace-temps et de causalité.

Le signal est une notion clé pour les automates cellulaires. Il s'agit là d'un motif qui se répète périodiquement et définit ainsi la vitesse comme pour les machines à signaux et les automates de Mondrian.

Il existe d'autres pendants continus des automates cellulaires [32, 49, par exemple] et d'autres modèles qui auraient pu être présentés, mais il ne s'agit pas ici de faire un catalogue.

Dans l'auto-assemblage, géométrie et synchronisation sont indissociables. Le signal y est — informellement ici, même si des définitions plus précises existent — un vecteur de transmission d'information, mais doit également construire son propre milieu de propagation.

Bibliographie

[1] L. Adleman, Q. Cheng, A. Goel, M.-D. Huang, and H. Wasserman. Linear self-assemblies : Equilibria, entropy and convergence rates. In *In Sixth International Conference on Difference Equations and Applications*. Taylor and Francis, 2001.

[2] H. Andréka, I. Németi, and P. Németi. General relativistic hypercomputing and foundation of mathematics. *Nat. Comput.*, 8(3) :499–516, 2009.

[3] E. Asarin and O. Maler. Achilles and the Tortoise climbing up the arithmetical hierarchy. In *FSTTCS '95*, number 1026 in LNCS, pages 471–483, 1995.

[4] E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoret. Comp. Sci.*, 138(1) :35–65, 1995.

[5] F. Becker, M. Chapelle, J. Durand-Lose, V. Levorato, and M. Senot. Abstract geometrical computation 8: Small machines, accumulations & rationality. Submitted, 2013.

[6] C. H. Bennett. Notes on the history of reversible computation. *IBM J. Res. Dev.*, 32(1) :16–23, 1988.

[7] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and real computation*. Springer, New York, 1998.

[8] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc.*, 21(1) :1–46, 1989.

[9] O. Bournez. Some bounds on the computational power of piecewise constant derivative systems (extended abstract). In *ICALP '97*, number 1256 in LNCS, pages 143–153, 1997.

[10] O. Bournez. Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoret. Comp. Sci.*, 210(1) :21–71, 1999.

[11] O. Bournez. Some bounds on the computational power of piecewise constant derivative systems. *Theory Comput. Syst.*, 32(1) :35–67, 1999.

[12] J.-C. Carrega. *Théorie des corps - La règle et le compas*. Hermann, 1981.

[13] H.-L. Chen, D. Doty, and S. Seki. Program size and temperature in self-assembly. In *ISAAC 2011 : Proceedings of the 22nd International Symposium on Algorithms and Computation*, volume 7074 of *Lecture Notes in Computer Science*, pages 445–453. Springer-Verlag, 2011.

[14] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15 :1–40, 2004.

[15] J.-P. Delahaye. *Information, complexité et hasard*. Hermès science publications, 1999.

[16] E. D. Demaine, M. J. Patitz, T. A. Rogers, R. T. Schweller, S. M. Summers, and D. Woods. The two-handed tile assembly model is not intrinsically universal. In *ICALP : 40th International Colloquium on Automata, Languages and Programming*, volume 7965 of *LNCS*, pages 400–412, Riga, Latvia, July 2013. Springer. Arxiv preprint : arXiv:1306.6710.

[17] D. Doty, J. H. Lutz, M. J. Patitz, R. T. Schweller, S. M. Summers, and D. Woods. The tile assembly model is intrinsically universal. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 439–446, Oct. 2012. Arxiv preprint : arXiv:1111.3097.

[18] D. Duchier, J. Durand-Lose, and M. Senot. Fractal parallelism: Solving SAT in bounded space and time. In C. Otfried, K.-Y. Chwa, and K. Park, editors, *Int. Symp. on Algorithms and Computation (ISAAC '10)*, number 6506 in LNCS, pages 279–290. Springer, 2010.

[19] D. Duchier, J. Durand-Lose, and M. Senot. Computing in the fractal cloud: modular generic solvers for SAT and Q-SAT variants. In M. Agrawal, B. S. Cooper, and A. Li, editors, *Theory and Applications of Models of Computations (TAMC '12)*, number 7287 in LNCS, pages 435–447. Springer, 2012.

[20] J. Durand-Lose. *Calculer géométriquement sur le plan – machines à signaux*. Habilitation à Diriger des Recherches, École Doctorale STIC, Université de Nice-Sophia Antipolis, 2003. In French.

[21] J. Durand-Lose. Abstract geometrical computation for black hole computation (extended abstract). In M. Margenstern, editor, *Machines, Computations, and Universality (MCU '04)*, number 3354 in LNCS, pages 176–187. Springer, 2005.

[22] J. Durand-Lose. Abstract geometrical computation 1: Embedding black hole computations with rational numbers. *Fund. Inf.*, 74(4) :491–510, 2006.

[23] J. Durand-Lose. Abstract geometrical computation and the linear Blum, Shub and Smale model. In B. S. Cooper, B. Löwe, and A. Sorbi, editors, *Computation and Logic in the Real World, 3rd Conf. Computability in Europe (CiE '07)*, number 4497 in LNCS, pages 238–247. Springer, 2007.

[24] J. Durand-Lose. The signal point of view: from cellular automata to signal machines. In B. Durand, editor, *Journées Automates cellulaires (JAC '08)*, pages 238–249, 2008.

[25] J. Durand-Lose. Abstract geometrical computation and computable analysis. In J. F. Costa and N. Dershowitz, editors, *Int. Conf. on Unconventional Computation 2009 (UC '09)*, number 5715 in LNCS, pages 158–167. Springer, 2009.

[26] J. Durand-Lose. Abstract geometrical computation 4: small Turing universal signal machines. *Theoret. Comp. Sci.*, 412 :57–67, 2011.

[27] J. Durand-Lose. Abstract geometrical computation 5: embedding computable analysis. *Nat. Comput.*, 10(4) :1261–1273, 2011. Special issue on Unconv. Comp. '09.

[28] J. Durand-Lose. Geometrical accumulations and computably enumerable real numbers (extended abstract). In C. S. Calude, J. Kari, I. Petre, and G. Rozenberg, editors, *Int. Conf. Unconventional Computation 2011 (UC '11)*, number 6714 in LNCS, pages 101–112. Springer, 2011.

[29] J. Durand-Lose. Abstract geometrical computation 6: a reversible, conservative and rational based model for black hole computation. *Int. J. Unconventional Computing*, 8(1) :33–46, 2012.

[30] J. Durand-Lose. Irrationality is needed to compute with signal machines with only three speeds. In P. Bonizzoni, V. Brattka, and B. Löwe, editors, *CiE '13, The Nature of Computation*, number 7921 in LNCS, pages 108–119. Springer, 2013. Invited talk for special session *Computation in nature*.

[31] G. Etesi and I. Németi. Non-Turing computations via Malament-Hogarth space-times. *Int. J. Theor. Phys.*, 41(2) :341–370, 2002.

[32] M. Hagiya. Discrete state transition systems on continuous space-time : A theoretical model for amorphous computing. In C. Calude, M. J. Dinneen, G. Păun, M. J. Pérez-Jiménez, and G. Rozenberg, editors, *Unconventional Computation, 4th Int. Conf., UC '05, Sevilla, Spain, October 3-7, 2005, Proceedings*, volume 3699 of *LNCS*, pages 117–129. Springer, 2005.

[33] M. L. Hogarth. Deciding arithmetic using SAD computers. *Brit. J. Philos. Sci.*, 55 :681–691, 2004.

[34] U. Huckenbeck. Euclidian geometry in terms of automata theory. *Theoret. Comp. Sci.*, 68(1) :71–87, 1989.

[35] U. Huckenbeck. A result about the power of geometric oracle machines. *Theoret. Comp. Sci.*, 88(2) :231–251, 1991.

[36] G. Jacopini and G. Sontacchi. Reversible parallel computation: an evolving space-model. *Theoret. Comp. Sci.*, 73(1) :1–46, 1990.

[37] M.-Y. Kao and R. T. Schweller. Reducing tile complexity for self-assembly through temperature programming. In *SODA*, pages 571–580. ACM Press, 2006.

[38] Y. Lecerf. Machines de Turing réversibles. Récursive insolubilité en $n \in \mathbb{N}$ de l'équation $u = \theta^n u$, où θ est un isomorphisme de codes. *Comptes rendus des séances de l'académie des sciences*, 257 :2597–2600, 1963.

[39] K. Meer and C. Michaux. A survey on real structural complexity theory. *Bulletin of the Belgian Mathematical Society*, 4 :113–148, 1997.

[40] P.-E. Meunier, M. J. Patitz, S. M. Summers, G. Theyssier, A. Winslow, and D. Woods. Intrinsic universality in tile self-assembly requires cooperation. In *SODA*, pages 752–771. SIAM, 2014.

[41] K. Morita, A. Shirasaki, and Y. Gono. A 1-tape 2-symbol reversible Turing machine. *Transactions of the IEICE*, E 72(3) :223–228, Mar. 1989.

[42] M. J. Patitz and S. M. Summers. Self-assembly of decidable sets. In *Proceedings of The Seventh International Conference on Unconventional Computation (Vienna, Austria, August 25-28, 2008)*, pages 206–219, 2008.

[43] M. J. Patitz and S. M. Summers. Self-assembly of discrete self-similar fractals. *Natural Computing*, 1 :135–172, 2010.

[44] S. Peyronnet, editor. *Informatique Mathématique — une photographie en 2014*. Presses Universitaires de Perpignan, 2014.

[45] R. M. Robinson. Undecidability and nonperiodicity for tilings on the plane. *Inventiones Mathematicae*, 12(3) :177–209, september 1971.

[46] M. Senot. *Geometrical model of computation : fractals and complexity gaps*. Thèse de doctorat, Université d'Orléans, June 2013.

[47] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Co., Boston, Massachusetts, 1997.

[48] D. Soloveichik and E. Winfree. Complexity of self-assembled shapes. In C. Ferretti, G. Mauri, and C. Zandron, editors, *DNA*, volume 3384 of *Lecture Notes in Computer Science*, pages 344–354. Springer, 2004.

[49] I. Takeuti. Transition systems over continuous time-space. *Electr. Notes Theor. Comput. Sci.*, 120 :173–186, 2005.

[50] H. Wang. Proving theorems by pattern recognition—ii. *Bell System Tech. Journal*, 40(1) :1 – 41, 1961.

[51] K. Weihrauch. *Introduction to computable analysis*. Texts in Theoretical computer science. Springer, Berlin, 2000.

[52] E. Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, Caltech, 1998.

[53] E. Winfree and R. Bekbolatov. Proofreading tile sets : Error correction for algorithmic self-assembly. In J. Chen and J. H. Reif, editors, *DNA*, volume 2943 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.

[54] D. Woods and T. Neary. The complexity of small universal Turing machines : A survey. *Theoret. Comp. Sci.*, 410(4–5) :443–450, 2009.