Rapport de DEA
Signaux rapides en plusieurs dimensions

Jean-Christophe Dubacq

7 juillet 1998

Résumé

Les automates cellulaires ont des propriétés très différentes selon qu'ils soient en dimension 1 ou 2 d'un point de vue de la calculabilité. Après avoir défini une notion de signal, on essaye ici d'analyser la puissance de calcul intrinsèque des automates cellulaires en une ou deux dimensions d'après la forme des signaux qu'il est possible de générer.

Mots clef : Automates cellulaires, signaux, géomètrie discrète

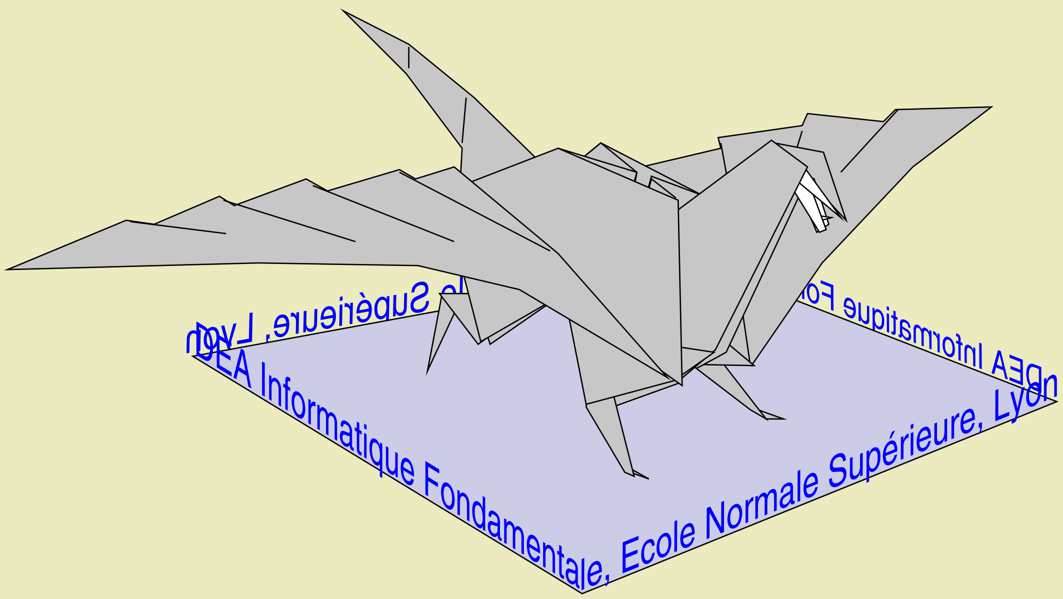


Table des matières

1 Introduction **3**

2 Définitions **5**
- 2.1 Automates cellulaires 5
- 2.2 Diagramme espace-temps 6
- 2.3 Signaux 6
- 2.4 Fonctions Fischer-constructibles 7
- 2.5 Quelques fonctions et notions utilisées 7

3 Signaux en dimension 1 **9**
- 3.1 Outils techniques 9
- 3.2 Fischer-constructibilité et signaux 11
- 3.3 Signaux en dimension 1 14
- 3.4 Fischer-construction du $\log^\star$ 14

4 Signaux bidimensionnels **17**
- 4.1 Représentation d'algorithmes cellulaires plans 17
- 4.2 Transformer une fonction Fischer-constructible 1D en signal 2D . . . 18
- 4.3 Hiérarchie de Grzegorczyk 18
- 4.4 Génération du $\log^\sharp$ en 2D 18
- 4.5 Limitations en 2D 21

5 Conclusion **25**

Table des figures

1 Voisinages de Moore 5
2 Détournement des signaux par un délai 9
3 Détournement des signaux par un décalage 10
4 Transformation Fischer/signal 12
5 Calcul de la réciproque 12
6 Calcul de la réciproque - signaux réels 13
7 Calcul de la réciproque 14
8 Calcul du $\log^\star$ 16
9 Calcul du $\log^\star$, phase initiale 16
10 Voisinages & temps réel 17
11 Transformation Fischer 1D/Quasi-temps réel 2D 19
12 Vue latérale 20
13 La première couche de calcul 22
14 Synchronisation des couches de calcul 23
15 Schéma des dépendances des diagonales $\mathfrak{D}$ 24

1 Introduction

Le travail qui a été effectué au Laboratoire de l'Informatique du Parallélisme dans le cadre du stage de DEA a porté sur un modèle simple de parallélisme massif, les automates cellulaires. Les automates cellulaires sont les modèles fondamentaux du *parallélisme synchrone*. Ils sont utilisés pour simuler des systèmes dynamiques en physique, ou en biologie de l'évolution des populations. Ils se rapprochent aussi beaucoup des tableaux itératifs et des systèmes systoliques dans leur mode de fonctionnement.

De manière informelle, un automate cellulaire est un ensemble de cellules (généralement organisé en une grille de dimension fixée d, ligne, plan, espace $\mathbb{Z}^3$), qui contiennent chacune un automate fini, tous identiques, dont l'état change de façon synchrone en fonction de l'état de ses voisins. Par synchrone, on entend l'existence d'une horloge globale qui commande le changement d'état. Parmi les automates cellulaires, on peut citer le jeu de la vie de Conway. Une configuration est l'état de toutes les cellules à un instant donné, et un automate cellulaire transforme des configurations en configurations. Avec cette façon d'envisager la dynamique émerge tout de suite une notion de temps discret.

Les propriétés algorithmiques des automates cellulaires sont très différentes selon que l'on se place dans un modèle linéaire (unidimensionnel) ou dans un modèle plan (bidimensionnel), voire de dimensions encore supérieures. C'est l'étude de cette frontière qui a été la partie la plus importante de ce stage.

La frontière entre le plan et la ligne est réellement fondamentale. Ainsi, du point de vue de la calculabilité, savoir si un automate cellulaire est bijectif est décidable en temps quadratique pour un automate cellulaire de dimension 1 (voir [?]) et indécidable pour un automate de dimension 2 comme expliqué dans [?]. C'est donc un point très sensible que cette étude, qui est fondamentale parce que mal comprise.

On s'est intéressé plus particulièrement dans ce rapport à la notion de signal, qui est une notion *naturelle*, et à sa formalisation sur des automates cellulaires. Ce genre de formalisation a déjà été utilisé, dans [?, ?, ?, ?] et a été étudié plus en détail dans [?]. Le signal est une notion forte dans les automates cellulaires : c'est en quelque sorte un motif de progression dans les états des cellules. On donnera une définition précise des signaux, qui servira de support à tout ce rapport. Beaucoup d'algorithmes peuvent être vus comme de la transmission de signaux, avec des interactions n'ayant lieu que lors de leur rencontre (absorption, émission de nouveaux signaux).

Pour étudier la puissance des automates cellulaires selon leur dimension, le choix s'est porté sur la forme des signaux qu'ils sont capables de générer. Si l'étude en dimension 1 de ces capacités a déjà été amorcée (dans [?] et [?]), l'ensemble exact en est encore mal connue, bien que l'on connaisse à la fois des choses possibles et impossibles à faire. En dimension 2, aucun résultat n'avait été avancé à ce jour.

Le jugement que l'on porte sur cette approche doit tenir compte du fait que ces automates cellulaires ont un fonctionnement particulier sans entrée, qui fait que l'on n'évalue pas la capacité de décodage de l'entrée par l'automate, mais réellement sa puissance de calcul intrinsèque.

L'étude de ce qui est possible en dimension 1 s'est révélée indispensable pour la suite. En effet, bien des méthodes classiques peuvent être étendues à plusieurs dimensions. De plus, il était indispensable de bien connaître les limites de ce que l'on sait faire en dimension 1 afin de faire remarquer la différence ultérieurement. A la fin de cette analyse, on trouvera notamment un algorithme original et utile par la suite permettant de construire la tour d'exponentielle de la même façon que Fischer construisait la fonction qui à n associe le $n^{\text{ième}}$ nombre premier dans [?].

L'approche des résultats en dimension 2 a donné des résultats nombreux, dont le moindre n'est pas une conjecture qui permettrait de classer exactement les automates cellulaires en fonction de leur dimension. Cette conjecture pose un lien entre la hiérarchie de fonctions décrite par Grzegorczyk dans [?] par exemple, et la dimension nécessaire pour calculer la fonction sur un automate cellulaire.

La hiérarchie de Grzegorczyk est en fait une hiérarchie des fonctions primitives récursives en fonction du nombre minimal de récurrences nécessaire pour calculer la fonction. Les exemples typiques des différents échelons de cette hiérarchie sont les fonctions d'Ackermann.

L'algorithmique cellulaire bidimensionnelle a elle-même amené des problèmes de représentations. Si l'on utilisait habituellement le plan pour exposer les algorithmes unidimensionnels, ce n'est plus directement possible en 2D, qu'il fallait représenter en 3D. Des solutions ont été trouvées, et appuient les autre preuves de cette section.

On a cherché, pour appuyer la conjecture, à prouver différents résultats à la fois de limitation et de possibilités sur les automates cellulaires plans. On a notamment construit en 2D un signal qu'il n'est pas possible de construire en dimension 1, à savoir un signal de vitesse $n + \log^\star(n)$. On a par ailleurs montré certains résultats d'impossibilité en dimension 2, et essayé de présenter certaines généralisations à une dimension quelconque.

Un résultat annexe mais toutefois intéressant s'est dégagé sur la différence de voisinages : lorsque l'on prend en dimension 2 un voisinage élémentaire du type de von Neumann, la vitesse maximale des signaux non-linéaires comprend au moins une composante supérieure au $\log\log$. Tandis que l'on peut faire des signaux à des vitesses beaucoup plus proches du linéaire avec un voisinage élémentaire du type de Moore, avec une composante seulement en $\log^\star$. Cette différence montre que dans un réseau en grille 4-connecté on est beaucoup plus limité (en puissance de calcul) que dans un réseau en grille 8-connecté. Ceci tient certainement en partie à la nature du modèle mais provient aussi sans doute d'une différence entre les interconnexions sus-citées.

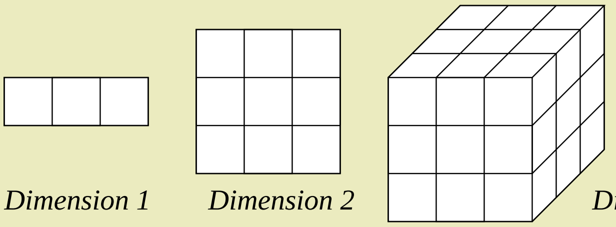


FIG. 1 - *Voisinages de Moore*

2 Définitions

Dans cette partie, on va donner l'ensemble des définitions qu'il est utile de bien connaître pour comprendre la suite. Outre la définition formelle, on y trouvera également une explication de certains choix qu'il a fallu faire lors de la définition.

2.1 Automates cellulaires

Définition 1 (Automate cellulaire) *Les automates cellulaires sont des quadruplets* $(d, \mathcal{S}, \mathcal{N}, \phi)$ *où* d *est la* dimension *de l'AC,* $\mathcal{S}$ *est un ensemble fini d'états,* $r \in \mathbb{N}$ *est le nombre de voisins,* $\mathcal{N} = \{z_1, z_2, \ldots, z_r\}$ *est un sous-ensemble fini de* $\mathbb{Z}^d$*,* voisinage *de l'AC, et* ϕ *est la* fonction de transition locale *définie de* $\mathcal{S}^{\mathcal{N}}$ *dans* $\mathcal{S}$

Pour d'autres formalismes, on pourra se reporter à [?].

On distinguera par la suite dans l'ensemble d'états un état quiescent, telle que $\phi(q, \ldots, q) = q$.

Dans la suite de ce rapport, le voisinage considéré pour les automates cellulaires linéaire est le voisinage $(-1, 0, 1)$, et pour les automates de dimension supérieure, le voisinage est l'hypercube de rayon 1 et de dimension d (voisinage de Moore en dimension 2). Le voisinage est donc l'hypercube de dimension d de coté 3 (voir Figure 1).

Les automates cellulaires travaillent sur une configuration (fonction de $\mathbb{Z}^d$ dans $\mathcal{S}$), et la transforment en une autre configuration, l'état de chaque nouvelle cellule étant calculé en appliquant la fonction de transition locale au voisinage relatif à la cellule, de façon synchrone. Formellement,

$$G(c(i)) = \Phi(c(i + z_1), \ldots, c(i + z_r))$$

. On peut donc établir une notion de temps discret, traduisant l'évolution successive des configurations. L'instant 0 est défini comme celui de la configuration d'origine.

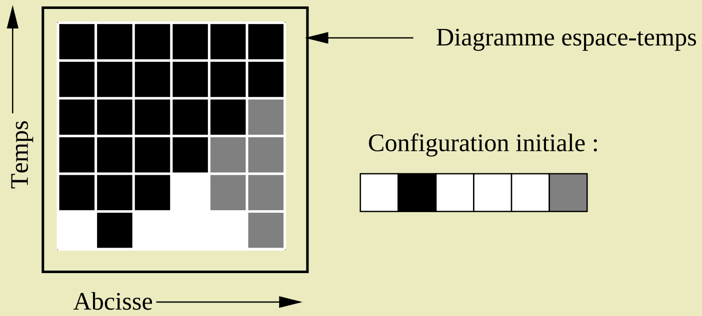
On veut en fait faire calculer des fonctions à un automate cellulaire, en lui faisant décrire un signal épousant la forme de la fonction. Mais comme un automate cellulaire travaillant sur une entrée, il faut éviter que la fonction soit encodée dans la configuration initiale de l'automate, réduisant alors le travail de l'automate à un travail de décodage. On considérera donc uniquement le cas où à l'instant 0 la configuration d'origine est réduite à un ensemble de cellules quiescentes sauf en un point (souvent appelé origine, et qui sera en général pris comme origine du système de coordonnées). En d'autres termes, cet ensemble fini sera par la suite en général réduit à un seul point. On nomme parfois ces automates cellulaires *automates cellulaires à impulsion.*

On définit donc la fonction $s : \mathbb{Z}^d \times \mathbb{N} \longrightarrow \mathcal{S}$ qui à une cellule (un jeu de coordonnées dans $\mathbb{Z}^d$) et à un instant t donnés associe l'état de la cellule au temps t. Un jeu de coordonnées (z, t) est un *site*: c'est en fait un des éléments du diagramme espace-temps du fonctionnement de l'automate.

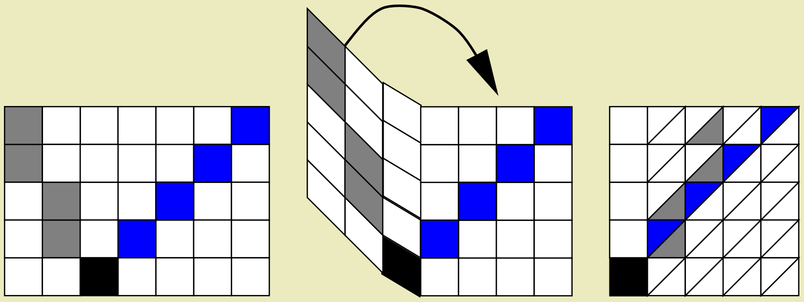
2.2 Diagramme espace-temps

La notion de diagramme espace-temps est indispensable pour bien faire comprendre les algorithmes employés, et aussi pour les décrire. Une notion de diagonale, par exemple, référera souvent à l'état d'une cellule dont l'abscisse évolue de façon linéaire avec le temps, car c'est *graphiquement* une diagonale.

Un diagramme espace-temps est simplement la superposition (graphique) de toutes les configurations. Voici un diagramme espace-temps très simple d'un automate cellulaire qui prend comme état le plus foncé de ses voisins :



Un bon nombre de manipulations de bases sont possibles avec les diagrammes espace-temps. Toutes ces manipulations sont reflétées par une transformation systématique de l'automate qui génère le diagramme. Par exemple, le *pliage* est une opération qui consiste à plier (littéralement) le diagramme en deux le long d'un axe vertical. Il suffit pour cela que chaque cellule du côté vers lequel on plie simule à la fois la cellule d'origine et la cellule pliée. La cellule qui, développée dans le temps (c'est une seule cellule qui forme un axe), est l'axe de pliage simule une seule cellule et n'utilise en fait qu'un seul de ses voisins, comme dans le schéma suivant :



De telles transformations seront utilisées souvent et sont abondantes dans la littérature. Leur exactitude ne sera pas prouvée ici. Un certain nombre d'entre elles sont expliquées plus loin.

Il faut toutefois faire attention avec ce genre de transformations, car leur nombre doit rester fini. Sans cela, le nombre d'états nécessaire pour faire l'automate serait infini (ainsi, on ne peut pas faire réaliser le calcul d'un automate cellulaire par un nombre fini de cellules, car il faudrait pour cela replier un nombre infini de fois le diagramme espace-temps).

2.3 Signaux

Un support de signal $\mathfrak{S}$ est un ensemble de sites $\{(z_0, t_0), (z_1, t_0+1), \ldots, (z_k, t_k)\}$ tel que z_{i+1} soit dans le voisinage de z_i. Un automate cellulaire $\mathcal{A}$ génère un signal $\mathfrak{S}$ si et seulement si il existe un sous-ensemble $\mathcal{S}_{\mathfrak{S}}$ de ses états et un voisinage du support $\mathfrak{V}$ (sur-ensemble du support) tels que :

$$\forall u \in \mathfrak{V}, s(u) \in \mathcal{S}_{\mathfrak{S}} \Longleftrightarrow u \in \mathfrak{S}$$

La plupart des algorithmes cellulaires peuvent être considérés comme des envois de signaux, les règles n'étant définies que pour gérer les intersections de signaux. L'étude des signaux constructibles est très importante, car c'est ainsi que l'on bâtit les algorithmes usuels.

Un signal est d'une certaine *pente* lorsqu'il se déplace dans le temps suivant une certaine fonction. Un signal de pente rationnelle se déplace suivant une fonction linéaire du temps : par exemple, un signal de pente $2/3$ se déplace deux coups à droite et reste un coup sur place. On peut définir des signaux de pente f (où f est une fonction) comme un support de signal $\mathfrak{S} = (f(0), 0), \ldots, (f(n), n), \ldots)$, donc un signal se déplaçant à la vitesse de la fonction.

Theorème 1 *Les signaux à pente rationnelle, de forme exponentielle ou de forme* $\sqrt{n} + n$ *ou polynômiale sont constructibles par automate cellulaire.*

Une autre notion très importante est la notion de *temps réel*, qui est l'ensemble des sites atteints par un signal se déplaçant à vitesse maximale dans toutes les directions. On pourra regarder la Figure 10 pour une illustration de diverses notions de temps réel en dimension 2.

2.4 Fonctions Fischer-constructibles

Définition 2 (Fischer-constructibilité) *Soit* f *une fonction croissante de* $\mathbb{N} \to \mathbb{N}$*. On dit que* f *est d-Fischer-constructible s'il existe un automate cellulaire de dimension* d *et un sous-ensemble* D *des états de l'automate tel que :*

$$s(0, i) \in D \Longleftrightarrow \exists! n \in \mathbb{N}, f(n) = i.$$

Dans la définition précédente, $s(0, i)$ désigne l'état de la cellule 0 à l'instant i. On déduit de la définition que f est *strictement* croissante. Il s'agit de marquer (par un état ou un sous-ensemble d'états) sur la cellule 0 toutes les images de la fonction qui est alors dite Fischer-constructible.

La définition originelle de Fischer était en dimension 1. Le choix qui a été fait pour l'extension à plusieurs dimensions a été de calculer dans tout l'espace, et non pas sur une ligne. On s'attend donc naturellement à une augmentation de la puissance de calcul.

2.5 Quelques fonctions et notions utilisées

On va utiliser dans la suite de ce rapport quelques notations pas toujours usitées, et dont on donne ici la définition :

Fonction primitive récursive Les fonctions primitives récursives sont l'image par clôture algébrique de l'identité, de la projection et du successeur par composition et récurrence.

Réciproque On définit la réciproque d'une fonction strictement croissante comme suit :

$$f^{-1}(y) = max\{x \mid f(x) \leq y\}.$$

Couple Fischer-constructible Puisque l'on ne va essayer de Fischer-construire que des fonctions strictement croissantes, on définit la notion de co-Fischer-constructibilité. Par exemple, le logarithme est dit co-Fischer-constructible parce que l'exponentielle est Fischer-constructible. Il faut remarquer qu'une marque étant faite sur l'axe lors de la construction de la fonction Fischer-constructible à chaque fois que l'on a un antécédent différent, cela correspond exactement à marquer sur l'axe toute augmentation de la fonction dans le cas de la co-Fischer-constructibilité.

Notation des itérées On notera $f^{(i)}$ la fonction définie par $f^{(0)} = Id$ et $f^{(i+1)} = f \circ f^{(i)}$.

Quelques fonctions primitives récursives **Ackermann** La fonction d'Ackermann $\mathfrak{A}$ est une fonction à deux variables, définie par récurrence :

- $\mathfrak{A}(n, 0) = 1$;
- $\mathfrak{A}(0, m) = 1$;
- $\mathfrak{A}(n, m) = \mathfrak{A}(\mathfrak{A}(n-1, m), m-1)$.

On voit ainsi que $\mathfrak{A}(., 1)$ est l'identité, $\mathfrak{A}(., 2)$ est la multiplication, $\mathfrak{A}(., 3)$ est l'exponentiation, etc. La fonction $n \mapsto \mathfrak{A}(n, n)$ n'est pas primitive récursive, mais récursive. Le point intéressant est que chacun des niveaux selon le deuxième indice de cette fonction correspond à un étage différent de la hiérarchie de Grzegorczyk.

$\log^\star$ Le $\log^\star$ est la fonction de $\mathbb{N}$ dans $\mathbb{N}$ définie par : $\log^\star(n) = max\{i, \log^{(i)} \geq 1\}$. C'est la fonction réciproque de la tour d'exponentielle ($\mathfrak{A}(., 4)$).

$\log^\sharp$ Le $\log^\sharp$ est la fonction de $\mathbb{N}$ dans $\mathbb{N}$ définie par : $\log^\sharp(n) = max\{i, \log^{\star(i)} \geq 1\}$. C'est la fonction réciproque de $\mathfrak{A}(., 5)$.

$\log^{\sharp^\sharp}$ Le $\log^{\sharp^\sharp}$ est la fonction de $\mathbb{N}$ dans $\mathbb{N}$ définie par : $\log^{\sharp^\sharp}(n) = max\{i, \log^{\sharp(i)} \geq 1\}$. C'est la fonction réciproque de $\mathfrak{A}(., 6)$.

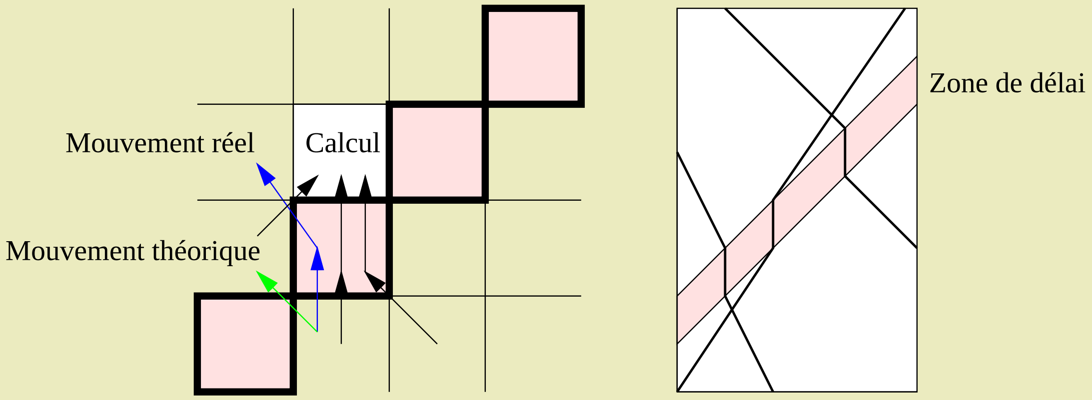


Fig. 2 - *Détournement des signaux par un délai*

3 Signaux en dimension 1

Bien que l'étude des signaux en dimension 1 soit le domaine le plus connu et le plus simple, son étude reste très importante. Bon nombre des outils de base opèrent en dimension 1 et ont une construction hautement généralisable à des dimensions supérieures. Comme on va le voir très rapidement, fonctions Fischer-constructibles et signaux sont fortement liés. On va donc d'abord parler de certaines des propriétés des fonctions Fischer-constructibles, on étudiera ensuite les liens entre Fischer-constructibilité et signaux, puis on étudiera plus précisément les limites de ce qu'il est possible de faire en dimension 1.

Une des principales bases de travail est [?], dans lequel on trouve le théorème très important suivant :

Theorème 2 (Stabilité) *L'ensemble des fonctions Fischer-constructibles en dimension 1 est stable par addition, multiplication par un scalaire rationnel, multiplication et composition.*

Le principe de chacune des preuves est une construction géométrique. Ce théorème fonde l'étude des signaux, car il donne des propriétés de stabilité de classe. Ces propriétés restent vraies en dimension supérieure, car la construction employée ne dépend pas de la dimension.

3.1 Outils techniques

On va présenter ici un certain nombre de lemmes et de notions techniquement utiles, qui non seulement permettent d'introduire des méthodes de calcul cellulaire sur les signaux, mais sont en eux-mêmes souvent utilisés.

Lemme 1 (Signal de délai) *Il est possible de définir un signal de délai, d'origine (z_0, t_0), qui se propage à vitesse maximale vers la droite, qui retarde de 1 la progression de tous les signaux situés dans les sites $\{(z_0 + t_1, t_0 + t_2), t_1 < t_2\}$ par rapport aux signaux situés dans les sites $\{(z_0 + t_1, t_0 + t_2), t_1 \geq t_2\}$.*

Preuve. L'idée générale de la preuve est de propager un signal à vitesse maximale vers la droite. On s'arrange ensuite pour que tous les signaux qui croisent ce signal (le temps étant discrétisé, on ne manque pas d'intersections) soient retardés d'un top, ce qui se traduit par une augmentation simple du nombre d'états (il suffit de pouvoir les mémoriser pendant un tour). Le signal est ensuite réémis avec un temps de retard dans la même direction. L'algorithme est détaillé sur la Figure 2. À la

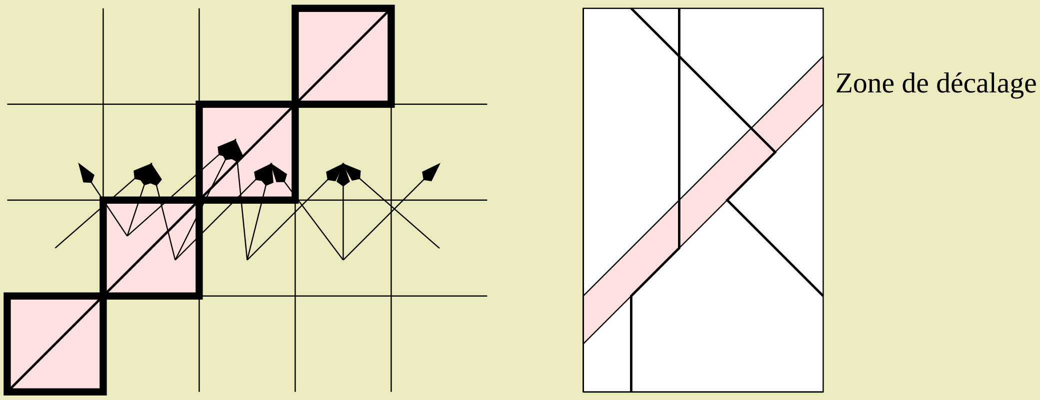


Fig. 3 - *Détournement des signaux par un décalage*

première étape où une cellule reçoit le signal de gel, elle :

– reçoit les signaux provenant de la droite et du dessous ;
– intercepte les signaux provenant de dessous allant vers la gauche ;
– renvoie les signaux interceptés vers la gauche ; ceci retarde bien ces signaux puisqu'ils auraient du être sur la cellule de gauche (voir Schéma 2) ;
– transmet le signal de délai vers la droite, et mémorise le reste.

À l'étape suivante, une cellule gelée :

– reçoit les signaux provenant de la gauche et ne reçoit pas les signaux provenant de la droite (interceptés) ;
– les combine avec les signaux mémorisés et en déduit les actions à faire ;

□

Lemme 2 (Signal de décalage) *Il est possible de définir un signal de décalage, d'origine (z_0, t_0), qui se propage à vitesse maximale vers la droite, qui décale vers la droite de 1 cellule tous les signaux $\{(z_0 + t_1, t_0 + t_2), t_1 < t_2\}$ par rapport aux signaux situés dans les sites $\{(z_0 + t_1, t_0 + t_2), t_1 \geq t_2\}$*

Preuve. Le principe est ici de pousser une partie du diagramme espace-temps vers la droite. En fait, chaque cellule recevant le signal de décalage va simuler deux cellules : celle située immédiatement à gauche (normalement) et elle-même. Une fois cette phase de décalage passée, la cellule continue son travail en se comportant comme si elle avait été la cellule de gauche des deux simulées. Quant à l'autre cellule simulée, elle est elle-même prise en charge par la cellule de droite qui reçoit le signal de décalage et se retrouve ensuite dans la colonne de droite.

L'algorithme est simple. La cellule recevant le signal de compression charge les états indiqués par les flèches sur la Figure 3, et s'en sert pour passer dans un état composé de l'état normal qui serait obtenu pour la cellule de gauche et celui obtenu pour la cellule de droite.

Au top suivant, la cellule récupère uniquement la première partie de l'état comme étant son état antérieur, la deuxième partie de l'état comme étant son voisin de droite et son voisin de gauche pour compléter. □

On utilise beaucoup ce genre de transformations sur les diagrammes espace-temps qui offrent l'avantage d'être des petites 'briques' élémentaires avec lesquelles on peut écrire les algorithmes.

Un autre genre de signaux utilisés : les signaux de destruction. On peut détruire tous les signaux (ou un certain sous-ensemble des signaux) dans le cône ayant le point d'initiation des signaux pour origine. on envoie simplement un signal se propageant à vitesse maximale et détruisant tous les signaux (ou un certain sous-ensemble de signaux) qu'il croise.

3.2 Fischer-constructibilité et signaux

Theorème 3 *Soit f strictement croissante telle qu'il existe $\alpha > 0$, tel que $f \geq n(1+\alpha)$. On a alors :*

1. *Si f est Fischer-constructible, et qu'il existe $\alpha > 0$, tel que $f \geq n(1+\alpha)$, alors on sait construire le signal de pente f ;*

2. *Si f croît plus vite qu'une exponentielle asymptotiquement, et qu'il existe un signal de pente f, alors $2f$ est Fischer-constructible (et donc f).*

Preuve. On va traiter les deux cas successivement. Les preuves font toutes deux référence à la Figure 4.

La première partie du théorème a déjà été utilisée dans [?]. On peut supposer, et ce n'est pas restrictif, que tout le calcul s'effectue dans la partie droite de l'automate (sinon, on s'y ramène par pliage). Ce calcul va Fischer-construire une fonction, c'est à dire que la cellule 0 va passer dans un état spécial à chaque fois que la fonction à un antécédent différent. Le graphe de la fonction se décale alors d'une cellule sur la droite en même temps qu'il monte. On va se servir des signaux de compression sus-mentionnés pour transformer l'axe des temps en le graphe du signal. L'automate est exactement le même que celui qui Fischer-construit la fonction, excepté qu'une cellule dans un des états spéciaux génère un signal de compression se propageant vers la droite. Comme toute autre marque aurait été faite après un tel signal, d'après la description qui en a été donnée, la marque sera faite une case plus à droite. Donc le graphe de la fonction se lit comme indiqué sur la Figure 4.

Notons que ceci ne fonctionne pas dans le cas où la fonction est en dessous de $(1+\alpha)n$ pour tout α positif (ce qui n'est pas par hypothèse), car il y a alors un nombre non borné de cellules qui doivent être compressées.

Dans le deuxième cas (qui peut certainement être amélioré), on cherche à fabriquer une fonction Fischer-constructible à partir d'un signal. On suppose que la pente du signal croît plus vite qu'une exponentielle. Cela se traduit par le fait que la fonction ne se décale pas avant un temps double du temps actuel. L'algorithme employé est clair et ne nécessite pas plus d'explications : on initie un signal se dirigeant vers la gauche, de pente f. Au temps $2n$, le signal entre en contact pour la première fois avec l'axe, comme expliqué sur la Figure 4. On se sert de ceci pour faire une marque. Pendant ce temps, l'automate a continué à calculer normalement. On envoie également un signal de destruction en même temps que la copie du signal pour détruire les copies précédentes du signal. On Fischer-construit ainsi le double de la fonction désirée ; comme le produit d'une fonction Fischer-constructible par un scalaire reste Fischer-constructible, f est donc Fischer-constructible.

$\square$

Theorème 4 (inversion) *Si f est une fonction strictement croissante, et qu'il existe un signal de pente f, il est possible de construire un signal de pente $2(n + f^{-1}(n))$.*

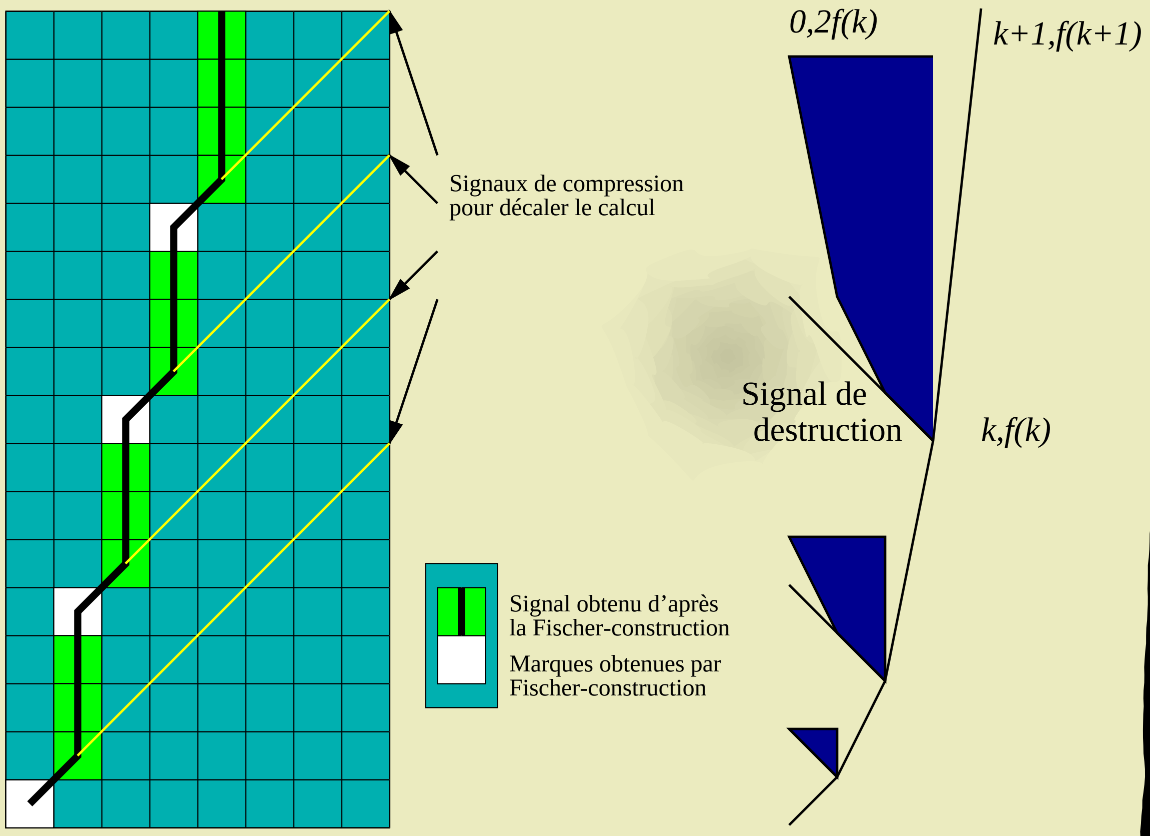


FIG. 4 - *Transformation Fischer/signal*

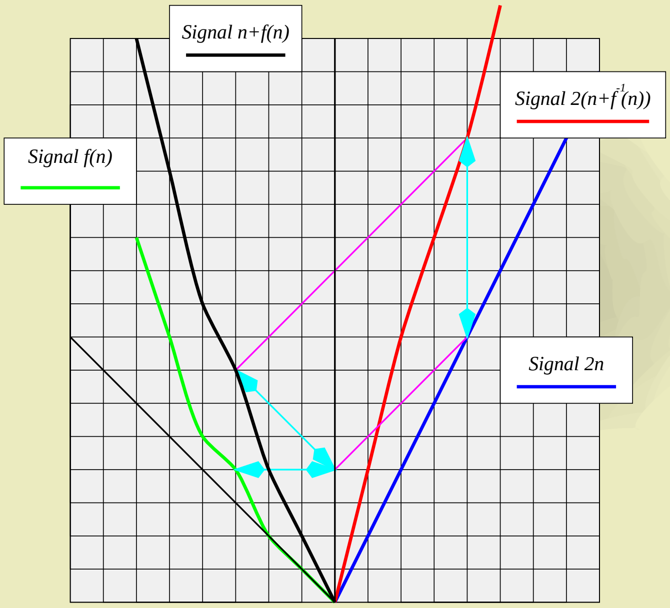


FIG. 5 - *Calcul de la réciproque*

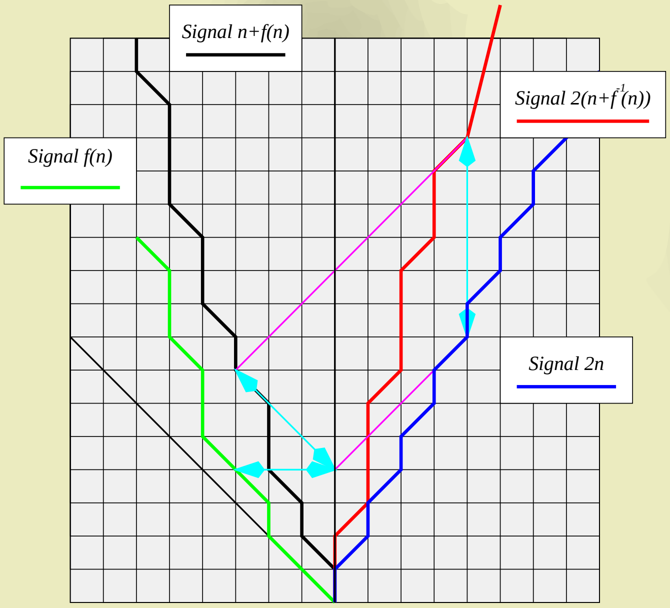


FIG. 6 - *Calcul de la réciproque - signaux réels*

Preuve. Le principe de cette preuve est de pencher les calculs ; l'écart entre l'axe des origines et le signal étant l'inverse de la fonction, on s'arrange pour reprojeter cette distance en la tournant. Pour cela, on présupposera (ce qui n'est pas restrictif) que l'ensemble du calcul se fait dans la partie gauche du diagramme espace-temps ; on pliera le calcul si besoin est. La façon de pencher le calcul est de décaler une fois sur deux toute la bande vers la droite pendant que l'on calcule la fonction à inverser plus n. Il y a deux figures correspondant à cet algorithme : la figure 5 qui est le diagramme espace-temps de l'algorithme idéalisé, et la figure 6 qui représente le même algorithme avec des signaux se déplaçant en suivant le réseau de cellules. [1]

L'algorithme complet est comme suit : étant donné un automate qui construit le signal de pente f, on en déduit un automate qui construit le signal de pente $f(n)+n$, [2] et l'on gère un compteur binaire sur toutes les cellules : un temps sur deux, on procède à un décalage vers la droite de l'état de l'automate.

Si l'on dénote le signal de pente $f(n)+n$ par $\mathfrak{S} = \{(0, f(0)), \ldots, (-k, f(k)+k), \ldots\}$ (les coordonnées sont négatives car on travail sur la demi-ligne gauche), l'image transformée sera en $\{(0, f(0)), \ldots, (f(k)+k-k, 2(f(k)+k)), \ldots\}$; le temps étant en tout point du signal égal à $f(k)+k$, on ajoute cette valeur aux deux composantes : à la composante de temps parce qu'on la double, à l'abscisse parce qu'on décale la moitié du temps. Le signal transformé est donc l'ensemble de sites $\mathfrak{S}' = \{(0, f(0)), \ldots, (f(k), 2(f(k)+k)), \ldots\}$. On opère enfin une substitution sur les variables pour mettre en évidence la construction : si l'on dénote par y la valeur de $f(k)$, et que l'on opère la substitution, on a que tous les sites forment l'ensemble $\{(f^{-1}(0), f(0)), \ldots, (y, 2(f^{-1}(y)+y)), \ldots\}$, ce qui est exactement le signal de pente $2(n+f^{-1}(n))$. $\square$

Remarquons que la méthode utilisée ci-dessus fait que la partie droite du calcul (qui ne contient que des états quiescents) est projetée sur la partie du plan sous la droite de pente 2 dirigée vers la droite. Ainsi, toute cette partie du plan sera

1. Attention : dans ces deux schémas, les cellules sont représentées par les intersections des lignes, et non par les cases.

2. C'est toujours possible (utilisation de signaux de délais).

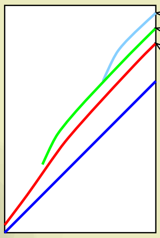


FIG. 7 - *Calcul de la réciproque*

quiescente (voir Figure 5)

3.3 Signaux en dimension 1

On arrive là à la frontière de ce que l'on sait faire en dimension 1. On dispose en effet du théorème suivant :

Theorème 5 (Mazoyer) *S'il existe un signal de pente f, alors soit $f(n) - n$ devient une constante, soit il existe α, tel que $f(n) \geq \log_\alpha(n)$.*

On ne peut donc pas espérer calculer l'inverse de 2^{2^n}, ou plutôt la fonction $n + \log\log n$. On sait calculer en temps $2n + f$, sans utiliser la surface en dessous de la droite de pente 2.

C'est une limitation importante. Ce stage a prouvé que la limitation ne tenait plus dès lors que l'on calculait en dimension supérieure.

Il est également possible de voir que l'on ne peut pas rendre le calcul de $n + f^{-1}$ Fischer-constructible en général pour f Fischer-constructible ; en effet, des fonctions aussi grandes que l'on veut sont Fischer-constructibles (stabilité par composition).

3.4 Fischer-construction du $\log^\star$

Afin de trouver une véritable différence entre les fonctions calculables en dimension 1 et en dimension 2, on va d'abord avoir besoin de Fischer-construire une fonction qui est vraiment plus grande : la tour d'exponentielles. Comme on sait calculer un signal en $n + f^{-1}(n)$ pour f 1-Fischer-constructible, on montre ainsi qu'il y a une différence stricte entre les deux dimensions.

Le principe de la preuve est que le $\log^\star$ augmente de 1 à chaque fois que l'une des fonctions $\{\log, \log\circ\log, \log\circ\log\circ\log, \ldots\}$ atteint la valeur 2 pour la première fois. On va donc co-Fischer-construire le $\log^\star$ (ce qui est équivalent) à l'aide d'une superposition des fonctions $\log^{(i)}$ (comme montré Figure 7), ce qui donnera les signaux de la forme générale :

$$\sum_{0}^{\log^\star(n)} \log^{(i)}(n) = n + \log n + \log\circ\log n + \ldots$$

.

Rappelons que $1 \leq \log^{(\log^\star(n))}(n) < 2$. Pour $i > \log^\star(n)$, on pose $\log^{(i)}(n) = 0$.

On va donc s'attacher à construire toute cette série de signaux (que l'on nommera aussi *couches* car ils sont disposés les uns au-dessus des autres), qui se déduisent les uns des autres (afin d'avoir un nombre d'états finis). On fera ensuite quelque chose de spécial pour exploiter le fait que chaque création d'une nouvelle couche doit se traduire par une augmentation du $\log^\star$: un signal part de toute génération de nouvelle couche (à partir de la valeur 2 de la fonction précédente) et vient en contact avec la droite de pente 2. Au point d'intersection avec cette droite, on renvoie le signal à vitesse maximale vers l'axe. Si le signal est généré dans la colonne n, il y reste jusqu'au temps $2n$. L'intersection du signal avec l'axe se fait donc au temps $3n$, générant ainsi la co-Fischer-construction de la fonction $\log^\star/3$.

Comme on peut multiplier par un scalaire les fonctions constructibles en restant dans la même classe, on peut donc co-Fischer-construire le $\log^\star$, donc Fischer-construire la tour d'exponentielles.

Il est conseillé pour la suite de la démonstration de se référer à la Figure 9.

On est assuré de la correction de cette partie de l'algorithme : chaque nouvelle couche envoie un signal rose, qui est ensuite dévié par la droite de pente 2 (en rouge). La création de la couche est émise sur la cellule n, et atteint donc l'axe après réflexion au temps $3n$. La somme des fonctions $\sum_0^{\log^\star(n)} \log^{(i)}(n)$ peut être facilement majorée par $2n$ (car inférieure ou égale à $n+\log^2(n)$), donc le signal rose rencontre effectivement le signal rouge.

Il reste à trouver une méthode de construction des couches. En fait, les signaux ne seront pas exactement les fonctions décrites ; mais ce que l'on garantit est que l'on sera au même endroit lorsque l'on générera les couches. La différence provient du fait que lorsque l'une des itérées augmente (par exemple, le $\log\log\log$), toutes les itérées d'ordre inférieur augmentent aussi, ce qui entraînerait des difficultés pour le décalage à opérer, si bien que l'on préfère retarder les courbes un peu avant (afin d'anticiper les délais). On se sert pour cela des signaux de délai définis précédemment, représentés en noir, et émis par chaque couche à chaque fois qu'elle augmente. Pour voir comment cela fonctionne, on peut regarder le Schéma 8 qui représente deux couches successives loin de l'origine.

On sait donc produire des courbes d'écart constant, et, à partir de là, les interpréter pour en déduire la co-Fischer-construction du $\log^\star$. Il reste à donner l'algorithme qui les construit.

Toutes les phases de cet algorithme apparaissent Figure 9. L'essentiel est l'existence d'un algorithme qui, disposant en entrée d'un top à chaque fois que la fonction de base augmente de 2, en déduit le logarithme de la fonction de base. Cet algorithme a déjà été utilisé par Mazoyer dans [?]. Le principe en est simple : on compte en binaire le nombre de tops que l'on a reçu et le premier 1 (désignant le nombre de tops comptés) désigne exactement le logarithme de la fonction d'entrée. Ce système est représenté graphiquement par les flèches noires (une flèche étant un bit à 1 dans le compte), et, de façon caché, un compteur binaire.[3]

Pour des contraintes d'uniformité, et à cause des cas particuliers, il est nécessaire de distinguer les fonctions lorsqu'elles sont encore proches ; notamment pour bien gérer l'intersection du signal bleu clair et du signal gris (qui représente une fonction en régime 'permanent'), mais on peut ensuite utiliser les même signaux dès que l'écart entre la nouvelle fonction et l'ancienne atteint 2 ; ce qui correspond d'ailleurs exactement au moment où l'on génère la nouvelle fonction.

La nouvelle fonction est ainsi représentée en bleu clair tant qu'elle ne s'est pas décalée de 1 ; elle se décale en même temps que la fonction du dessous, et devient bleu foncé. Lorsque le signal bleu foncé touche le signal gris, le signal gris lui transmet une flèche (il n'en a pas transmis au bleu clair), et donc la fonction augmente bien d'un cran. Elle passe alors en régime permanent, et génère une nouvelle couche. Ainsi, on a bien les signaux voulus dès lors que l'on est aux endroits où de nouvelles couches sont créées. Donc, l'algorithme permet effectivement la co-Fischer-construction du $\log^\star$.

3. Puisque on compte le double des flèches, il faut qu'une couche ne retransmette une flèche que lorsqu'elle a augmenté de 2 sans émettre de flèche. Ceci alourdissant le dessin, cette partie a été cachée).

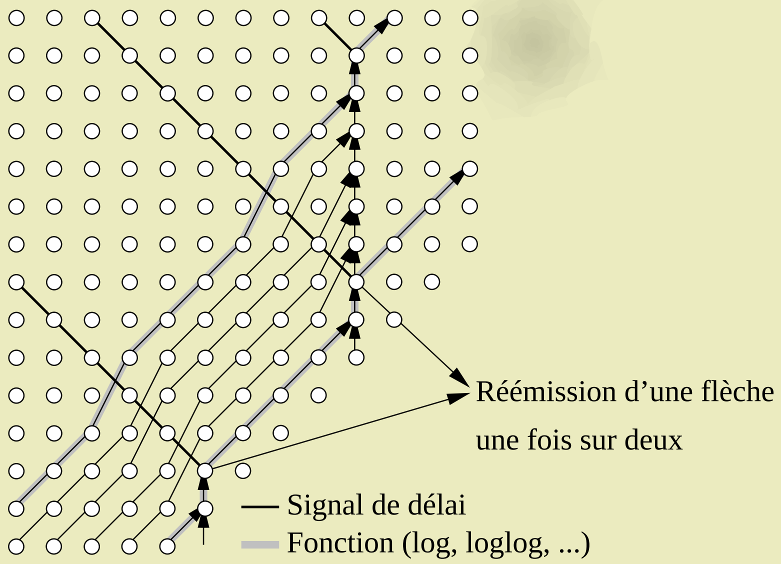


FIG. 8 - *Calcul du* $\log^\star$

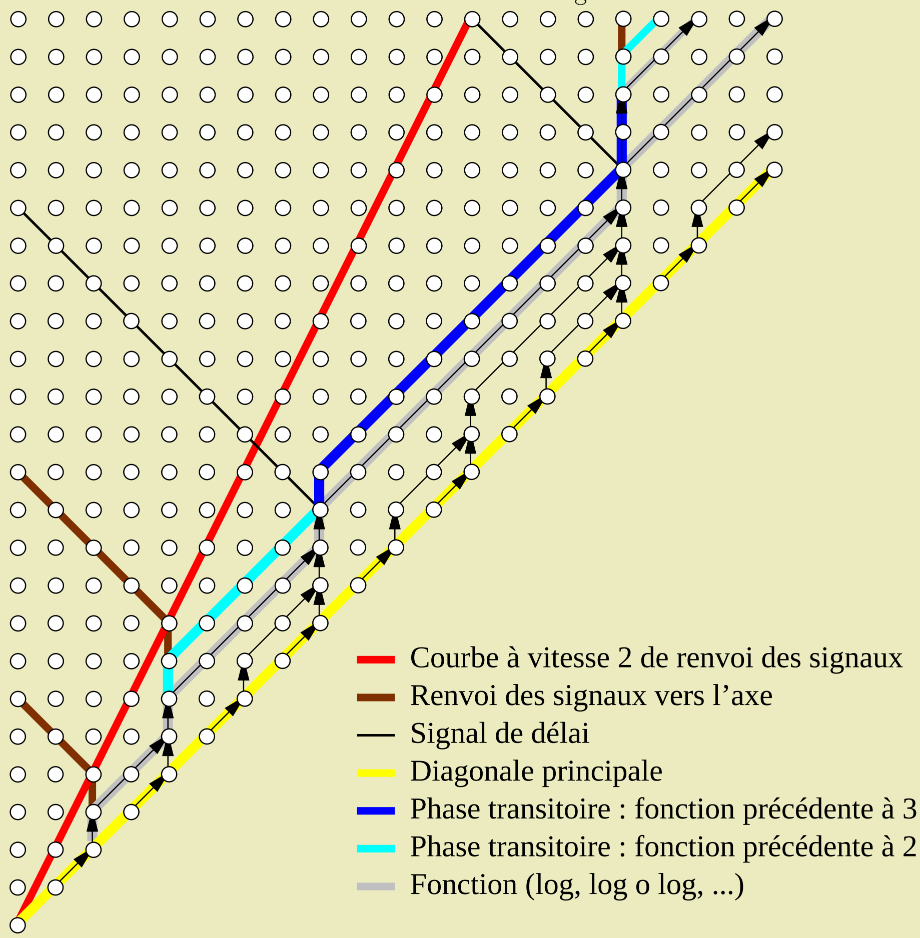


FIG. 9 - *Calcul du* $\log^\star$*, phase initiale*

4 Signaux bidimensionnels

L'introduction d'une dimension supplémentaire permet d'augmenter grandement les capacités des automates cellulaires, car il est possible de tordre beaucoup plus les calculs. Une des applications de ceci est la description d'un algorithme pour faire des signaux de ratio $log^{\star}$ en dimension 2, ce qu'il n'est pas possible de faire en dimension 1.

4.1 Représentation d'algorithmes cellulaires plans

Un des principaux problèmes a été la représentation des algorithmes que l'on construit. En effet, sur les dessins, on doit porter souvent non seulement les deux dimensions spatiales de l'automate, mais aussi son évolution dans le temps ; ce qui porte à 3 le nombre de dimensions à représenter. De plus, on doit différencier les différents sites, qui ont des rôles différents.

La solution adoptée est la représentation sous forme tridimensionnelle et en couleur des données.

Le temps devient la hauteur, et est, à l'instar des diagrammes espace-temps précédents, projeté vers le haut. Ainsi, on obtient pour les voisinages classiques le schéma suivant :

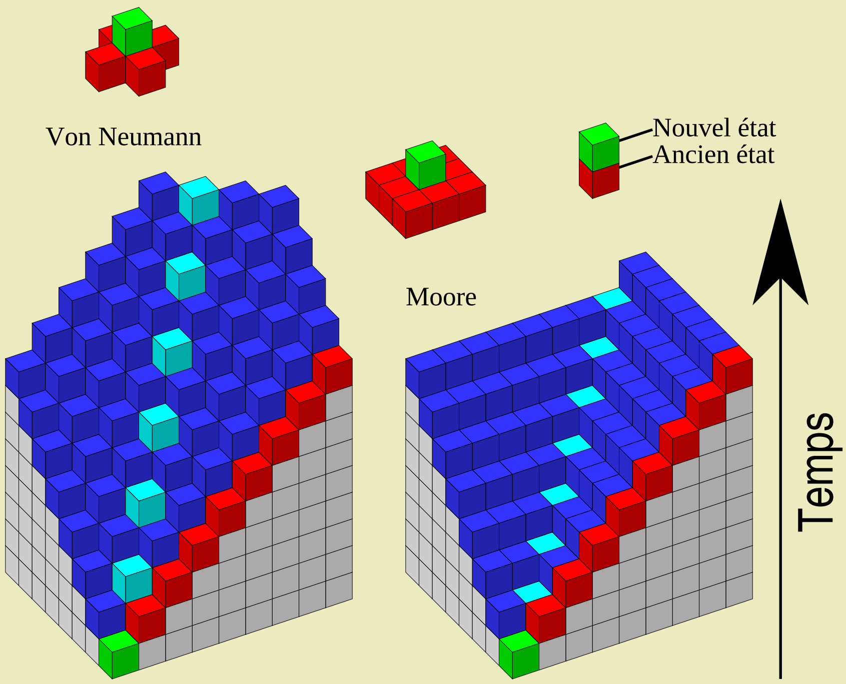


Fig. 10 - *Voisinages & temps réel*

Ce genre de représentation a ses limites. Par exemple, il n'est pas possible de représenter un bloc “plein”, utilisant entièrement un volume d'espace-temps. Mais le problème devient alors très difficile, nécessitant sans doute des techniques interactives.

Implémentation L'implémentation de la représentation a surtout été faite par une interface utilisateur pour générer facilement des dessins. Un système simple de programmation en PostScript permet de définir très facilement un type de cellule (taille, couleur principale, couleur auxiliaire, forme (rond ou cube)), et ensuite, d'entrer l'état des cellules sous forme d'un tableau. Par exemple, `h1 [0 0 1 1 1 1 1.5] def` définit `h1` comme étant le symbole d'une boule blanche et bleue de taille moyenne, et `[[zz qq][zb bz]] prepareplan 0 makeplan` indique de construire le carré constitué des cellules indiquées à l'altitude de référence, l'angle de vue et l'ombrage artificiel étant paramétrables.

4.2 Transformer une fonction Fischer-constructible 1D en signal 2D

On veut s'attaquer dans ce paragraphe au problème non résolu dans le cas 1D : réussir à construire un signal proche du temps réel, par exemple $n + f^{-1}(n)$, où f est une fonction qui croît rapidement (plus vite que l'exponentielle).

Le principe est d'utiliser la relation entre Fischer-constructibilité d'une fonction et co-Fischer-constructibilité de sa réciproque. Puisque chaque marque sur l'axe correspond à une augmentation de 1 de l'inverse, on va laisser des colonnes correspondant à ces marques, calculées en temps réel, pour que le signal se décale dès qu'il en rencontre une.

Cette construction se fait donc assez facilement en dimension 2 (voir Figure 11), contrairement à ce qui se passe en dimension 1.

4.3 Hiérarchie de Grzegorczyk

La hiérarchie de Grzegorczyk est une classification des fonctions récursives primitives en fonction du nombre minimal de récurrences nécessaires pour calculer la fonction. On la trouve pour la première fois dans [?]. Cette hiérarchie est stricte (il existe des fonctions de tous les niveaux) et est représentée en particulier par les fonctions successives d'Ackermann ($\mathfrak{A}(.,n)$ est de classe n dans la hiérarchie). Plusieurs réflexions nous ont donc amené à établir une conjecture, et une partie du travail effectué a tourné autour de la conjecture suivante :

Conjecture 1 *Les fonctions d-Fischer-constructibles avec $d \geq 2$, sont de classe $d+3$ dans la hiérarchie de Grzegorczyk.*

Parmi les différents éléments qui nous font croire à la véracité de cette conjecture, il y a une certaine idée de ce que peut être le calcul multidimensionnel. L'idée maîtresse est que l'on ne peut faire qu'une seule récurrence par dimension, les petites récurrences exceptées. Ainsi, rajouter une dimension permet de monter dans l'échelle des fonctions de Grzegorczyk.

La suite de ce chapitre cherche à montrer divers aspects par lesquels on peut penser que cette conjecture est vraie.

4.4 Génération du $\log^\sharp$ par automate cellulaire plan

On cherche à prouver que le $\log^\sharp$ peut être 2-co-Fischer-construit, c'est à dire que la tour de tour d'exponentielles est Fischer-constructible en 2D. La construction toujours valide utilisée pour transformer une fonction Fischer-constructible en permet facilement d'en déduire un signal de forme $\mathfrak{A}(n,3)$.

La méthode de construction est fondée sur une remarque simple : le $\log^\sharp$ augmente d'une unité à chaque fois que l'on atteint un nouvel étage de la tour d'exponentielle $2, 2^2, 2^{2^{2^2}}, \ldots$. On va donc générer tous les signaux (correspondants aux différentes itérées de la tour d'exponentielle) dans certaines couches de l'espace-temps, un peu comme on a montré que le $\log^\sharp$ était co-Fischer-constructible, et l'on fera une marque chaque fois qu'une nouvelle couche sera créée. On prend pour fonction f le $\log^\star$ dans le cas du $\log^\sharp$ (mais on utilisera f pour plus de généralité). Les

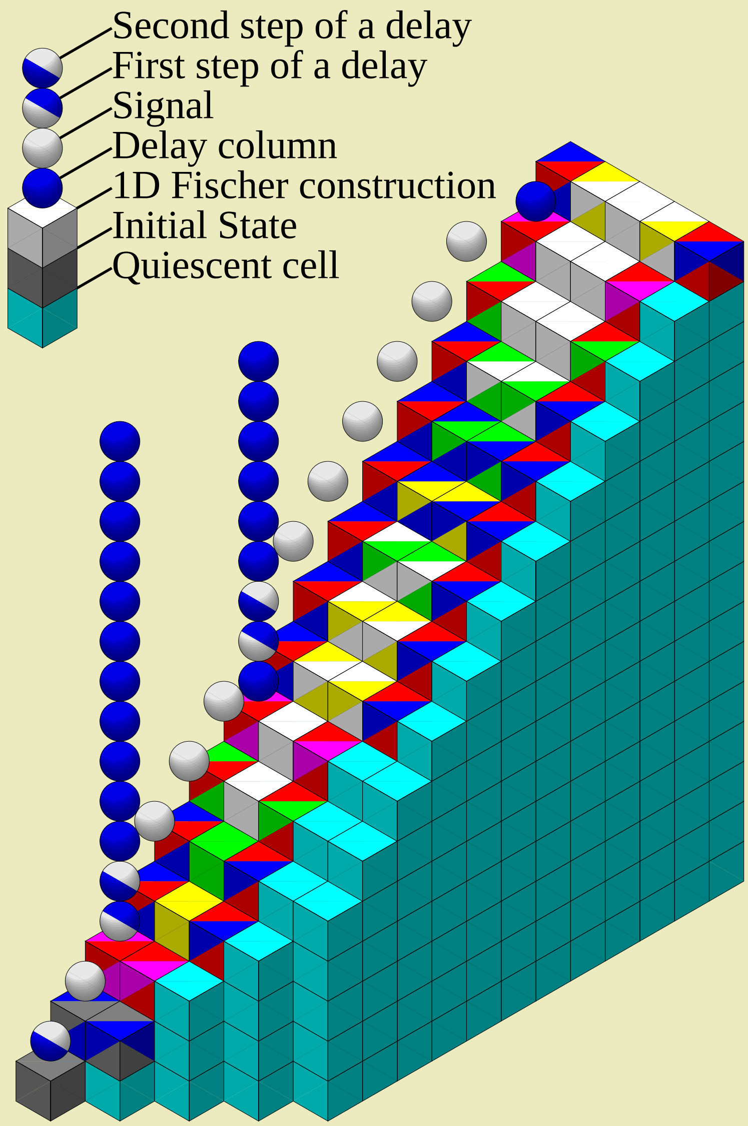


FIG. 11 - *Transformation Fischer 1D/Quasi-temps réel 2D*

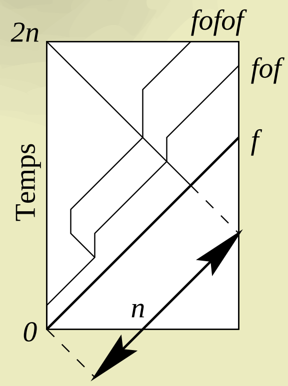


FIG. 12 - *Vue latérale*

dessins sont toutefois réalisés avec une fonction f qui montre plus les cas critiques, le $\log^\star$ évoluant trop lentement.

Dans la représentation latérale de l'espace de travail (Figure 12), on voit la méthode générale de formation des diverses couches. Contrairement au $\log^\star$, on construit les couches dans la diagonale. L'algorithme est de construire dans des hyperplans (le premier étant l'hyperplan du temps réel) la fonction f de base.

On ne va faire des étapes de calcul que lorsque l'on reçoit un signal de synchronisation provenant de l'hyperplan précédent, sinon on se contente de se décaler dans une direction commune (en s'éloignant de l'axe, voir la Figure 13). Lorsque la zone de calcul nécessaire pour la Fischer-construction croît, on surélève la couche de calcul en restant sur place, et l'on envoie un délai vers l'axe. Ensuite, un signal de synchronisation est envoyé vers l'axe principal et donc vers la couche suivante à chaque fois qu'une couche augmente pour lui signifier de faire une étape de calcul supplémentaire. Ainsi, la première couche calculera f uniquement lorsque f augmente. Donc, au temps n, la première couche calcule $f \circ f(n)$. La deuxième couche ne calcule que lorsque $f \circ f(n)$ augmente, et calcule donc $f \circ f \circ f(n)$. Les choses que l'on doit garantir pour que l'algorithme fonctionne sont que :

- la couche i calcule $\underbrace{f \circ \ldots \circ f(n)}_{i+1 \text{ fois}}$;
- l'espace entre deux couches successives est suffisant pour que le signal synchronise toute la couche supérieure. Cette espace doit donc être au moins égal à la largeur du calcul sur la couche supérieure plus un (car la couche doit pouvoir croître en largeur) ;
- la distance de la dernière couche au temps réel est inférieure ou égale à n ;
- le nombre d'états est fini.

Ces points découlent naturellement de l'algorithme employé. La couche i calcule bien ce qui est demandée, puisqu'elle applique l'algorithme de calcul de f uniquement lorsqu'elle reçoit un signal de la couche précédente, donc elle calcule $f \circ g(n)$, où g est la fonction calculée à la couche précédente. Par récurrence immédiate, ce point est donc prouvé.

Le deuxième point est facile à vérifier. L'algorithme spécifie que lorsqu'une couche s'élargit, elle se décale vers le haut. Ainsi, si l'écart est égal à la largeur nécessaire pour le calcul avant une étape de calcul, il est toujours égal après l'étape de calcul (voir figures 13 et 14). De plus si la couche inférieure monte, elle envoie un signal de délai (comme décrit plus haut, voir Figure 14). Donc deux couches ne se rapprochent pas. Il faut toutefois que la couche de base (temps réel) envoie des signaux moins d'un temps sur deux, sinon les calculs et les délais se chevaucheraient. C'est évidemment le cas pour $f = \log^\star$.

Une fois les deux résultats précédents acquis, le troisième point en découle. En effet, l'espace entre la dernière couche et le temps réel est donc (quand n est le temps auquel aurait été émis un signal vers l'axe principal qui atteint la couche) $f \circ f(n) + f \circ f \circ f(n) + \ldots + \underbrace{f \circ \ldots \circ f(n)}_{f^\sharp \text{ fois}}$. Or cette valeur est inférieure ou égale à $f^2(n)$. Donc si $f \leq \sqrt{n}$ (ce qui est le cas pour le $\log^\star$), les couches de calcul ne prennent pas trop de place.

Le nombre d'états nécessaires reste fini : outre les états nécessaires pour réaliser la Fischer-construction de f, multipliés pour les phases d'émissions de délai et d'attente (déplacement de la couche sans rien faire), il y a le signal de synchronisation (qui peut aussi être utilisé pour signifier la création d'une nouvelle couche et donc pour faire un top sur l'axe. Il faut un petit nombre de signaux supplémentaires, et un système de signaux de délais. Tout ceci utilise donc un nombre fini d'états.

La dernière étape de l'algorithme pour obtenir la Fischer-construction désirée est la suivante : dès qu'une nouvelle couche de calcul est créée, on envoie vers l'axe un signal non intercepté qui, en arrivant sur l'axe (donc au temps $2n$), déclenche un top. Conformément aux observations sur le $\log^\sharp$, cela correspond exactement à co-Fischer-construire la fonction $f^\star(n)/2$. Comme les fonctions Fischer-constructibles sont stables par multiplication par un scalaire, on sait construire $\log^\sharp(n)$.

On peut donc très vite donner une généralisation de ce théorème.

Theorème 6 *Si f est une fonction d-Fischer-constructible croissant moins vite que $\sqrt{n}$, $f \leq \sqrt{n}$, $f^\star$ est $d+1$ Fischer-constructible.*

Preuve. En fait, on applique exactement l'algorithme exposé ci-dessus. Les calculs de chacune des couches se font dans des hyperplans. Les conditions faites sur la fonction permettent d'appliquer l'algorithme sans modifications. □

Un corollaire immédiat est que le $\log^{\sharp^\sharp}$ est 3-co-Fischer-constructible.

4.5 Limitations en 2D

Il est important de bien réussir à percevoir que tout n'est pas possible en dimension 2. Dans le cadre du voisinage de Moore, le stage n'a pas réussi à faire émerger cette même notion qui permet de faire tourner la démonstration de non-constructibilité des signaux en dimension 1. Il faudrait en effet réussir à définir une notion de *périodicité* qui ne soit pas une périodicité, mais plutôt le grain de la fonction (dans l'hypothèse où la conjecture serait juste).

En revanche, la situation dans le cadre du voisinage de von Neumann (voir Figure 10) est légèrement différente. Le nombre plus réduit de voisin fait que sur les axes privilégiées, celles où l'on veut propager le signal, les dépendances sont beaucoup plus fortes que dans le cas du voisinage de Moore. On peut notamment remarquer que dans von Neumann, la ligne de temps réel est ultimement périodique, tandis que dans le cas du voisinage de Moore, c'est une fonction 1-Fischer-constructible.

On a donc la proposition suivante :

Theorème 7 (Limitation en 2D) *Pour tout signal $\mathfrak{S}$ de pente $n + f(n)$ se propageant dans le plan $x = 0$, ou bien il existe α tel que $f(n) \geq \log_\alpha(\log_\alpha(n))$, ou bien f est bornée.*

Preuve. Appellons $\mathfrak{D}(y, z)$ l'ensemble des sites $\{((x, y), z), x \in \mathbb{N}\}$. Ce sont les diagonales paralléles au plan d'équation $x = 0$. Ce que l'on va montrer, c'est que toutes

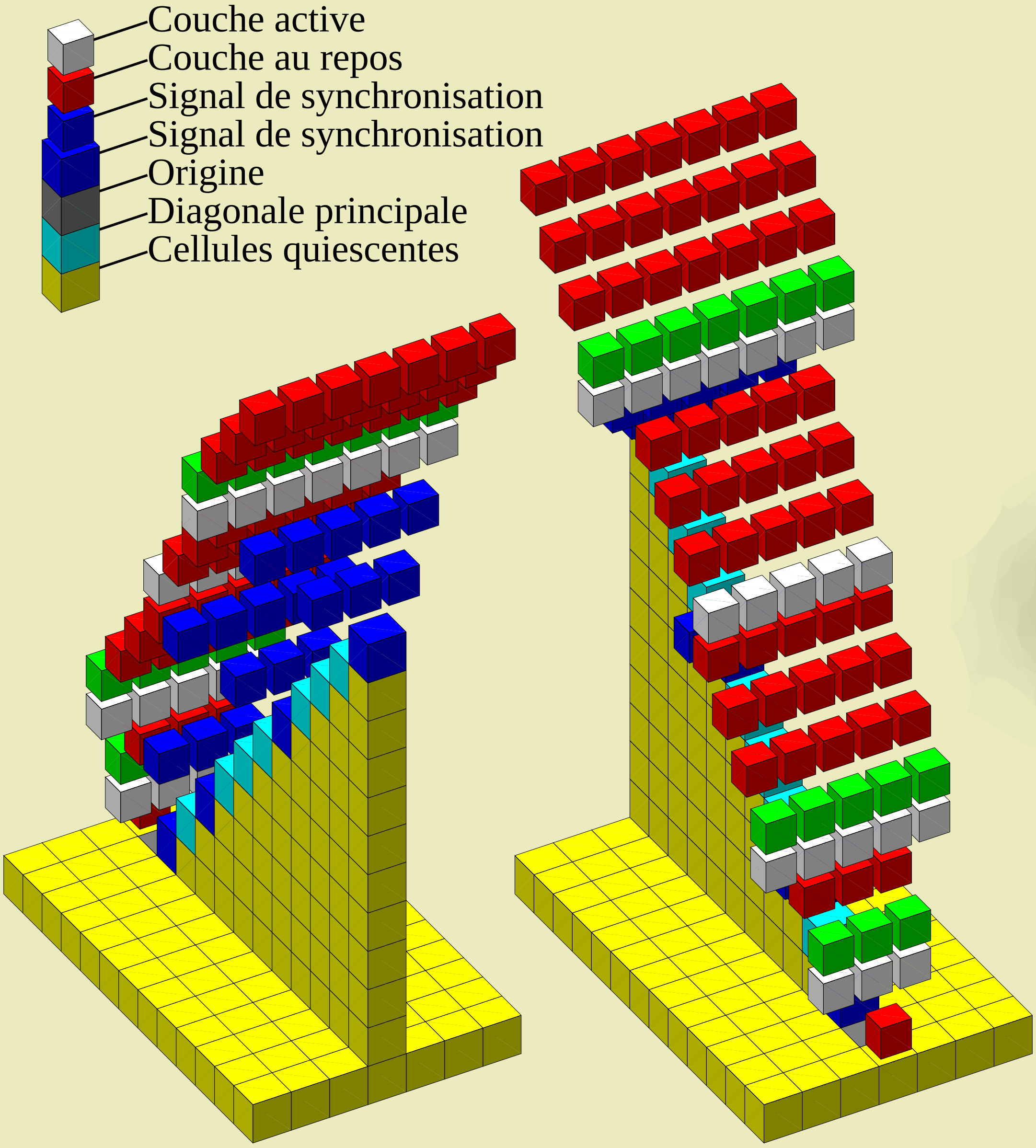


FIG. 13 - *La première couche de calcul*

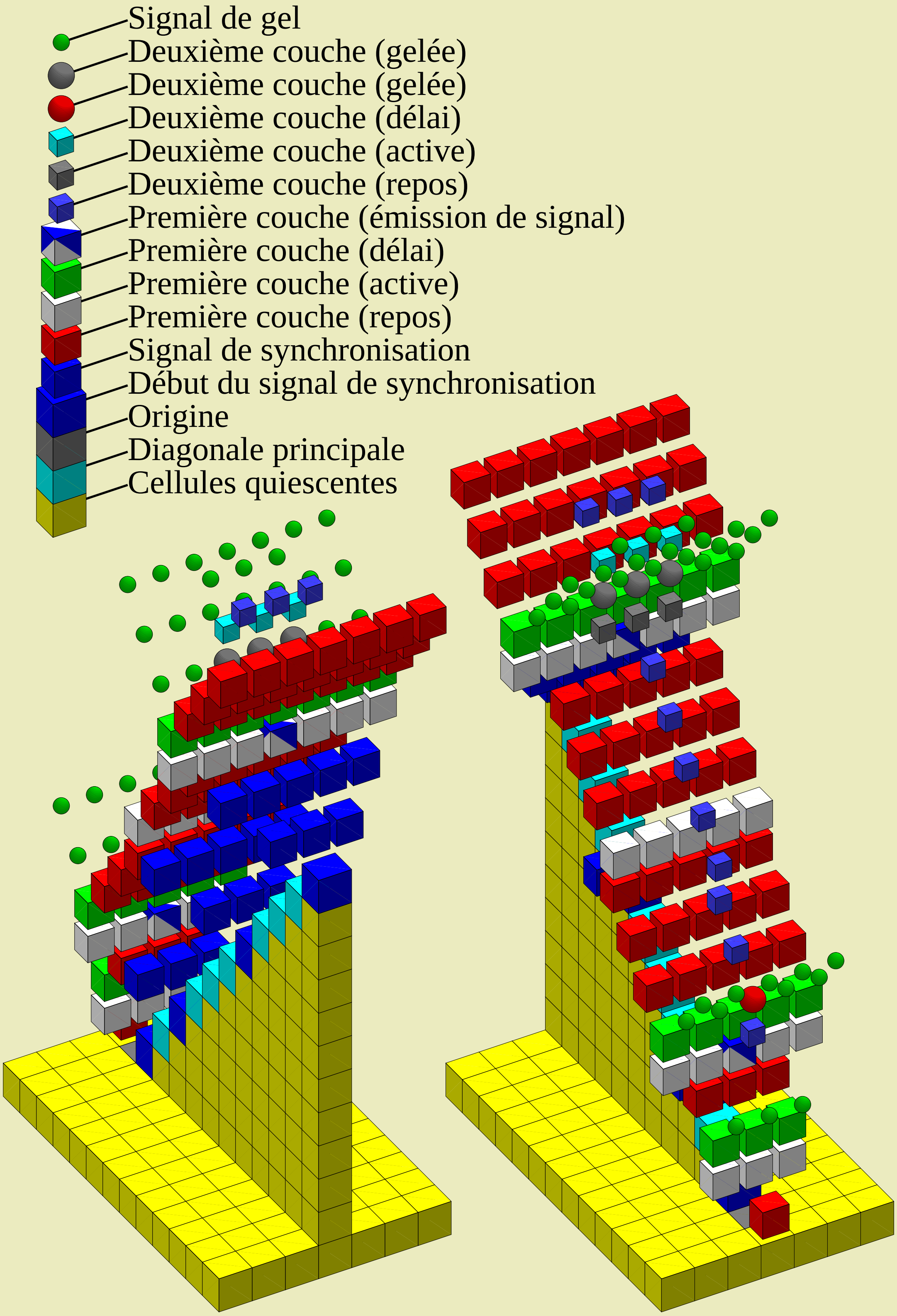


FIG. 14 - *Synchronisation des couches de calcul*

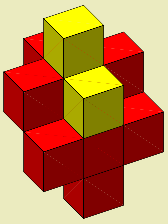


FIG. 15 - *Schéma des dépendances des diagonales $\mathfrak{D}$*

$z\backslash y$	-4	-3	-2	-1	0	1	2	3	4
4	5	14	36	56	68	56	36	14	5
3	0	4	9	19	21	19	9	4	0
2	0	0	3	5	8	5	3	0	0
1	0	0	0	2	2	2	0	0	0
0	0	0	0	0	1	0	0	0	0

TAB. 1 - *premiers éléments de* $\log_q \mathfrak{p}(y, z)$

ces diagonales sont ultimement périodiques, et que leur période ne dépend que des périodes des diagonales situées en dessous. En effet, si l'on suppose que toutes les diagonales $\mathfrak{D}(y, z)$ pour $z \leq z_0$ avec z_0 fixé, sont ultimement périodiques, de période $\mathfrak{p}(y, z)$, alors toute diagonale $\mathfrak{D}(y, z_0 + 1)$ est un automate fini à q états admettant pour entrée 4 suites ultimement périodiques (comme le montre le schéma des dépendances 15). La période $\mathfrak{p}(y, z + 1)$ peut être calculée par l'équation récurrente suivante[4] :

$$\mathfrak{p}(y, z + 1) = q\mathfrak{p}(y, z)\mathfrak{p}(y + 1, z)\mathfrak{p}(y - 1, z)\mathfrak{p}(y, z - 1)$$

Il reste donc à prouver que pour $z_0 = 0$ les diagonales sont ultimement périodiques, ce qui est immédiatement vrai puisqu'une seule diagonale n'est pas quiescente à $z = 0$, c'est la diagonale principale qui est de période q (elle ne dépend que de l'état d'origine et de diagonales quiescentes).

Donc toutes les diagonales $\mathfrak{D}(y, z)$ sont ultimement périodiques.

On va maintenant essayer de donner une majoration de la période. En effet, si le signal apparaît dans un intervalle supérieur à la période de la diagonale, il y apparaîtra périodiquement ; et étant donné les délais de propagation, on en déduit que f est bornée. On s'aperçoit rapidement que la période est une fonction strictement croissante en z dès qu'elle est différente de 1. Elle est également strictement décroissante en $|y|$, et elle est invariante par symétrie (si l'on change y en $-y$). Ces démonstrations sont laissées au lecteur, mais les premiers éléments de période peuvent aider à une compréhension générale (voir la Table 1). Pour $y = 0$, on majore immédiatement avec $\mathfrak{p}(0, z) < \mathfrak{p}^5(0, z - 1)$, ce qui donne $\mathfrak{p}(0, z) < q^{5^z}$. Ce qui donne bien la majoration attendue, la période est en $\mathcal{O}(2^{2^z})$, et le signal doit donc se propager plus lentement qu'une logarithme de logarithme. Pour $y \neq 0$, la même majoration tient, donc le résultat est vrai pour n'importe quelle tranche rectiligne dans laquelle doit se propager le logarithme. $\square$

4. sauf dans le cas où toutes les périodes valent 1. Dans ce cas-là, toutes les diagonales sont quiescentes, et donc la diagonale résultante aussi.

5 Conclusion

Dans ce rapport, on a exploré les puissances de calcul comparées des automates cellulaires en une et plusieurs dimensions, surtout du point de vue de la capacité à construire des signaux.

Les signaux sont une notion difficile à définir. La notion simple qui a été définie dans le cadre de ce rapport apparaît parfois peu satisfaisante, lorsque l'on essaye de pousser plus loin les interrogations sur l'universalité des signaux. Mais, d'un point de vue algorithmique, on est arrivé à une définition du signal qui est solide, de telle sorte que les signaux aient de bonnes propriétés.

La notion de Fischer-constructibilité et le lien qui peut être fait entre Fischer-constructibilité d'une fonction et un signal ayant pour pente la fonction ont été explorés. Il ressort que ces deux notions sont intrinsèquement liées, mais distinctes.

On a ensuite essayé de trouver de manière plus précise ce qui ne pouvait pas être fait en une dimension, et ce qui pouvait être fait. Dans le cadre de ce stage a été notamment démontré que l'on pouvait construire des réciproques de fonctions ($2n+f^{-1}(n)$ exactement), et qu'il n'était donc pas toujours possible de transformer un signal en Fischer-construction (alors que l'inverse est toujours possible). C'est une limitation importante. Un autre résultat connu était que l'on ne peut faire de signal allant plus vite que $n + \lceil\log(n)\rceil$ qui ne soit pas affine.

La deuxième partie du stage a été dédiée plus spécialement aux automates cellulaires plans. Après avoir vérifié que les algorithmes et les propriétés de classe étaient conservés, il a été montré que l'on pouvait faire des signaux à des vitesses supérieures à $n+\lceil\log(n)\rceil$, en laissant la possibilité d'inverser n'importe quelle fonction 1-Fischer-constructible. Un exemple est donné en montrant que l'on peut faire des signaux en $n + \log^{\star}(n)$.

L'étude des automates cellulaires plans a également donné lieu à une conjecture, sur un lien direct entre la hiérarchie de Grzegorczyk et la dimension nécessaire pour Fischer-construire une fonction. Plusieurs arguments viennent étayer cette hypothèse : les limitations du calcul en dimension 1, l'existence naturelle d'un représentant dans la bonne classe (la fonction d'Ackermann de niveau n) qui provient d'une généralisation de l'algorithme original utilisé en deux dimensions.

Enfin, certaines limitations du calcul sur automate cellulaire plan viennent reposer le problème du voisinage. En effet, on peut dés la dimension 2 distinguer une différence fondamentale entre les deux voisinages classiques, celui de Moore et celui de von Neumann

Ce travail débouche sur plusieurs problèmes ouverts. La définition du signal en elle-même reste un problème d'importance majeure dans les systèmes de communications synchrones, et il serait envisageable de voir quelle pourrait être la meilleure définition possible du signal, celle qui correspond le plus à l'idée intuitive que l'on s'en fait.

Un autre problème ouvert très important est la réponse à la question posée par la conjecture. Cette conjecture offre en outre de multiples extensions : dépendance par rapport au voisinage et à la métrique employés par exemple. La différence entre voisinage de von Neumann et voisinage de Moore au niveau de la puissance de calcul apparaît très clairement, donnant ainsi des résultats fondamentaux pour l'étude des systèmes de calculs itératifs. Étudier la puissance de calcul en fonction de la taille du voisinage, notamment pour tout ce qui concerne le temps réel et ce qui s'en approche, est un problème crucial qui devra être posé pour les ordinateurs de demain.

Enfin, la possibilité de transformer les automates cellulaires en méthode simple de programmation et de calcul parallèle passe par l'utilisation de signaux ; il reste donc crucial de continuer à étudier ce domaine.

Références

Remerciements

Je remercie pour leur aide précieuse durant toute la durée de mon stage mes encadreurs, Jacques Mazoyer dont les idées m'ont poussé en avant, et Bruno Durand, qui m'a empêché de tomber en allant trop vite. Je remercie aussi Éric Rémila et Maurice Margenstern, pour m'avoir fait découvrir l'univers des conférences d'un côté et de l'autre de la barrière. J'irais trop vite si j'oubliais Laure Tougne et Zsuzsanna Roka et leurs discussions ô combien instructives et reposantes, Marianne Delorme et Thomas Chaboud pour avoir relu avec patience mon rapport. Et enfin, je remercie Paul Pichaureau parce que c'est un bon copain et qu'il m'a donné les moyens de faire plus facilement mes parchemins, le cuisinier du restaurant l'Escargot à Roanne, parce que j'ai bien mangé, et également : Adobe Systems Inc., pour avoir créé POSTSCRIPT™, Sun Microsystems Inc., Franck pour avoir partagé une nuit blanche, Grégoire, tous mes correspondants pour leur soutien, un maître d'école, une prise à 5 broches, 14 alexandrins, dix doigts de la main, et trois ratons laveurs.

Je ne remercie pas Donald Knuth parce que je l'ai déjà fait l'année dernière, les gens qui ont mal votés dans certaines villes du sud de la France, et ceux que je ne remercie pas en général.