ORSAY
N° d'ordre : 4826

UNIVERSITE DE PARIS-SUD
U.F.R. SCIENTIFIQUE D'ORSAY

THESE
présentée
Pour obtenir

Le GRADE de DOCTEUR en SCIENCES
DE L'UNIVERSITE PARIS XI ORSAY

par
CHRISTOPH DÜRR

Sujet: **Automates cellulaires quantiques finis**

Soutenue le 6 juin 1997 devant la Commission d'examen

M JEAN-PIERRE JOUANNAUD	Président
M MICHEL COSNARD	
M ARTUR EKERT	
MME CLAIRE KENYON	Rapporteur
M MIKLOS SANTHA	Directeur de thèse
M TOMMASO TOFFOLI	Rapporteur

Table des matières

1

Remerciements

Je tiens à remercier Miklos Santha pour m'avoir initié à la recherche, et pour avoir accepté d'être co-auteur. Merci aussi à Alain Denise, Peter Høyer, Fouad Ibn-Majdoub et Hương LêThanh pour avoir cherché avec moi.
Je suis très reconnaissant à Catherine Reinhard pour avoir corrigé le français de cette thèse.
Un grand merci à Michel Cosnard, Artur Ekert et Jean-Pierre Jouannaud qui ont accepté d'être membre du Jury, et en particulier aux rapporteurs Claire Kenyon et Tom Toffoli qui ont fait un grand travail en évaluant ma thèse.
Pour des discussions fructueuses je tiens à remercier sincèrement Dorit Aharonov, Jean-Paul Allouche, Stephane Boucheron, Gilles Brassard, Richard Cleve, Bruno Durand, Wim van Dam, Dominique Gouyou-Beauchamps, Richard Jozsa, Ngọc-Minh Lê, Seth Lloyd, Bernd Loescher, Fréderic Magniez, Norm Margolus, Jacques Mazoyer, David Meyer, Laurent Rosaz, Umesh Vazirani et John Watrous.
Merci à Ion Filotti de m'avoir poussé à reprendre les études.
Ce travail a été soutenu par les groupes de travail ESPRIT 7097 RAND et ESPRIT RAND2 ainsi que par la Fondation ISI.

FIG. **1.1:** Des fleurs pour les personnes auxquelles je suis reconnaissant. Avec l'aimable autorisation de Marie-France Rivière. (http://www.sito.org/sito/-MASS/Riviere_MF/MFR092P.JPG)

2

Introduction

Le microprocesseur a eu 25 ans l'année dernière. Depuis le processeur 4004 de Intel fabriqué en 1971, les performances des microprocesseurs ont été continuellement améliorées. Ceci est dû à une avancée technologique et dans une moindre mesure technique. Le développement continu de la photo-lithographie a permis "d'imprimer" de plus en plus de transistors de plus en plus petits sur le silicium. Cependant cette progression a une limite pour deux raisons. D'une part il existe une taille minimale en nombre d'atomes pour un transistor. Et d'autre part la transformation croissante d'énergie électrique en chaleur pose des problèmes de refroidissement et d'alimentation.

Toutefois une nouvelle technologie qui serait construite à un niveau atomique et qui aurait une faible consommation d'énergie pourrait contrer ces problèmes.

Dans les années 60 Landauer a montré que toute opération de calcul irréversible (comme mettre une variable à zéro, par exemple) dégage forcément une quantité minimale d'énergie. On pensait alors que tout calcul intéressant doit être irréversible et qu'il existe donc une quantité d'énergie incontournable nécessaire au calcul. Bennet [Ben73] a montré dans les années 70 que tout calcul peut être simulé efficacement (avec un facteur de temps polynomial) par un calcul réversible. En d'autres termes, la dissipation d'énergie d'un ordinateur n'est due qu'à la technologie utilisée et n'est pas inhérente au calcul.

En essayant de simuler des systèmes quantiques sur ordinateur, Feynman, Benioff et Deutsch [Fey82, Ben82a, Ben82b, Deu85, Fey86] se sont posé la question de savoir s'il n'était pas possible de contrer le sûr-coût exponentiel de la simulation en construisant un ordinateur basé sur la mécanique quantique. Les lois de la physique imposent à une telle machine de n'évoluer que de manière bijective. Le support d'un ordinateur quantique pourrait être un système physique soumis aux lois de la mécanique quantique, par exemple des atomes, des électrons, etc.

Un tel ordinateur pourrait tirer profit des particularités des systèmes quantiques pour résoudre certains problèmes plus efficacement qu'une machine classique. Les premiers articles définissant formellement des modèles de calcul quantique (circuit quantique [Deu89, Yao93], machine de Turing quantique [Deu85, BV93]) contenaient l'intuition que ces machines seraient plus puissantes que les ordinateurs classiques. En 1994, Shor [Sho94] a trouvé un algorithme quantique efficace pour la factorisation, venant renforcer cette intuition, car on conjecture qu'il n'existe pas d'algorithme classique polynomial pour ce problème. D'ailleurs le système cryptographique RSA est basé sur cette conjecture. Ce résultat a suscité l'intérêt d'un grand nombre d'informaticiens pour trouver des algorithmes quantiques destinés à d'autres problèmes : la recherche générale [Gro96, BBHT96], son application à la recherche du minimum [DH96] (voir chapitre 9), la transformation de Fourier dans les groupes Abéliens [Kit95] et dans les groupes non-Abéliens [Høy97].

Tous ces algorithmes n'ont évidemment un intérêt pratique que s'il est possible de construire un ordinateur quantique et de le faire fonctionner avec des frais abordables. En 1993 Lloyd [Llo93, Llo94] a envisagé une possibilité de réaliser un ordinateur quantique sous forme d'un automate cellulaire quantique particulier. Un automate cellulaire [FAQ] est un ensemble de cellules disposées sur une grille. Chaque cellule se trouve dans un état qui change en fonction de l'état des voisins. Les automates cellulaires sont utilisés pour modéliser des phénomènes physiques [Mar87], mais sont aussi un modèle de calcul au même titre que les machines de Turing.[1]

Cet article de Lloyd nous a motivé à définir formellement une variante d'automate cellulaire quantique (ACQ fini) [DLS96] ; d'autres variantes ont été introduites indépendamment dans [Wat95, Dam96]. Mais tous ces modèles sont idéalisés, dans le sens où ils supposent que l'ordinateur quantique est coupé du reste du monde. Or l'interaction inévitable avec l'environnement produit un effet dit de *décohérence* qui empêche les mécanismes internes de la machine, et font qu'elle se comporte partiellement comme une machine classique, ce qui restreint l'intérêt de la machine quantique. Désormais la principale activité dans le domaine est consacrée aux codes correcteurs susceptibles de corriger dans une certaine mesure cet effet négatif.

En dehors de la question de savoir si oui ou non, un tel ordinateur est réalisable, les travaux concernant ce sujet ont relancé en physique la recherche sur la décohérence. En informatique ils permettront peut-être de comprendre quel est l'apport en pouvoir de calcul des schémas d'interférences, intervenant dans les ordinateurs quantiques.

Personnellement je suis plutôt intéressé par la question suivante. La version moderne de la thèse de Church-Turing postule que tous les modèles de calcul *raisonnables* peuvent se simuler mutuellement avec un surcoût polynomial. Est-ce que cette équivalence est toujours vérifiée pour les versions quantiques de ces modèles ? Les résultats existants semblent affirmer cette question, mais on ne sait toujours pas si les automates cellulaires quantiques finis peuvent être simulés par des machines de Turing quantique avec un surcoût uniquement polynomial.

Résultats personnels

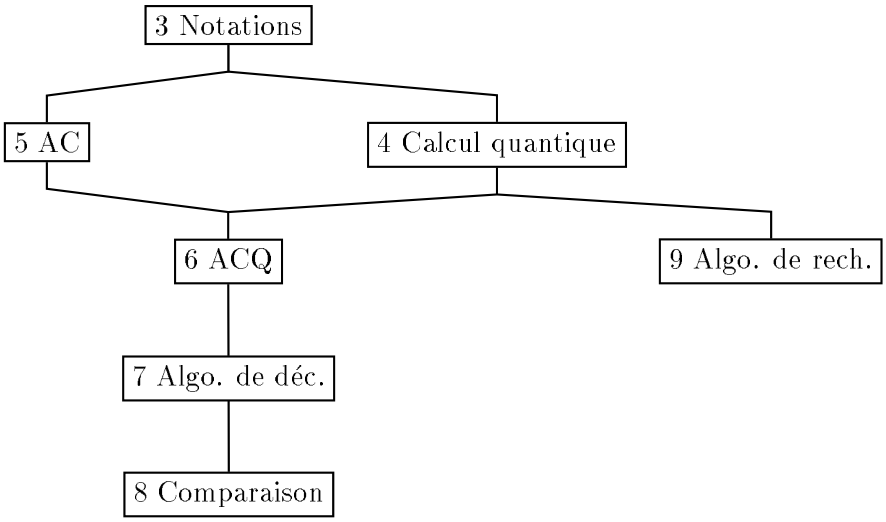
Motivé par un article de Lloyd [Llo93, Llo94] qui décrivait comment on pourrait éventuellement concevoir un ordinateur quantique par un automate cellulaire quantique particulier, nous avons généralisé avec Hương LêThanh et Miklos Santha le modèle des automates cellulaires finis dans [DLS96]. L'idée qu'un ordinateur quantique (ou même le monde) serait peut-être un automate cellulaire quantique est déjà dans l'article fondateur de Feynman [Fey82]. Pour qu'un modèle soit raisonnable il faut pouvoir caractériser ses instances valides. C'est ce que nous avons fait en [DLS96, DS96] en donnant un algorithme qui vérifie si une instance donnée du modèle est valide.

[1] Il est facile de construire pour toute machine de Turing un AC qui le simule sans surcoût. À l'inverse comme une machine de Turing ne peut changer qu'une seule cellule à la fois, elle ne peut simuler une transition complète d'un AC. Par contre elle peut bien sûr calculer le nouvel état d'une cellule donnée. Donc l'état d'un ensemble fini de cellules au bout d'un nombre fini de transitions de l'AC peut être calculé en temps fini par une machine de Turing.

Grover [Gro96] a trouvé un algorithme quantique de recherche générale. Avec Peter Høyer nous avons utilisé cet algorithme pour résoudre de manière optimale le problème de recherche du minimum, auquel il ne s'appliquait pas directement.

Nous donnons également des preuves plus courtes d'une part pour le premier théorème de [BV93], qui établit que l'isométrie et l'unitarité sont des propriétés identiques pour les opérateurs d'évolution des machines de Turing quantiques, et d'autre part pour le premier théorème de [Wat95] qui caractérise les automates cellulaires quantiques partitionnés dont l'opérateur d'évolution est unitaire.

Plan de lecture



⋮ ••••➤ 3

Notations

Dans ce chapitre nous introduisons les notations utilisées tout au long de la thèse.

Ensembles

Nous notons par $\mathbb{N}, \mathbb{Z}, \mathbb{R}$ et $\mathbb{C}$ respectivement l'ensemble des entiers naturels, des entiers, des réels et des complexes. Les entiers modulo k sont notés $\mathbb{Z}_k = \{0, 1, \ldots, k-1\}$. Les réels positifs ou nuls sont notés $\mathbb{R}_+$. Le conjugué de $\alpha \in \mathbb{C}$ est noté $\overline{\alpha}$, et le module $|\alpha|$.

Un intervalle fini sur les entiers est noté $[j, k]$ pour $j \leq k$ et définit l'ensemble fini d'entiers consécutifs $\{j, j+1, \ldots, k\}$.

Soit Q un ensemble. Le cardinal de Q est noté $\mathrm{card}(Q)$.

Soient E et E' des ensembles dénombrables. L'ensemble des vecteurs dont le domaine d'indices est E et dont les composantes sont en Q est noté Q^E. Pour $v \in Q^E$, la composante d'indice $i \in E$ est notée v_i. Pour tout $n \geq 1$, nous abrégeons $Q^{[1,n]}$ par Q^n.

L'ensemble des matrices de Q, dont le domaine d'indices des colonnes est E et celui des lignes est E' est noté $Q^{E \times E'}$. Pour $M \in Q^{E \times E'}$, la composante de la colonne $i \in E$ et ligne $j \in E'$ est notée $M(j, i)$. Si $E = [1, m]$ et $E' = [1, n]$ alors $Q^{E \times E'}$ est noté $Q^{m \times n}$. Pour tout $M \in \mathbb{C}^{E \times E'}$, nous notons $M^\dagger \in \mathbb{C}^{E' \times E}$ la matrice transposée conjuguée de M.

Espaces de vecteurs

Soit $\mathcal{C}$ un ensemble dénombrable. Tout élément u de $\mathbb{C}^{\mathcal{C}}$ est un vecteur complexe dont l'ensemble des indices est $\mathcal{C}$. Pour préciser s'il s'agit d'un vecteur ligne ou d'un vecteur colonne nous le notons respectivement par $\langle u|$ ou $|u\rangle$. En particulier, pour tout $c \in \mathcal{C}$, nous notons $\langle c|$ (respectivement $|c\rangle$), le *vecteur de base* qui contient 1 à l'indice c et 0 ailleurs.

La norme (L_2) d'un vecteur $u \in \mathbb{C}^{\mathcal{C}}$ est définie par

$$\|u\| = \sqrt{\sum_{c \in \mathcal{C}} |u_c|^2}.$$

Nous notons $\ell_2(\mathcal{C})$ l'ensemble des vecteurs de norme finie. L'ensemble $\ell_2(\mathcal{C})$ muni de la norme L_2 est un espace de Hilbert.

Pour tous vecteurs u et v de $\ell_2(\mathcal{C})$ nous notons le produit scalaire par

$$\langle u|v\rangle = \sum_{c\in\mathcal{C}} u_c\overline{v_c}.$$

Nous remarquons que pour $u \in \mathbb{C}^{\mathcal{C}}$ et $c \in \mathcal{C}$ nous avons $\langle u|c\rangle = u_c$, et en particulier pour c et d de $\mathcal{C}$, $\langle c|d\rangle = 1$ si $c = d$ et $\langle c|d\rangle = 0$ sinon. Nous avons aussi $\|u\|^2 = \langle u|u\rangle$.

Si $\langle u|v\rangle = 0$, on dit que u et v sont *orthogonaux* et on note $u \perp v$.

Une matrice $M \in \mathbb{C}^{\mathcal{C}\times\mathcal{C}}$ est une application linéaire sur $\mathbb{C}^{\mathcal{C}}$. La matrice est *isométrique* si pour tout vecteur u nous avons $\|Mu\| = \|u\|$. On dit que M est *unitaire* si

$$MM^\dagger = M^\dagger M = I,$$

où I est l'identité. L'*isométrie* est le fait d'être isométrique et l'*unitarité* le fait d'être unitaire pour une matrice.

Pour tout $c \in \mathcal{C}$ nous notons $M(\cdot, c)$ et $M(c, \cdot)$ le vecteur colonne et respectivement le vecteur ligne de M d'indice c.

La norme (L_1) d'un vecteur $u \in \mathbb{R}_+^{\mathcal{C}}$ est définie par

$$\|u\|_1 = \sum_{c\in\mathcal{C}} u_c.$$

Nous notons $\ell_1(\mathcal{C})$ l'ensemble des vecteurs de $\mathbb{R}_+^{\mathcal{C}}$ de norme (L_1) finie. Soit une matrice $M \in \mathbb{R}_+^{\mathcal{C}\times\mathcal{C}}$. M est une *matrice stochastique* si pour tout vecteur $u \in \ell_1(\mathcal{C})$ nous avons $\|Mu\|_1 = \|u\|_1$.

Par la suite nous n'utiliserons plus la notation $\|\cdot\|_1$, il n'y a donc pas de confusion possible avec la norme L_2.

Matrices unitaires

Nous allons dans cette section énumérer des faits importants concernant les matrices unitaires et les matrices isométriques.

Fait 3.1 *Soit $M \in \mathbb{C}^{\mathcal{C}\times\mathcal{C}}$ un opérateur linéaire. Alors les trois conditions suivantes sont équivalentes :*

1. *M est isométrique.*
2. *Pour tout $c, d \in \mathcal{C}$ le produit scalaire des vecteurs colonnes associés satisfait*

$$\langle M(\cdot, c)|M(\cdot, d)\rangle = \langle c|d\rangle.$$

3. *$M^\dagger M = I$.*

Preuve :

(1) $\Rightarrow$ (2) Soit M isométrique et u et v deux vecteurs orthogonaux. Nous avons d'une part

$$\|M(u + v)\|^2 = \|Mu\|^2 + \|Mv\|^2 + \langle Mu|Mv\rangle + \langle Mv|Mu\rangle$$

et d'autre part

$$\|M(u+v)\|^2 = \|u+v\|^2 = \|u\|^2 + \|v\|^2 = \|Mu\|^2 + \|Mv\|^2,$$

ce qui implique

$$\langle Mu|Mv\rangle + \langle Mv|Mu\rangle = 0.$$

Le même raisonnement pour iu et v implique

$$\langle Miu|Mv\rangle + \langle Mv|Miu\rangle = i\langle Mu|Mv\rangle - i\langle Mv|Mu\rangle = 0.$$

Soient $a, b \in \mathbb{R}$ tel que $\langle Mu|Mv\rangle = a + ib$. Nous obtenons les équations

$$\begin{array}{rcl} (a+ib)+(a-ib) & = & 0 \\ i(a+ib)-i(a-ib) & = & 0. \end{array}$$

La première équation implique $a = 0$ et la deuxième $b = 0$. Donc Mu est orthogonal à Mv. En posant $u = |c\rangle$ et $v = |d\rangle$ pour deux configurations distinctes c et d, nous obtenons

$$\langle M(\cdot, c)|M(\cdot, d)\rangle = 0.$$

De plus

$$\langle M(\cdot, c)|M(\cdot, c)\rangle = \|M|c\rangle\|^2 = \|\,|c\rangle\|^2 = 1.$$

(2) $\Rightarrow$ (1) Soit $u = \sum_{c\in\mathcal{C}} \alpha_c|c\rangle$ un vecteur arbitraire. Soit M tel que pour tout $c, d \in \mathcal{C}$, $\langle M(\cdot, c)|M(\cdot, d)\rangle = \langle c|d\rangle$. Alors par linéarité de M nous avons

$$\begin{array}{rcl} \|Mu\|^2 & = & \langle Mu|Mu\rangle \\ & = & \left\langle M\sum_c \alpha_c|c\rangle \middle| M\sum_d \alpha_d|d\rangle \right\rangle \\ & = & \left\langle \sum_c \alpha_c M|c\rangle \middle| \sum_d \alpha_d M|d\rangle \right\rangle \\ & = & \sum_{c,d} \alpha_c\overline{\alpha_d}\langle M(\cdot, c)|M(\cdot, d)\rangle \\ & = & \sum_{c,d} \alpha_c\overline{\alpha_d}\langle c|d\rangle \\ & = & \sum_{c'} |\alpha_{c'}|^2, \end{array}$$

ce qui est par définition la norme de u au carré.

(2) $\Leftrightarrow$ (3) Pour tous vecteurs de base $u = |c\rangle$ et $v = |d\rangle$, nous avons

$$\begin{array}{rcl} \langle M(\cdot, c)|M(\cdot, d)\rangle & = & \langle Mu|Mv\rangle \\ & = & \langle M^\dagger Mu|v\rangle \\ & = & (M^\dagger M)(d, c). \end{array}$$

Il s'en suit que pour $M^\dagger M = I$ si et seulement si pour tout c, d, $\langle M(\cdot, c)|M(\cdot, d)\rangle = \langle c|d\rangle$.

□

Fait 3.2 *Soit $M \in \mathbb{C}^{\mathcal{C}\times\mathcal{C}}$ un opérateur linéaire. Si M est une isométrie alors les vecteurs lignes sont de norme au plus 1.*

Preuve : Soit c une configuration arbitraire et $u = \overline{M(c,\cdot)} = M^\dagger|c\rangle$ le conjugué du vecteur ligne de M à l'indice c. Nous avons

$$\langle c|Mu\rangle = \Big\langle M^\dagger|c\rangle \Big| u\Big\rangle = \langle u|u\rangle.$$

La projection de Mu sur $|c\rangle$ a donc la norme $\|u\|^2$. Puisque la projection d'un vecteur v sur un vecteur de norme 1 ne peut pas avoir une norme plus grande que v, nous avons (inégalité de Schwarz)

$$|\langle c|Mu\rangle| \leq \|Mu\|.$$

Par hypothèse M préserve la norme, alors $\|Mu\| = \|u\|$. En utilisant l'égalité ci-haut, ceci nous mène à

$$\|u\|^2 \leq \|u\|$$

et donc $\|u\| \leq 1$. □

Fait 3.3 *Soit $M \in \mathbb{C}^{\mathcal{C}\times\mathcal{C}}$ un opérateur linéaire. M est unitaire si et seulement si M est isométrique et les vecteurs lignes de M sont de norme 1.*

Preuve : Si M est unitaire, alors par définition $MM^\dagger = I = M^\dagger M$. Donc $M^\dagger$ est isométrique, et par le fait 3.1 les vecteurs lignes de M sont de norme 1.

Pour la partie "si", soit c une configuration arbitraire et $u = M^\dagger|c\rangle$. Par le fait précédent nous avons $\langle c|Mu\rangle = \langle u|u\rangle$, et par hypothèse $\|u\| = 1$. Alors la projection de Mu sur c est de norme 1. Par hypothèse M préserve la norme, donc Mu est aussi de norme 1, et la projection de Mu sur tout autre vecteur de base doit être 0. Donc pour toute configuration d nous avons que $\langle d|Mu\rangle$ est égal à 1 si $d = c$ et 0 sinon. Comme $Mu = MM^\dagger|c\rangle$ ceci équivaut à $MM^\dagger = I$, ce qui conclut la preuve. □

4

Calcul quantique

Dans ce chapitre nous introduisons les ordinateurs quantiques en général et définissons en particulier le modèle de *machine de Turing quantique*.

4.1 Mécanique quantique

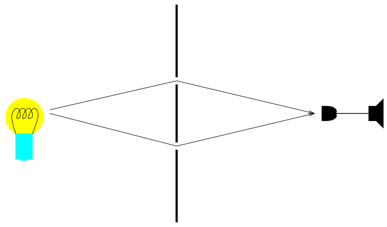


FIG. **4.1:** Expérience à 2 fentes.

Pour introduire le calcul quantique il nous semble important de fournir au préalable les notions de base de la mécanique quantique. Le livre de vulgarisation “Quantum Electronic dynamics” de Feynman [Fey85] en est une introduction plus complète. Nous allons décrire l'expérience principale dont traite cet ouvrage.
Considérons l'expérience illustrée en figure 4.1. Soit un mur avec deux fentes laissant passer la lumière. D'un côté on place une faible source de lumière. De l'autre côté on dispose d'un détecteur de photon avec un haut-parleur qui émet un “clic” à chaque fois que le détecteur intercepte un photon. La lumière a alors deux chemins pour atteindre l'appareil de mesure, un par la fente du haut et un par celle du bas.

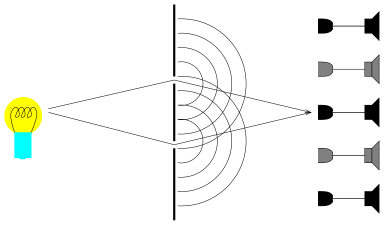


FIG. **4.2:** Phénomène décrit par le modèle des ondes.

Nous remarquons qu'il y a des positions où le détecteur n'intercepte aucun photon et d'autres où il en intercepte de temps en temps. Le phénomène s'explique par le *modèle d'onde* de la lumière, comme illustré en figure 4.2. Quand la lumière passe par une fente elle se propage ensuite comme une onde. Les ondes émises par la fente du haut et celles émises par la fente du bas se superposent, et s'amplifient à certains endroits et s'annulent à d'autres. Le fait que l'appareil intercepte la lumière ou pas dépend de la différence des distances entre le détecteur et chacune des fentes.

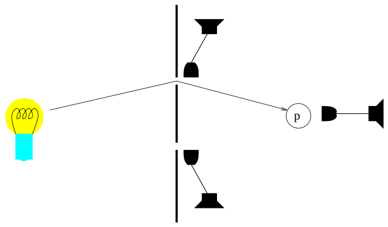


FIG. **4.3:** Phénomène décrit par le modèle de photons.

Maintenant changeons l'expérience un petit peu. Nous plaçons derrière chaque fente un détecteur de photon, qui émet un clic sur un haut-parleur dès qu'un photon passe, sans toutefois l'intercepter. On peut alors observer les phénomènes suivants. Les deux haut-parleurs derrière les fentes ne cliquent jamais en même temps. Si le détecteur initial intercepte un photon, il y a exactement un des autres détecteurs qui émet simultanément un clic. Ceci s'explique par le *modèle de particules* de la lumière. Un photon emprunte un seul chemin pour aller de la source vers le détecteur initial. Par contre dans cette expérience il n'existe plus de positions où cet appareil n'intercepte plus rien. La lumière ne se comporte plus comme une onde.
Nous avons alors deux modèles de la lumière, dont chacun explique un phénomène particulier. La mécanique quantique est un modèle qui explique aussi bien la première expérience que la deuxième.

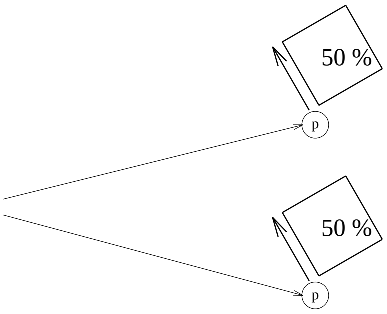


FIG. **4.4:** Un photon en superposition.

Quand un photon se déplace de la source vers le détecteur, il emprunte les deux chemins en même temps. On dit qu'à chaque moment il est en *superposition* entre deux positions. A chacune des positions nous associons une *amplitude*, un nombre complexe, que nous pouvons représenter par une flèche dans le plan, comme en figure 4.4. Si nous observons le photon, nous ne le détecterons qu'à un seul des endroits. Il choisira alors au hasard une des positions, avec une probabilité qui est le carré du module de l'amplitude associée. Après une observation l'amplitude associée à la position où il a été observé sera de module 1 et l'amplitude associée à l'autre position sera 0. Ceci explique l'expérience de la figure 4.3. Un photon qui atteint l'appareil a été *forcé* d'emprunter une des deux fentes par les appareils de mesure situés juste derrière le mur.

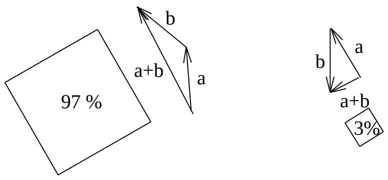


FIG. **4.5:** Additions de deux amplitudes.

Expliquons maintenant l'expérience de la figure 4.2 à la page d'avant avec le modèle de la mécanique quantique. Dans ce modèle, à tout moment il existe une instance du photon qui a emprunté le chemin par le haut et une qui a pris le chemin du bas. A chaque instance nous associons une amplitude. Au fur et à mesure que les instances avancent sur leur trajectoire, la phase de leur amplitude associée change proportionnellement avec la distance parcourue. Ceci se traduit par un changement d'angle de la flèche par laquelle nous représentons graphiquement l'amplitude. Elle tourne alors comme les aiguilles d'une montre, au cours du déplacement.
Quand deux instances du même photon arrivent au même endroit — en l'occurrence ici sur le détecteur — l'amplitude avec laquelle le photon s'y trouve est la somme des amplitudes de ses instances. Contrairement aux probabilités, les amplitudes peuvent s'annuler ou s'amplifier, comme l'illustre la figure 4.5. C'est alors la différence des distances respectives entre les fentes et le détecteur qui détermine si oui ou non l'appareil interceptera des photons.

Le modèle de la mécanique quantique ne s'applique pas qu'aux photons, mais aussi à toutes les autres particules, les électrons, les ions, les atomes non-chargés, etc.

4.2 Conception physique du calcul probabiliste

Soit $\mathcal{C}$ l'ensemble dénombrable des configurations d'une machine, par exemple d'une machine de Turing. L'état d'une machine probabiliste est décrit par un vecteur $u \in \ell_1(\mathcal{C})$ de norme (L_1) 1, qu'on appelle une *distribution.*

Observation

L'observation totale de la machine révélera une configuration $c \in \mathcal{C}$ avec une probabilité u_c. La restriction aux vecteurs de norme 1 assure que la somme des probabilités vaut 1. L'état après observation sera $|c\rangle$, c'est-à-dire la distribution qui associe probabilité 1 à c et 0 ailleurs. De manière plus générale, une observation partielle révèle une information, avec une probabilité qui est la somme des probabilités associées aux configurations compatibles avec cette information. Par exemple l'observation d'une cellule du ruban d'une machine de Turing probabiliste révèle une lettre x avec une probabilité qui est la somme des probabilités de toutes les configurations qui ont la lettre x dans cette cellule. Après observation, l'état de la machine est la distribution normalisée, restreinte aux configurations compatibles avec le résultat de l'observation. C'est-à-dire, si nous répétons immédiatement l'observation, nous obtiendrons le même résultat avec certitude, et l'état de la machine n'en sera pas modifié.

Évolution

L'évolution d'une machine probabiliste est décrite par un opérateur linéaire qui préserve la norme L_1, c'est-à-dire une matrice stochastique $M \in \mathbb{R}_+^{\mathcal{C}\times\mathcal{C}}$. Pour toutes les configurations c et d, l'entrée $M(d, c)$ indique la probabilité de faire une transition de c vers d. Si la machine se trouve dans une distribution u à un moment donné, par définition elle se trouvera en Mu à l'instant suivant.

Ce fait motive la restriction des matrices d'évolution aux matrices stochastiques, car il est nécessaire que l'image d'une distribution par l'opérateur d'évolution soit aussi une distribution, et pour cela l'opérateur doit préserver la norme L_1.

4.3 Généralisation quantique

Une machine quantique est similaire en de nombreux points à une machine probabiliste. Nous commencerons à exposer ce qu'il y a en commun, avant d'en discuter les différences.

L'état d'une machine quantique est un vecteur $u \in \ell_2(\mathcal{C})$ de norme (L_2) 1, qu'on appelle *superposition.* Les composantes de ce vecteur s'appellent des *amplitudes.* On dit que la machine se trouve dans la configuration c avec amplitude u_c.

Observation

L'observation totale de la machine révèle une configuration $c \in \mathcal{C}$ avec une probabilité $|u_c|^2$. La restriction aux vecteurs de norme 1 assure que la somme des probabilités vaut 1. Une observation partielle révèle une information, avec une probabilité qui est la somme des carrés des modules des amplitudes associées aux configurations compatibles avec cette information. Après observation, l'état de la machine est la superposition normalisée, restreinte aux configurations compatibles avec le résultat de l'observation.

Évolution

L'évolution d'une machine probabiliste est décrite par une matrice $M \in \mathbb{C}^{\mathcal{C}\times\mathcal{C}}$, qu'on nomme *opérateur d'évolution*. Nous disons que la machine est *valide* (ou *unitaire*) si M est unitaire. Pour toutes les configurations c et d, l'entrée $M(d,c)$ indique l'amplitude de faire une transition de c vers d. Si la machine se trouve dans une superposition u à un moment donné, par définition elle se trouvera en Mu à l'instant suivant.

Il est nécessaire que l'image d'une superposition par l'opérateur d'évolution soit aussi une superposition, et pour cela l'opérateur doit préserver la norme L_2. Cependant la condition d'unitarité imposée aux opérateurs d'évolution est une restriction plus forte, qui a une motivation physique.

Généralisation

La machine de Turing quantique peut simuler efficacement une machine de Turing probabiliste, et est dans ce sens une généralisation de ce modèle. La simulation repose sur quatre résultats.

1. Sans perte de généralité on peut supposer qu'une machine de Turing probabiliste est une machine déterministe qui a un ruban d'entrée supplémentaire, comportant des bits aléatoires.

2. Toute machine de Turing déterministe peut être simulée par une machine de Turing réversible, c'est-à-dire dont la fonction de transition globale est injective [Ben73] (précédé par [Lec63]).

3. Si la fonction de transition globale d'une machine de Turing est injective, alors elle est aussi bijective [BV93].

4. Il existe une opération de *tirage quantique*[1], qui permet à une machine de Turing quantique entre autres de produire à partir d'un ruban tout blanc un ruban contenant des bits aléatoires.

4.4 Modèles d'ordinateurs quantiques

Pour formaliser des algorithmes quantiques ou comparer la puissance du calcul quantique avec celle du calcul classique, il faut un modèle formel informatique (par opposition à "modèle physique") d'un ordinateur quantique. Différents modèles ont été définis à ce jour : les machines de Turing quantiques [Ben82a, Ben82b], les circuits quantiques [Deu89], les automates cellulaires quantiques (voir chapitre 6), et même le lambda-calcul quantique [May96].

Ce dernier modèle ne nous semble pas "raisonnable" pour les raisons que nous donnons en section 4.5, et donc nous l'ignorons pour l'instant.

Les problèmes naturels qui se posent avec ces modèles sont :

1. Pour chacun de ces modèles caractériser les instances valides, c'est-à-dire les instances dont l'opérateur d'évolution est unitaire.

[1] en anglais *Quantum Flip*

2. Pour chacun de ces modèles trouver une instance universelle, c'est-à-dire une instance valide qui simule efficacement toutes les autres instances valides, c'est-à-dire avec un coût de simulation uniquement polynomial.

3. Comparer les modèles entre eux, c'est-à-dire décider si pour toute instance valide d'un premier modèle il existe une instance valide d'un deuxième modèle qui la simule efficacement.

Certains de ces problèmes ont été résolus :

1. Bernstein et Vazirani [BV93] ont donné des conditions locales et facilement vérifiables caractérisant les machines de Turing valides. Dürr, LêThanh et Santha [DLS96, DS96] ont décrit un algorithme qui pour un automate cellulaire fini quantique donné décide si l'opérateur d'évolution associé est unitaire. Van Dam [Dam96] a fourni un algorithme similaire pour les automates cellulaires quantiques périodiques, et Watrous [Wat95] pour les automates cellulaires quantiques partitionnés. Pour des circuits quantiques une telle caractérisation est inutile, car ils sont par définition unitaires.

2. Deutsch [Deu85] a donné une première machine de Turing quantique universelle, mais dont le coût de simulation était exponentiel. Bernstein et Vazirani [BV93] ont ensuite décrit une machine universelle efficace. Van Dam [Dam96] a construit un automate cellulaire quantique périodique universel. A ce jour on se sait pas s'il existe un automate cellulaire quantique fini universel. Une porte quantique est universelle si tout circuit peut être simulé par un circuit qui n'est construit qu'avec cette porte. Deutsch a décrit une porte quantique universelle [Deu89].

3. Yao [Yao93] a montré que toute machine de Turing valide peut être simulée par un circuit valide. La simulation inverse est triviale. Watrous [Wat95] a décrit comment un automate cellulaire quantique partitionné peut simuler une machine de Turing quantique valide et vice-versa.

   Van Dam [Dam96] a présenté une simulation efficace d'un automate cellulaire quantique périodique par un circuit.

   Les relations entre les différentes variantes d'automates cellulaires quantiques seront décrites au chapitre 8.

   La question de savoir si un automate cellulaire quantique fini peut être simulé par les autres modèles de calcul quantique reste ouverte.

4.5 Lambda-calcul quantique

Dans cette section nous présentons la raison pour laquelle le modèle de lambda-calcul introduit par Maymin ne nous semble pas raisonnable.

Maymin représente des superpositions par des collections — des multi-ensembles hétérogènes — dont chaque élément peut être doté d'un signe de phase. Le module de l'amplitude associée à un élément est défini comme l'inverse de la racine du nombre d'éléments dans cette collection. Par exemple la collection $\{a, b\}$ représente la superposition $(|a\rangle + |b\rangle)/\sqrt{2}$. Si le terme b de cette collection se réduit à $\{-c, d\}$, le résultat est $\{a, \{-c, d\}\}$. Maymin a défini une règle qui réduit ce dernier résultat à $\{a, -c, d\}$, correspondant à la superposition

$(|a\rangle - |c\rangle + |d\rangle)/\sqrt{3}$, alors que, intuitivement, nous devrions obtenir $(2|a\rangle - |c\rangle + |d\rangle)/\sqrt{4}$. Cette particularité lui permet de résoudre SAT efficacement, et rend son modèle très différent des autres modèles de calcul quantique.

4.6 Machine de Turing Quantique

Les machines de Turing sont importantes pour le domaine de la complexité. Les principales classes de complexité sont définies sur ce modèle. Nous allons alors considérer maintenant sa généralisation quantique.

Une machine de Turing quantique (MTQ) est un triplet (Σ, Q, δ), où Σ est un alphabet fini, comportant un symbole *blanc* $\sqcup$, Q est un ensemble fini d'états et δ une fonction de transition locale de la forme

$$\delta : Q \times \Sigma \times \Sigma \times Q \times \{\lhd, \perp, \rhd\} \to \mathbb{C}.$$

La machine comporte un ruban infini dans les deux sens, composé de cellules indicées par $\mathbb{Z}$. Chaque cellule contient une lettre de Σ, mais seulement un nombre fini de cellules diffèrent du symbole blanc $\sqcup$. De plus la machine a une tête de lecture/écriture qui pointe sur une des cellules du ruban, et qui se trouve dans un état de Q. La fonction locale δ indique que si la machine est dans l'état q et lit la lettre x, alors avec amplitude $\delta(q, x, y, p, d)$ elle écrit y sur la cellule pointé par la tête, change dans l'état p et déplace sa tête dans la direction $d \in \{\lhd, \perp, \rhd\}$. Ceci est illustré en figure 4.6. Plus précisément la direction $\lhd$ indique que la tête se déplace d'une cellule vers la gauche, $\perp$ indique que la tête reste sur place et $\rhd$ précise que la tête bouge d'une cellule vers la droite.

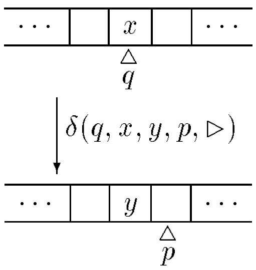


FIG. **4.6:** Une transition d'une MTQ.

Une configuration de la machine est un triplet composé du contenu du ruban, de l'état et de la position de la tête de lecture/écriture. Nous notons $\mathcal{C}$ l'ensemble des configurations. Cet ensemble est clairement dénombrable. L'opérateur d'évolution associé est une matrice $U \in \mathbb{C}^{\mathcal{C} \times \mathcal{C}}$. L'entrée $U(c', c)$ désigne l'amplitude de faire une transition de la configuration c vers c'. Cette valeur est 0 si c' n'est pas accessible à partir de c en une étape, et sinon est l'amplitude $\delta(q, x, y, p, d)$ pour des valeurs q, x, y, p, d appropriées.

Nous notons $\delta(\cdot, \cdot, y, p, d)$ le vecteur $\sum_{q,x} \delta(q, x, y, p, d)|q, x\rangle \in \ell_2(Q \times \Sigma)$. De manière similaire nous utiliserons cette notation pour $\delta(q, x, \cdot, \cdot, \cdot)$ et $\delta(q, x, y, \cdot, d)$.

Contrairement à nous, Bernstein et Vazirani ajoutent à leur définition de la machine de Turing quantique les restrictions suivantes.

- Les seules amplitudes autorisées sont celles pour lesquelles il existe une machine de Turing (classique) qui calcule le i-ème bit des parties réelle et imaginaire en temps polynomial en i.

- Le type et le moment des observations à effectuer sur la machine sont déterminés par un algorithme classique polynomial en la taille de l'entrée.

- La tête ne peut pas rester sur place, les seules directions possibles sont $\{\lhd, \rhd\}$.

Les deux premières restrictions sont cruciales pour leur preuve de l'existence d'une machine universelle. Depuis l'article de Yao [Yao93] la dernière restriction est devenue inutile. Cependant nous n'entrerons pas en ces considérations, car il n'y a que la caractérisation des machines valides qui nous concerne. Nous l'exposons maintenant.

Théorème 4.1 ([BV93, théorème 2] généralisé par [Mag95]) *Soit* (Q, Σ, δ) *une machine de Turing quantique et* U *l'opérateur d'évolution associé. Alors* U *est isométrique si et seulement si les conditions suivantes sont satisfaites :*

$$\forall q \in Q, x \in \Sigma : \|\delta(q, x, \cdot, \cdot, \cdot)\| = 1 \tag{4.1}$$

$$\forall q, q' \in Q, x, x' \in \Sigma : (q, x) \neq (q', x') \Rightarrow \langle \delta(q, x, \cdot, \cdot, \cdot) | \delta(q', x', \cdot, \cdot, \cdot) \rangle = 0 \tag{4.2}$$

$$\forall q, q' \in Q, x, x', y, y' \in \Sigma : \\ \langle \delta(q, x, y, \cdot, \rhd) | \delta(q', x', y', \cdot, \bot) \rangle + \langle \delta(q, x, y, \cdot, \bot) | \delta(q', x', y', \cdot, \lhd) \rangle = 0 \tag{4.3}$$

$$\forall q, q' \in Q, x, x', y, y' \in \Sigma : \langle \delta(q, x, y, \cdot, \rhd) | \delta(q', x', y', \cdot, \lhd) \rangle = 0 \tag{4.4}$$

Preuve : Soit c une configuration arbitraire, q son état et x la lettre sous la tête. Par définition de l'opérateur d'évolution nous avons $\|U(\cdot, c)\| = \|\delta(q, x, \cdot, \cdot, \cdot)\|$. Donc les vecteurs colonnes de U sont de norme 1 si et seulement si condition 4.1 est satisfaite.
Nous montrons maintenant que les autres conditions caractérisent l'orthogonalité des vecteurs colonnes. Soient c, c' deux configurations distinctes arbitraires. Alors $\langle U(\cdot, c) | U(\cdot, c') \rangle$ est différent de 0 seulement s'il existe une configuration c'' qui est accessible à la fois de c et de c' en une seule transition. Soient A_0, A_1 et A_2 les ensembles des paires de configurations (c, c') telles que les rubans de c et de c' soient identiques en dehors des cellules pointées par leurs têtes respectives, et que les positions de leurs têtes diffèrent respectivement d'aucune, d'une ou de deux cellules. Par définition des machines de Turing quantiques, seulement les paires $(c, c') \in A_0 \cup A_1 \cup A_2$ peuvent accéder à une configuration commune, comme illustré en figure 4.7.

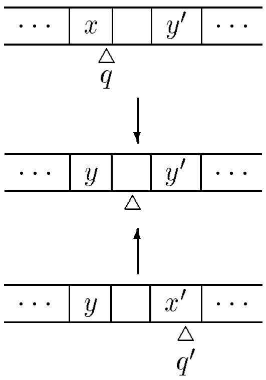


Fig. **4.7:** Deux configurations de A_2 accédant à une même configuration.

Nous avons alors pour toute paire de configurations (c, c') de A_0 (respectivement A_1 et A_2), $\langle U(\cdot, c) | U(\cdot, c') \rangle = 0$ si et seulement si condition 4.2 (respectivement condition 4.3 et 4.4) est satisfaite. □

Théorème 4.2 ([BV93, théorème 1]) *Soit* (Q, Σ, δ) *une machine de Turing quantique et* U *l'opérateur d'évolution associé. Alors* U *est unitaire si et seulement si* U *est isométrique.*

Par définition de l'unitarité, pour prouver ce théorème il suffit de supposer que U est isométrique et de montrer qu'alors U est unitaire. Il y a eu plusieurs preuves. Dans la version conférence de [BV93] le théorème est montré en prouvant que U est surjectif, et dans la version journal [BV97] en prouvant que les lignes de U sont de norme 1. Les deux preuves argumentent avec une sous-matrice de U. Nous allons présenter une troisième preuve plus directe.

Preuve : Dans toute la preuve les domaines des variables suivantes sont : $q, p \in Q$, $x, y \in \Sigma$ et $d \in \{\lhd, \perp, \rhd\}$. Supposons que U soit isométrique. Soit c une configuration arbitraire, p son état et $yy'y''$ les trois lettres centrées autour de la tête, comme illustré en figure 4.8. La ligne d'indice c dans U contient les amplitudes de faire une transition vers c. Le carré de la norme de cette ligne est

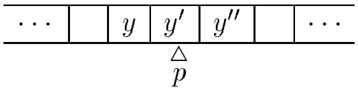


FIG. **4.8:** Une configuration.

$$\begin{aligned}
\|U(c,\cdot)\|^2 &= \sum_{c'} |U(c,c')|^2 \\
&= \sum_{q,x} |\delta(q,x,y,p,\rhd)|^2 + \sum_{q,x} |\delta(q,x,y',p,\perp)|^2 + \sum_{q,x} |\delta(q,x,y'',p,\lhd)|^2 \\
&= \|\delta(\cdot,\cdot,y,p,\rhd)\|^2 + \|\delta(\cdot,\cdot,y',p,\perp)\|^2 + \|\delta(\cdot,\cdot,y'',p,\lhd)\|^2.
\end{aligned}$$

Par le fait 3.3 nous avons pour toutes lettres y, y', y'' et tout état p,

$$\|\delta(\cdot,\cdot,y,p,\rhd)\|^2 + \|\delta(\cdot,\cdot,y',p,\perp)\|^2 + \|\delta(\cdot,\cdot,y'',p,\lhd)\|^2 \leq 1,$$

et nous devons montrer qu'il y a égalité.

Soient σ et π deux permutations arbitraires sur Σ. Alors la cardinalité de $Q \times \Sigma$ est

$$\begin{aligned}
\sum_{q,x} 1 &= \sum_{q,x} \|\delta(q,x,\cdot,\cdot,\cdot)\|^2 \qquad \text{par le thm. 4.1} \\
&= \sum_{q,x,y,p,d} |\delta(q,x,y,p,d)|^2 \\
&= \sum_{y,p} \|\delta(\cdot,\cdot,y,p,\rhd)\|^2 + \|\delta(\cdot,\cdot,y,p,\perp)\|^2 + \|\delta(\cdot,\cdot,y,p,\lhd)\|^2 \\
&= \sum_{y,p} \|\delta(\cdot,\cdot,y,p,\rhd)\|^2 + \|\delta(\cdot,\cdot,\sigma(y),p,\perp)\|^2 + \|\delta(\cdot,\cdot,\pi(y),p,\lhd)\|^2.
\end{aligned}$$

Donc pour toute lettre y et état p nous avons

$$\|\delta(\cdot,\cdot,y,p,\rhd)\|^2 + \|\delta(\cdot,\cdot,\sigma(y),p,\perp)\|^2 + \|\delta(\cdot,\cdot,\pi(y),p,\lhd)\|^2 = 1,$$

ce qui conclut la preuve. □

4.7 Choix des amplitudes de transition

Adleman, DeMarrais et Huang [ADH97] ont évalué le pouvoir de calcul de ce modèle en fonction des amplitudes utilisées par la fonction de transition locale. De manière très simplifiée, en codant en une amplitude réelle α un ensemble d'entiers tel que le i-ème bit est 1 si et seulement i est élément de cet ensemble, il est possible de résoudre de manière probabiliste le problème de l'arrêt avec une machine de Turing quantique.

Il est donc nécessaire de restreindre les modèles de calcul quantique à n'utiliser que des amplitudes de transition d'un certain sous-ensemble de $\mathcal{C}$, pour que ces modèles soient *raisonnables*.

Cependant comme nous ne comparons pas dans cette thèse le pouvoir de calcul des ordinateurs quantiques avec les ordinateurs classiques, nous nous permettons d'omettre cette restriction. Notre algorithme de décision pour des ACQ quantiques unitaires est insensible à une telle restriction.

4.8 Produit tensoriel

Dans cette section nous définissons d'abord le produit tensoriel de deux superpositions. Ensuite après avoir exposé l'intuition de cette notion et prouvé une propriété importante, nous généralisons le produit tensoriel et la propriété à plusieurs superpositions.

Produit tensoriel de deux superpositions

Soient Q et Q' des ensembles dénombrables. Alors pour tous vecteurs $u \in \ell_2(Q), u' \in \ell_2(Q')$ nous définissons le *produit tensoriel* entre u et u',

$$u \otimes u' \in \ell_2(Q \times Q'),$$

pour tout $xx' \in Q \times Q'$ par

$$(u \otimes u')_{xx'} = u_x u'_{x'},$$

ce qui peut aussi être noté par

$$\langle u \otimes u' | xx' \rangle = \langle u|x\rangle \cdot \langle u'|x'\rangle.$$

Soit Q l'ensemble d'états d'une "cellule", et $v \in \ell_2(Q \times Q)$ l'état de deux cellules alors on dit qu'elles sont *indépendantes* s'il existe $u \in \ell_2(Q), u' \in \ell_2(Q)$ tel que $v = u \otimes u'$, sinon on dit que les cellules sont *enchevêtrées*[2]. L'intuition de ces deux notions peut être illustrée par des observations totales de la première puis de la deuxième cellule. Dans le cas des cellules indépendantes — où $v = u \otimes u'$ — la probabilité d'observer un $x' \in Q$ dans la deuxième cellule sera toujours $|u'_{x'}|^2$, quelque soit l'état qui a été observé dans la première cellule. Ceci n'est pas le cas pour des cellules enchevêtrées. Prenons l'exemple de $Q = \{0, 1\}$ et $v = (|00\rangle + |11\rangle)/\sqrt{2}$. L'observation de la première cellule révélera un $x \in Q$ avec probabilité uniforme. L'état des deux cellules deviendra alors $|xx\rangle$, et donc le résultat d'une observation de la deuxième cellule sera x avec certitude.

Lemme 4.1 *Soient les superpositions $u, v \in \ell_2(Q)$ et $u', v' \in \ell_2(Q')$. Alors*

$$\langle u \otimes u' | v \otimes v' \rangle = \langle u|v\rangle \cdot \langle u'|v'\rangle.$$

Preuve :

$$\begin{aligned}
\langle u \otimes u' | v \otimes v' \rangle &= \sum_{xx' \in Q \times Q'} \langle u \otimes u' | xx' \rangle \cdot \overline{\langle v \otimes v' | xx' \rangle} \\
&= \sum_{xx' \in Q \times Q'} \langle u|x\rangle \cdot \langle u'|x'\rangle \cdot \overline{\langle v|x\rangle} \cdot \overline{\langle v'|x'\rangle} \\
&= \left(\sum_{x \in Q} \langle u|x\rangle \cdot \overline{\langle v|x\rangle} \right) \cdot \left(\sum_{x' \in Q'} \langle u'|x'\rangle \cdot \overline{\langle v'|x'\rangle} \right) \\
&= \langle u|v\rangle \cdot \langle u'|v'\rangle.
\end{aligned}$$

[2] En anglais : *entangled*

□

Ce corollaire suit immédiatement.

Corollaire 4.1 *Soient les superpositions* $u \in \ell_2(Q)$ *et* $u' \in \ell_2(Q')$. *Alors*

$$\|u \otimes u'\| = \|u\| \cdot \|u'\|.$$

Produit tensoriel de plusieurs superpositions

Nous établissons ces notions dans un cadre plus général. Pour tout ensemble fini Q et tout intervalle fini $I = [a, b]$ soit le produit tensoriel entre $u^a, u^{a+1}, \ldots, u^b$,

$$\bigotimes_{i \in I} u^i \in \ell_2(Q^I)$$

défini pour tout $w \in Q^I$ par

$$\left\langle \bigotimes_{i \in I} u^i \middle| w \right\rangle = \prod_{i \in I} \langle u^i | w_i \rangle.$$

Ce produit satisfait les relations suivantes.

Lemme 4.2 *Pour tout ensemble fini* Q*, tout intervalle fini* $I = [a, b]$ *et tout* $u^a, \ldots, u^b, v^a, \ldots, v^b \in \ell_2(Q)$ *nous avons*

$$\left\langle \bigotimes_{i \in I} u^i \middle| \bigotimes_{i \in I} v^i \right\rangle = \prod_{i \in I} \langle u^i | v^i \rangle.$$

Preuve : par induction sur la taille de I. Pour $\mathrm{card}(I) = 1$ l'équation est triviale. Supposons qu'elle est satisfaite pour tout intervalle fini I tel que $\mathrm{card}(I) = n$. Soit $I = [a, a+n]$ un intervalle arbitraire. Alors

$$\begin{aligned}
\left\langle \bigotimes_{i \in I} u^i \middle| \bigotimes_{i \in I} v^i \right\rangle &= \left\langle u^a \otimes \left(\bigotimes_{i \in [a+1, a+n]} u^i \right) \middle| v^a \otimes \left(\bigotimes_{i \in [a+1, a+n]} v^i \right) \right\rangle \\
&= \langle u^a | v^a \rangle \cdot \left\langle \bigotimes_{i \in [a+1, a+n]} u^i \middle| \bigotimes_{i \in [a+1, a+n]} v^i \right\rangle \qquad (4.5) \\
&= \prod_{i \in I} \langle u^i | v^i \rangle. \qquad (4.6)
\end{aligned}$$

L'équation 4.5 est justifiée par le lemme 4.1 et l'équation 4.6 par l'hypothèse d'induction. □

Ce corollaire découle du lemme précédent.

Corollaire 4.2 *Pour tout ensemble fini* Q*, tout intervalle fini* $I = [a, b]$ *et tout* $u^a, \ldots, u^b \in \ell_2(Q)$ *nous avons*

$$\left\| \bigotimes_{i \in I} u^i \right\| = \prod_{i \in I} \|u^i\|.$$

5

Automates cellulaires classiques

Les automates cellulaires (AC) sont un modèle qui réalise de manière discrète les notions du temps, de l'espace et de la localité de la physique classique. Cette propriété permet aux automates cellulaires d'être un outil de modélisation qui trouve son application en physique, biologie, sociologie, etc.

Sans vouloir flatter un des rapporteurs, un bon résumé sur l'histoire des automates cellulaires nous semble être celui qui est présenté dans [TM87, section 1.4].

Informellement un automate cellulaire consiste en un ensemble de cellules, disposées sur une grille. Toute cellule se trouve dans un état particulier. A chaque coup d'horloge les cellules changent de manière synchrone leur état en fonction de leur propre état et de l'état de leur voisins. Dans le modèle que nous considérons

- les cellules sont indicées par $\mathbb{Z}^d$ où d est la *dimension*,
- l'état d'une cellule est élément d'un ensemble fini d'états,
- chaque cellule a le même voisinage relatif,
- la changement d'état des cellules est indépendant de la position dans la grille et du temps. Il est déterminé par une fonction qui envoie pour chaque cellule l'état de son voisinage au nouvel état.

Cette *fonction de transition locale* induit naturellement une fonction de *transition globale* qui envoie une configuration de l'automate cellulaire à son successeur. Dans ce modèle des règles locales très simples peuvent alors définir un comportement global complexe.[1]

5.1 Exemple d'automate cellulaire : Le jeu de la vie

L'exemple le plus connu est sans doute le *jeu de la vie*, inventé par Conway et vulgarisé par Gardner [Gar70]. (Tapez `M-x life` sous l'éditeur Emacs pour voir une réalisation de cet automate.)

Cet automate consiste en une grille carrée à 2 dimensions, où chaque cellule peut être soit morte, soit vivante. A chaque instant l'état d'une cellule change en fonction de son propre

[1] Ceci n'est pas propre aux automates cellulaires. Voir la course à la plus petite machine de Turing universelle [Min67, section 14.8].

état et de l'état des huit voisins immédiats. Si le nombre de voisins immédiats vivants est exactement 3, la cellule devient vivante, si ce nombre est 2, elle reste dans son état précédent, et sinon elle meurt.

Dans ce modèle il existe des *glisseurs*, un ensemble de cellules vivantes, qui au bout de quelques étapes se retrouvent décalées dans la grille. Figure 5.1 illustre un de ces glisseurs. Il existe aussi des miroirs de glisseurs, des générateurs de glisseurs, des absorbeurs, etc. En somme il est possible de simuler avec ce modèle une machine de Turing, et codant 1 et 0 par la présence et l'absence d'un glisseur, et en utilisant des constructions particulières pour réaliser les opérations booléennes. Les détails de cette simulation ont été établis par Roka [Rok96].

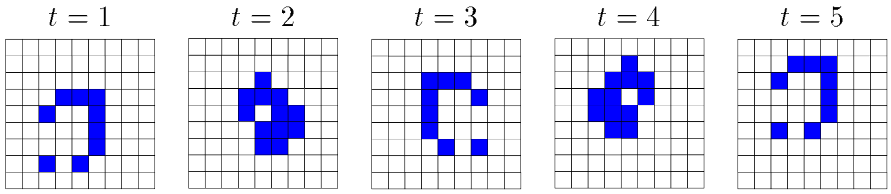


FIG. 5.1: Différentes étapes d'un glisseur dans le jeu de la vie de Conway

5.2 Définition du modèle et de variantes

Nous allons maintenant définir formellement les automates cellulaires dont les cellules sont dans un espace à une dimension. Nous nous restreignons à un type de voisinage particulier, où le nouvel état d'une cellule dépend de son état courant et de celui des $r-1$ cellules immédiatement à droite. Nous décrivons des voisinages plus généraux en section 6.3 à la page 37.

Définition 5.1 (Automate cellulaire) *Un automate cellulaire (AC) est un triplet* $A = (Q, r, f)$ *où* Q *est un ensemble fini non-vide* d'états des cellules*,* r *la taille du voisinage et* $f : Q^r \to Q$ *une fonction de transition locale.*

Une configuration est une fonction des entiers vers les états. L'ensemble des configurations est $\mathcal{C}_A = \{c : \mathbb{Z} \to Q\}$*. L'état de la cellule* i *dans* c *est noté* c_i*. Le voisinage de* i *est* $(c_i, c_{i+1}, \ldots, c_{i+r-1}) \in Q^r$ *et est noté* $c_{i+[r]}$*. La* fonction globale *associée à l'automate est* $F_A : \mathcal{C}_A \to \mathcal{C}_A$ *et envoie tout* $c \in \mathcal{C}_A$ *sur une configuration* d *tel que pour tout* $i \in \mathbb{Z}$*,* $d_i = f(c_{i+[r]})$*.*

Pour illustrer cette définition, soit l'automate XOR $= (\{0,1\}, 2, + \pmod 2)$. Un exemple d'exécution est visualisé dans un *diagramme espace-temps* en figure 5.2.
Chaque ligne représente une partie d'une configuration. Les cellules dans l'état 0 sont blanches, celles dans l'état 1 sont colorées. Dans ce schéma, le temps croît vers le bas, c'est-à-dire que le successeur d'une configuration est représenté par la ligne suivante. L'état d'une cellule à un instant est fonction de l'état de cette même cellule et de sa voisine de droite à l'instant d'avant.
Cette définition se généralise pour des automates à n-dimensions, simplement en remplaçant les entiers — indices des cellules — par des vecteurs d'entiers de dimension n. Mais dans

ce qui suit nous allons uniquement travailler avec des automates cellulaires à une dimension.
Un automate cellulaire peut être considéré comme un modèle de calcul, dans le même esprit qu'une machine de Turing. L'entrée de l'automate est une configuration initiale, et la sortie est une configuration finale. Une convention raisonnable consiste à fixer une cellule particulière qui indique si une configuration est finale.
Il s'agit alors d'un modèle de calcul massivement parallèle, car chaque cellule peut représenter une machine à mémoire bornée qui communique avec ses voisins immédiats. Ce qui est atypique dans ce modèle, c'est qu'il existe une infinité de cellules qui *calculent* toutes en même temps. Il serait donc impossible de simuler un tel modèle avec une machine de Turing en général.

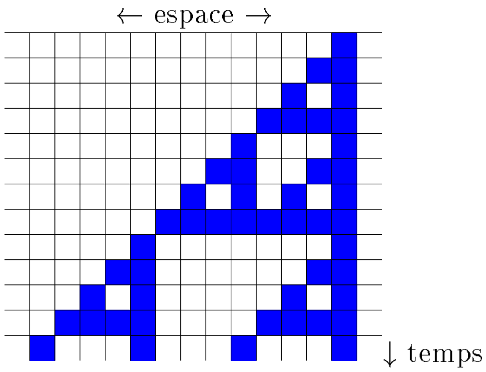


FIG. **5.2:** Un exemple d'automate cellulaire

C'est pourquoi nous sommes plus intéressé par des variantes finies des automates cellulaires. Nous considérons deux modèles : Les automates sur des *configurations finies*, et les automates *périodiques*. Dans le premier modèle nous nous restreignons à un sous-ensemble des configurations telles qu'à tout moment, uniquement un nombre fini de cellules peuvent changer d'état. Dans la deuxième variante les cellules se trouvent sur un support fini. (Voir figure 5.3).

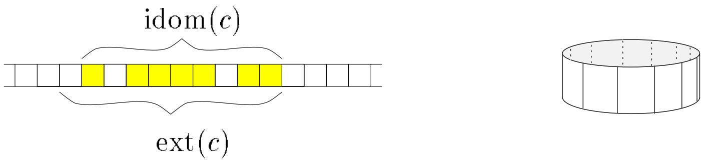


FIG. 5.3: Une configuration finie (c) et une configuration périodique. Dans le premier cas, le voisinage est de taille 2 et l'état quiescent est représenté en blanc.

Définition 5.2 (État quiescent) *Un état $q \in Q$ est* quiescent *pour un automate $A = (Q, r, f)$ si $f(q^r) = q$.*

Définition 5.3 (AC fini) *Un automate cellulaire fini est un couple composé d'un AC $A = (Q, r, f)$ et d'un état quiescent q pour A. L'ensemble des configurations est $\mathcal{C}_A^q = \{c \in \mathcal{C}_A : \mathrm{card}\{i : c_i \neq q\} < \infty\}$. Nous les appelons les configurations finies. La fonction de transition globale est la restriction de F_A sur cet sous-ensemble de configurations. Nous la notons F_A^q.*

Pour toute configuration finie c, il existe un intervalle fini I tel que la partie non-quiescente de c réside en I, c'est-à-dire pour tout $i \notin I$ nous avons $c_i = q$. Soit *l'intervalle de domaine* $\mathrm{idom}(c)$ le plus petit intervalle qui satisfait cette contrainte, à l'exception de la configuration toute quiescente, dont par convention l'intervalle de domaine est l'intervalle $[0, 0]$. Nous définissons l'extension de cet intervalle par $\mathrm{ext}(c) = [j - (r - 1), k]$ si $\mathrm{idom}(c) = [j, k]$, à l'exception de la configuration toute quiescente, dont l'extension de l'intervalle de domaine est aussi $[0, 0]$. Il est clair que pour tout $i \notin \mathrm{ext}(c)$ nous avons $c_{i+[r]} = q^r$. Il s'en suit que si $F_A^q(c) = d$ alors $\mathrm{idom}(d) \subseteq \mathrm{ext}(c)$, ce qui prouve que d est aussi une configuration finie. Donc F_A^q est bien défini.

La notion d'intervalle de domaine et son extension est importante pour cette thèse. Elle est illustrée en figure 5.3 à la page précédente.

Par exemple ($\textsc{Xor}, 0$) est un automate fini, et ($\textsc{Xor}, 1$) en est un autre.

Pour définir les automates cellulaires périodiques, nous allons d'abord introduire la notion de *période* d'une configuration.

Définition 5.4 (Période d'une configuration) *Une configuration c a une période k si pour tout $i \in \mathbb{Z}$ nous avons $c_i = c_{i+k}$. Pour tout automate cellulaire A, une configuration $c \in \mathcal{C}_A$ est* périodique *s'il existe un k, tel que c a une période k.*

Définition 5.5 (AC périodique, support infini) *Un automate cellulaire périodique est un automate cellulaire (tout court) $A = (Q, r, f)$, dont l'ensemble des configurations est l'ensemble des configurations périodiques. Nous notons $\mathcal{C}_A^{P'}$ cet ensemble et $F_A^{P'}$ la fonction de transition globale associée.*

Toute configuration $c : \mathbb{Z} \to Q$ de période k admet une description finie de la forme $c' : \mathbb{Z}_k \to Q$, et l'image de c a aussi la période k. Ceci motive la variante suivante.

Définition 5.6 (AC périodique, support fini) *Un automate cellulaire sur tore est un automate cellulaire (tout court) $A = (Q, r, f)$, dont l'ensemble des configurations est $\mathcal{C}_A^P = \bigcup_{k \geq 1} \{c : \mathbb{Z}_k \to Q\}$. La fonction de transition globale est définie de la manière suivante : Elle préserve la taille des tores, et l'image d'une configuration est — similaire aux autres variantes — la composition des images de la fonction de transition locale. Pour $k \geq 1$ et toute configuration sur tore $c : \mathbb{Z}_k \to Q$, l'image de c par F_A^P est une configuration $d : \mathbb{Z}_k \to Q$ tel que pour tout $i \in \mathbb{Z}_k$ nous ayons $d_i = f(c_{i+[r]})$.*

Pour toute configuration sur tore $c : \mathbb{Z}_k \to Q$ nous notons par $\text{taille}(c)$ la taille k du tore.

La représentation des configurations périodiques infini par des configurations périodiques sur tore n'est pas unique, car une configuration infini de période k a aussi les périodes $2k, 3k$, etc. Cependant le lemme suivant nous assure que cette représentation préserve des propriétés importantes. Aussi à partir de maintenant nous ne considérons plus que les configurations périodiques sur tore.

Lemme 5.1 *Pour tout automate cellulaire A les propriétés suivantes sont équivalentes :*

$$F_A^P \text{ injective, } F_A^P \text{ bijective, } F_A^{P'} \text{ injective, } F_A^{P'} \text{ bijective.}$$

Preuve : Soit pour toute configuration c (de $\mathcal{C}_A^P$ ou de $\mathcal{C}_A^{P'}$) la *période minimale* la plus petite période de c.

Soit S_k l'ensemble des configurations périodiques infinies de période minimale au plus k. Alors S_k est fini, car $\text{card}(S_k) = \text{card}(Q)^k$. L'image d'une configuration de S_k ne peut pas avoir une période minimale supérieure à k, et est donc aussi dans S_k. En conséquence si $F_A^{P'}$ est injective, sa restriction sur S_k l'est aussi. Comme S_k est fini, elle est même bijective. Le choix de k était arbitraire, donc dans ce cas $F_A^{P'}$ est aussi bijective, prouvant $F_A^{P'}$ injectif $\Leftrightarrow$ $F_A^{P'}$ bijectif.

La même argumentation s'applique aussi aux configurations sur des tore en fixant arbitrairement une taille de tore k, prouvant F_A^P injectif $\Leftrightarrow$ F_A^P bijectif.

Soit $c' \in \mathcal{C}_A^{P'}$ une configuration arbitraire, $c \in \mathcal{C}_A^P$ une de ses représentations finies. Par les définitions des opérateurs d'évolution respectifs, $F_A^{P'}$ préserve la période minimale de c' si et seulement si F_A^P préserve celle de c. Ceci prouve F_A^P injectif $\Leftrightarrow$ $F_A^{P'}$ injectif. $\square$

Nous voulons généraliser ces modèles pour le calcul quantique. Nous avons donc besoin d'une caractérisation des automates dont la fonction de transition globale est injective (respectivement bijective). La section suivante regroupe les résultats existants dans le cas classique.

5.3 Caractérisation des automates bijectifs

Amoroso et Patt [AP72] ont donné en 1972 un algorithme polynomial qui décide si la fonction de transition globale d'un automate cellulaire simple à une dimension donnée est injective. Ils pensaient que leur algorithme se généralise pour des automates à plusieurs dimensions et c'est seulement presque 20 ans plus tard qu'il a été montré que ce problème est indécidable pour des automates à deux dimensions ou plus. Ce résultat de Kari [Kar90] est basé sur une réduction d'un problème indécidable de pavage du plan. Il a été ensuite généralisé pour les automates périodiques par Durand [Dur94]. Par contre ce problème reste ouvert pour les automates sur les configurations finies. C'est ce résultat d'indécidabilité qui nous a motivé à nous intéresser seulement aux automates cellulaires à une dimension. Car les automates classiques sont des cas particuliers des automates quantiques, les automates quantiques sont par définition valides seulement si la fonction de transition globale est bijective, et nous ne voulions pas travailler dans un modèle pour lequel il nous serait impossible d'en reconnaître les instances valides.

L'algorithme de Amoroso et Patt a été amélioré par Sutner [Sut91] en utilisant la représentation élégante des automates cellulaires par des graphes de de Bruijn. C'est ce dernier algorithme que nous présentons maintenant.

Automates cellulaires triviaux

Un automate cellulaire $A = (Q, r, f)$ est appelé *trivial* si $r = 1$. Dans ce cas les cellules évoluent de manière indépendante les unes des autres. La fonction de transition locale est alors de la forme $f : Q \to Q$. Si l'état d'une cellule est $x \in Q$ à un moment donné il sera $f(x)$ à l'instant d'après. Clairement F_A est bijective (respectivement injective ou surjective) si et seulement si f l'est aussi. Donc l'injectivité, la surjectivité et la bijectivité de F_A sont des propriétés équivalentes.

A partir de maintenant nous ne considérons que les automates cellulaires non triviaux.

Graphes de de Bruijn dans le cadre des automates cellulaires

Les automates cellulaires ont été décris par des graphes de de Bruijn la première fois dans [Wol84]. C'est cette représentation élégante qui est la base d'un algorithme de Sutner pour décider l'injectivité de la fonction globale d'un automate donné.

Étant donné un automate cellulaire simple $A = (Q, r, f)$ nous lui associons le graphe de de Bruijn étiqueté aux arcs $G_A(V, E, g)$, défini par $V = Q^{r-1}$ et $E = \{(xw, wy) : x, y \in Q, w \in Q^{r-2}\}$. A chaque arc (xw, wy) nous associons le voisinage xwy et l'étiquette $g((xw, wy)) = f(xwy)$. Un exemple de ce graphe est donné en figure 5.4 à la page 31 pour l'automate XOR.

Un *chemin infini dans les deux sens* est une fonction $p : \mathbb{Z} \to V$ tel que $(p(i), p(i+1)) \in E$, pour tout i. Un *cycle* est une fonction $p : \mathbb{Z}_k \to V$ pour un $k \geq 1$ tel que pour tout $i \in \mathbb{Z}_k$, $(p(i), p(i+1)) \in E$.

Nous étendons g sur les chemins et sur les cycles. Soit p un chemin infini dans les deux sens. Alors $g(p)$ est une configuration c tel que $c_i = g(p(i), p(i+1))$ pour tout $i \in \mathbb{Z}$. Soit p un cycle de longueur k, alors $g(p)$ est une configuration périodique c de taille k, avec $c_i = g(p(i), p(i+1))$ pour tout $i \in \mathbb{Z}_k$.

Nous définissons les fonctions h de $\mathcal{C}_A$ vers l'ensemble des chemins infinis dans les deux sens dans G_A et h^P de $\mathcal{C}_A^P$ vers l'ensemble des cycles : Pour toute configuration c, $h(c)$ est le chemin infini dans les deux sens $p : i \mapsto c_{i+[r-1]}$. Pour toute configuration périodique d de taille k, $h^P(d)$ est un cycle $p : i \mapsto d_{i+[r-1]}$.

Alors clairement ces fonctions sont bijectives et satisfont

$$\begin{aligned} \forall c \in \mathcal{C}_A : g(h(c)) &= F_A(c) \\ \forall d \in \mathcal{C}_A^P : g(h^P(d)) &= F_A^P(c). \end{aligned}$$

Soit q un état quiescent. Alors en particulier nous avons aussi

$$\forall c \in \mathcal{C}_A^q : g(h(c)) = F_A^q(c).$$

Un algorithme de décision pour des automates cellulaires injectifs

Décider si la fonction globale d'un automate cellulaire est injective, revient à vérifier qu'il n'existe pas deux configurations différentes qui ont la même image. Ceci revient à vérifier qu'il n'existe pas deux différents chemins infinis dans les deux sens qui ont la même étiquette. Pour décider cela, Sutner [Sut91] construit un graphe fini, tel qu'un chemin infini dans les deux sens corresponde à deux configurations ayant la même image (lemme 5.2). L'injectivité de l'opérateur d'évolution correspond alors à une propriété du graphe (lemme 5.3) qui est facilement vérifiable (théorème 5.1). Ce résultat de Sutner s'applique aussi aux automates cellulaires périodiques et finis.

Soit $H_A = (V', E')$ le graphe produit $G_A \times G_A$ auquel on a enlevé certains arcs, plus précisément $V' = V^2$ et

$$E' = \{((xw, x'w'), (wy, w'y')) : x, y, x', y' \in Q, w, w' \in Q^{r-2} : f(xwy) = f(x'w'y')\}.$$

Les arcs de ce graphe sont en bijection avec les couples de voisinages ayant la même image. Par extension, les chemins infinis dans les deux sens correspondent à des couples de configurations qui ont la même image. Précisons cette observation.

Pour H_A nous considérons les mêmes notions de cycle et de chemin infini dans les deux sens, comme pour G_A, et considérons un troisième type : Un *q-cycle* est un cycle $p : \mathbb{Z}_k \to V'$ tel que $p(0) = (q^{r-1}, q^{r-1})$, et si $k > 1$ alors $p(k-1) \neq p(0) \neq p(1)$.

Pour tout chemin p nous notons p_1 et p_2 les chemins de G_A, tel que les domaines de p, p_1, et p_2 sont les mêmes et que pour tout i de ce domaine, $p(i) = (p_1(i), p_2(i))$.

Lemme 5.2 *Les équivalences suivantes sont vérifiées.*

1. *La fonction F_A est injective si et seulement si pour tout chemin p infini dans les deux sens dans H_A nous avons $p_1 = p_2$.*

2. *La fonction F_A^P est injective si et seulement si pour tout cycle p nous avons $p_1 = p_2$.*

3. *La fonction F_A^q est injective si et seulement si pour tout q-cycle p nous avons $p_1 = p_2$.*

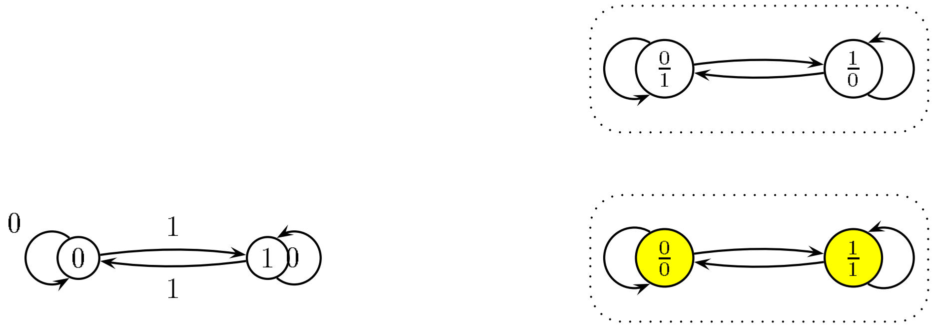


FIG. 5.4: Les graphes G_{Xor} (à gauche) et H_{Xor} (à droite). Les éléments diagonaux sont colorés et les composantes fortement connexes entourées par des traits pointillés.

Preuve : Nous ne prouverons que la dernière partie, les preuves des deux autres étant similaires.

Soit $L = \{(c,d) \in \mathcal{C}_A^q \times \mathcal{C}_A^q : F_A^q(c) = F_A^q(d)\}$ et soit T l'ensemble des q-cycles dans H_A. Nous allons définir une application $M : L \to T$. Pour $(c,d) \in L$ soit $I = [j,k]$ le plus petit intervalle qui contient aussi bien $\mathrm{ext}(c)$ que $\mathrm{ext}(d)$. Soit $t = \mathrm{card}(I)$. Alors par définition $M(c,d) = p$ avec $p : \mathbb{Z}_t \to V'$ et $p : i \mapsto (c_{i+j+[r-1]}, d_{i+j+[r-1]})$. Comme I a été choisi assez grand nous avons $p(0) = (q^{r-1}, q^{r-1})$. Comme I est minimal, si $t > 1$ alors $p(t-1) \neq p(0) \neq p(1)$.

Puisque $F_A^q(c) = F_A^q(d)$ nous avons pour tout i, $f(c_{i+[r]}) = f(d_{i+[r]})$ et donc p est bien un q-cycle dans H_A.

Nous montrons maintenant que M est surjective. Soit $p \in T$ un q-cycle arbitraire et t sa longueur. Soient les configurations c et d, telles que pour tout $i \notin \mathbb{Z}_t$, $c_i = d_i = q$ et pour tout $i \in \mathbb{Z}_t$, c_i est la première lettre de w et d_i la première lettre de w' avec $p(i) = (w, w')$. Alors par construction $M(c,d) = p$.

Finalement $c \neq d$ si et seulement si $M(c,d)_1 \neq M(c,d)_2$, car ces deux conditions sont équivalentes avec l'existence d'un $i \in I$ tel que $c_i \neq d_i$. □

Nous réduisons maintenant les caractérisations énoncées dans le dernier lemme à des propriétés du graphe H_A qui sont facilement vérifiables.

Un sous-ensemble de sommets V_0 d'un graphe est *fortement connexe* si pour tout $u, v \in V_0$ il existe un chemin (éventuellement vide) de u à v. Une *composante fortement connexe* d'un graphe est un sous-ensemble de sommets qui est maximal et fortement connexe. Un sommet de H_A est *diagonal* s'il est de la forme (w, w) pour un $w \in Q^{r-1}$. Une composante fortement connexe est *triviale* si elle n'est composée que d'un unique sommet u et qu'il n'y a pas d'arc de u à u.

Lemme 5.3 *Soient les propriétés suivantes du graphe H_A.*

1. *Pour tout chemin p infini dans les deux sens, $p_1 = p_2$.*

2. *Pour tout cycle p, $p_1 = p_2$.*

3. *Toutes les composantes fortement connexes non-triviales ne contiennent que des sommets diagonaux.*

4. *Pour tout q-cycle p, $p_1 = p_2$.*

5. *La composante fortement connexe qui contient* (q^{r-1}, q^{r-1}) *ne contient que des sommets diagonaux.*

Alors les propriétés 1, 2 et 3 sont équivalentes. Les propriétés 4 et 5 sont aussi équivalentes et la propriété 3 implique 5.

Preuve : Nous commençons par remarquer que pour un chemin (respectivement un cycle) p, $p_1 = p_2$ si et seulement si p ne passe que par des sommets diagonaux. Nous prouvons maintenant les différentes implications du lemme.

(1) $\Rightarrow$ **(2)** Soit p un cycle et k sa longueur. Nous lui associons le chemin p' infini dans les deux sens, défini pour tout $i \in \mathbb{Z}$ par $p'(i) = p(i \bmod k)$. Clairement si $p'_1 = p'_2$ alors $p_1 = p_2$. Le choix de p étant arbitraire, ceci prouve l'implication.

(2) $\Rightarrow$ **(3)** Pour toute composante fortement connexe non-triviale et deux de ses sommets u, v, il existe un cycle qui passe par u et par v. Donc si pour tout cycle p, nous avons $p_1 = p_2$, alors toute composante fortement connexe non-triviale ne contient que des sommets diagonaux.

(3) $\Rightarrow$ **(1)** Supposons que toutes les composantes fortement connexes non-triviales ne contiennent que des sommets diagonaux. Nous remarquons que l'ensemble des sommets diagonaux est fortement connexe. Donc il existe une unique composante fortement connexe non-triviale. Par conséquent tout chemin p infini dans les deux sens ne passe que par des sommets de cette composante, qui sont par hypothèse diagonaux et donc $p_1 = p_2$.

(3) $\Rightarrow$ **(5)** La composante fortement connexe qui contient (q^{r-1}, q^{r-1}) contient aussi tous les sommets diagonaux et n'est donc pas triviale. L'implication suit immédiatement.

(4) $\Rightarrow$ **(5)** Nous montrons cette implication par l'absurde. Supposons que pour tout q-cycle p, $p_1 = p_2$ et supposons que que la composante fortement connexe qui contient (q^{r-1}, q^{r-1}) contient un sommet u qui n'est pas diagonal. Soient v, v' deux sommets diagonaux dans cette composante tels que $v \neq (q^{r-1}, q^{r-1}) \neq v'$, et tels qu'il existe un arc de (q^{r-1}, q^{r-1}) à v et un arc de v' à (q^{r-1}, q^{r-1}). Alors en composant ces arcs avec les chemins de v à u et de u à v' on obtient un q-cycle p, avec $p_1 \neq p_2$, ce qui contredit l'hypothèse.

(5) $\Rightarrow$ **(4)** Soit un q-cycle arbitraire et u, v deux de ses sommets. Alors il existe un chemin de u à v, de v à (q^{r-1}, q^{r-1}) et de (q^{r-1}, q^{r-1}) à u. Par conséquent u, v et (q^{r-1}, q^{r-1}) sont dans une même composante fortement connexe. Le choix du q-cycle, de u et de v était arbitraire, ce qui implique que tous les sommets des q-cycles sont dans la composante fortement connexe qui contient (q^{r-1}, q^{r-1}). Ceci conclut l'implication.

□

Ce lemme implique le corollaire suivant.

Corollaire 5.1 *Pour tout AC A et tout état quiescent q :*

$$\begin{aligned} F_A \text{ injective} &\Leftrightarrow F_A^P \text{ injective} \\ F_A^P \text{ injective} &\Rightarrow F_A^q \text{ injective.} \end{aligned}$$

L'automate XOR illustré en figure 5.4 montre que la dernière implication est stricte.

Avant de donner l'algorithme de décision pour les AC injectifs, nous devons préciser ce que sera la taille de son entrée. La description d'un AC (Q, r, f) est clairement déterminée par la table de transition qui spécifie f. Nous convenons que la taille d'un état de Q est 1. Donc la taille de l'automate cellulaire est $n = \mathrm{card}(Q)^r$.

Théorème 5.1 ([Sut91, thérorème 2.1]) *Il existe un algorithme qui pour un automate cellulaire A décide en temps $O(n^2)$ si F_A est injectif, où n est la taille de A.*

Preuve : Soit l'AC $A = (Q, r, f)$. L'algorithme construit le graphe H_A et calcule les composantes fortement connexes. Ensuite il vérifie la condition 3 donnée en lemme 5.3 à la page 31 par un simple parcours des sommets. Le nombre de sommets de H_A est $\mathrm{card}(V') = \mathrm{card}(Q)^{2(r-1)}$, et le nombre d'arcs est au plus $\mathrm{card}(E') = \mathrm{card}(Q)^{2r}$. Les composantes fortement connexes peuvent être calculées avec l'algorithme de parcours en profondeur de Tarjan [Tar75], en temps $O(\mathrm{card}(E')) = O(n^2)$. La construction de H_A et la vérification des composantes peut être réalisée en temps linéaire en $\mathrm{card}(V') + \mathrm{card}(E')$, ce qui conclut la preuve. □

Il suffit de modifier la vérification des composantes dans l'algorithme précédent pour obtenir le corollaire suivant.

Corollaire 5.2 ([Sut91, théorème 2.2]) *Il existe un algorithme qui pour un automate cellulaire fini (A, q) décide en temps $O(n^2)$ si F_A^q est injectif.*

5.4 Automates cellulaires partitionnés

Nous avons une procédure de décision pour les automates cellulaires bijectifs, mais ceci ne nous permet pas d'en construire directement. Des variantes de ce modèle s'imposent qui seraient bijectives par construction. Une description de différentes variantes est donnée dans la thèse de Margolus [Mar87, section 2.2] et de manière équivalente dans [TM87, chapitre 14] ou encore dans [TM90]. Nous n'en décrivons qu'une seule : *les automates cellulaires partitionnés*. Le terme a été introduit par Margolus, mais le modèle est plus ancien. Il a été inventé indépendamment par Pomeau [Har76], Toffoli [Tof77a, Tof77b], Margolus [Mar84], Iacopini [Jac90] et Kari [Kar96].

Il est important de noter que les AC partitionnés (injectives) sont un cas particulier des AC générales (injectives), mais que l'inverse n'est pas vrai.

Dans ce modèle, chaque cellule se décompose en r sous-cellules. Une transition s'effectue en deux étapes. Lors de la première, les informations contenues dans les sous-cellules sont décalées de manière bijective dans les sous-cellules respectives des cellules voisines. Pendant la deuxième étape chaque cellule est soumise à une bijection sur les états. La première étape est identique pour tous les automates cellulaires partitionnés, ce n'est que la bijection sur les états intervenant durant la deuxième étape, qui détermine le comportement particulier de l'automate.

Ce modèle est un cas particulier des automates cellulaires (tout court) et est bijectif par construction. Formalisons le.

Définition 5.7 (Automate cellulaire partitionné) *Un automate cellulaire partitionné est un triplet $A = (Q, r, f)$ où l'ensemble des états est composé par $Q = Q_1 \times \cdots \times Q_r$ et $f : Q \to Q$*

est une fonction. L'automate cellulaire induit est $A' = (Q, r, f')$ avec $f' : Q^r \to Q$, définie pour tout $x_1, \ldots, x_r \in Q^r$ par

$$f'(x_1, \ldots, x_r) = f(x_{1,1}, \ldots, x_{r,r}),$$

où pour tout $1 \leq i \leq r$, $x_{i,i} \in Q_i$ est la i-ème composante de x_i. L'ensemble des configurations et la fonction de transition globale de A sont identiques avec ceux de A'.

Lemme 5.4 *La fonction globale de l'automate cellulaire partitionné $A = (Q, r, f)$ est bijective si et seulement si f est une bijection. De plus pour la fonction globale, injectivité, surjectivité et bijectivité sont des propriétés équivalentes.*

Preuve : Soient les automates cellulaires $A' = (Q, r, f')$ et $A'' = (Q, 1, f)$ défini par

$$f'(x_1, \ldots, x_r) = (x_{1,1}, \ldots, x_{r,r}).$$

Alors clairement pour toute configuration c, nous avons $F_A(c) = F_{A''}(F_{A'}(c))$. Comme $F_{A'}$ est une bijection par construction, le lemme suit des observations concernant les automates cellulaires triviaux en section 5.3 à la page 29. □

Exemple d'automate cellulaire partitionné

Soit $(\{0,1\}^2, 2, f)$ un automate cellulaire partitionné avec f définie pour tout $(x, y) \in \{0,1\}^2$ par $f((x, y)) = (x, x + y \mod 2)$.

Considérons la configuration suivante, où nous avons représenté 0 et 1 respectivement par □ et ■.

...	□	□	□	■	■	□	■	■	■	□	□	...
	□	□	■	□	■	□	□	■	□	□	□	

Dans la première étape d'une transition la deuxième partie de chaque cellule est décalée à gauche.

...	□	□	□	■	■	□	■	■	■	□	□	...
←	□	■	□	■	□	□	■	□	□	□	□	←

Ensuite la bijection f est appliquée à chaque cellule.

...	□	□	□	■	■	□	■	■	■	□	□	...
	□	■	□	□	■	□	□	■	■	□	□	

6

Automates cellulaires quantiques

Le même terme d'*automate cellulaire quantique* (ACQ) a été utilisé pour désigner des modèles très différents. Pour mettre bien au clair ce dont il est question, nous allons brièvement les exposer :

- Porod, Lent et Tougaw [LTPB93, LTP94] désignent par automate cellulaire quantique un circuit classique composé de portes AND, OR et NOT. Les portes ainsi que les fils les reliant sont entièrement réalisés par des "*quantum dots*", qui sont des cellules pouvant être en superposition entre 0 et 1, et dont l'état influe sur l'état des voisins immédiats et est influencé par eux. Leur motivation est de créer une technologie qui permettrait de continuer à réaliser des circuits classiques à un taille tellement réduite qu'il faut tenir compte des lois de la mécanique quantique. Cependant leur intérêt n'est pas de tirer profit de ces lois, pour réaliser un modèle de calcul quantique : leurs circuits ne sont pas réversibles, puisque par exemple la porte AND a un degré d'entrée 2 et un degré de sortie 1.

- Margolus et Biafore [Mar94, Bia94] définissent des automates cellulaires avec le formalisme des Hamiltoniens. En gros il s'agit d'un modèle à temps continu, alors que dans le nôtre le temps est discret.

- Pour Watrous [Wat95], Dürr, LêThanh, Santha [DLS96, DS96] et van Dam [Dam96] les automates cellulaires quantiques sont une généralisation directe des automates cellulaires classiques basée sur le formalisme des opérateurs d'évolution.

- Un modèle intermédiaire entre les deux derniers est l'automate cellulaire quantique partitionné [Wat95, Llo93, Llo94]. Nous en discuterons en section 6.5. Le modèle décrit par Lloyd est basé sur le travail important de Mahler et al. [Obe88, parties 1 et 2] qui traite l'aspect physique des ACQ.

C'est le troisième modèle qui est le sujet de ce chapitre.

6.1 Définitions

L'ensemble des configurations d'un automate cellulaire n'est pas dénombrable. Par conséquent une généralisation quantique de ce modèle serait basée sur la théorie de mesure. Pour éviter ce formalisme seules les variantes finies des automates cellulaires ont été généralisées au calcul quantique. Nous discuterons en conclusion nos intuitions sur le modèle général (infini).

De manière similaire au cas classique, les variantes quantiques ont en commun la définition d'un ensemble d'états, d'un vecteur de voisinage, d'une fonction de transition locale et diffèrent uniquement dans le choix des configurations auxquelles elles s'appliquent. Nous regroupons la partie commune dans le triplet suivant.

Définition 6.1 (Triplet ACQ) $\textsc{Acq}$ *est l'ensemble des triplets* (Q, r, f) *où* Q *est un ensemble fini, non-vide d'états,* r *est la taille du voisinage, et* $f : Q^r \to \mathbb{C}^Q$ *est une fonction de transition locale qui satisfait pour tout* $w \in Q^r : \|f(w)\| > 0$.

Les deux modèles suivants sont une généralisation directe de variantes des automates cellulaires classiques. Les ACQ finis ont étés introduits de manière indépendante par Watrous [Wat95] et Dürr, LêThanh et Santha [DLS96]. Les ACQ périodiques ont été définis par van Dam [Dam96].

Généralisation 6.2 (ACQ périodique) *Un automate cellulaire quantique périodique est un triplet* $A = (Q, r, f) \in \textsc{Acq}$. *L'ensemble des configurations est* $\mathcal{C}_A^P$ *et l'opérateur d'évolution associé* U_A^P *est défini pour tout* $c, d \in \mathcal{C}_A^P$ *de manière naturelle par*

$$U_A^P(d, c) = \begin{cases} \prod_{i\in\mathbb{Z}_k} \langle f(c_{i+[r]})|d_i\rangle & si\ \exists k : k = \text{taille}(c) = \text{taille}(d), \\ 0 & sinon. \end{cases}$$

Rappelons nous que $c_{i+[r]}$ est une notation pour $c_i c_{i+1} \dots c_{i+r-1}$.

Généralisation 6.3 (État quiescent) *Un état* $q \in Q$ *est quiescent pour* $(Q, r, f) \in\textsc{Acq}$ *si* $f(q^r) = |q\rangle$.

Généralisation 6.4 (ACQ fini) *Un automate cellulaire quantique fini est un couple* (A, q) *avec* $A = (Q, r, f) \in \textsc{Acq}$ *et* q *quiescent pour* A. *L'ensemble des configurations est* $\mathcal{C}_A^q$ *et l'opérateur d'évolution associé* U_A^q *est défini pour tout* $c, d \in \mathcal{C}_A^q$ *de manière naturelle par*

$$U_A^q(d, c) = \prod_{i\in\mathbb{Z}} \langle f(c_{i+[r]})|d_i\rangle.$$

Nous rappelons que les instances des modèles ci-haut sont par définition *valides* (ou *unitaires*) si leur opérateurs d'évolution sont unitaires. Il nous faut donc caractériser les instances valides et c'est le sujet du chapitre suivant.

6.2 Exemple d'un automate cellulaire quantique fini

Soit QFlip= $(\{0, 1\}, 2, f) \in \textsc{Acq}$ avec l'état quiescent 0 et f définie par

$$f(x, y) = \begin{cases} |x\rangle & \text{si } y = 0, \\ U|x\rangle & \text{sinon,} \end{cases}$$

où U est l'opérateur de tirage quantique

$$U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Le graphe de la figure 6.1 illustre l'exécution de cet automate. Chaque ligne décrit l'état de l'automate à une étape donnée. Les arcs indiquent les transitions à amplitude non-nulle et sont étiquetés par l'amplitude associée. Notons $\ldots x_0 x_1 x_2 x_3 x_4 \ldots$ la configuration $c \in \mathcal{C}^0_{\mathrm{Qflip}}$ tel que pour tout $i \in [0,4]$, $c_i = x_i$ et $c_i = 0$ ailleurs. Nous partons de la superposition $|\ldots 00010 \ldots\rangle$, obtenons à l'étape d'après $(|\ldots 00010 \ldots\rangle + |\ldots 00110 \ldots\rangle)/\sqrt{2}$, et ainsi de suite. La figure comporte les probabilités (arrondies) d'observation des configurations au bout de 2 étapes. Cet exemple montre alors une interférence constructive vers la configuration $\ldots 00010 \ldots$ et une interférence destructive vers $\ldots 00110 \ldots$.

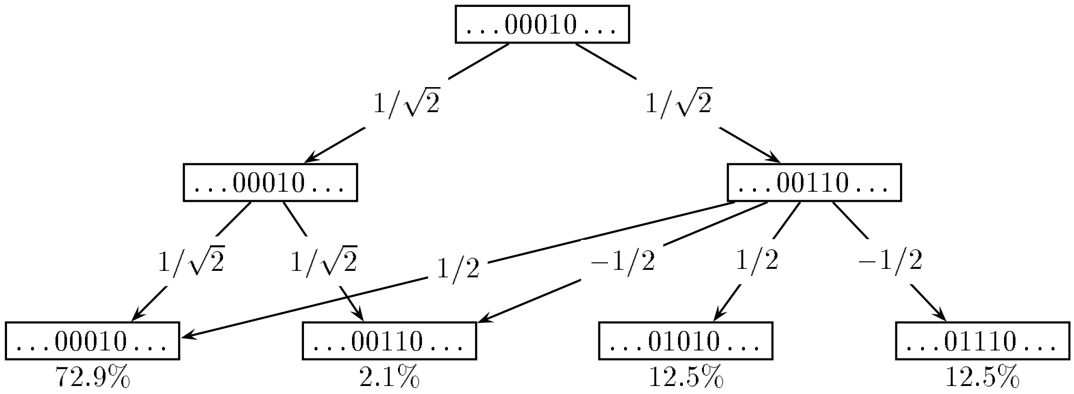


FIG. 6.1: Exemple d'une exécution de l'ACQ fini (QFlip, 0). (La somme des probabilités d'observation donne bien sûr 100%. Mais ici les valeurs sont arrondies.) Nous ne pouvons pas comme dans le cas classique représenter l'évolution d'un automate par un diagramme espace-temps, mais devons utiliser cet arbre d'évolution, car à un moment donné l'automate est en superposition entre plusieurs configurations. Ça ne veut pas dire qu'une cellule est en superposition entre différents états, car les cellules peuvent être enchevêtrées, comme c'est le cas au temps 2 dans cet exemple.

6.3 Autres voisinages

Certains auteurs [Wol94] restreignent dans leur définition des automates cellulaires le voisinage à $(-1, 0, 1)$, c'est-à-dire les voisins d'une cellule sont les cellules immédiatement à gauche et à droite et elle même. Le type de voisinage le plus courant [Mar87, FAQ] est un voisinage continue centré autour de la cellule concernée. Cependant il existe une définition plus générale dans [Dur94, Wat95] : Le voisinage est déterminé par une suite finie monotone croissante d'entiers $N = (a_1, \ldots, a_r)$ telle que le voisinage de la cellule i soit composé des cellules $i + a_1, \ldots, i + a_r$. Appelons ce type de voisinage *éparpillé* et celui que nous avons utilisé dans ce document *simple*.

Nous présentons dans cette thèse des algorithmes qui vérifient pour un automate cellulaire (quantique) simple donné si la fonction globale est injective (unitaire). Comment pouvons nous changer ces procédures de décisions pour pouvoir les appliquer à des automates cellulaires à voisinage éparpillé ?

Il suffit de transformer tout automate à voisinage éparpillé en un automate à voisinage simple, dont la fonction de transition globale (respectivement l'opérateur d'évolution) satisfait les mêmes propriétés que l'algorithme est censé de vérifier, et d'exécuter la procédure de décision sur ce nouvel automate. Cette transformation a pour effet d'agrandir la taille de l'automate et donc la complexité de l'algorithme. Nous allons maintenant chiffrer cette augmentation.

Soit $A = (Q, N, f)$ un automate cellulaire classique au voisinage éparpillé $N = (a_1, a_2, \dots, a_r)$. Nous transformons A en un automate à voisinage simple en deux étapes. D'abord nous rendons le voisinage continu, puis nous le décalons.

Soit $A' = (Q, N', f')$ l'automate défini par le voisinage continue $N' = (a_1, a_1 + 1, \dots, a_r)$ et f', une extension de f qui est insensible aux nouveaux paramètres. C'est-à-dire pour tout $w \in Q^{[a_1,a_r]}$ nous avons $f'(w) = f(w_{a_1}, w_{a_2}, \dots, w_{a_r})$. Clairement les ensembles de configurations des automates A et A' sont identiques ainsi que leurs fonctions de transition globales.

Soit maintenant $A'' = (Q, s, f')$ l'automate à voisinage simple défini par $s = \text{card}[a_1, a_r]$ et la bijection σ sur $\mathcal{C}_A$ suivante. L'image de tout $c \in \mathcal{C}_A$ par σ est une configuration d tel que pour tout $i \in \mathbb{Z}$ nous ayons $d_i = c_{i+a_1}$. Alors pour toute configuration $c \in \mathcal{C}_A$, $\sigma(F_A(c)) = F_{A''}(\sigma(c))$. Donc F_A est injective si et seulement si $F_{A''}$ l'est aussi.

Cette transformation s'applique également aux automates cellulaires quantiques finis. Les définitions de A' et de A'' pour un $A \in$ACQ sont les mêmes. Pour une superposition u sur $\mathcal{C}_A^q$ nous notons $\sigma(u) = \sum_c u_c|\sigma(c)\rangle$. Alors pour toute configuration finie c, $\sigma(U_A|c\rangle) = U_{A''}|\sigma(c)\rangle$, en d'autres termes pour tout $c, d \in \mathcal{C}_A^q$, $U_A(\sigma(d), c) = U_{A''}(d, \sigma(c))$. Donc U_A est unitaire si et seulement si $U_{A''}$ l'est aussi.

Nous allons maintenant analyser la complexité des algorithmes de décisions sur les automates à voisinage éparpillé.

Automates cellulaires classiques

Pour le voisinage éparpillé $N = (a_1, \dots, a_r)$, soit $s = \text{card}[a_1, a_r]$ le *couvrement* de N. Nous avions convenu en section 5.3 que la taille d'un état est 1 et que la taille de A est $n = \text{card}(Q)^r$. Alors la taille de A'' est $n'' = \text{card}(Q)^s$. Nous définissons le *facteur d'expansion* de A par $e = s/r$. Donc tout algorithme qui prendra un temps $O(n^c)$ sur les automates à voisinages simples, prendra un temps $O(n^{ce})$ sur un automate à voisinage éparpillé, où e est son facteur d'expansion, car $n'' = n^e$.

Automates cellulaires quantique finis

Nous travaillons dans un modèle algébrique où l'espace occupé (sur le ruban d'une machine de Turing) par un nombre complexe est constant. À la différence des automates cellulaires classiques, la taille de la description d'un automate cellulaire quantique fini (A, q) à voisinage éparpillé $N = (a_1, \dots, a_r)$ est $\text{card}(Q)^{r+1}$. Nous définissons donc le facteur d'expansion de A par $e = (s + 1)/(r + 1)$. Ainsi un algorithme dont la complexité serait $O(n^c)$ pour des automates quantiques finis à voisinage simple s'exécute sur A en temps $O(n^{ce})$.

6.4 Automates cellulaires quantiques finis triviaux

Un automate cellulaire quantique fini (A, q) avec $A = (Q, r, f)$ est appelé *trivial* si $r = 1$. Dans ce cas les cellules évoluent de manière indépendante les unes des autres. L'état global de l'automate sera alors toujours un produit tensoriel sur les superpositions d'états de chaque cellule, si les transitions commencent par des configurations, plus précisément par des superpositions associant amplitude 1 à une configuration et 0 ailleurs. Alors par le fait 4.2,

U_A est isométrique si et seulement f l'est aussi. Dans ce cas f est aussi unitaire, et $f^\dagger$ définit l'automate $A' = (Q, 1, f^\dagger)$ tel que $U^q_{A'}$ inverse U^q_A et vice-versa. Alors U^q_A est unitaire si et seulement si U^q_A est isométrique.

Cette caractérisation ne s'applique pas aux automates non-triviaux, car un opérateur de $\mathbb{C}^{Q^r}$ dans $\mathbb{C}^Q$ ne peut être isométrique, si $r > 1$. Aussi à partir de maintenant nous supposons que les automates ne sont pas triviaux.

6.5 Automates cellulaires quantiques partitionnés

Les automates cellulaires quantiques partitionnés sont une généralisation des automates cellulaires classiques partitionnés. Ils ont été introduits par Watrous [Wat95].

Définition 6.5 (ACQ partitionné) *Un automate cellulaire quantique partitionné est un quadruplet $A = (Q, r, f, q)$, où l'ensemble des états est une composition de $Q = Q_1 \times \cdots \times Q_r$, pour des ensembles finis $Q_1, \ldots, Q_r$, r est la taille du voisinage, f est une application linéaire sur $\ell_2(Q)$, et q est un état de Q qui satisfait $f|q\rangle = |q\rangle$. Le triplet* Acq *induit est $A' = (Q, r, f')$ avec f' définie pour tout $x_1, \ldots, x_r \in Q^r$ par*

$$f'(x_1, \ldots, x_r) = f|x_{1,1}, \ldots, x_{r,r}\rangle,$$

où $x_{i,i} \in Q_i$ est la i-ème composante de x_i. L'ensemble des configurations et la fonction de transition globale de A sont par définition ceux de l'ACQ fini (A', q).

L'intérêt principal de ce modèle est que ces instances sont valides par construction, comme l'établit le lemme suivant.

Lemme 6.1 ([Wat95, théorème 1]) *La fonction globale d'un automate cellulaire partitionné $A = (Q, r, f, q)$ est unitaire si et seulement si f est unitaire.*

Preuve : Soient les triplets Acq $A' = (Q, r, f')$ et $A'' = (Q, 1, f)$ défini par

$$f'(x_1, \ldots, x_r) = |x_{1,1}, \ldots, x_{r,r}\rangle.$$

Alors clairement pour toute configuration c, nous avons $F_A(c)^q = F^q_{A''}(F^q_{A'}(c))$. Comme $F_{A'}$ est une unitaire par construction, le lemme suit des observations concernant les automates cellulaires quantiques triviaux en section 6.4. $\square$

7

Un algorithme de décision pour des automates cellulaires quantiques finis unitaires

Dans ce chapitre nous donnons explicitement un algorithme qui décide si un ACQ fini donné est valide. C'est-à-dire il décide si l'opérateur d'évolution d'un automate cellulaire décrit par une fonction locale donnée est unitaire.

Ce problème est la version quantique d'un problème classique connu qui a été étudié sous différentes formes pendant longtemps, voir p.ex. [AP72, Sut91]. Nous décidons donc si une instance d'un modèle de calcul quantique particulier satisfait une contrainte imposée par la mécanique quantique. Il s'agit d'un modèle théorique général et non d'un modèle physique concret, comme par exemple l'automate décrit par Lloyd [Llo93, Llo94].

Lors d'une transition d'une machine de Turing quantique (Q, Σ, δ) il n'y a qu'une cellule du ruban qui peut changer de contenu en une transition. L'opérateur d'évolution U est en étroite relation avec la fonction de transition δ, car toute entrée de U est par définition soit 0, soit $\delta(q, x, y, p, d)$ pour des valeurs $(q, x, y, p, d) \in Q \times \Sigma \times \Sigma \times Q \times \{\triangleleft, \perp, \triangleright\}$.

Cette propriété a permis à Bernstein et Vazirani de donner des conditions sur la fonction de transition locale qui sont satisfaites si et seulement si l'opérateur d'évolution est unitaire.

La situation est bien différente pour les automates cellulaires quantiques finis, car en une transition il y a un nombre non-borné de cellules qui peuvent changer d'état. Les entrées de l'opérateur d'évolution sont définies par un produit des amplitudes de transitions locales. De plus les vecteurs lignes de l'opérateur d'évolution peuvent avoir un nombre infini d'entrées non-nulles. C'est pour cette raison qu'une caractérisation locale des ACQ finis valides n'a pas pu être établie, comme pour les MTQ. Par contre il nous a été possible de fournir un algorithme de décision pour des ACQ finis dont l'opérateur d'évolution est unitaire. C'est le sujet de ce chapitre.

7.1 Reformulation des amplitudes de transition

Si nous exécutons une étape d'un automate quantique fini sur une configuration — plus précisément sur une superposition associant amplitude 1 à une configuration particulière et 0 ailleurs — alors nous obtenons une superposition de configurations où chaque cellule a fait une transition à amplitude indépendante. En d'autres termes, la superposition résultante peut être décrite par un produit tensoriel : Pour tout ACQ fini (A, q) avec $A = (Q, r, f)$, pour toute configuration $c \in \mathcal{C}_A^q$, pour tout intervalle fini I avec $I \supseteq \mathrm{ext}(c)$ et pour toute configuration $d \in \mathcal{C}_A^q$ avec $\mathrm{idom}(d) \subseteq I$ nous avons

$$\begin{aligned} U_A^q(d,c) &= \prod_{i\in\mathbb{Z}} \langle f(c_{i+[r]}) | d_i \rangle \\ &= \prod_{i\in I} \langle f(c_{i+[r]}) | d_i \rangle \\ &= \left\langle \bigotimes_{i\in I} f(c_{i+[r]}) \middle| d_I \right\rangle, \end{aligned}$$

où d_I est la restriction de d sur I.

Donc pour toute configuration $d' \in \mathcal{C}_A^q$ nous avons

$$U_A^q(d', c) = \begin{cases} \left\langle \bigotimes_{i\in I} f(c_{i+[r]}) \middle| d'_I \right\rangle & \text{si } \mathrm{idom}(d') \subseteq I, \\ 0 & \text{sinon.} \end{cases}$$

Cette reformulation de l'opérateur d'évolution est la base du lemme suivant :

Lemme 7.1 *Pour tout ACQ fini (A, q) avec $A = (Q, r, f)$, pour toutes configurations $c, c' \in \mathcal{C}_A^q$ et pour tout intervalle fini I avec $I \supseteq \mathrm{ext}(c)$ et $I \supseteq \mathrm{ext}(c')$ nous avons*

$$\langle U_A^q(\cdot, c) | U_A^q(\cdot, c') \rangle = \prod_{i\in I} \langle f(c_{i+[r]}) | f(c'_{i+[r]}) \rangle.$$

Preuve :

$$\begin{aligned} \langle U_A^q(\cdot, c) | U_A^q(\cdot, c') \rangle &= \sum_{d\in\mathcal{C}_A^q} U_A^q(d,c) \cdot \overline{U_A^q(d,c')} && (7.1) \\ &= \sum_{d\in\mathcal{C}_A^q, \mathrm{idom}(d)\subseteq I} \left\langle \bigotimes_{i\in I} f(c_{i+[r]}) \middle| d_I \right\rangle \cdot \overline{\left\langle \bigotimes_{i\in I} f(c'_{i+[r]}) \middle| d_I \right\rangle} && (7.2) \\ &= \sum_{d'\in Q^I} \left\langle \bigotimes_{i\in I} f(c_{i+[r]}) \middle| d' \right\rangle \cdot \overline{\left\langle \bigotimes_{i\in I} f(c'_{i+[r]}) \middle| d' \right\rangle} && (7.3) \\ &= \left\langle \bigotimes_{i\in I} f(c_{i+[r]}) \middle| \bigotimes_{i\in I} f(c'_{i+[r]}) \right\rangle && (7.4) \\ &= \prod_{i\in I} \langle f(c_{i+[r]}), f(c'_{i+[r]}) \rangle && (7.5) \end{aligned}$$

Ces égalités sont justifiées de la manière suivante :

(7.1) par définition du produit interne,

(7.2) par le choix de I et la reformulation de l'opérateur U_A^q,

(7.3) par identification de d_I avec d',

(7.4) par définition du produit tensoriel,

(7.5) par le lemme 4.2 à la page 24.

□

Le corollaire suivant est une conséquence immédiate de ce lemme.

Corollaire 7.1 *Soit (A, q) un automate cellulaire quantique fini, c une configuration et I un intervalle fini assez grand pour contenir $ext(c)$. Alors nous avons*

$$\|U_A^q(\cdot, c)\| = \prod_{i \in I} \|f(c_{i+[r]})\|.$$

Nous ne connaissons pas de conditions nécessaires et suffisantes pour l'unitarité de l'opérateur d'évolution d'un ACQ fini, même si l'automate est classique. Néanmoins il existe un algorithme de décision pour l'unitarité de l'opérateur d'évolution que nous présentons maintenant.

7.2 L'algorithme

D'après les faits 3.1 et 3.3 à la page 14, pour décider si l'opérateur d'évolution est unitaire il suffit de vérifier que les vecteurs colonnes de l'opérateur sont de norme 1, et sont mutuellement orthogonaux et que les vecteurs lignes sont de norme 1. Chacune des sections suivantes est dédiée à un de ces trois sous-problèmes. Notre théorème principal est alors une conséquence directe du théorème 7.2 à la page 45, et des théorèmes 7.4 et 7.5 à la page 52.

Théorème 7.1 *Il existe un algorithme R qui décide si l'opérateur d'évolution d'un ACQ $((Q, r, f), q)$ donné est unitaire. La complexité de l'algorithme est $O(n^{\frac{3r-1}{r+1}})$.*

7.3 Exemples

Pour illustrer les algorithmes pour chacun des sous-problèmes nous définissons les ACQ finis suivants.[1]

Soit l'ACQ fini $B = ((\{a, b\}, 3, f), a)$ avec f défini pour tout $x, y, z \in \{a, b\}$ par

$$f(x, y, z) \begin{cases} \frac{1}{2}|y\rangle & \text{si } x = a, z = b, \\ 2|y\rangle & \text{si } x = b, z = a, \\ |y\rangle & \text{sinon.} \end{cases}$$

Soit l'ACQ fini QFLIP$= ((\{a, b\}, 2, f), 0)$ défini pour tout $x, y \in \{a, b\}$ par

$$f(x, y) = \begin{cases} U|x\rangle & \text{si } y = 1, \\ |x\rangle & \text{sinon,} \end{cases}$$

où U est l'opérateur de tirage quantique

$$U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

[1] Un même exemple pour tous les algorithmes aurait été soit trop simple pour le premier sous-problème, soit trop compliqué pour le deuxième.

7.4 Norme des vecteurs colonnes

7.4.1 Notre algorithme

Dans cette section nous allons donner un algorithme qui décide si tous les vecteurs colonnes de l'opérateur d'évolution sont de norme 1. Soit (A, q) (avec $A = (Q, r, f)$) un ACQ fini. Nous définissons une version pondérée du graphe de de Bruijn orienté de la section 5.3 à la page 29. Soit $G_A = (V, E, g)$ avec l'ensemble des sommets $V = Q^{r-1}$, l'ensemble des arcs $E = \{(xw, wy) : x, y \in Q, w \in Q^{r-2}\}$ et la fonction de poids $g : E \to \mathbb{R}_+$ définie par $g((xw, wy)) = \|f(xwy)\|$. Ce graphe est illustré en figure 7.1.

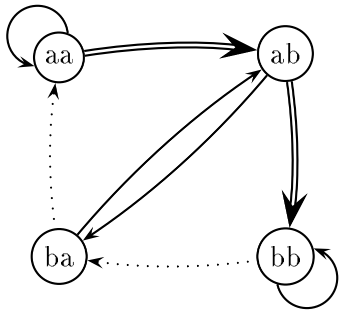


FIG. **7.1:** Le graphe G_B. Les arcs de poids 2 sont en double-ligne, et ceux de poids 1/2 en pointillé.

Un *chemin* est une séquence non-vide $p = (v_0, \dots, v_k)$ de sommets tel que pour tout $0 \leq i \leq k-1$, nous avons $(v_i, v_{i+1}) \in E$. Le *poids* $g(p)$ d'un chemin p est

$$\prod_{i=0}^{k-1} g((v_i, v_{i+1})).$$

Nous appelons le chemin $(v_0, \dots, v_k)$ un *cycle* si $v_0 = v_k$ et $k > 0$. Si de plus $v_0 = q^{r-1}$ alors nous l'appelons un q-cycle. Notre algorithme est basé sur le lemme suivant.

Lemme 7.2 *Les vecteurs colonnes de U_A^q sont de norme* 1 *si et seulement si le poids de tous les q-cycles dans G_A est* 1.

Preuve : Soit T l'ensemble des q-cycles de G_A. Nous définissons une application $M : C_A \to T$. Soit c une configuration avec le domaine d'intervalle $I = [j, k]$. Alors par définition

$$M(c) = (q^{r-1}, q^{r-2}c_j, q^{r-3}c_jc_{j+1}, \dots, c_{i+[r-1]}, \dots, c_kq^{r-2}, q^{r-1}).$$

Nous obtenons alors

$$\|U_A^q(\cdot, c)\| = \|f(q^{r-1}c_j)\| \cdot \|f(q^{r-2}c_jc_{j+1})\| \cdot \dots \cdot \|f(c_kq^{r-1})\| \tag{7.6}$$

$$= \|f(q^r)\| \cdot \|f(q^{r-1}c_j)\| \cdot \|f(q^{r-2}c_jc_{j+1})\| \cdot \dots \cdot \|f(c_kq^{r-1})\| \cdot \|f(q^r)\| \tag{7.7}$$

$$= g(M(c)). \tag{7.8}$$

Les équations sont justifiées comme suit :

(7.6) par le corollaire 7.1,

(7.7) par la définition d'un état quiescent,

(7.8) par définition de M et du poids d'un chemin.

Le lemme suit du fait que l'application M est surjective. □

Vérifier que tous les vecteur colonnes de U_A^q sont de norme 1 revient alors à vérifier que tous les q-cycles de G_A ont un poids 1. C'est précisément ce qui est fait par l'algorithme que nous décrivons maintenant.

Théorème 7.2 ([DLS96]) *Il existe un algorithme R_1 qui prend en entrée un ACQ fini (A, q) et décide si les vecteurs colonnes de l'opérateur d'évolution U_A^q sont tous de norme 1. La complexité de l'algorithme est $O(n^{\frac{2r-1}{r+1}})$.*

Preuve : L'algorithme R_1 construit le graphe G_A puis détermine s'il a un q-cycle de poids différent de 1. Ceci va être réalisé par deux algorithmes R_{11} et R_{12}, tel que le premier vérifie s'il existe un cycle de poids (une colonne de norme) strictement inférieure à 1 et le deuxième vérifie s'il existe un cycle de poids (une colonne de norme) strictement supérieure à 1. Il s'agit dans les deux cas de modifications de l'algorithme Bellman-Ford (BF) de recherche du plus court chemin à source unique [Bel58, FF62, voir aussi [CLR90]] en fixant la source des chemins à q^{r-1}. Nous utilisons le fait que BF détecte les cycles négatifs passant par la source.[2]

Algorithme R_{11}

> **entrée** : Le graphe $G_A = (V, E, g)$ et un sommet v
> **sortie** : décider s'il existe un cycle de poids inférieur à 1 autour de v
>
> Compteur : $V \to \mathbb{N}$, initialisé à 0
> Pile : pile de sommets, initialisée à (v).
> distance : $V \to \mathbb{R}_+^\infty$ initialisée à 1 pour v et ∞ ailleurs
> **Tant que** Pile n'est pas vide
> $u :=$ sommet de Pile, dépiler Pile
>   Compteur$[u]$:= Compteur$[u] + 1$
>   **Si** Compteur$[u] >$ card(V)
>   arrêter et retourner "Non"
>   **Pour toute** arête e sortant de u
> $d :=$ min (distance$[u]\cdot g(e)$, distance$[v]$)
>   **Si** $d \neq$ distance$[v]$
>   distance$[v] := d$
>   **Si** v n'est pas dans Pile
>   empiler v dans Pile
> retourner "Oui"

Algorithme R_{12}

> **entrée** : Le graphe $G_A = (V, E, g)$ et un sommet v
> **sortie** : décider s'il existe un cycle de poids supérieur à 1 autour de v
>
> Même algorithme que R_{11}, avec comme seules différences max à la place de min, et 'distance' initialisée à 1 pour v et 0 ailleurs.

L'algorithme R_{11} est obtenu à partir de BF en remplaçant toute opération de somme de BF par l'opération produit et en initialisant l'estimation du plus court chemin pour la source à 1 (au lieu de 0 comme dans BF) et infini pour les autres sommets (comme dans BF). De cette manière R_{11} calcule le plus court chemin quand le poids d'un chemin est défini comme

---

[2] En fait, pour nos besoins nous aurions pu utiliser n'importe quel algorithme de recherche du plus court chemin, qui utilise seulement la somme et le minimum comme opérations arithmétiques et qui détecte les cycles négatifs. Par exemple l'algorithme de Floyd est un candidat, mais pas celui de Dijkstra.

le produit du poids de ses arcs. Pour s'en convaincre, soit G'_A le même graphe que G_A à la différence que le poids des arcs est remplacé par leur logarithme. Alors le poids du plus court chemin dans G'_A donné par BF va être le logarithme du poids du plus court chemin de G_A donné par R_{11}.

Pour les mêmes raisons, les cycles négatifs passant par la source dans G'_A correspondent à des q-cycles dans G_A avec un poids strictement inférieur à 1, et qui seront donc détectés par R_{11}.

L'algorithme R_{12} remplace toute opération min dans R_{11} par max, l'estimation initiale du plus court chemin par 0, et l'exécute sur G_A. De cette manière il calcule le plus court chemin, quand le poids d'un chemin est défini comme le produit de l'inverse du poids des arcs. Si nous définissons G'_A avec comme poids le *négatif* du logarithme alors les cycles négatifs dans G'_A correspondent au cycles dans G_A avec un poids strictement supérieur à 1, qui seront alors détectés par R_{12}.

La complexité de BF est $O(\text{card}(V) \cdot \text{card}(E))$. En G_A nous avons $V = Q^{r-1}$. Tout sommet a un degré de sortie $\text{card}(Q)$ et donc $\text{card}(E) = \text{card}(Q)^r$. Ceci établit la complexité. □

7.4.2 Algorithme de Høyer

Høyer [Høy96] a proposé un algorithme optimal pour décider si les vecteurs colonnes de l'opérateur d'évolution sont tous de norme 1. Son amélioration repose sur le lemme 7.3 qui établit qu'il suffit pour cela de vérifier pour un petit ensemble D' de configurations que pour tout $c \in D'$, le vecteur colonne dans U_A^q associé à c soit de norme 1. Pour cela nous introduisons un graphe pondéré et un ensemble P de chemins et montrons (lemme 7.4) qu'il existe une bijection $M : D' \to P$ telle que pour tout $c \in D'$, le poids de $M(c)$ est $\|U_A^q(\cdot, c)\|^2$. Vérifier que les vecteurs colonnes dans U_A^q associés au configurations de D' sont tous de norme 1, revient alors à vérifier que tous les chemins de P ont le poids 1. La structure particulière du graphe permet de vérifier cela en temps linéaire en n (lemme 7.5).

Lemme 7.3 ([Høy96]) *Soit $((Q, r, f), q)$ un ACQ fini. Si pour toute configuration c, avec $\text{card}(idom(c)) \leq r$, nous avons $\|U_A^q(\cdot, c)\| = 1$, alors tous les vecteurs lignes de l'opérateur d'évolution U_A^q sont de norme 1.*

Preuve : Soit l'ensemble des configurations $D = \{c \in \mathcal{C}_A^q : \|U_A^q(\cdot, c)\| \neq 1\}$. Si $D = \emptyset$, alors l'implication suit. Supposons $D \neq \emptyset$.

Appelons la *taille* d'une configuration c, la cardinalité de $\text{idom}(c)$. Soit c une configuration de D de taille minimale. Pour raisonner par l'absurde supposons que cette taille soit strictement supérieure à r.

Soit $\text{idom}(c) = [j, k]$ et soient les configurations c', c'' et d définies pour tout i par

$$c'_i = \begin{cases} q & \text{si } i = j, \\ c_i & \text{sinon}, \end{cases}$$

$$c''_i = \begin{cases} q & \text{si } i = k, \\ c_i & \text{sinon}, \end{cases}$$

et

$$d_i = \begin{cases} q & \text{si } i = j, \\ q & \text{si } i = k, \\ c_i & \text{sinon.} \end{cases}$$

Ces définitions sont illustrées en figure 7.2. Par la définition d'un état quiescent, le vecteur

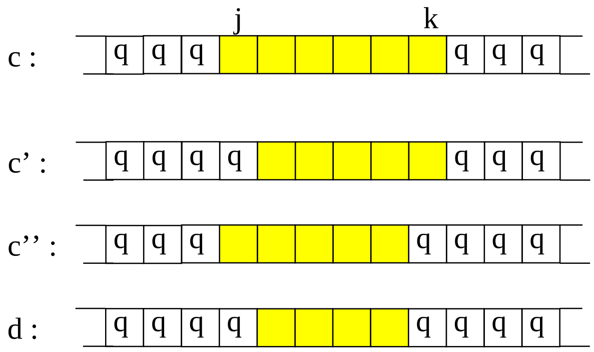


FIG. 7.2: Les configurations c', c'' et d associées à c.

.

colonne indicé par la configuration toute quiescente est de norme 1 et n'est donc pas dans D. Par conséquent $\mathrm{ext}(c) = [j-(r-1), k]$. Nous décomposons maintenant cet intervalle dans les intervalles $I_1 = [j-(r-1), j]$, $I_2 = [j+1, k-r]$ et $I_3 = [k-(r-1), k]$. Par hypothèse $\mathrm{card}(\mathrm{idom}(c)) = k - j + 1 > r$, ce qui assure que I_1 et I_3 sont disjoints.

Soient les valeurs x_1, x_2, x_3 définies pour tout $\ell \in \{1, 2, 3\}$ par

$$x_\ell = \prod_{i \in I_\ell} \|f(c_{i+[r]})\|,$$

et x'_1, x'_3

$$x'_1 = \prod_{i \in I_1} \|f(c'_{i+[r]})\|,$$
$$x'_3 = \prod_{i \in I_3} \|f(c''_{i+[r]})\|.$$

Alors

$$\begin{aligned} \|U_A^q(\cdot, c)\| &= x_1 x_2 x_3, \\ \|U_A^q(\cdot, c')\| &= x'_1 x_2 x_3, \\ \|U_A^q(\cdot, c'')\| &= x_1 x_2 x'_3. \end{aligned}$$

et

$$\|U_A^q(\cdot, d)\| = x'_1 x_2 x'_3 = \frac{\|U_A^q(\cdot, c')\| \cdot \|U_A^q(\cdot, c'')\|}{\|U_A^q(\cdot, c)\|}.$$

Or par construction les tailles de c', c'' et de d sont strictement inférieures à celle de c, et par le choix de c nous avons $\|U_A^q(\cdot, c')\| = \|U_A^q(\cdot, c'')\| = \|U_A^q(\cdot, d)\| = 1$. Ceci contredit l'équation précédente.

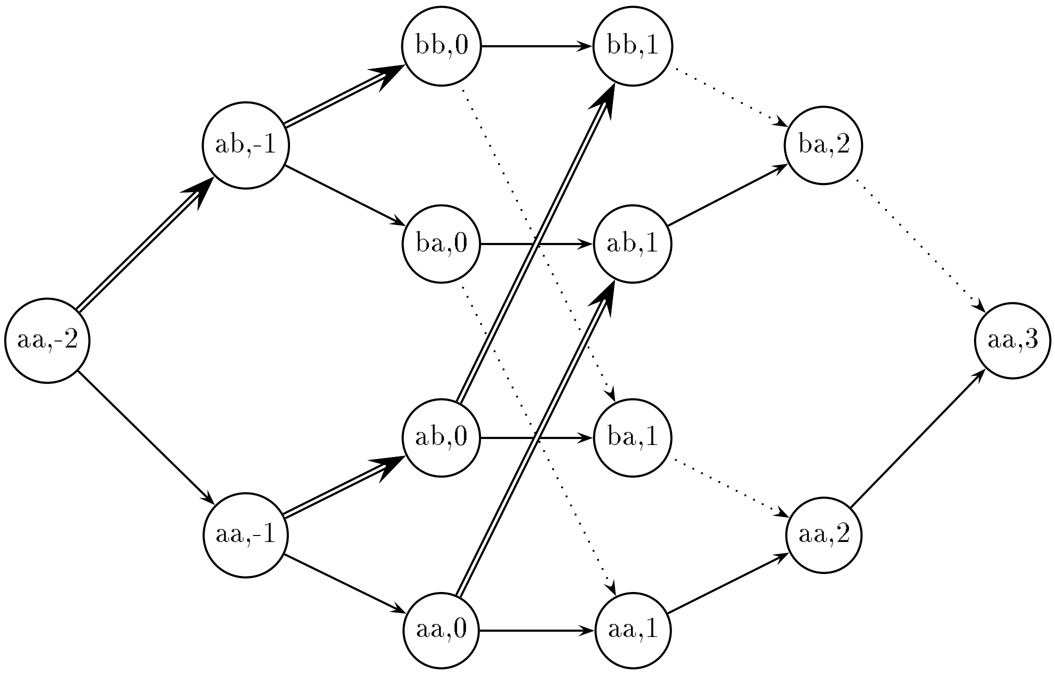


FIG. 7.3: Le graphe G'_B pour B défini en section 7.3 à la page 43. Les arcs de poids 2 sont en double-ligne, et ceux de poids 1/2 en pointillé.

Nous avons donc établit que si $D \neq \emptyset$ alors il existe une configuration de D de taille inférieure ou égale à r, ce qui conclut la preuve. □

Par le lemme précédent il suffit de vérifier $\|U_A^q(\cdot, c)\| = 1$ pour toute configuration c de taille au plus r. Comme la norme des lignes est invariante aux translations sur les configurations, il suffit de vérifier cette propriété pour toutes les configurations dont l'intervalle de domaine est inclus en $[0, r-1]$, à l'exception de la configuration toute quiescente. Soit D' l'ensemble de ces configurations.

Nous remarquons que pour tout $c \in D$, $\mathrm{ext}(c) \subseteq [1-r, r-1]$. Soit le graphe $G'_A = (V, E, g)$ avec

$$V = \{(c_{i+[r-1]}, i) : c \in D', i \in [1-r, r]\}$$

et

$$E = \{((xw, i), (wy, i+1)) : x, y \in Q, w \in Q^{r-2}, (xw, i), (wy, i+1) \in V\}$$

Le poids de l'arc $e = ((xw, i), (wy, i+1))$ est défini par $g(e) = \|f(xwy)\|$. Un chemin est une séquence non-vide de sommets, telle que les sommets successifs sont reliés par un arc. Un chemin vide est une séquence composée d'un unique sommet. Le poids d'un chemin non-vide est le produit du poids des arcs respectifs et le poids d'un chemin vide est 1. Ce graphe est illustré en figure 7.3. Nous appelons le niveau i du graphe l'ensemble des sommets de la forme (w, i) pour $w \in Q^{r-1}$. Nous remarquons que tout arc relie un sommet d'un niveau à un sommet du niveau suivant.

Soient les sommets $v_{\min} = (q^{r-1}, 1-r)$ et $v_{\max} = (q^{r-1}, r)$ et soit P l'ensemble des chemins de $v_{\min}$ à $v_{\max}$.

L'idée de l'algorithme de Høyer est qu'il existe une bijection préservant le poids entre l'ensemble P de chemins et l'ensemble D' de configurations. En conséquence il suffit de vérifier que tous les chemins dans P sont de poids 1, ce qui peut être fait en temps linéaire par un parcours en largeur du graphe.

Lemme 7.4 *Il existe une bijection $M : D' \to P$, telle que pour tout $c \in D'$, le poids de $M(c)$ est égal à $\|U_A^q(\cdot, c)\|^2$.*

Preuve : L'image d'une configuration $c \in D'$ est définie par

$$M(c) = ((c_{1-r+[r-1]}, 1-r), (c_{2-r+[r-1]}, 2-r), \ldots, (c_{r+[r-1]}, r)).$$

Cette fonction satisfait

$$\begin{aligned} g(M(c)) &= \prod_{i=1-r}^{r-1} g((c_{i+[r-1]}, i), (c_{i+1+[r-1]}, i+1)) \\ &= \prod_{i=1-r}^{r-1} \|f(c_{i+[r]})\| \\ &= \|U_A^q(\cdot, c)\|^2. \end{aligned}$$

Si deux configurations c et c' diffèrent en cellule i, alors par construction leurs images $M(c)$ et $M(c')$ diffèrent au niveau i. Donc M est injective.

Pour conclure la preuve nous devons montrer que M est aussi surjective. Pour cela nous commençons par remarquer que tous les sommets de niveau $i \in [1-r, 0]$ sont de la forme

$$(q^{-i} x_0 x_1 \ldots x_{r-1+i}, i)$$

et tous les sommets de niveau $i \in [1, r]$ sont de la forme

$$(x_i x_i + 1 \ldots x_{r-1} q^{i-1}, i)$$

pour des états $x_0, x_1, \ldots, x_{r-1} \in Q$. Par conséquent tout chemin $p \in P$ est de la forme

$$p = ((q^{r-1}, 1-r), (q^{r-2} x_0, 2-r), \ldots, (x_0 \ldots x_{r-2}, 0), (x_1 \ldots x_{r-1}, 1), \ldots, (x_{r-1} q^{r-2}, r-1), (q^{r-1}, r)).$$

Il s'en suit qu'il existe une configuration $c \in D'$ avec $c_i = x_i$ pour tout $i \in [0, r-1]$ telle que $M(c) = p$, ce qui conclut la preuve. □

Nous venons de montrer que pour vérifier que tous les vecteurs colonnes de U_A^q sont de norme 1, il suffit de vérifier que tous les chemins dans P ont le poids 1. Nous donnons maintenant un algorithme pour le dernier problème.

Lemme 7.5 *Il existe un algorithme qui vérifie si tous les chemins de $v_{\min}$ à $v_{\max}$ dans le graphe $G'_A(V, E, g)$ donné sont de poids 1. La complexité est $O(\mathrm{card}(E) \cdot \mathrm{card}(Q))$.*

Preuve : L'algorithme calcule pour tout sommet v, le poids $h(v)$ de tous les chemins de $v_{\min}$ à v.

$h : V \to \mathbb{R}$
$h(v_{\min}) = 1$
Pour tout niveau i de $2 - r$ à r
  Pour tout sommet v de niveau i
    Si pour toutes les arêtes $(u, v) \in E$, la valeur $h(u)g((u, v))$ est la même
      $h(v) = h(u)g((u, v))$
    Sinon
      Arrêter et répondre “Non”
Si $h(v_{\max}) = 1$
  Répondre “Oui”
Sinon
  Répondre “Non”

Supposons que pour un sommet v il existe deux chemins de $v_{\min}$ à v de poids différents. Dans ce cas ces chemins peuvent être prolongés vers $v_{\max}$ dont au moins un est de poids différent de 1. L'algorithme rejette l'entrée dans ce cas, ce qui établit la correction.

L'algorithme parcourt tous les sommets, et pour chaque sommet, tous les arcs entrants. Pour chaque arc (u, v) il calcule $h(u)g((u, v))$, ce qui est déterminé par le calcul de $g(u, v)$, et prend un temps $O(\mathrm{card}(Q))$. Sa complexité est donc $O(\mathrm{card}(E) \cdot \mathrm{card}(Q))$. □

Théorème 7.3 ([Høy96]) *Il existe un algorithme R'_1 qui vérifie si tous les vecteurs lignes de U^q_A sont de norme 1. Sa complexité est $O(n)$.*

Preuve : L'existence d'un algorithme suit des lemmes 7.3 à 7.5. Pour établir la complexité, nous allons déterminer la cardinalité de E.

Par construction de G'_A, pour tout arc $((xw, i), (wy, i + 1))$ il existe une configuration $c \in D'$, tel que $xwy = c_{i+[r+1]}$. Le nombre d'arcs entre le niveau i et le niveau $i + 1$ est alors $\mathrm{card}(Q)^{r-1+i}$ pour $i \in [1 - r, 0]$ et $\mathrm{card}(Q)^{r-1-i}$ pour $i \in [1, r - 1]$. Le nombre total d'arcs est donc

$$\mathrm{card}(E) = \sum_{i' \in [0,r-1]} \mathrm{card}(Q)^{i'} + \sum_{i' \in [0,r-2]} \mathrm{card}(Q)^{i'} < 2\mathrm{card}(Q)^r.$$

Donc la complexité de R'_1 est $O(\mathrm{card}(Q)^r \cdot \mathrm{card}(Q))$, ce qui est $O(n)$, car par convention $n = \mathrm{card}(Q)^{r+1}$. □

7.5 Orthogonalité des vecteurs colonnes

Nous montrons maintenant que l'algorithme de Sutner décrit en section 5.3 à la page 29 peut être adapté pour décider si les vecteurs colonnes de l'opérateur d'évolution sont mutuellement orthogonaux.

L'algorithme de Sutner pour vérifier que la fonction globale d'un AC fini $((Q, r, f), q)$ est injective (théorème 5.1 à la page 33) repose sur le fait que pour toutes configurations finies c et d,

$$F^q_A(c) \neq F^q_A(d) \Leftrightarrow \exists i : f(c_{i+[r]}) \neq f(d_{i+[r]}).$$

Or le lemme 7.1 à la page 42 implique que pour tout ACQ finis $((Q, r, f), q)$ et toutes configurations c et d,

$$U^q_A(\cdot, c) \perp U^q_A(\cdot, d) \Leftrightarrow \exists i : f(c_{i+[r]}) \perp f(d_{i+[r]}).$$

L'idée est donc de remplacer l'inégalité par l'orthogonalité pour obtenir un algorithme de décision pour les ACQ finis isométriques.

Nous définissons le graphe $H_A = (V, E)$ avec — comme dans le cas des AC classiques — l'ensemble des sommets $V = Q^{r-1} \times Q^{r-1}$ par contre l'ensemble des arcs est maintenant

$$E = \{((xw, x'w'), (wy, w'y')) : x, x', y, y' \in \Sigma, w, w' \in \Sigma^{r-2}, f(xwy) \not\perp f(x'w'y')\}.$$

Ce graphe est illustré en figure 7.4. Nous gardons les mêmes notions de cycle et de q-cycles pour H_A.

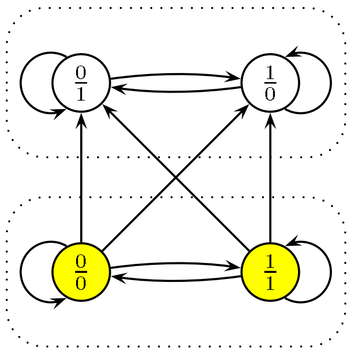


FIG. 7.4: Le graphe H_{Qflip}. Les composants fortement connexes sont visualisées par un contour pointillé. Les sommets diagonaux sont colorés.

Lemme 5.2 à la page 30 (sauf équivalence 1), lemme 5.3 à la page 31 (sauf propriété 1) et la deuxième partie du corollaire 5.1 à la page 32 se généralisent directement :

Lemme 7.6 (version quantique du lemme 5.2) *Les équivalences suivantes sont vérifiées pour tout $A \in$ACQ et tout état quiescent q :*

1. *L'opérateur U_A^P est isométrique si et seulement si pour tout cycle p nous avons $p_1 = p_2$.*
2. *L'opérateur U_A^q est isométrique si et seulement si pour tout q-cycle p nous avons $p_1 = p_2$.*

Lemme 7.7 (version quantique du lemme 5.3) *Soient les propriétés suivantes du graphe H_A.*

1. *Pour tout cycle p, $p_1 = p_2$.*
2. *Toutes les composantes fortement connexes non-triviales ne contiennent que des sommets diagonaux.*
3. *Pour tout q-cycle p, $p_1 = p_2$.*
4. *La composante fortement connexe qui contient (q^{r-1}, q^{r-1}) ne contient que des sommets diagonaux.*

Alors les propriétés 1 et 2 sont équivalentes. Les propriétés 3 et 4 sont aussi équivalentes et la propriété 2 implique 4.

Corollaire 7.2 (version quantique du corollaire 5.1) *Pour tout $A \in$ACQ et tout état quiescent q :*

$$U_A^P \text{ isométrique } \Rightarrow U_A^q \text{ isométrique.}$$

Les automates XOR et QFLIP montrent que l'implication est stricte.

Ces lemmes permettent d'affirmer le théorème suivant :

Théorème 7.4 (version quantique du corollaire 5.2 [DLS96]) *Il existe un algorithme R_2 qui prend en entrée un ACQ fini (A, q) avec $A = (Q, r, f) \in$ ACQ et qui décide si les vecteurs colonnes de l'opérateur d'évolution U_A^q sont orthogonaux. La complexité de cet algorithme est $O(n^{\frac{2r}{r+1}})$.*

Preuve : L'algorithme est le même que dans le cas classique. Par contre ici, $n = \mathrm{card}(Q)^{r+1}$, ce qui prouve la complexité. □

7.6 Norme des vecteurs lignes

Nous décrivons maintenant un algorithme qui prend en entrée un ACQ fini A avec la promesse que U_A^q est isométrique et qui décide si U_A^q est unitaire.

Le plan de cette sous-section est le suivant : D'abord nous décrivons une réduction en plusieurs étapes de notre problème vers un problème de graphe puis vers un problème d'algèbre linéaire.

Nous définissons un graphe de de Bruijn infini, dont les chemins infinis sont en bijection avec les configurations (de colonnes). Toute configuration (de ligne) induit un poids sur les chemins infinis du graphe, tel que le poids total des chemins correspond à la norme de la ligne. Ce poids total peut être décomposé d'une certaine manière en trois parties : le poids de génération de la partie toute quiescente et infinie à gauche, le poids de génération de la partie finie non-quiescente et finalement le poids de la génération de la partie toute quiescente et infinie à droite. Les poids des parties quiescentes peuvent être représentés par des vecteurs dans un espace vectoriel réel de dimension finie. Nous les appelons les vecteurs de bord. Nous montrons qu'ils sont indépendants de la configuration (de ligne) et montrons que leur calcul se résume à calculer le poids total de tous les chemins finis dans un graphe fini. Le poids de la génération de la partie non-quiescente d'une configuration dépend justement de cette configuration et il peut être représenté par une séquence d'opérateurs linéaires dans ce même espace vectoriel. Notre problème se résume finalement au problème d'algèbre linéaire suivant : Étant donné un hyperplan affine, un vecteur et un ensemble fini d'opérateurs linéaires, est-il vrai que toute composition finie des opérateurs envoie le vecteur dans l'hyperplan affine ?

Le théorème suivant est une conséquence directe des théorèmes 7.7 et 7.8 aux pages 58 et 61.

Théorème 7.5 *Il existe un algorithme R_3 qui prend un automate cellulaire quantique, fini $((Q, r, f), q)$ en entrée avec la promesse que son opérateur d'évolution est orthogonal et qui décide si les vecteurs de ligne sont de norme 1. La complexité de cet algorithme est $O(n^{\frac{3r-1}{r+1}})$.*

7.6.1 Réduction

Les différentes étapes de la réduction sont illustrées dans les figures 7.5 à 7.7.

En résumé le problème est le suivant. Il s'agit de décider si la norme de toute ligne de l'opérateur d'évolution d'une ACQ fini donné est 1, avec la promesse, que l'opérateur est isométrique. L'approche naïve échoue, car il faudrait calculer la norme pour un nombre infini de lignes. Pour chaque ligne il peut exister un nombre infini d'entrées non-nulles et de surcroît chaque entrée est définie par un produit dans lequel interviennent un nombre non-borné de valeurs. Il nous faut donc réduire notre problème, vers un problème fini. Cette réduction est le sujet de cette section.

Le *graphe de configurations* est le graphe infini, orienté $G_\infty(V, E)$ défini par $V = Q^{r-1} \times \mathbb{Z}$ et $E = \{((xw, i), (wy, i+1)) : x, y \in Q, w \in Q^{r-2}, i \in \mathbb{Z}\}$. A notre connaissance, ce type de graphe a été utilisé la première fois par Sutner et Maas [SM88] pour montrer qu'un problème particulier de déplacement d'un robot en présence d'obstacles en mouvement est PSPACE-dur. Il a été réutilisé par la suite par Sutner [Sut91] pour prouver que toute pré-image d'une configuration récursive (dont les cellules sont décrites par une fonction récursive) est aussi récursive.

Une séquence non-vide (éventuellement infinie à gauche, à droite ou dans les deux sens) de sommets $(\ldots, (w_i, i), \ldots)$ dans G_∞ est un *chemin* si seulement pour un nombre fini d'indices i nous avons $w_i \neq q^{r-1}$ et s'il y a un arc entre tous les sommets successifs. Précisons qu'une séquence composée d'un unique sommet est déjà un chemin. Nous notons l'ensemble des chemins finis, infini à gauche, infini à droite et dans les deux sens respectivement par M, L, R et P. Figure 7.5 illustre un chemin infini dans les deux sens.

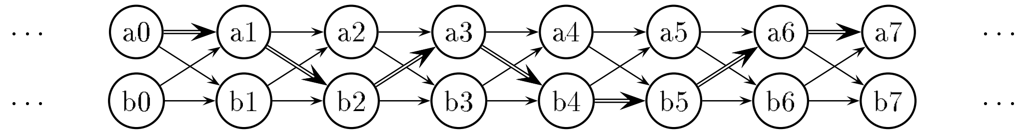


FIG. 7.5: Le graphe de configuration de l'automate (QFLIP, a). Le chemin correspondant à la configuration $\ldots aababbaa \ldots$ est visualisé par des arcs doubles.

Nous disons que deux chemins p_1 et p_2 sont compatibles si le dernier sommet de p_1 et le premier sommet de p_2 existe, et s'ils sont les mêmes. Dans ce cas la composition $p_1 \otimes p_2$ est la concaténation des deux séquences après identification des deux sommets extrêmes. Le résultat est clairement aussi un chemin. Si P_1 et P_2 sont deux ensembles de chemins alors nous définissons

$$P_1 \otimes P_2 = \{p_1 \otimes p_2 : p_1 \in P_1, p_2 \in P_2, p_1 \text{ et } p_2 \text{ sont compatibles}\}.$$

Soit d une configuration arbitraire. Elle induit une fonction *de poids* g_d sur les arcs de G_∞, définie par $g_d((xw, i), (wy, i+1)) = |\langle f(xwy)|d_i\rangle|^2$. Nous étendons la fonction de poids pour des chemins et des ensembles de chemins. Le poids d'un chemin composé d'un unique sommet (chemin vide) est 1 et le poids d'un chemin non-vide est le produit du poids des arcs respectifs. Le poids de l'ensemble vide est 0 et le poids d'un ensemble non-vide de chemin est la somme du poids des chemins respectifs. Nous notons le graphe ainsi pondéré par G_∞^d. La figure 7.6 à la page suivante illustre ce graphe.

Bien que le poids d'un chemin infini soit un produit infini, il est bien défini, car seulement un nombre fini d'arcs ont un poids différent de 1. Le lemme suivant établit une relation importante entre le poids d'un chemin infini dans G_∞^d et les entrées de la matrice d'évolution.

Lemme 7.8 *Il y a une bijection h entre l'ensemble des configurations $\mathcal{C}_A^q$ et l'ensemble des chemins infini dans les deux sens dans G_∞^d, telle que pour toutes configurations c et d nous ayons*

$$g_d(h(c)) = |U_A^q(d, c)|^2.$$

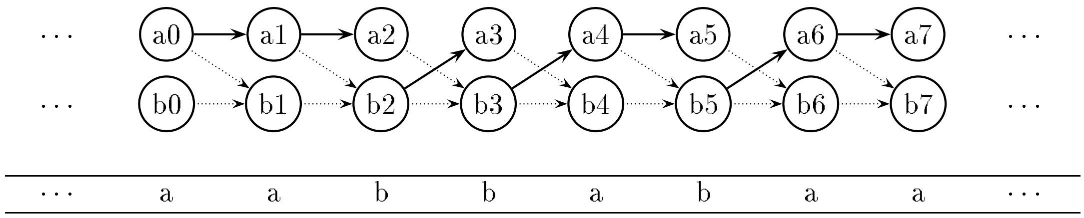


FIG. 7.6: Le graphe de configurations de (QFLIP, a) pondéré par la configuration $d = \ldots aabbabaa \ldots$. Les arcs de poids 1 sont dessinés normalement, ceux de poids 1/2 en pointillé, et ceux de poids 0 sont omis.

Preuve : Soit $h : \mathcal{C}_A^q \to P$ définie pour toute configuration c par

$$h(c) = (\ldots, (c_{i+[r-1]}, i), \ldots).$$

Alors h est une bijection et les équations suivantes concluent la preuve.

$$\begin{aligned}
&g_d(h(c)) = \\
&\quad = \prod_{i \in \mathbb{Z}} g_d((c_{i+[r-1]}, i), i), (c_{i+1+[r-1]}, i+1)) \\
&\quad = \prod_{i \in \mathbb{Z}} |\langle f(c_{i+[r]}) | d_i \rangle|^2 \\
&\quad = \left| \prod_{i \in \mathbb{Z}} \langle f(c_{i+[r]}) | d_i \rangle \right|^2 \\
&\quad = |U_A^q(d, c)|^2.
\end{aligned}$$

□

Nous venons d'établir la réduction suivante de notre problème.

Corollaire 7.3 *Pour tout automate cellulaire quantique fini (A, q) tel que U_A^q soit isométrique et que pour toute configuration d nous ayons $g_d(P) \leq 1$. De plus l'opérateur d'évolution est unitaire si et seulement si pour tout d, $g_d(P) = 1$.*

Fixons une configuration d avec $\mathrm{idom}(d) = [j, k]$. Cette configuration induit des sous-ensembles de L, M et R. Pour tout $w, w' \in Q^{r-1}$ nous posons

$$\begin{aligned}
L_w^d &= \{p \in L : \text{le dernier sommet de } p \text{ est } (w, j)\}, \\
M_{w,w'}^d &= \left\{ p \in M : \begin{array}{l} \text{le premier sommet de } p \text{ est } (w, j) \text{ et} \\ \text{le dernier sommet de } p \text{ est } (w', k) \end{array} \right\}, \\
R_{w'}^d &= \{p \in R : \text{le premier sommet de } p \text{ est } (w', k)\}.
\end{aligned}$$

L'ensemble des chemins infinis dans les deux sens peut être composé par

$$P = \bigcup_{w, w' \in Q^{r-1}} L_w^d \otimes M_{w,w'}^d \otimes R_{w'}^d,$$

et donc

$$g_d(P) = \sum_{w,w' \in Q^{r-1}} g_d(L_w^d) \cdot g_d(M_{w,w'}^d) \cdot g_d(R_{w'}^d).$$

Les lemmes suivants montrent que $g_d(L_w^d)$ et $g_d(R_{w'}^d)$ sont indépendants de d.

Lemme 7.9 *Pour toutes configurations d et d' et tout $w \in Q^{r-1}$ nous avons*

$$g_d(L_w^d) = g_d(L_w^{d'}) \quad \textit{et} \quad g_d(R_{w'}^d) = g_d(R_{w'}^{d'})$$

Preuve : Nous ne prouvons que la première égalité, la preuve pour la deuxième étant similaire. Soit $\mathrm{idom}(d) = [j,k]$, $\mathrm{idom}(d') = [j',k']$ et $m = j'-j$. Nous définissons une bijection de L_w^d vers $L_w^{d'}$ qui préserve le poids. Si $p = (\dots,(w_i,i),\dots,(w_{j'-m},j))$ est un chemin dans L_w^d alors par définition son image est $p' = (\dots,(w_{i-m},i),\dots,(w_j,j'))$. Ceci est clairement une bijection et nous avons par ailleurs $g_d(p) = g_d(p')$, puisque pour tout $i < j$, $d_i = q$ et pour tout $i < j'$, $d'_i = q$. □

Nous définissons les vecteurs de *bord gauche* et de *bord droit*, par $\vec{l} = (l_w)_{w\in Q^{r-1}}$ et respectivement $\vec{r} = (r_w)_{w\in Q^{r-1}}$ comme suit : Pour tout $w \in Q^{r-1}$,

$$l_w = g_d(L_w^d) \text{ et } r_w = g_d(R_w^d),$$

où d est une configuration arbitraire. Le prochain lemme assure que $\vec{l}$ et $\vec{r}$ sont dans $\mathbb{R}^{Q^{r-1}}$.

Lemme 7.10 *Pour tout $w \in Q^{r-1}$, l_w et r_w sont finis.*

Preuve : Supposons qu'il existe un w tel que $l_w = \infty$. (Le cas r_w est symétrique.) Nous voulons montrer qu'il existe alors une configuration, telle que le vecteur ligne associé soit de norme infinie, contredisant par le fait 3.2 l'hypothèse que l'opérateur soit isométrique.

Soit w' tel que $r_{w'} > 0$. Il en existe forcément un, car par exemple $r_{q^{r-1}} \geq 1$. Soient $x_1, x_2, x_{2r-2} \in Q$ tel que $w = x_1 \dots x_{r-1}$ et $w' = x_r \dots x_{2r-2}$. Nous notons pour $i = 1,\dots,r$, $w_i = x_i x_{i+1} \dots x_{i+r-2}$ et pour $i = 1,\dots,r-1$, $v_i = x_i x_{i+1} \dots x_{i+r-1}$. Remarquons que $w_1 = w$ et $w_r = w'$. Pour $i = 1,\dots,r-1$ soit $y_i \in Q$ tel que $\langle f(w'_i)|y_i\rangle \neq 0$. Soit j un entier arbitraire et posons $k = j+r-2$. Nous définissons une configuration d qui est quiescente en dehors de $[j,k]$ et qui est $d_i = y_{i-j+1}$ pour tout $i \in [j,k]$. Alors dans G_∞^d rien que la partie des chemins passant par les sommets $(w_1,j),\dots,(w_r,k+1)$ a un poids infini. Comme les poids ne sont pas négatifs, P a aussi un poids infini, ce qui avec le corollaire 7.3 à la page précédente contredit l'hypothèse que l'opérateur soit isométrique. □

La première partie de notre algorithme consiste à calculer les vecteurs de bords $\vec{l}$ et $\vec{r}$. Pour la seconde partie, nous réduisons maintenant notre problème à une question d'algèbre linéaire.

Pour tout $a \in Q$ soit $F_a \in \mathbb{R}^{Q^{r-1}\times Q^{r-1}}$ un opérateur linéaire dont la matrice est définie pour tout $v, v' \in Q^{r-1}$ par

$$F_a(v,v') = \begin{cases} |\langle f(xwy)|a\rangle|^2 & \text{si } v = xw \text{ et } v' = wy \text{ pour des } x,y \in Q,\ w \in Q^{r-2}, \\ 0 & \text{sinon.} \end{cases}$$

Nous étendons cette définition pour tout mot fini sur Q. Si ϵ dénote le mot vide, alors F_ϵ est l'opérateur d'identité. Soit $s > 1$ un entier et $b = b_1 \dots b_s$ un élément de Q^s. Alors par définition

$$F_b = F_{b_s} \times \cdots \times F_{b_1}.$$

Lemme 7.11 *Soit d une configuration avec $idom(d) = [j,k]$. Alors*

$$g_d(P) = \langle F_{d_j \dots d_k} \vec{l} | \vec{r} \rangle.$$

Preuve : Nous avons

$$\begin{aligned} g_d(P) &= \sum_{w,w' \in Q^{r-1}} g_d(L^d_w) \cdot g_d(F^d_{w,w'}) \cdot g_d(R^d_{w'}) \\ &= \sum_{w,w' \in Q^{r-1}} l_w \cdot g_d(F^d_{w,w'}) \cdot r_{w'} \\ &= \sum_{w'' \in Q^{r-1}} F_{d_j \dots d_k} l_{w''} \cdot r_{w''} \\ &= \langle F_{d_j \dots d_k} \vec{l} | \vec{r} \rangle. \end{aligned}$$

□

Soit Q^* l'ensemble des mots finis sur Q. Pour tout $b \in Q^*$, il existe une configuration dont la partie indicée par son domaine d'intervalle est b. Ce fait, le corollaire 7.3 et le lemme 7.11 prouvent le corollaire suivant.

Corollaire 7.4 *Pour tout automate cellulaire quantique fini (A, q) tel que U^q_A soit isométrique, l'opérateur U^q_A est unitaire si et seulement si pour tout $b \in Q^*$, $\langle F_b \vec{l} | \vec{r} \rangle = 1$.*

En particulier nous devons avoir $\langle \vec{l} | \vec{r} \rangle = 1$, ce qui est équivalent à la condition que le vecteur ligne de l'opérateur d'évolution indicé par la configuration toute quiescente soit de norme 1. Cette condition peut être facilement vérifiée, étant donné $\vec{l}$ et $\vec{r}$. Aussi à partir de maintenant supposons que ce soit le cas. Ceci simplifiera la dernière étape de notre réduction.

Soit $m = \text{card}(Q)^{r-1}$. Les vecteurs de bords peuvent être considérés comme des éléments de $\mathbb{R}^m$, et les éléments de l'ensemble $\mathcal{F} = \{F_a : a \in Q\}$ peuvent à leur tour être considérés comme des applications linéaires dans $\mathbb{R}^m$. Fixons quelques notations concernant l'espace vectoriel $\mathbb{R}^m$. Soit $\vec{u} \in \mathbb{R}^m$ un vecteur, V un ensemble fini de vecteurs et $\mathcal{S} \subset \mathbb{R}^{m \times m}$ une famille finie d'applications linéaires. Nous notons $V + \vec{u} = \{\vec{v} + \vec{u} : \vec{v} \in V\}$ et $\mathcal{S}V = \{s(\vec{v}) : \vec{v} \in V, s \in \mathcal{S}\}$. Le sous-espace généré par V est noté $\langle V \rangle$, et soit $H_{\vec{u}}$ l'hyperplan dont le vecteur normal est $\vec{u}$, c'est-à-dire $H_{\vec{u}} = \{\vec{v} : \langle \vec{v} | \vec{u} \rangle = 0\}$.

Nous définissons par induction sur i, pour $i \geq 0$ les ensembles $\mathcal{S}^i(V)$. Soit $\mathcal{S}^0(V) = V$ et $\mathcal{S}^{i+1}(V) = \mathcal{S}^i(V) \cup \mathcal{S}(\mathcal{S}^i(V))$. Nous disons que V *englobe* $\vec{u}$ *sous* $\mathcal{S}$ si $\bigcup_{i=0}^{\infty} \mathcal{S}^i(\{\vec{u}\}) \subseteq V$.

Puisque par hypothèse $\langle \vec{l} | \vec{r} \rangle = 1$, l'hyperplan affine $H_{\vec{r}} + \vec{l}$ est l'ensemble des vecteurs dont le produit scalaire avec $\vec{r}$ vaut 1, c'est-à-dire

$$H_{\vec{r}} + \vec{l} = \{\vec{u} : \langle \vec{u} | \vec{r} \rangle = 1\}.$$

Clairement pour tout $b \in Q^*$, $\langle F_b \vec{l} | \vec{r} \rangle = 1$ si et seulement $\langle \vec{l} | \vec{r} \rangle = 1$, et que $H_{\vec{r}} + \vec{l}$ englobe $\vec{l}$ sous $\mathcal{F}$. Nous pouvons désormais résumer nos étapes de réduction dans ce théorème :

Théorème 7.6 *Les vecteurs lignes de l'opérateur d'évolution d'un ACQ fini, isométrique sont tous de norme 1 si et seulement si $\langle \vec{l} | \vec{r} \rangle = 1$, et que $H_{\vec{r}} + \vec{l}$ englobe $\vec{l}$ sous $\mathcal{F}$.*

Cette caractérisation est illustrée en figure 7.7.

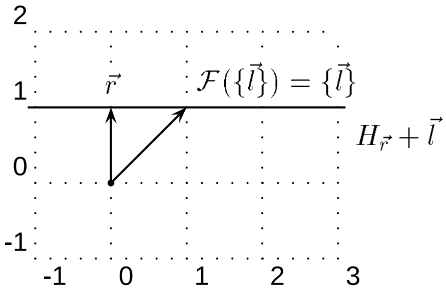


FIG. 7.7: Toujours pour l'ACQ fini (QFLIP, a), les vecteurs de bords $\vec{l}$ et $\vec{r}$, ainsi que l'hyperplan affine $H_{\vec{r}} + \vec{l}$. Dans cet exemple $\vec{l}$ est invariant aux applications F_a et F_b, ce qui montre que U^a_{Qflip} est unitaire.

7.6.2 Calcul des vecteurs de bord

Dans cette section nous donnons un algorithme pour calculer les vecteurs de bord. Par symétrie il est suffisant de ne donner que l'algorithme pour le vecteur de bord gauche. L'objet principal dans ce calcul sera le *graphe de bord pondéré*. Il s'agit du graphe G_A, le graphe utilisé en section 7.6 à la page 52, mais avec une autre fonction de poids. Dans G_A le poids des arcs est défini comme la norme des superpositions d'états résultant d'une transition, alors qu'ici il sera le carré du module de l'amplitude de l'état quiescent dans ces superpositions.

Le graphe de bord (gauche) est le graphe fini, orienté, pondéré $G_l = (V, E, g)$. L'ensemble des sommets est $V = Q^{r-1}$. L'ensemble des arcs est $E = \{(xw, wy) : x, y \in Q, w \in Q^{r-2}\}$. La fonction de poids est définie pour tout $(xw, wy) \in E$ par

$$g((xw, wy)) = |\langle f(xwy)|q\rangle|^2.$$

Un chemin dans G_l est une séquence finie d'au moins deux sommets tel qu'il existe un arc entre deux sommets consécutifs. Remarquons qu'ici un unique sommet ne forme pas encore un chemin. Comme d'habitude, le poids d'un chemin est le poids des arcs et le poids d'un ensemble la somme des poids de chacun de ses chemins. Le poids de l'ensemble vide est 0.

Pour tout $w \in Q^{r-1}$, nous définissons P_w comme étant l'ensemble des chemins en G_l dont le premier sommet est q^{r-1}, le deuxième est différent de q^{r-1} et le dernier est w.

Lemme 7.12 *Pour tout $w \in Q^{r-1}$, nous avons*

$$l_w = \begin{cases} g(P_w) & si\ w \neq q^{r-1}, \\ g(P_w) + 1 & si\ w = q^{r-1}. \end{cases}$$

Preuve : Soit d une configuration avec l'intervalle de domaine $[j, k]$, et soit

$$p_q = (\ldots, (q^{r-1}, i), \ldots, (q^{r-1}, j)).$$

Nous posons $L'_w = L^d_w \backslash \{p_q\}$. Nous allons donner une bijection de L'_w à P_w qui associe p à p' tout en préservant le poids. Soit $p = (\ldots, (w_i, i), \ldots, (w_j, j))$ un élément de L'_w, avec $w_j = w$. Soit h le plus grand entier $h \leq j$ tel que pour tout $i \leq h$ nous ayons $w_i = q^{r-1}$. Alors nous définissons $p' = (q^{r-1}, w_{h+1}, \ldots, w_j)$. Ceci est clairement une injection, et aussi une surjection, car par le choix de h, $w_{h+1} \neq q^{r-1}$. Cette application préserve aussi le poids, car les arcs de p avant le sommet (w_h, h) sont tous de poids 1. Le lemme suit du fait que $g_d(p_q) = 1$. □

Théorème 7.7 *Il existe un algorithme R_{31} qui calcule les vecteurs de bord en temps $O(n^{\frac{3(r-1)}{r+1}})$.*

Preuve : Par le lemme 7.12 il est suffisant de calculer $g(P_w)$ pour tout $w \in Q^{r-1}$. Ce qui rend difficile le calcul de $g(P_w)$ est le fait que les chemins de P_w sont définis par une contrainte qui impose au deuxième sommet d'être différent de q^{r-1}. La solution que nous proposons consiste à coder cette contrainte dans le graphe, que nous allons étendre d'un sommet à ce but. Ensuite nous calculons pour tout couple de sommets i, j, le poids total des chemins de i à j. Pour cela nous adaptons l'algorithme standard qui construit l'expression régulière équivalente à un automate fini donné.

Soit $G'_l = (V', E', g')$ avec $V' = V \cup \{sq^{r-2}\}$ pour un $s \notin Q$, $E' = E \cup \{(sq^{r-2}, q^{r-2}x) : x \in Q\backslash\{q\}\}$ et $g'(e) = g(e)$ pour tout arc e de E et $g'((sq^{r-2}, q^{r-2}y)) = g((q^{r-1}, q^{r-2}y))$. Ce graphe est illustré en figure 7.8. Pour tout $w \in Q^{r-1}$ soit P'_w l'ensemble des chemins dans G'_l tel que le premier sommet soit sq^{r-2} et le dernier sommet soit w. Clairement, P_w est en bijection avec P'_w et $g(P_w) = g'(P'_w)$.

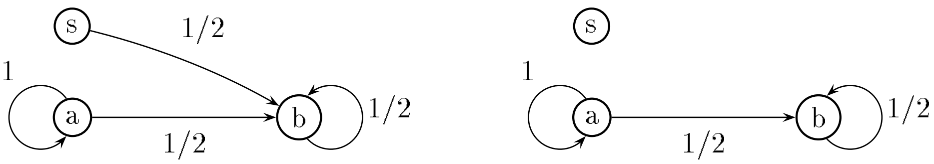


FIG. 7.8: Les graphes de bord G'_l (gauche) et G'_r (droite), associés à l'ACQ fini (QFLIP, a). Ils permettent de calculer $\vec{l} = (1,1)^\dagger$ et $\vec{r} = (1,0)^\dagger$.

Bien que les vecteurs de bords n'aient pas de composantes infinies, pour les calculer nous devons étendre les réels non-négatifs par ∞. Soit $\mathbb{R}_+^\infty$ cet ensemble. Nous définissons les règles de calcul suivants concernant ∞ : pour tout $c \in \mathbb{R}_+^\infty$,

$$\infty + c = c + \infty = \infty,$$

pour tout réel $c > 0$,

$$\infty \cdot c = c \cdot \infty = \infty \cdot \infty = \infty$$

ainsi que

$$\infty \cdot 0 = 0 \cdot \infty = 0,$$

et $\infty^0 = 1$. Nous définissons aussi c^* pour tout $c \in \mathbb{R}_+^\infty$ comme suit

$$c^* = \begin{cases} 1/(1-c) & \text{si } 0 \leq c < 1, \\ \infty & \text{sinon.} \end{cases}$$

Soit $\{v_1, v_2, \ldots, v_{\mathrm{card}(V)}\}$ une énumération arbitraire des sommets de G_l. Pour $1 \leq i, j \leq \mathrm{card}(V)$ et pour $0 \leq k \leq \mathrm{card}(V)$, nous définissons $P_k(i,j)$ comme l'ensemble des chemins débutant en v_i, terminant en v_j et dont tous les autres sommets intermédiaires sont d'indice inférieur ou égal à k. Notons $W_k(i,j)$ pour $g(P_k(i,j))$. Alors nous affirmons que $W_k(i,j)$ satisfait la récursion suivante pour $1 \leq i,j \leq \mathrm{card}(V)$ et $1 \leq k \leq \mathrm{card}(V)$:

$$\begin{aligned} W_0(i,j) &= \begin{cases} g((v_i,v_j)) & \text{si } (v_i,v_j) \in E, \\ 0 & \text{sinon.} \end{cases} \\ W_k(i,j) &= W_{k-1}(i,j) + W_{k-1}(i,k) \cdot (W_{k-1}(k,k))^* \cdot W_{k-1}(k,j). \end{aligned}$$

Nous prouvons notre affirmation par induction sur k. Dans $P_0(i,j)$ le seul chemin est l'arc entre v_i et v_j s'il existe, ce qui établit le cas de base.

Supposons que l'équation soit vraie pour $k-1$. Nous observons que pour chaque chemin de $P_k(i,j)$, il existe un unique entier e, tel que le chemin passe exactement e fois par le sommet v_k. Nous pouvons donc écrire

$$P_k(i,j) = P_{k-1}(i,j) \cup \bigcup_{e=1}^{\infty} P_{k-1}(i,k) \otimes \underbrace{P_{k-1}(k,k) \otimes \cdots \otimes P_{k-1}(k,k)}_{e-1} \otimes P_{k-1}(k,j),$$

où les unions sont disjointes, et l'opérateur $\otimes$ est la composition de chemin (définie en section 7.6.1 à la page 52). Par hypothèse d'induction le poids total de $P_k(i,j)$ est donc

$$W_k(i,j) = W_{k-1}(i,j) + \sum_{e=0}^{\infty} (W_{k-1}(i,k) \cdot (W_{k-1}(k,k))^e \cdot W_{k-1}(k,j)),$$

ce qui conclut la récursion.

La complexité de l'algorithme est $O(\mathrm{card}(Q)^{3(r-1)}) = O(n^{\frac{3(r-1)}{r+1}})$. □

7.6.3 Hyperplans affines englobants

Dans cette section nous allons donner un algorithme polynomial au problème suivant :

> Hyperplans affines englobants
>
> **entrée** les vecteurs $\vec{l}, \vec{r}$ et un ensemble d'applications linéaires $\mathcal{F} = \{F_a : a \in Q\}$
>
> **promesse** $\langle \vec{l} | \vec{r} \rangle = 1$
>
> **décider** si $H_{\vec{r}} + \vec{l}$ englobe $\vec{l}$ sous $\mathcal{F}$, c'est-à-dire si pour tout $b \in Q^*$ nous avons $F_b \vec{l} \in H_{\vec{r}} + \vec{l}$.

Nous rappelons que $m = \mathrm{card}(Q)^{r-1}$ et soit $t = \mathrm{card}(Q)$. Pour simplifier la notation soit $H = H_{\vec{r}}$ et $H' = H_{\vec{r}} + \vec{l}$. Soit $E_i = \mathcal{F}^i(\{\vec{l}\})$.

La principale difficulté pour décider si $\bigcup_{i=0}^{\infty} E_i \subseteq H'$, est que $\bigcup_{i=0}^{\infty} E_i$ peut contenir un nombre infini de vecteurs et que H' n'est pas un sous-espace. Plus précisément, il n'y pas *à priori* de point fixe E_j tel que si E_j est inclus dans H' alors $\bigcup_{i=0}^{\infty} E_i$ l'est aussi. Il n'est donc pas évident qu'il existe un algorithme fini pour décider ce problème.

Par contre il serait beaucoup plus facile si la question consistait à décider si un hyperplan — plutôt qu'un hyperplan affine — englobe $\vec{l}$ sous des application linéaires. Car il nous sera alors possible de travailler avec les sous-espaces générés $\langle E_i \rangle$, et l'existence d'un point fixe suivra par des arguments de dimension, comme nous allons le décrire dans cette section. L'idée principale de notre algorithme consiste à remplacer dans le problème hyperplan affine par hyperplan et applications linéaires par applications linéaires affines, et d'appliquer ensuite l'approche de point fixe. Précisons maintenant cette idée.

Pour tout $F \in \mathcal{F}$, nous définissons une application $F' : \mathbb{R}^m \to \mathbb{R}^m$ par

$$F'(\vec{v}) = F(\vec{v} + \vec{l}) - \vec{l},$$

et nous posons $\mathcal{F}' = \{F' : F \in \mathcal{F}\}$. Observons que $\mathcal{F}'$ est une transformation affine, car $F'(\vec{v}) = F(\vec{v}) + F'(\vec{0})$. De plus $F'(\vec{0}) = F(\vec{l}) - \vec{l}$. Nous posons $E'_i = E_i - \vec{l}$.

Lemme 7.13 *$E'_0 = \{\vec{0}\}$ et pour tout $i \geq 0$, nous avons $E'_{i+1} = E'_i \cup \mathcal{F}'(E'_i)$.*

Preuve : Par définition $E'_0 = \{\vec{l}\} - \vec{l} = \{\vec{0}\}$. Pour tout $i \geq 0$ nous avons

$$\begin{aligned}
E'_{i+1} &= E_{i+1} - \vec{l} \\
&= (E_i \cup \mathcal{F}(E_i)) - \vec{l} \\
&= (E_i - \vec{l}) \cup (\mathcal{F}(E_i) - \vec{l}) \\
&= E'_i \cup (\mathcal{F}(E_i - \vec{l} + \vec{l}) - \vec{l}) \\
&= E'_i \cup (\mathcal{F}(E'_i + \vec{l}) - \vec{l}) \\
&= E'_i \cup \mathcal{F}'(E'_i).
\end{aligned}$$

□

Corollaire 7.5 *H' englobe $\vec{l}$ sous $\mathcal{F}$ si et seulement si H englobe $\vec{0}$ sous $\mathcal{F}'$.*

Ce corollaire suggère déjà qu'il existe un algorithme fini pour le problème : nous calculons le point fixe E'_j — c'est-à-dire le plus petit entier j tel que $\langle E'_j \rangle = \langle E'_{j+1} \rangle$ — et décidons si $E'_j \subseteq H$. Clairement j est au plus m, la dimension de l'espace. Le problème est que E'_j peut contenir un nombre exponentiel de vecteurs. Donc au lieu de calculer E_i nous ne calculerons qu'une base B_i de $\langle E'_i \rangle$, pour tout i, jusqu'à ce que nous obtenions un point fixe. Comme la base contient au plus m éléments, notre algorithme sera polynomial. La correction de cette approche est basée sur le lemme suivant.

Lemme 7.14 *Soit $X, Y \subseteq \mathbb{R}^m$ tel que $\langle X \rangle = \langle Y \rangle$, et $\mathcal{F}'(\{\vec{0}\}) \subseteq \langle X \rangle$. Alors nous avons*

$$\langle X \cup \mathcal{F}'(X) \rangle = \langle Y \cup \mathcal{F}'(Y) \rangle.$$

Preuve : Par symétrie il suffit de montrer $X \cup \mathcal{F}'(X) \subseteq \langle Y \cup \mathcal{F}'(Y) \rangle$, qui implique immédiatement $\langle X \cup \mathcal{F}'(X) \rangle \subseteq \langle Y \cup \mathcal{F}'(Y) \rangle$.

L'hypothèse $\langle X \rangle = \langle Y \rangle$ implique $X \subseteq \langle Y \rangle \subseteq \langle Y \cup \mathcal{F}'(Y) \rangle$. Soit $\vec{v} \in \mathcal{F}'(X)$, alors il existe $F' \in \mathcal{F}'$ et $\vec{u} \in X$ tel que $\vec{v} = F'(\vec{u})$. Puisque $X \subseteq \langle Y \rangle$, il existe un k, et pour tout $1 \leq j \leq k$

il existe $\vec{u}_j \in Y$ et $\alpha_j \in \mathbb{R}$, tel que $\vec{u} = \sum_{j=1}^{k} \alpha_j \vec{u}_j$. Ceci nous mène à

$$\begin{aligned}
\vec{v} &= F'(\vec{u}) \\
&= F(\vec{u}+\vec{l}) - \vec{l} \\
&= F(\vec{u}) + F(\vec{l}) - \vec{l} \\
&= F\left(\sum_{j=1}^{k} \alpha_j \vec{u}_j\right) + F(\vec{0}+\vec{l}) - \vec{l} \\
&= \sum_{j=1}^{k} \alpha_j F(\vec{u}_j) + F'(\vec{0}) \\
&= \sum_{j=1}^{k} \alpha_j F(\vec{u}_j+\vec{l}) - \left(\sum_{j=1}^{k} \alpha_j\right) F(\vec{l}) + F'(\vec{0}) \\
&= \sum_{j=1}^{k} \alpha_j F'(\vec{u}_j) - \left(\sum_{j=1}^{k} \alpha_j\right) (F(\vec{l}) - \vec{l}) + F'(\vec{0}) \\
&= \sum_{j=1}^{k} \alpha_j F'(\vec{u}_j) - \left(\sum_{j=1}^{k} \alpha_j\right) F'(\vec{0}) + F'(\vec{0}) \\
&= \sum_{j=1}^{k} \alpha_j F'(\vec{u}_j) + \left(1 - \sum_{j=1}^{k} \alpha_j\right) F'(\vec{0}).
\end{aligned}$$

Maintenant $F'(\vec{u}_j) \in \mathcal{F}'(Y)$ implique $\sum_{j=1}^{k} \alpha_j F'(\vec{u}_j) \in \langle \mathcal{F}'(Y) \rangle$. Par hypothèse $(1-\sum_{j=1}^{k} \alpha_j)F'(\vec{0})$ est inclus dans $\langle Y \rangle$. Donc la somme des deux vecteurs est en $\langle Y \cup \mathcal{F}'(Y) \rangle$. □

Théorème 7.8 *Il existe un algorithme R_{32} qui décide si H' englobe $\vec{l}$ sous $\mathcal{F}$ en temps $O(n^{\frac{3r-1}{r+1}})$.*

Preuve : Par le corollaire 7.5 nous allons décider si H englobe $\vec{0}$ sous $\mathcal{F}'$ avec l'algorithme suivant :

> $B_1 :=$ une base arbitraire de $\langle \mathcal{F}'(\{\vec{0}\}) \rangle$
> $i \ := 1$
> **tant que** $\langle B_i \rangle \neq \langle B_i \cup \mathcal{F}'(B_i) \rangle$
> $B_{i+1} :=$ une base de $\langle B_i \cup \mathcal{F}'(B_i) \rangle$
> $i \ := i+1$
> $B := B_i$
> **si** $B \subseteq H$ accepter
> **sinon** rejeter

A chaque itération la dimension de $\langle B_i \rangle$ croît et donc l'algorithme termine au bout d'au plus $m-1$ itérations. Nous prouvons maintenant qu'il est correct.

Comme $\mathcal{F}'(\{\vec{0}\}) \subseteq \langle B_1 \rangle$, et $\langle B_i \rangle \subseteq \langle B_{i+1} \rangle$, nous avons $\mathcal{F}'(\{\vec{0}\}) \subseteq \langle B_i \rangle$ pour tout $i \geq 1$. Nous prouvons $\langle B_i \rangle = \langle E_i' \rangle$ pour tout $i \geq 1$ par induction. Pour $i = 1$, ceci est vrai par définition, et l'induction découle des lemmes 7.13 et 7.14. Comme B est un point fixe, c'est-à-dire $\langle B\rangle = \langle B \cup \mathcal{F}'(B)\rangle$, ceci implique que $\langle B\rangle \subseteq \langle \bigcup_{i=0}^{\infty} E'_i\rangle$ et établit ainsi la correction.

Nous considérons maintenant la complexité. Puisque $\mathrm{card}(\mathcal{F}') = t$ et $\mathrm{card}(B_i) \leq m$ pour tout i, nous avons $\mathrm{card}(\mathcal{F}'(B_i)) \leq tm$. Calculer $F'u$ pour une application $F' \in \mathcal{F}'$ et un vecteur $u \in B_i$ prend un temps $O(tm)$, car l'addition et la soustraction avec $\vec{l}$ sont linéaires en m, et que pour la matrice F, il existe $t = \mathrm{card}(Q)$ entrées à des positions bien définies par ligne qui peuvent contenir les seules entrées non-nulles. Donc calculer $\mathcal{F}'(B_i)$ prend un temps $O(t^2m^2)$. Vérifier pour chaque vecteur de $\mathcal{F}'(B_i)$ s'il dépend des vecteurs de base, et l'y ajouter dans le cas échéant, est réalisé en temps $O(m^2)$ par l'algorithme décrit dans la section suivante. Donc le temps par itération est $O(t^2m^2)$, et le temps total est $O(t^2m^3)$. Le théorème suit de $t = n^{\frac{1}{r+1}}$ et $m = n^{\frac{r-1}{r+1}}$. □

7.6.4 Base d'un sous-espace

Dans cette section nous allons donner un algorithme polynomial au problème suivant :

BASE D'UN SOUS-ESPACE VECTORIEL

entrée Un vecteur $u \in \mathbb{R}^m$ et un ensemble B de d vecteurs indépendants de $\mathbb{R}^m$.

sortie Retourner $B \cup \{u\}$ si $u \notin \langle B\rangle$ et rien sinon.

Pour ce problème il faut à la fois trouver un algorithme qui le résout et une représentation adéquate d'une base d'un sous-espace.

Une solution simple à ce problème serait de représenter B par la matrice composée des vecteurs de B, et d'appliquer l'algorithme d'élimination de Gauss (voir par ex. [Len90]) pour décider si $u \in \langle B\rangle$. Ceci prendrait un temps $O(m^2d)$.

C'est la mise en forme triangulaire qui est le goulot d'étranglement de l'algorithme de Gauss. L'amélioration que nous proposons évite cette étape, par une représentation plus adaptée de B.

Théorème 7.9 *Il existe un algorithme R_{33} qui résout* BASE D'UN SOUS-ESPACE VECTORIEL *en temps $O(m(m-d))$.*

Preuve : Nous représentons B par un couple (d, M) avec $d \in [0, m]$ et $M \in \mathbb{R}^{m\times m}$ tel que pour tout $u \in \mathbb{R}^m$, $u \in \langle B\rangle$ si et seulement si $Mu \in \mathbb{R}^d \times \{0\}^{m-d}$.

L'algorithme vérifie si les $m-d$ dernières composantes de Mu sont 0. Le calcul de ces composantes prend un temps $O(m(m-d))$, par la multiplication standard de matrices.

Le cas échéant l'algorithme R_{33} applique une étape de l'algorithme d'élimination de Gauss à u. Cette transformation est décrite par deux opérateurs linéaires sur $\mathbb{R}^m$: une matrice de permutation P et un opérateur *d'élimination* E, que nous définissons maintenant.

Soit k un entier de $[d+1, m]$ tel que $(Mu)_k \neq 0$. Il existe un tel k, car par hypothèse $Mu \notin \langle B\rangle$. Si $k = d+1$, P est l'identité, sinon P est la matrice de permutation qui a $P(d+1, k) = 1$, $P(k, d+1) = 1$, $P(i,i) = 1$ pour tout i différent de k et de $d+1$, et qui est 0 ailleurs.

L'opérateur d'élimination est défini pour tout i, j par

$$E(i,j) = \begin{cases} 1 & \text{si } i = j, \\ -(Mu)_i/(Mu)_k & \text{si } j = d+1 \text{ et } i > d+1, \\ 0 & \text{sinon.} \end{cases}$$

L'algorithme R_{33} retourne alors $(d+1, EPM)$. La correction de l'algorithme de Gauss assure que ceci est un codage de $B \cup \{u\}$. Pour calculer EPM, il faut éventuellement (si $k \neq d+1$) échanger deux lignes de M pour obtenir PM, et ensuite pour tout $i \in [d+2, m]$ ajouter la $(d+1)$-ème ligne pondérée par $E(i, d+1)$ à la i-ème ligne. Ceci prend un temps $O(m(m-d))$, ce qui conclut la preuve. □

Pour que cet algorithme soit appliquable dans notre cas nous avons besoin du lemme suivant.

Lemme 7.15 *Étant donné une base B composée d'un unique vecteur, une représentation de B est calculée en temps $O(m^2)$, où m est la dimension de l'espace vectoriel.*

Preuve : Soit $(1, M)$ une réprésentation de B. Le temps est déterminé par l'écriture de M, qui est de taille $m \times m$. □

8

Comparaison de modèles d'automates cellulaires

Dans ce chapitre nous comparons les modèles d'automates cellulaires périodiques et finis en ce qui concerne l'isométrie et unitarité de l'opérateur d'évolution dans le cas quantique, et injectivité et bijectivité de la fonction de transition globale dans le cas classique. Nous donnerons un contre-exemple pour montrer qu'un des résultats du monde classique n'a pas son équivalent dans le monde quantique.

8.1 Similitudes

Soit un entier $k \geq 1$ arbitraire. Par définition, l'image d'une configuration périodique de taille k par la fonction globale d'un AC est aussi une configuration périodique de taille k. L'ensemble des configurations périodiques de taille k est fini.

Ceci implique pour tout automate cellulaire A (lemme 5.1 à la page 28)

$$F_A^P \text{ injective } \Leftrightarrow F_A^P \text{ bijective,}$$

et de manière similaire pour tout $A \in$ACQ,

$$U_A^P \text{ isométrique } \Leftrightarrow U_A^P \text{ unitaire.}$$

Ces équivalences ainsi que les implications du lemme 8.1, des corollaires 5.1 à la page 32 et 7.2 à la page 51 et du théorème 8.1 à la page suivante sont illustrées en figure 8.1 à la page suivante.

Le lemme et le contre-exemple suivants sont dûs à Durand.

Lemme 8.1 ([Dur94, proposition 4.3]) *Pour tout automate cellulaire A et tout état quiescent q,*

$$F_A^P \textit{ bijective } \Rightarrow F_A^q \textit{ bijective.}$$

La courte preuve du lemme utilise une topologie sur l'ensemble des configurations que nous ne voulons pas introduire ici.

L'automate suivant prouve que l'implication est stricte. Soit XorB$= (\{q, 0, 1\}, 2, f)$ défini pour tout $x, y \in \{q, 0, 1\}$ par

$$f(x, y) = \begin{cases} x & \text{si } x = q \text{ ou } y = q, \\ (x + y) \bmod 2 & \text{sinon.} \end{cases}$$

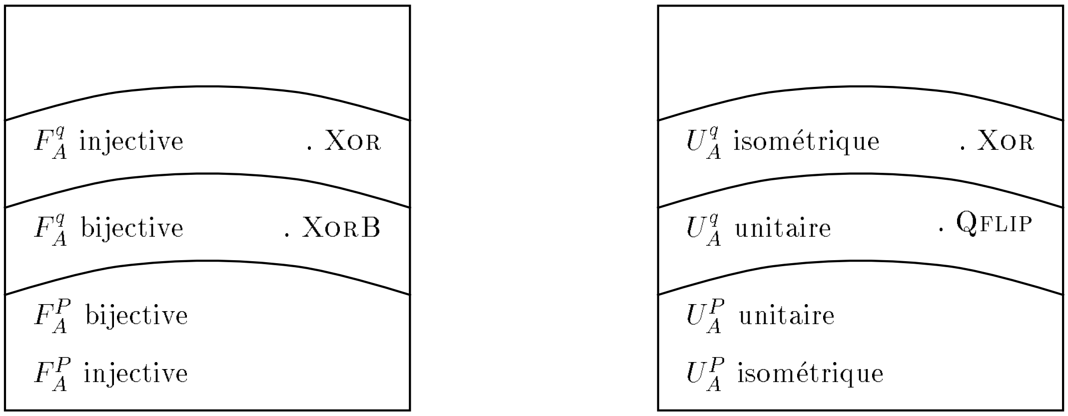


FIG. 8.1: Implications des propriétés de la fonction de transition globale (à gauche) et de l'opérateur d'évolution (à droite). Le carré illustre l'ensemble de tous les couples (A, q). Les zones indiquées par une propriété illustrent le sous-ensemble des couples qui la satisfont.

Soient c_0, c_1 deux configurations périodiques de même taille, dont la première est 0 partout et la deuxième 1 partout. Alors $F_A^P(c_0) = F_A^P(c_1) = c_0$, ce qui montre que F_A^P n'est pas injective. Par contre F_A^q est bijective, ce qui peut être vérifié en appliquant l'algorithme décrit au chapitre 7.

Nous montrons maintenant que le dernier lemme reste toujours vrai dans le cas des ACQ.

Théorème 8.1 *Pour tout $A \in$ACQ et tout état quiescent q,*

$$U_A^P \text{ unitaire } \Rightarrow U_A^q \text{ unitaire.}$$

Preuve : Soit (A, q) un ACQ fini, tel que U_A^P est unitaire. Pour simplifier les notations supposons $A = (Q, 2, f)$. La preuve se généralise aisément pour des voisinages plus grands. Par le corollaire 7.2 à la page 51, il suffit de montrer que U_A^q est surjectif. Soit $[j, j']$ un intervalle non-vide, et c une configuration finie tels que $\mathrm{idom}(c) \subseteq [j, j']$. Nous devons montrer qu'il existe une superposition u, telle que $U_A^q u = |c\rangle$. L'idée de la preuve est la suivante. Soient

$$\begin{aligned} a &= \mathrm{card}(Q) + 1 \\ I &= [j - a, j' + a] \\ L &= \{c \in \mathcal{C}_A^q : \mathrm{idom}(c) \in I\} \\ k &= \mathrm{card}(I). \end{aligned}$$

Nous donnons une bijection M entre L et l'ensemble des configurations périodiques de taille k. Soit $\hat{c} = M(c)$ et soit la superposition $\hat{u}$ telle que $U_A^P \hat{u} = |\hat{c}\rangle$. Nous allons montrer que la superposition $u = \sum_{d \in L} \langle \hat{u} | M(d) \rangle |d\rangle$ est une pré-image pour $|c\rangle$. Comme le choix de i, j' et c était arbitraire ceci impliquerait que U_A^q est unitaire et conclurait la preuve.

$$\begin{tikzcd} |c\rangle \arrow[r, "M"] & |\hat{c}\rangle \\ u \arrow[u, "U_A^q"] & \hat{u} \arrow[l, "M^{-1}"] \arrow[u, "U_A^P"'] \end{tikzcd}$$

FIG. **8.2:** Idée de la preuve

Nous définissons M de la manière suivante : l'image de $c' \in L$ est $\hat{c}'$, telle que pour tout $i \in \mathbb{Z}_k$, $\hat{c}'_i = c'_{j-a+i}$. La définition de L assure que $\hat{c}'$ contient toute la partie non-quiescente de c', donc M est injective. De plus $\mathrm{card}(L) = \mathrm{card}(Q)^k$ ce qui est le nombre de configurations périodiques de taille k, en conséquence M est aussi bijective.

La preuve simple mais fastidieuse est décomposée en trois lemmes. Soit $\hat{d}$ une configuration périodique arbitraire, telle que $\langle \hat{u} | \hat{d} \rangle \neq 0$.

Lemme 8.2 *Il existe $l \in [0, a-1]$ et $l' \in [k-a, k-1]$ tel que $\hat{d}_l = \hat{d}_{l'} = q$.*

Preuve : (illustrée en figure 8.3) Par symétrie il suffit de montrer l'existence de l tel que $\hat{d}_l = q$. Comme card(Q) est fini, et par le choix de a, il existe $0 \leq n < m \leq a-1$ tel que $\hat{d}_n = \hat{d}_m$. Soit $\hat{d}'$ la configuration périodique de taille card$[n, m-1]$, définie pour tout i par $\hat{d}'_i = \hat{d}_{i+n}$. Soit $\hat{e}$ la configuration périodique toute quiescente de même taille que $\hat{d}'$.

Par construction pour tout $i \in [n, m-1]$, $\hat{d}'_{i-m+[2]} = \hat{d}_{i+[2]}$. L'inégalité $U_A^P(\hat{d}, \hat{c}) \neq 0$ implique que pour tout $i \in [0, a-2] \cup [k-a, k-1]$, $\langle f(\hat{d}_{i+[2]}) | q \rangle \neq 0$, et donc en particulier

$$U_A^P(\hat{e}, \hat{d}') = \prod_{i \in [n, m-1]} \langle f(\hat{d}_{i+[2]}) | q \rangle \neq 0.$$

Or par définition d'un état quiescent, $U_A^P(\hat{e}, \hat{e}) = 1$, et comme par hypothèse U_A^P est unitaire, ceci implique $\hat{d}' = \hat{e}$. Nous pouvons donc choisir $l = n$ et obtenons $\hat{d}_l = \hat{d}'_0 = q$. □

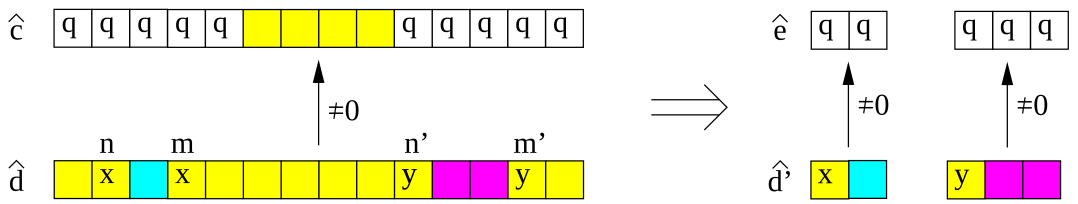


FIG. 8.3: Plan du lemme 8.2.

Lemme 8.3 *Nous avons $\hat{d}_0 = q$.*

Preuve : (illustrée en figure 8.4) Par le lemme précédent il existe $l \in [0, a-1]$ et $l' \in [k-a, k-1]$ tel que $\hat{d}_l = \hat{d}_{l'}$. Soit $\hat{d}'$ la configuration périodique de taille card$([0, l-1] \cup [l', k-1])$, tel que

$$\hat{d}'_i = \begin{cases} \hat{d}_i & \text{si } i \in [0, l-1], \\ \hat{d}_{i-l+l'} & \text{sinon.} \end{cases}$$

Soit $\hat{e}$ la configuration toute quiescente de même taille que $\hat{d}'$. Alors pour les mêmes raison que dans la preuve du lemme précédent, nous avons $\hat{d}' = \hat{e}$, et en particulier $\hat{d}'_0 = q$. □

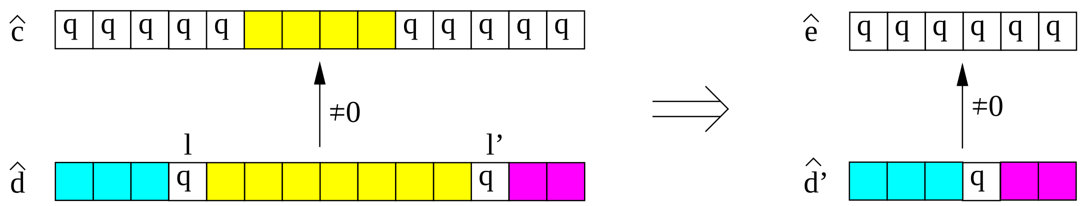


FIG. 8.4: Plan du lemme 8.3.

Lemme 8.4 *Soit $d = M^{-1}(\hat{d})$. Alors pour tout $d' \in L$ et $\hat{d'} = M(d')$, nous avons $U_A^q(d', d) = U_A^P(\hat{d'}, \hat{d})$.*

Preuve : (illustrée en figure 8.5) Le dernier lemme assure que pour tout $i \in [0, k-1]$, $\hat{d}_{i+[2]} = d_{i+j-a+[2]}$. Donc

$$\begin{aligned} U_A^q(d', d) &= \prod_{i \in I} \langle f(d_{i+[2]}) | d'_i \rangle \\ &= \prod_{i \in \mathbb{Z}_k} \langle f(\hat{d}_{i+[2]}) | \hat{d}'_i \rangle \\ &= U_A^P(\hat{d'}, \hat{d}). \end{aligned}$$

□

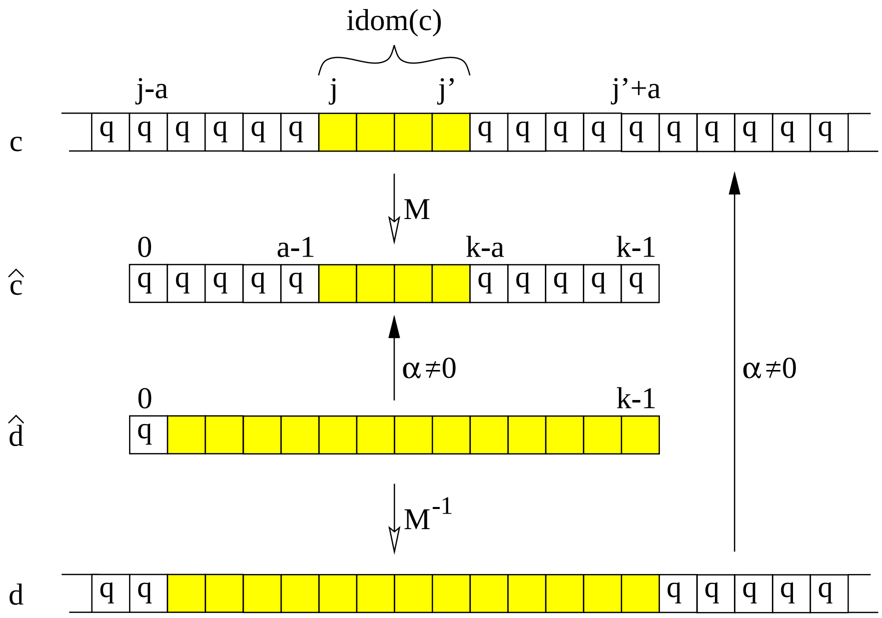


Fig. 8.5: Plan du lemme 8.4. La flèche pleine indique une amplitude de transition α non-nulle.

Nous obtenons donc

$$\begin{aligned} \langle U_A^q u | c \rangle &= \sum_d \langle u | d \rangle U_A^q(c, d) \\ &= \sum_{d \in L} \langle \hat{u} | M(d) \rangle U_A^q(c, d) && \text{par définition de } u \\ &= \sum_{\hat{d}} \langle \hat{u} | \hat{d} \rangle U_A^P(\hat{c}, \hat{d}) && \text{par le lemme précédent} \\ &= 1. && \text{par définition de } \hat{u}. \end{aligned}$$

Le corollaire 7.2 à la page 51 et le fait 3.2 à la page 14 impliquent que pour tout $c' \neq c$, $\langle U_A^q u | c' \rangle = 0$. Donc $U_A^q u = |c\rangle$ ce qui conclut le théorème. □

8.2 Différences

Un théorème de Richardson [Ric72] caractérise les fonctions sur les configurations qui sont des fonctions de transition globales d'automates cellulaires. Il implique le corollaire suivant.

Corollaire 8.1 *Pour tout AC A, si F_A est bijectif, alors il existe un AC B tel que $F_B F_A$ est l'identité.*

Pour tout AC fini (A, q), si F_A^q est bijectif, alors il existe un AC (B, q) tel que $F_B^q F_A^q$ est l'identité.

La deuxième partie de ce corollaire n'est plus vraie dans le cas des ACQ. Considérons l'ACQ QFLIP défini en section 7.3 à la page 43. Pour tout $k \geq 0$, soit c_k la configuration qui est à 1 dans l'intervalle $[-k, -1]$ et 0 ailleurs. Soit la superposition

$$u = \sum_{k \geq 1} 1/\sqrt{2^k} |c_k\rangle.$$

Alors

$$\begin{aligned}
\langle U^0_{\text{Qflip}} u | c_1 \rangle &= \sum_{k \geq 1} 1/\sqrt{2^k} U^0_{\text{Qflip}}(c_1, c_k) \\
&= \sum_{k \geq 1} 1/\sqrt{2^k} \prod_{i=-k-1}^{-2} 1/\sqrt{2^k} \\
&= \sum_{k \geq 1} 1/2^k \\
&= 1.
\end{aligned}$$

Puisque U^0_{Qflip} est unitaire (voir chapitre 7) et $\|u\| = 1$ nous avons $U^0_{\text{Qflip}} u = |c_1\rangle$.

Soit U tel que UU^0_{Qflip} est l'identité. Nous remarquons que c_0 est la configuration toute quiescente. Alors $U|c_0\rangle = |c_0\rangle$ et $U|c_1\rangle = u$. Par conséquent, en une transition par U, toutes les cellules d'indice négatif dépendent de l'état de la cellule d'indice -1. Il ne peut pas exister un automate B tel que $U^0_B = U$, car B devrait avoir un voisinage infiniment grand. De plus en u, les cellules sont enchevêtrées, ce qui ne peut pas être le cas après une unique transition d'un ACQ à partir d'une configuration. (voir section 7.1 à la page 41)

9

Un algorithme quantique de recherche du minimum

Après avoir défini des modèles d'ordinateurs quantiques il fallait comparer leur pouvoir de calcul avec celui des modèles classiques. Pour cela des problèmes artificiels ont été inventés [DJ92, BV93, Sim94] ainsi que des algorithmes quantiques qui les résolvent plus efficacement qu'une machine classique. Shor [Sho94] fut le premier à avoir trouvé un algorithme quantique pour un problème concret, la factorisation d'un nombre N en nombres premiers dans un temps polynomial en $\log N$, ce qui constitue une amélioration par rapport aux algorithmes classiques connus. L'importance de ce résultat vient du fait qu'il rend potentiellement vulnérable le système de cryptographie RSA, basé sur la conjecture qu'il n'existe pas un algorithme efficace pour factoriser un nombre.

Pendant les deux ans qui ont suivi cette découverte, les seuls algorithmes quantiques qui ont été trouvés [Kit95, Høy97], étaient des variations de la transformation de Fourier quantique, la base de l'algorithme de Shor. On pouvait se demander si la puissance des machines quantiques se résume à la capacité de réaliser efficacement la transformation de Fourier.

9.1 Un algorithme quantique de recherche générale

En 1996 Grover a présenté un algorithme quantique pour le problème de recherche générale [Gro96]. L'analyse de cet algorithme a ensuite été affiné par Boyer, Brassard, Høyer et Tapp [BBHT96]. Le problème en question est formalisé comme suit.

Recherche Générale

entrée Un entier N et une fonction $f : \mathbb{Z}_N \to \{0,1\}$ sous forme d'oracle.

sortie Un i tel que $f(i) = 1$, s'il en existe, et rien sinon.

Soit t le nombre solutions, c'est-à-dire le nombre de i tel que $f(i) = 1$. Alors si on n'impose pas de conditions particulières à f, un algorithme classique probabiliste ne pourra pas trouver une solution avec une probabilité supérieure à $1/2$ en moins de $O(N/t)$ appels à l'oracle. L'algorithme proposé par Boyer et al. retourne une solution après un nombre espéré de $O(\sqrt{N/t})$ appels à l'oracle si $t > 0$ et sinon ne termine jamais. Le nombre d'appels à l'oracle de cet algorithme est optimal à une constante près dû à une borne inférieure dans [BBBV97].

Théorème 9.1 ([BBHT96, théorème 3]) *Si le nombre de solutions t est non-nul, alors l'algorithme de recherche quantique retourne uniformément une solution après un nombre espéré d'au plus $\frac{9}{2}\sqrt{N/t}$ appels à l'oracle. Si N est une puissance de 2 alors le temps de calcul est $O(\log N)$ fois le nombre d'appels, sinon le facteur est $O(\log^2 N)$.*

S'il n'existe aucune solution, alors l'algorithme ne termine jamais.

L'espérance exacte du nombre d'appels à l'oracle est implicite dans le théorème. Nous nous référons maintenant à l'analyse du nombre d'appels dans la preuve du théorème 3 dans [BBHT96]. Le nombre espéré avant que l'algorithme atteigne *l'étape critique* est $3\sqrt{N/t}$. Une fois cette étape atteinte, le nombre espéré avant que l'algorithme trouve une solution est $\frac{3}{2}\sqrt{N/t}$. La somme de ces nombres établit la complexité.

Le facteur $O(\log N)$ dans le temps de calcul est déterminé par la transformation de Fourier Walsh-Hadamard intervenant dans l'algorithme. Dans le cas où N n'est pas une puissance de 2, une autre transformation est utilisée (par exemple la transformation de Fourier décrite dans [Cle94]) qui prend un temps $O(\log^2 N)$. Par contre on peut toujours supposer sans perte de généralité que N est une puissance de 2, mais dans ce cas on doit accepter un facteur de $\sqrt{2}$ dans le temps de calcul espéré.

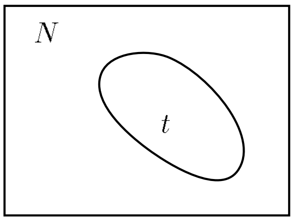


FIG. **9.1:** La recherche générale consiste à trouver une des t solutions parmi N éléments

Tous les problèmes qui sont efficacement vérifiables peuvent être raisonnablement modélisés par un problème de recherche générale. Par exemple un entier i peut coder une assignation de n variables booléennes et f l'évaluation d'une formule logique du premier ordre. Soit $F(n)$, le temps nécessaire à l'évaluation de f. L'algorithme de recherche quantique trouve une assignation satisfaisant la formule (si elle est satisfiable) en temps $O(F(n) \cdot n \cdot 2^{n/2})$. Ceci est encore exponentiel en n, mais améliore le temps $O(F(n) \cdot 2^n)$ de la recherche exhaustive classique, si $F(n)$ reste polynomial en n. Cependant pour certains problèmes le temps de vérification est trop important. Par exemple pour vérifier de manière déterministe si un entier i minimise une fonction $g : \mathbb{Z}_N \to \mathbb{R}$, il n'y a pas d'autre choix que de rechercher de manière exhaustive le minimum. L'algorithme de recherche générale ne s'y applique alors pas tel quel.

9.2 La recherche du minimum

Dans [DH96] nous nous sommes posé la question de savoir s'il est possible d'appliquer l'algorithme de recherche de Grover à un problème qui n'est pas facilement vérifiable. C'est le cas de la recherche du minimum dans un tableau. Formalisons notre problème.

Soit $T : \mathbb{Z}_N \to E$ un tableau (non-trié) de N éléments d'un ensemble E muni d'un ordre total. Pour simplifier supposons que tous les éléments soient distincts. Le problème de recherche du minimum consiste à trouver l'indice y tel que $T[y]$ soit minimum. Résoudre ce problème requiert clairement un nombre linéaire d'accès au tableau dans une machine probabiliste ; il faut au moins $N/2$ accès pour donner la bonne réponse avec une probabilité d'au moins $1/2$.

Nous donnons ici un algorithme quantique simple qui résout avec grande probabilité ce problème n'utilisant que $O(\sqrt{N})$ accès au tableau. Il fait principalement des appels à l'algorithme de recherche quantique. Le nombre d'appels à l'oracle de notre algorithme est optimal à une constante près, dû à une borne inférieure donnée en [BBBV97].

9.3 L'algorithme

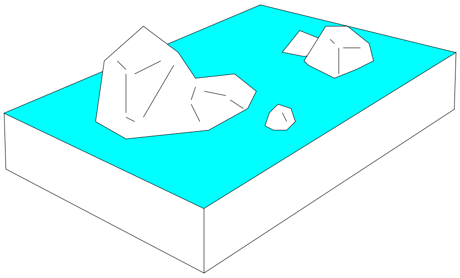


FIG. **9.2:** L'idée de notre algorithme.

La variable principale de notre algorithme est un *indice de seuil* qui détermine un *élément de seuil*. Nous utilisons l'algorithme de recherche quantique [Gro96, BBHT96] comme sous-routine pour trouver l'indice d'un élément qui est plus petit que l'élément de seuil. Le résultat représente ensuite le nouvel indice de seuil. Ce processus est répété jusqu'à ce que la probabilité que l'élément seuil soit le minimum, soit assez large.
L'idée de notre algorithme est illustrée en figure 9.2. Pour rechercher le point maximum dans un paysage, nous en choisissons un au hasard, et versons de l'eau, afin que le niveau d'eau l'atteigne. Ensuite nous recherchons un autre point, parmi le paysage encore découvert, et nous recommençons.
Soit l'algorithme suivant :

Algorithme $A(m)$

1. Choisir uniformément un indice de seuil $y \in \mathbb{Z}_N$.
2. **Tant que** le nombre d'accès au tableau ne dépasse pas m (interrompre éventuellement la sous-routine) **faire**
   (a) Appliquer l'algorithme de recherche quantique de [BBHT96] avec l'oracle f, défini par $f(i) = 1$ si $T[i] < T[y]$ et 0 sinon.
   (b) En cas de succès, affecter y avec la réponse de la sous-routine.
3. Retourner y.

Soit $\hat{m} = \lceil 15\sqrt{N} \rceil$.

Théorème 9.2 *L'algorithme $A(2\hat{m})$ trouve le minimum avec une probabilité au moins $1/2$. Son temps de calcul est $O(\log N\sqrt{N})$ et le nombre d'accès au tableau est $O(\sqrt{N})$.*

Preuve : Par construction, le nombre d'accès au tableau de $A(m)$ est exactement m. Par le théorème 9.1 le temps est $O(\log N \cdot m)$.

Soit $A(\infty)$, la version infinie de l'algorithme. Nous allons montrer que $\hat{m}$ est le nombre espéré d'accès au tableau avant que $A(\infty)$ ne trouve le minimum. Ceci impliquera alors par l'inégalité de Markov, que $A(2\hat{m})$ succède avec une probabilité d'au moins $1/2$.

Nous définissons le rang d'un élément du tableau : le rang du minimum est 1, celui du deuxième plus petit élément est 2, etc.

A tout moment l'algorithme infini cherche le minimum parmi les — disons t — éléments inférieurs à $T[y]$. Pendant le reste de l'exécution chacun de ces éléments pourra être choisi comme indice de seuil. Soit $p(t, r)$ la probabilité que l'élément de rang r est choisi à un moment pendant la recherche du minimum parmi t éléments. Clairement $p(t, 1) = 1$, et $p(t, t) = 1/t$. Nous montrons maintenant que cette probabilité est l'inverse du rang et est indépendante de la taille du tableau.

Lemme 9.1 *Si $r \leq t$ alors $p(t,r) = 1/r$, sinon $p(t,r) = 0$.*

Preuve : Le cas $r > t$ est trivial. Le cas $r \leq t$ est prouvé par induction sur t pour un r fixé : Le cas de base $p(r,r) = 1/r$ suit du fait que la distribution de sortie de l'algorithme de recherche quantique est uniforme. Supposons que pour tout $k \in [r,t]$ l'égalité $p(k,r) = 1/r$ soit vraie. Si l'algorithme recherche le minimum parmi $t+1$ éléments, alors après le premier appel à la sous-routine, y est choisi uniformément parmi eux, et peut être l'indice de l'élément de rang r, de rang plus grand ou plus petit que r. Uniquement les deux premiers cas contribuent à la somme suivante :

$$\begin{aligned} p(t+1,r) &= \frac{1}{t+1} + \sum_{k=r+1}^{t+1} \frac{1}{t+1} p(k-1,r) \\ &= \frac{1}{t+1} + \frac{t+1-r}{t+1} \cdot \frac{1}{r} \\ &= \frac{1}{r}. \end{aligned}$$

□

Par le théorème 9.1 le nombre d'accès espéré qu'il faut à l'algorithme infini pour trouver le minimum est au plus

$$\begin{aligned} \sum_{r=2}^{N} p(N,r) \frac{9}{2} \sqrt{\frac{N}{r-1}} &= \frac{9}{2}\sqrt{N} \sum_{r=1}^{N-1} \frac{1}{r+1} \frac{1}{\sqrt{r}} \\ &\leq \frac{9}{2}\sqrt{N} \left(\frac{1}{2} + \sum_{r=2}^{N-1} r^{-3/2} \right) \\ &\leq \frac{9}{2}\sqrt{N} \left(\frac{1}{2} + \int_{r=1}^{N-1} r^{-3/2} dr \right) \\ &= \frac{9}{2}\sqrt{N} \left(\frac{1}{2} + \left[-2r^{-1/2} \right]_{r=1}^{N-1} \right) \\ &\leq 15\sqrt{N}. \end{aligned}$$

□

9.4 Remarques finales

Si les valeurs dans T ne sont pas distinctes, nous utilisons le même algorithme que dans le cas simplifié. L'analyse reste identique pour ce cas, sauf que dans le lemme 9.1 l'égalité $p(t,r) = 1/r$ devient alors l'inégalité $p(t,r) \leq 1/r$. Donc la borne inférieure pour la probabilité de succès donnée en théorème 9.2 pour le cas simplifié est aussi une borne inférieure pour le cas général.

Si nous exécutons c fois l'algorithme $A(2\hat{m})$, alors la probabilité que le minimum de toutes les c réponses ne soit pas la solution est au plus $1/2^c$. Mais ces exécutions sont indépendantes et ne tiennent pas compte des résultats précédents. De ce fait la probabilité de succès de $A(2c\hat{m})$ ne peut être que meilleure et est donc au moins $1 - 1/2^c$.

Si jamais on arrive à construire un ordinateur quantique, il serait peut-être difficile d'alterner des périodes d'observations et d'évolution de la machines. Il est possible d'implanter cet algorithme de telle manière qu'il existe une unique observation à la fin du calcul. Pour cela il suffit d'écrire (de manière réversible, avec XOR) les bits à observer sur un ruban particulier, et ne plus y toucher jusqu'à la fin du calcul. L'observation des bits écrits sur ce ruban fournit alors le même résultat que si on avait observé ces bits au fur et à mesure au cours de l'évolution de la machine.

10

Conclusion

Comparaison des ACQ finis avec les machines de Turing quantiques

Le principal problème que nous laissons ouvert dans cette thèse est celui de la simulation (avec un surcoût polynomial) des automates cellulaires quantiques finis par des machines de Turing quantiques. Considérons l'ensemble des modèles de calcul qui sont équivalents (par simulation mutuelle avec un surcoût polynomial) aux machines de Turing classiques. La version moderne de la thèse de Church et Turing établit que cet ensemble est équivalent à l'ensemble des modèles de calcul "raisonnables". Alors il nous semblerait réconfortant que les versions quantiques de ces modèles soient aussi équivalentes aux machines de Turing quantiques.

L'existence d'une simulation des ACQ finis par les machines de Turing quantiques viendrait alors soutenir cette *version quantique* de la version moderne de la thèse de Church et Turing. Par contre nous avions montré en section 8.2 à la page 69 qu'il existe au moins une différence importante entre les ACQ finis et les AC finis classiques : pour toute instance valide des ACQ finis il n'existe pas forcément un ACQ fini qui l'inverse. Il ne serait donc pas surprenant qu'il existe d'autres différences, par exemple l'existence d'automates cellulaires quantiques finis qui ne peuvent pas être simulés avec un surcoût polynômial par des machines de Turing quantiques.

Critiques

Certains physiciens nous ont fait part de leurs remarques concernant le modèle des ACQ finis :

Richard Jozsa reproche à ce modèle que toutes les transitions se fassent en même temps. En effet, ceci nécessite une horloge globale commune à toutes les cellules, ce qui n'est pas physiquement réalisable.

Dorit Aharonov regrette quem dans notre modèlem il ne soit pas facile de construire des instances valides. Les automates cellulaires quantiques partitionnés ou les circuits quantiques sont beaucoup plus adaptés pour concevoir en détail un programme quantique.

Norm Margolus et Seth Lloyd nous signalent qu'un modèle de calcul quantique ne peut avoir de réalité physique que s'il peut être décrit par des *Hamiltoniens locaux* et donc les

ACQ finis ainsi que les machines de Turing quantiques ne sont pas des modèles réalistes.

Il s'avère donc que notre modèle n'est (malheureusement) qu'une construction dont la raison d'être ne peut être que théorique : Nous pensons qu'il nous permettrait de vérifier (ou de réfuter) la version moderne et quantique de la thèse Church-Turing.

Ce n'est pas parce qu'un modèle de calcul quantique ne satisfait pas la condition requise mentionnée par Norm Margolus et Seth Lloyd ci-haut, qu'il n'est pas raisonnable d'y concevoir en détail des algorithmes quantiques. La simulation de Yao des machines de Turing quantiques par des circuits quantiques rend au premier modèle sa raison d'être. Ceci fournit une raison de plus pour résoudre le problème ouvert mentionné au début de cette conclusion.

Automates cellulaires quantiques sans restriction

Les automates cellulaires (sans restriction sur les configurations) n'ont pas encore été généralisés au calcul quantique. Le problème est que dans ce cas l'ensemble des configurations n'est pas dénombrable, et donc on ne peut utiliser le formalisme des opérateurs d'évolution. C'est ce que qui a été évité en ne considérant que des ACQ finis ou périodiques. Notre intuition est que dans le cas général la similarité entre la différence des états (ou des configurations) et l'orthogonalité des superposition d'états (ou des superpositions de configurations) qui a été exploitée dans cette thèse reste toujours vraie. Nous pensons donc que — comme dans le cas classique — un ACQ est valide dans le cas général si et seulement s'il l'est dans le cas des configurations périodiques.

Aussi faudrait-il faire le lien entre ces deux derniers modèles et le modèle des automates cellulaires quantiques basé sur le formalisme de Hamiltoniens. Quelles sont les instances des deux premiers modèles qui peuvent être décrites par le dernier et vice-versa ? Quel est alors le rapport entre les deux descriptions ?

Bibliographie

[ADH97] L. Adleman, J. DeMarrais, M.D.A. Huang, Quantum computability, à paraître dans *SIAM Journal on Computing*, `huang@pollux.usc.edu`, 1997.

[AP72] S. Amoroso et Y. Patt, Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures, *Journal of Computer and System Sciences* **6**, 448–464, 1972.

[BBBV97] C.H. Bennett, E. Bernstein, G. Brassard et U. Vazirani, Strengths and weaknesses of quantum computing, à paraître dans *SIAM Journal on Computing*, 1997.

[BBHT96] M. Boyer, G. Brassard, P. Høyer et A. Tapp, Tight bounds on quantum searching, *Proceedings of the 4th Workshop on Physics and Computation*, 1996.

[Bel58] R. Bellman, On a routing problem, *Quarterly of Applied Mathematics* **16**, (1):87–90, 1958.

[Ben73] C.H. Bennet, Logical reversibility of computation, *IBM J. Res. Develop.* **17**, 525–532, 1973.

[Ben82a] P. Benioff, Quantum mechanical hamiltonian models of Turing machines, *J. Stat. Phys.* **29**, 515–546, 1982.

[Ben82b] P. Benioff, Quantum mechanical hamiltonian models of Turing machines that dissipates no energy, *Physical Review Letters* **48**, 1581–1585, 1982.

[Bia94] M. Biafore, Can computers have simple hamiltonians? *Proceedings of the 3rd Workshop on Physics and Computation*, 63–69, 1994.

[BV93] E. Bernstein et U. Vazirani, Quantum complexity theory, *Proceedings of the 25th ACM Symposium on the Theory of Computing*, 11–20, 1993.

[BV97] E. Bernstein et U. Vazirani, Quantum complexity theory, (version journal de [BV93]), à paraître dans *SIAM Journal on Computing*, `vazirani@cs.berkeley.edu`, 1997.

[Cle94] R. Cleve, A note on computing Fourier transforms by quantum programs, *manuscrit*, `cleve@cwi.nl`, 1994.

[CLR90] T. Cormen, C. Leiserson et R. Rivest, Introduction to Algorithms, *The MIT Press*, 1990.

[Dam96] W. van Dam, A universal quantum cellular automaton, *Fourth Workshop on Physics and Computation*, 1996.

[Deu85] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer, *Proceedings of the Royal Society of London*, (A400):97–117, 1985.

[Deu89] D. Deutsch, Quantum computational networks, *Proceedings of the Royal Society of London*, (A425):73–90, 1989.

[DH96] C. Dürr, P. Høyer, A Quantum algorithm for finding the minimum, manuscrit, `http://xxx.lanl.gov/quant-ph/abs/9607014`, 1996.

[DJ92] D. Deutsch et R. Jozsa, Rapid solution of problems by quantum computation, *Proceedings of the Royal Society of London*, A439:553–558, 1992.

[DLS96] C. Dürr, H. LêThanh et M. Santha, A decision procedure for well-formed linear quantum cellular automata, *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, 281–292, 1996.

[DS96] C. Dürr et M. Santha, A decision procedure for unitary linear quantum cellular automata, *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, 38–45, 1996.

[Dur94] B. Durand, Automates cellulaires: réversibilité et complexité, *Thèse à l'École normale supérieur de Lyon*, 1994.

[FAQ] Contributions diverses, édité par H. Gutowitz, Frequently asked questions about cellular automata, `http://alife.santafe.edu/alife/topics/ca`, mis-à-jour régulièrement.

[Fey82] R. Feynman, Simulating physics with computers, *International Journal of Theoretical Physics* **21**, 467–488, 1982.

[Fey85] R. Feynman, QED – The Strange Theory of Light and Matter, *Princeton University Press*, 1985.

[Fey86] R. Feynman, Quantum mechanical computers, *Foundations of Physics* **16**, 507, 1986.

[FF62] L. Ford et D. Fulkerson, Flows in networks, *Princeton University Press*, 1962.

[Gar70] M. Gardner, The fantastic combinations of John Conway's new solitaire game 'Life', *Scientific American* **223**, (4):120–123, 1970.

[Gro96] L.K. Grover, A fast quantum mechanical algorithm for database search, *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 212–219, 1996.

[Har76] J. Hardy, O. de Pazzis et Yves Pomeau, Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions, *Physical Review* A, **13**, 1949–1960, 1976.

[Høy96] P. Høyer, Note on linear quantum cellular automata, *manuscrit*, `u2pi@imada.-ou.dk`, 1996.

[Høy97] P. Høyer, Efficient Quantum Transforms, *manuscrit*, `http://xxx.lanl.gov/-abs/quant-ph/9702028`, 1997.

[Jac90] G. Jacopini et G. Sontacchi, Reversible parallel computation: an evolving space model, *Theoretical Computer Science*, **75**, 1990.

[Kar90] J. Kari, Reversability of 2D cellular automata is undecidable, *Physica*, D **45**, 379–385, 1990.

[Kar96] J. Kari, Representation of reversible cellular automata with block permutations, *Mathematical Systems Theory* **29**, 47–61, 1996.

[Kit95] A.Y. Kitaev, Quantum measurements and the Abelian stabilizer problem, *manuscrit*, `http://xxx.lanl.gov/abs/quant-ph/9511026`, 1995.

[Lec63] Y. Lecerf, Machines de Turing réversibles. Récursive insolubilité en $n \in \mathbb{N}$ de l'équation $u = \rho^n$, où ρ est un 'isomorphisme de codes', *Comptes Rendus* **19**, 2597–2600, 1963.

[Len90] A.K. Lenstra et H.W. Lenstra Jr., Algorithms in number theory, dans *Handbook of Theoretical Computer Science* **A**, Elsevier Science Publishers et The Mit Press, 683–685, 1990.

[Llo93] S. Lloyd, A potentially realizable quantum computer, *Science* **261**, 1569–1571, 1993.

[Llo94] S. Lloyd, Envisioning a quantum supercomputer, *Science* **263**, 695, 1994.

[LTP94] C.S. Lent, P.D. Tougaw, W. Porod, Quantum cellular automata: The physics of computing with quantum dot molecules, *Proceedings of the Workshop on Physics and Computing*, 1994.

[LTPB93] C.S. Lent, P.D. Tougaw, W. Porod et G.H. Bernstein, Quantum cellular automata, *Nanotechnology* **4**, 49, 1993.

[Mag95] F. Magniez, Sur le calcul quantique, *Rapport de stage de DEA*, Université Paris-Sud, Laboratoire de Recherche en Informatique, 1995.

[Mar84] N. Margolus, Physics-like models of computation, *Physica* D **10**, 81–95, 1984

[Mar87] N. Margolus, Physics and Computation, *PhD in Physics* Massachusetts Institute of Technology, 1997.

[Mar94] N. Margolus, Parallel quantum computation, *Complexity, Entropy and the Physics of Information*, Addison-Wesley, 273, 1994.

[May96] P. Maymin, Extending the lambda calculus to express randomized and quantumized algorithms, *manuscrit*, `http://xxx.lanl.gov/abs/quant-ph/9612052`, 1996.

[Min67] M. Minsky, Computation: Finite and Infinite Machines, Prentice Hall, 1967.

[Obe88] K. Oberteich, W.G. Teich et G. Mahler, Structural basis of multistationary quantum systems, *Physical review* B **37**, 8096–8121, 1988.

[Ric72] D. Richardson, Tesselations with local transformations, *Journal of Computer and System Sciences* **6**, 373–388, 1972.

[Rok96] Z. Roka, Jeu de la vie, en préparation, `roka@iut.univ-metz.fr`.

[Sim94] D. Simon, On the power of quantum computation, *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, 116–123, 1994.

[Sho94] P. Shor, Algorithms for quantum computation: discrete log and factoring *Proceedings of the 26th ACM Symposium on the Theory of Computing*, 124–134, 1994.

[SM88] K. Sutner et W. Maass, Motion planning among time-dependent obstacles, *Acta Informatica* **26**, 93–122, 1988.

[Sut91] K. Sutner, De Bruijn graphs and linear cellular automata, *Complex Systems* **5**, (1):19–30, 1991.

[Tar75] R. Tarjan, Depth first search and linear graph algorithms, *SIAM Journal on Computing* **1**, (2):146–160, 1972.

[TM87] T. Toffoli et N. Margolus, Cellular Automata Machines, *MIT Press*, 1987.

[TM90] T. Toffoli et N. Margolus, Invertible cellular automata: a review, *Physica* D **45**, 229–253, 1990.

[Tof77a] T. Toffoli, Cellular Automata Mechanics, *Rapport technique 208*, Comp. Comm. Sci. Dept., University of Michigan, 1977.

[Tof77b] T. Toffoli, Computation and construction universality of reversible cellular automata, *Journal of Computer System Science* **15**, 213–231, 1977.

[Wat95] J. Watrous, On one dimensional quantum cellular automata, *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, 528–537, 1995.

[Wol84] S. Wolfram, Computation theory of cellular automata, *Communications in Mathematical Physics* **96**, (1):15–57, 1984.

[Wol94] S. Wolfram, Cellular Automata and Complexity: Collected Papers, *Addison-Wesley*, 1994.

[Yao93] A. Yao, Quantum circuit complexity, *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, 352–361, 1993.