Aix*Marseille université

THÈSE D'HABILITATION À DIRIGER DES RECHERCHES

Soutenue à Aix-Marseille Université
le 25 janvier 2022 par

Kévin PERROT

Études de la complexité algorithmique des réseaux d'automates

Discipline
Informatique

École doctorale
ED 184 Mathématiques et Informatique

Équipe Calcul Naturel du Laboratoire d'Informatique et Systèmes UMR CNRS 7020

Composition du jury

Julio ARACENA Professeur à l'Université de Concepción Département d'Ingénierie Mathématique	Rapporteur
Olivier BOURNEZ Professeur à l'École Polytechnique LIX	Président
Nadia CREIGNOU Professeure à Aix-Marseille Université LIS	Examinatrice
Emmanuel JEANDEL Professeur à l'Université de Lorraine LORIA	Rapporteur
Jarkko KARI Professeur à l'Université de Turku Département de Mathématiques	Rapporteur
Loïc PAULEVÉ Chargé de recherche du CNRS LaBRI	Examinateur
Sylvain SENÉ Professeur à Aix-Marseille Université LIS	Tuteur

LIS LABORATOIRE D'INFORMATIQUE & SYSTÈMES

Je soussigné, Kévin Perrot, déclare par la présente que le travail présenté dans ce manuscrit est mon propre travail, réalisé dans le respect des principes d'honnêteté, d'intégrité et de responsabilité inhérents à la mission de recherche. Les travaux de recherche et la rédaction de ce manuscrit ont été réalisées dans le respect à la fois de la charte nationale de déontologie des métiers de la recherche et de la charte d'Aix-Marseille Université relative à la lutte contre le plagiat.

Ce travail n'a pas été précédemment soumis en France ou à l'étranger dans une version identique ou similaire à un organisme examinateur.

Résumé

Les réseaux d'automates modélisent toute dynamique discrète finie. Chaque automate possède un état, qui évolue en étapes de temps discrètes, selon l'état des autres automates. La structure des interactions est décrite par un graphe et des fonctions locales de transition. Une dynamique globale est obtenue avec un mode de mises à jour, décrivant quels automates appliquent leur fonction locale pour changer d'état, de façon déterministe ou non déterministe. Cette grande expressivité permet la modélisation de phénomènes observés dans la nature, où le paradigme du passage des règles locales au comportement global capture l'essence des systèmes « complexes ».

Un réseau d'automates est encodé par les circuits (booléens) de ses fonctions locales. Puisque l'espace est fini, toute la dynamique peut être calculée. Nous proposons une compréhension des variantes du modèle, à travers la caractérisation des complexités algorithmiques de problèmes : liés au calcul du graphe des interactions, liés à la dynamique asymptotique d'un réseau d'automates donné, liés à la dynamique asymptotique de tous les réseaux possibles sur une structure d'interactions donnée, et liés à plusieurs modes de mises à jour. Il s'agira indirectement d'appréhender la capacité à implémenter du calcul dans les différentes facettes du modèle.

Nous dégagerons des éléments clés sur la structure des interactions, le principe de localité, l'importance des alphabets d'états, et les encodages. Ce travail aboutira à des perspectives sur le long terme, guidées par l'extension d'informations partielles sur un réseau, et l'énoncé de théorèmes « à la Rice » formulés grâce à la théorie des modèles.

Mots clés. Complexité algorithmique, systèmes dynamiques discrets (en espace et en temps), réseaux d'automates, graphe d'interaction.

Abstract

Automata networks model all finite discrete dynamics. Each automaton has a state, evolving in discrete time steps, according to the states of other automata. The structure of interactions is described by a graph and local transition functions. A global dynamics is obtained with an update schedule, describing which automata apply their local function to change state, deterministically or nondeterministically. This rich expressivity allows to modelize phenomena observed in nature, where the paradigm of going from local rules to global behavior captures the essence of "complex" systems.

An automata network is encoded by the (Boolean) circuits of its local functions. Since space is finite, the whole dynamics can be computed. We intend to understand variants of the model, through characterizing the computational complexities of problems: related to computing the graph of interactions, related to the asymptotic dynamics of a given automata network, related to the asymptotic dynamics of all possible networks on a given structure of interactions, and related to update schedules. It indirectly deals with the capacity to embed computation in different aspects of the model.

We will draw key elements on the structure of interactions, the principle of locality, the importance of state alphabets, and encodings. This work will result in long term perspectives, on the extension of some partial information on a network, and the statement of "Rice-like" theorems formalized using model theory.

Keywords. Algorithmic complexity, discrete dynamical systems (in time and space), automata networks, interaction graph.

Table des matières

Notations

$[\![n]\!], [n]$	Nombres entiers de 0 à $n-1$, et de 1 à n
$\bot, \top$	Faux et vrai
$0^n, 1^n$	Configuration dont les n composantes valent $0, 1$
$\circ$	Composition de fonctions, $(f \circ g)(x) = f(g(x))$
$\oplus, \leftrightarrow$	Ou exclusif (xor) et équivalence ($a \leftrightarrow b \equiv \neg(a \oplus b)$)
$\sqcup$	Union disjointe d'ensembles et de graphes
$\forall x \in X : \exists y \in Y : \varphi$	Délimiteur « : » pour les quantifications
$X \models \varphi$	L'objet X vérifie la propriété φ
$\leq_{\mathtt{m}}^{\mathsf{C}}$	Réduction many-one dans FC
$\leq_{\mathtt{tt}}^{\mathsf{C}}$	Réduction truth-table dans FC
$\leq_{\mathtt{T}}^{\mathsf{C}}$	Réduction Turing dans C (avec oracle)
$\leq_{\mathtt{parci}}^{\mathsf{P}}$	Réduction many-one parcimonieuse dans FP
$\Delta(G)$	Degré entrant maximum du graphe orienté G
$e_i, x + e_i \in \{0,1\}^n$	i-ème vecteur de base, addition modulo 2 (*flip* état i)
$G[V']$	Sous-graphe induit par $V' \subseteq V$ dans $G = (V, A)$
$\log n, \log\log n$	$\log_2(n)$ le logarithme en base 2, $\log\big(\log(n)\big)$
$\mathbb{N}_+$	Entiers naturels strictement positifs, $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$
NC, P, NP, $\#\mathsf{P}$...	Classes de complexité (voir chapitre 2)
$\Omega(f(n))$, $\mathcal{O}(f(n))$, $\Theta(f(n))$	Bornes asymptotiques (inf., sup., exacte)
$\mathcal{P}(X)$	Ensemble des sous-ensembles de l'ensemble X
x_I	Restriction de $x \in \{0,1\}^n$ aux composantes de $I \subseteq [n]$

$\beta^{\mathtt{par}}$	Mode de mise à jour parallèle
$\mathfrak{B}_f(x)$	Bassin d'attraction de x dans f, $\mathfrak{B}_f(x) = \{y \mid x \in \mathfrak{O}_f(y)\}$
$\mathtt{BS}_n$	Ensemble des modes de mises à jour bloc-séquentiels sur n automates (partitions ordonnées de $[n]$)
$\mathfrak{C}_f^k$	Ensemble des cycles limites de taille k dans f
$f^{[(B)]}$	Mise à jour d'un bloc B sur f
$f^{[\beta]}$	Mode de mise à jour β appliqué à f
$\mathcal{F}_G$	Ensemble des $f : \{0,1\}^n \to \{0,1\}^n$ tels que $G_f = G$
$\mathcal{F}_G(i)$	Ensemble des $f_i : \{0,1\}^n \to \{0,1\}$ pour $f \in \mathcal{F}_G$
$G_f^{\pm}, G_f$	Graphe d'interaction signé, non signé, de f
G_f^{β}	Graphe des mises à jour de f pour $\beta \in \mathtt{BS}_n$
$\mathscr{G}_f$	Graphe de la fonction f
$\mathcal{N}_G(i)$	Ensemble des voisins entrants de i dans G
$\mathcal{N}_G^+(i), \mathcal{N}_G^-(i), \mathcal{N}_G^{\pm}(i)$	Partition de $\mathcal{N}_G(i)$ selon les arcs $+, -, \pm$
Ω_f	Ensemble limite de f, $\Omega_f = \bigcap_{t\in\mathbb{N}} f^t(\{0,1\}^n)$
$\mathfrak{O}_f(x)$	Orbite de x dans f, $\mathfrak{O}_f(x) = \{f^t(x) \mid t \in \mathbb{N}\}$
$\mathfrak{P}_f$	Ensemble des points fixes de f
$\mathfrak{P}^{\mathtt{max}}(G), \mathfrak{P}^{\mathtt{min}}(G)$	Nombre max., min. de points fixes d'un $f \in \mathcal{F}_G$

1. Introduction

« *Pour qu'une chose soit intéressante, il suffit de la regarder longtemps.* »
Gustave Flaubert (1821–1880)

J'aime beaucoup cette citation, qui de mon point de vue s'applique tout à fait à l'informatique théorique, dans un sens très positif. Il ne s'agit pas de dire que tout se vaut et donc rien n'a de valeur, mais au contraire que tout peut être sujet à des discussions intéressantes. L'astrophysicien Aurélien Barrau a résumé ma pensée lors d'une intervention à *TEDxMarseille* (2018) :

« *La science, contrairement à ce que l'on dit parfois, ce ne sont pas des réponses triviales à des questions complexes ; c'est toujours une invitation à la subtilité.* »

Le mathématicien Mickaël Launey (chaîne YouTube *Micmaths*) écrit de même en épilogue de son livre *Le grand roman des maths* (Flammarion, 2018) :

« *C'est une chose qu'il faut accepter dès que l'on fait de la science :*
plus on en sait sur un sujet, plus on mesure l'étendue de notre ignorance. »

Nous connaissons de grandes théories et de beaux théorèmes, très élégants et profonds, mais il y a tellement de questionnements qui restent à élucider ! Cette idée d'une quête de l'impossible, d'un combat perdu d'avance, c'est un peu ma vision de la science informatique. Mais attention, je trouve pour autant le voyage extrêmement enthousiasmant, et l'on progresse ! Bien que j'aie la conviction que l'on ne saura jamais « tout », comprendre et partager de nouvelles connaissances est un réel plaisir, et le vocabulaire de l'esthétisme que nous employons naturellement en est un témoignage.

Les raisonnements de Kurt Gödel sur la longueur des preuves en 1936 [Göd36 ; Por10] soutiennent formellement ces propos : il existe des questions simples (courtes) dont les réponses sont très très très complexes (longues). Il ne s'agit pas d'une observation, mais bien d'un théorème démontré mathématiquement : pour tout système formel contenant au moins l'arithmétique, la fonction $\mathcal{L} : \mathbb{N} \to \mathbb{N}$ qui à n associe la taille de la plus longue des plus courtes preuves des énoncés de taille n, croît plus vite que toute fonction calculable. Les conjectures de Collatz-Syracuse [a] (1937) et Goldbach [b] (1742) en sont d'excellentes illustrations : malgré leur apparente simplicité et de très importants efforts concrétisés par quantité d'articles scientifiques, aucune

a. La suite $n \mapsto \begin{cases} n/2 & \text{si } n \text{ est pair} \\ 3n+1 & \text{si } n \text{ est impair} \end{cases}$ converge-t-elle vers le cycle $1, 4, 2$ pour tout $n \in \mathbb{N}$?

b. Est-ce que tout nombre entier pair ≥ 3 peut s'écrire comme la somme de deux nombres premiers ?

démonstration ni contre-exemple n'ont pour l'heure été trouvés. Ainsi, on doute que ces conjectures puissent être démontrées ou réfutées par des raisonnements d'une simplicité proportionnée à leur énoncé.

L'informatique théorique nous invite donc à la prudence et à la délicatesse. Étant à la frontière entre l'informatique et les mathématiques, elle offre néanmoins un cadre formel qui permet d'énoncer des vérités claires et indiscutables (théorèmes et démonstrations), et c'est une chance de pouvoir se reposer sur un socle solide et inamovible (matéralisé par les articles et ouvrages de la littérature scientifique). Nous nous abstiendrons ici de nous lancer dans de longues discussions sur les fondements des mathématiques, mais nous encourageons chaleureusement le·a lect·eur·rice intéressé·e à parcourir la captivante bande dessinée *Logicomix* [Dox+10].

1.1. Complexité algorithmique et calcul naturel

La thèse que je soutiens dans le présent document est la suivante : la complexité algorithmique est un outil pertinent pour discuter formellement de la « complexité » des systèmes « complexes », et celle-ci repose sur leur capacité à calculer.

Notre sujet d'étude sera les réseaux d'automates, qui sont des systèmes « complexes » inspirés par des phénomènes naturels, dans lesquels plusieurs entités interagissent localement pour donner une dynamique globale. La notion de calcul jouera un rôle essentiel, car on constate que la propriété d'être capable de calculer est omniprésente dans de tels modèles [Moo90 ; Lan90 ; Kau90]. La thèse de Church-Turing postule que tout mécanisme « raisonnable » permet au mieux de calculer tout ce qu'un ordinateur (une machine de Turing) peut calculer, mais jamais davantage. Elle n'a pas été contredite depuis l'invention des premières machines à calculer universelles dans les années 1930, mais une myriade d'exemples la soutenant ont été découverts indépendamment les uns des autres (voir par exemple [MM11, chapitre 7.6 Computation Everywhere]).

La complexité algorithmique s'intéresse à la résolution de problèmes, et au meilleur algorithme pouvant résoudre un problème donné. Il y a plusieurs façons de définir le sens des mots « meilleur » et « résoudre », la plus commune étant le plus *rapide* et qui donne toujours la réponse *exacte*. Les premières questions théoriques posées à sa genèse dans les années 1970, qui sont au cœur de la théorie, restent ouvertes. On pense bien évidement à la distinction entre classes de complexité, dont l'emblématique [c] : est-ce que P (l'ensemble des problèmes pour lesquels on peut *trouver* efficacement une solution) est égale à NP (l'ensemble des problèmes pour lesquels on peut *vérifier* efficacement si une solution donnée est correcte) ? Il est surprenant d'imaginer que trouver une solution ne soit pas fondamentalement plus difficile que vérifier si une solution donnée est correcte [d], mais la question est toujours ouverte : on ne sait pas.

c. « Efficacement » signifie *en temps polynomial en la taille de l'entrée* (thèse de Cobham [Cob65]).

d. Prenons l'exemple du jeu Sudoku : compléter par nous-même une grille semble fondamentalement plus difficile que de vérifier une grille déjà complétée par notre voisin...

La démarche scientifique, c'est aussi ne pas s'arrêter aux obstacles mais les englober dans nos réflexions pour faire progresser nos connaissances, et les notions de réduction et de complétude en théorie de la complexité algorithmique en sont de très bons exemples. Pour des problèmes dont on sait vérifier efficacement une solution (dans NP), on aimerait démontrer qu'il est impossible de trouver efficacement une solution (pas dans P). Face à ce verrou (qui distinguerait P et NP), a été inventée la notion de *réduction* (inspirée de la théorie de la calculabilité) qui permet de comparer la complexité de deux problèmes $\mathcal{A}$ et $\mathcal{B}$, et dire formellement :

« $\mathcal{A}$ est au moins aussi difficile que $\mathcal{B}$ ».

Vient ensuite la *complétude* (ou *difficulté*) qui permet de dire formellement :

« $\mathcal{A}$ est au moins aussi difficile que tous les autres problèmes ».

La quantification universelle du « tous les autres problèmes » est extrêmement forte, et la graine (théorème de Cook-Levine) est un résultat superbe. Dit autrement, si l'on découvre comment résoudre le problème $\mathcal{A}$ efficacement, alors on en déduit immédiatement une façon de résoudre *tous* les autres problèmes efficacement. Il s'agit d'un moyen d'énoncer quelque chose de pertinent sur la difficulté des problèmes (former une sous-famille des problèmes « les plus difficiles » parmi les problèmes de NP que nous ne savons pas résoudre efficacement, mais pour lesquelles nous ne savons pas démontrer qu'il est impossible de les résoudre efficacement), sans pour autant arriver à montrer que les classes P et NP sont distinctes. Garey et Johnson en donnent de sympathiques illustrations en introduction de leur livre [GJ79], conclues par un employé pouvant dire à son patron :

« *I can't find an efficient algorithm, but neither can all these famous people.* »

Ces raisonnements sur P et NP, les réductions polynomiales $\leq_{\mathrm{T}}^{\mathsf{P}}$, la NP-difficulté et NP-complétude, s'adaptent aux autres classes de complexité.

Appliquées à la complexité des systèmes dynamiques (en particulier des réseaux d'automates), les réductions montrent de fait comment embarquer du calcul, car pour une classe C le problème C-complet canonique est de la forme :

Étant donnés un algorithme $\mathcal{M}$ dans C, une entrée u et un temps de calcul $t \in \mathbb{N}$, est-ce que $\mathcal{M}$ accepte u en au plus t étapes ?

De la définition de la classe C dépendra le type d'algorithme et l'encodage de t (unaire, binaire, *etc.*, ou d'autres mesures que le temps), voir par exemple [Per14, proposition 3-M page 68]. Une réduction depuis *n'importe quel* problème C-difficile donnera par transitivité une réduction depuis ce problème canonique, c'est-à-dire une façon efficace d'encoder du calcul dans l'évolution du système considéré (du moins pour le type de questions que nous allons étudier, essentiellement liées à la dynamique).

1.2. Réseaux d'automates

Les réseaux d'automates sont des systèmes dynamiques discrets finis (la dynamique entière peut être calculée) sur une structure d'interactions locales. Dans le cas booléen, ils sont composés de n automates possédant chacun un état dans $\{0,1\}$, dont le comportement est décrit par des fonctions locales $f_i : \{0,1\}^n \to \{0,1\}$ indiquant l'état suivant de chaque automate $i \in [n]$ en fonction de l'état des autres automates, et dont la dynamique globale est $f : \{0,1\}^n \to \{0,1\}^n$ avec $\pi_i \circ f = f_i$ où $\pi_i : \{0,1\}^n \to \{0,1\}$ est la projection sur la i-ème composante. Il s'agit d'un modèle très expressif, qui possède plusieurs variantes importantes lorsqu'on généralise l'alphabet (de $\{0,1\}$ à $[\![q]\!]$) et l'ordre de mise à jour des automates (de déterministe synchrone donné par f, à une permutation de $[n]$, une partition ordonnée de $[n]$, une séquence de sous-ensembles de $[n]$, ou des modes non déterministes dont l'asynchrone parfait...).

Une motivation importante pour leur étude est la modélisation des processus de régulation génétique : dans une cellule, chaque gène est un automate qui peut être exprimé ou non (états $\{0,1\}$), et dont l'état évolue en fonction de l'état de certains autres automates du réseau. Les configurations stables (en particulier les points fixes) correspondent alors à des types de cellules : les mécanismes de différenciation cellulaire peuvent ainsi être capturés. La réalité biologique n'est bien entendu pas aussi élémentaire que ce modèle abstrait [Pau+20]. Cette approche sera discutée en introduction du chapitre 7. Dans le présent document, nous considérerons avant tout les réseaux d'automates comme un modèle de calcul *per se*, sans visée applicative.

Un intérêt marqué a été porté sur le graphe d'interaction des réseaux d'automates, qui capture les dépendances entre les entités du système. Pour un réseau $f : \{0,1\}^n \to \{0,1\}^n$ sur n automates, on aura n sommets et un arc du sommet i vers le sommet j lorsque la fonction locale de l'automate j dépend effectivement de l'état de l'automate i. Le graphe d'interaction donne une information partielle sur le réseau : la structure des échanges d'information entre automates (ces échanges étant binaires dans le cas booléen, avec un bit de mémoire pour chaque automate). Les relations entre ce graphe noté G_f, et la dynamique de f dont le graphe est noté $\mathcal{G}_f$, sont aux fondements des études sur le modèle (une revue bibliographique est proposée à la suite des définitions au chapitre 3.6). Observons que si le graphe d'interaction est complet (tous les arcs) alors les informations sont échangées globalement, alors qu'un graphe d'interaction dont les degrés sont bornés force une localité dans les échanges.

Le modèle des réseaux d'automates est très général : toute fonction ou relation sur $\{0,1\}^n$ est la dynamique d'un réseau. Cependant l'espace-temps est fini, et l'on navigue à l'intérieur d'un monde calculable. La théorie de la complexité algorithmique semble ainsi une lunette adéquate pour en cerner les contours. Notre approche dans ce document consistera à proposer une multitude de points de vues sur la complexité algorithmique des réseaux d'automates, qui mettront en lumière différents éléments qui les constituent. L'ensemble de ces résultats offrira une certaine compréhension globale du modèle, dont la discussion sera poursuivie en conclusion.

1.3. Plan détaillé

Des perspectives seront proposées progressivement en fin des chapitres.

Chapitre 2 Préliminaires. Nous commencerons par rappeler quelques considérations sur les modèles de calcul (**RAM**) et les circuits booléens. Nous introduirons les classes de complexité rencontrées dans nos études, de L à NEXPTIME en passant par les classiques P, NP, les niveaux Σ_i^P, Π_i^P de la hiérarchie polynomiale PH, ainsi que US (solution unique) et DP (entre NP, coNP et Δ_2^P). Ce sera également l'occasion de rappeler quelques lieux communs, et la quasi-absence de distinctions connues entre ces classes. Ces préliminaires présenteront également les réductions, problèmes de comptage (classes #P et $\#\cdot\mathsf{OptP}[\log n]$), et promesses.

Chapitre 3 Modèles des réseaux d'automates. Il s'agira d'introduire les différentes variantes du modèle suivant les alphabets et modes de mise à jour considérés. Nous définirons le graphe d'interaction (signé), le graphe de la dynamique (c'est-à-dire le graphe de transition du système, déterministe ou non déterministe), et les notions de dynamique limite, points fixes et cycles limites. Nous porterons une attention particulière à des aspects de bas niveau relatifs aux problèmes de décision : comment un réseau d'automates est-il encodé ? Ce sera l'occasion de constater que certains problèmes qui semblent élémentaires (comme le calcul des images d'une configuration) apportent déjà leur lot de subtilités. Nous donnerons plusieurs exemples en fin de chapitre (pages 41 à 45), et énoncerons de beaux résultats de la littérature traditionnelle, qui s'intéressent aux liens structurels entre le graphe d'interaction et la dynamique.

Chapitre 4 Calculer le graphe d'interaction G_f. La première contribution que nous présenterons concerne le calcul du graphe d'interaction d'un réseau d'automates donné. Des développements sans grande difficulté sur l'effectivité des dépendances nous mèneront à la classe DP. Ainsi, connaître le graphe d'intéraction n'est pas une information négligeable, et peut nécessiter la formulation d'une promesse.

Chapitre 5 Complexité de la dynamique asymptotique I : f en entrée. Nous parcourerons dans ce chapitre les questions typiques posées sur tout modèle de calcul, concernant sa dynamique asymptotique : existence et comptage des points fixes et cycles limites. Les problèmes liés aux points fixes bénéficient de constructions très claires qui nous feront voyager à tous les niveaux de PH, même sous la restriction d'un graphe d'interaction de degré entrant maximum $\Delta(G_f) \leq 2$. Le calcul des préimages (dynamique en arrière) sera l'objet de résultats nouveaux, et l'atteignabilité nous fera atteindre PSPACE. L'expression d'un lemme technique de simulation d'une machine de Turing par un réseau d'automates uniforme déterministe permettra de dériver la PSPACE-difficulté de plusieurs problèmes au sujet de la dynamique limite, dont savoir si l'ensemble limite est de taille 1 (nilpotence). Nous examinerons également quelques problèmes de décision coNP-complets, reliés à des familles très contraintes (décider si un réseau donné est bijectif, l'identité, ou une constante).

Chapitre 6. Complexité des questions FO. De nombreuses questions sont exprimables en logique du premier ordre (dont presque toutes celles du chapitre 5), et nous

présenterons dans ce chapitre un théorème de difficulté « à la Rice » : toute question exprimable au premier ordre sur la dynamique d'un réseau d'automates donné est soit triviale (répondue en $\mathcal{O}(1)$), soit NP- soit coNP-difficile. Partant d'une formule au premier ordre ψ, nous emploierons plusieurs techniques de théorie des modèles afin d'exhiber des éléments de dynamique (sous-graphes de la dynamique $\mathcal{G}_f$), qui sont assemblés en temps polynomial pour former un réseau d'automates implémentant une réduction depuis **SAT**. Nous verrons en outre que tous les niveaux de PH sont atteignables au premier ordre, et proposerons des perspectives sur la logique monadique du second ordre (avec un surprenant graphe Λ_m).

Chapitre 7 Complexité de la dynamique asymptotique II : G_f en entrée. La majorité des problèmes de décision étudiés prennent en entrée un réseau d'automates, mais que se passe-t-il si l'on connaît uniquement le graphe d'interaction du réseau ? Dans ce chapitre, nous présenterons de premiers éléments sur cette question, à travers l'étude des nombres maximum et minimum de points fixes que l'on peut obtenir pour un graphe d'interaction donné. Les constructions proposeront un regard nouveau car il faudra en permanence considérer toutes les fonctions locales possibles sur un graphe d'interaction donné. Pour contrôler la dynamique, on utilisera de petits degrés, et des liens entre points fixes et *feedback vertex sets*. Des restrictions nous feront visiter P, NP, $\mathsf{NP}^{\mathsf{NP}}$, $\mathsf{NP}^{\#\mathsf{P}}$, pour finalement atteindre NEXPTIME dans le cas le plus difficile.

Chapitre 8 Complexité des mises à jour. Comment varie la complexité des problèmes pour différents modes de mise à jour ? Nous porterons une attention particulière aux modes de mise à jour bloc-séquentiels, et aux problèmes de cycle limite de longueur au moins 2 (les points fixes sont invariants). Viendra ensuite une question naturelle de comptage #P-complète : combien de modes bloc-séquentiels non équivalents un réseau donné admet-il ? Et une étude de quelques cas polynomiaux obtenus par décomposition (diviser pour régner). Le paysage est très vaste et nous proposerons plusieurs perspectives à court et moyen termes.

Annexe A Solution unique $\exists!$ et PH. Au chapitre 3 nous rencontrerons la classe US (solution unique) et le quantificateur $\exists!$, pour un problème équivalent à $\forall\exists!$**-SAT** (est-ce que $\forall x : \exists! y : \varphi(x, y) = \top$?). Pour quelle classe ce problème est-il complet ?

Annexe B Classes au dessous de P. Dans les petites classes de complexité, les détails sont importants. Nous discuterons dans cette annexe de la robustesse de NC, et du modèle **CREW PRAM** de calcul parallèle (avec des exemples d'algorithmes). Nous rappellerons quelques éléments sur la P-complétude et les variantes de **CVP**.

Nomenclature.

- **Théorème** pour les résultats de la littérature.
- **Théorème** pour les résultats publiés auxquels j'ai participé.
- **Théorème** pour les résultats nouveaux inclus dans ce document.

1.4. Encadrements doctoraux

Cédric Bérenger (thèse soutenue en 2021) http://www.theses.fr/2021AIXM0024
Co-encadrement avec Peter Niebert (LIS, Marseille).

Sujet : *Grands réseaux maillés basse énergie : protocoles minimalistes pour la synchronisation, la mesure de distance et le partitionnement.*

Résumé : Les microcontrôleurs, ces petits ordinateurs embarqués à bas coût, nous permettent aujourd'hui de voir grand. Leur faible consommation ainsi que leur équipement leur permettent d'interagir avec l'environnement, tout en nous communiquant leurs agissements via les ondes. Ainsi, pourquoi ne pas former un grand maillage, pour administrer et surveiller une maison, un complexe industriel, voire une ville entière ? Un tel passage à l'échelle s'avère difficile, et même si de nombreuses solutions existent déjà pour la gestion de réseaux maillés, les impératifs de robustesse et d'efficacité ont poussé à faire certains compromis, relayant la simplicité d'implémentation, la basse consommation, ainsi que l'idéal d'un réseau maillé aux communications entièrement pair à pair, à un second plan. Dans cette thèse, nous contribuons à une solution alternative pour laquelle nous réalisons nos propres compromis : nous cherchons une méthode minimaliste permettant de construire des réseaux maillés à grande échelle, autonomes en énergie et capable de s'auto-gérer sans coordinateur.

Pacôme Perrotin (thèse soutenue en 2021) http://www.theses.fr/2021AIXM0011
Co-encadrement avec Sylvain Sené (LIS, Marseille).

Sujet : *Simulation entre modèles de calcul naturel et modularité des réseaux d'automates.*

Résumé : Nous explorons différentes généralisations concernant les modèles de calcul naturel. La plus théorique est la notion de simulation entre modèles, pour laquelle nous décrivons une série de propositions de définition, en discutant des intérêts et des failles de chacune d'elles. Nous profitons des définitions les plus prometteuses pour élargir le propos sur les possibles conséquences de la simulation en théorie de la complexité, comme la construction de nouvelles classes de complexité en proposant la substitution de la réduction polynomiale par la simulation.

Notre approche plus appliquée consiste en la généralisation des réseaux d'automates sous formes de modules, qui possèdent des entrées. Ce formalisme permet d'approcher les questions de la dynamique des réseaux d'interactions sous un nouvel angle : nous explorons son utilité en tant qu'outil modulaire propre à simuler de façon flexible de nombreux objets similaires, ainsi que l'expressivité des modules acycliques. Ceux-ci permettent la caractérisation de la dynamique des réseaux d'automates sous la forme de fonctions de sortie. Cette expressivité nous autorise la description d'un processus d'optimisation de réseaux d'automates, qui réduit certains réseaux en taille tout en conservant des attracteurs équivalents.

2. Préliminaires

Dans ce chapitre, nous présentons les notions de base employées dans la suite du document. La complexité algorithmique a trait aux ressources (le temps et l'espace pour nos considérations) nécessaires à la résolution de problèmes. Ainsi, la définition des classes de complexité repose sur la définition de modèles de calcul. Étant donnée la subtilité des questions ouvertes de théorie de la complexité que nous toucherons, il nous semble nécessaire de discuter brièvement en section 2.1 des modèles de calcul. Ce sera l'occasion en particulier de présenter les circuits booléens (section 2.1.2). La section 2.2 présente les classes de complexité rencontrées dans nos études.

2.1. Modèles de calcul

Dans la suite du document, un *algorithme* sera un algorithme séquentiel pour machine de Turing ou **RAM**, *déterministe* ou *non-déterminste*. Les *algorithmes parallèles* pour **CREW PRAM** (*déterministes* uniquement), sont discutés en annexe B.

2.1.1. Modèles séquentiels (RAM)

Pour notre propos, nous pourrons nous satisfaire de la notion intuitive d'un *algorithme* séquentiel (*déterministe* ou *non déterministe*), et présenterons de tels algorithmes sous la forme de pseudo-code ou simplement d'une description de la procédure en français. Les modèles formels sous-jacents sont les *machines de Turing* (**MT**) ou le modèle *Random Access Machine* (**RAM**) [Sip05]. Les classes de complexité de calcul séquentiel que nous considérons (P et au-dessus) sont suffisamment robustes aux détails du modèle de calcul pour qu'une définition précise ne soit pas requise ; elle sont dites « *model-independent* » [Emd90]. Nous nous en tiendrons à une citation de van Emde Boas : « *in the Turing machine model all possible types of instructions have been fused into a single read-test-write-move-goto instruction ; for models like the RAM the instruction set resembles a primitive assembly code* » (affectation, bloc d'instructions, conditionnelle, boucles). Pour les classes de complexité de la hiérarchie polynomiale, certains algorithmes seront équipés d'un *oracle* : une instruction supplémentaire consiste à appeler une « boîte noire » (l'oracle) qui répond en temps constant sur des instances d'un problème appartenant à (et généralement complet pour) la classe de complexité de l'oracle.

Une considération importante est la suivante : dans nos algorithmes, nous supposerons que les opérations de base (telles que l'addition et la soustraction) sont

effectuées en temps constant. Bien que, pour une **MT** à alphabet fixé et une **RAM** à taille de mot donnée, additionner et soustraire deux nombres entiers de valeurs $\mathcal{O}(n)$ prenne un temps $\mathcal{O}(\log n)$ (proportionnel à leur taille), dans la pratique il est d'usage d'effectuer cette simplification. Le point de vue que nous adoptons s'appelle *critère de coût uniforme* et est traditionnellement associé au modèle **RAM**, par opposition au *critère de coût logarithmique* associé au modèle **log-cost RAM**.

2.1.2. Circuits booléens

Les circuits booléens vont intervenir dans ce manuscrit comme encodage d'une fonction $\{0,1\}^n \to \{0,1\}$ dans un réseau d'automates. Ils sont également utilisés comme modèle de calcul pour la définition des étages NC^i à l'interieur de NC (annexe B). Un *circuit booléen* est un graphe orienté **acyclique** [a] tel que :

- les sommets de degré entrant 0 sont appelés *entrées* et sont étiquetés par le nom d'une variable x_i ou par la constante 0 ou 1,
- les autres sommets sont des *portes* étiquetées parmi $\{\neg, \wedge, \vee\}$. Précisons que $\neg$ n'est possible que pour les portes de degré entrant 1 (unaires), et que nous considérerons par défaut des circuits de degré entrant au plus 2 (on peut adjoindre toutes les portes d'arité au plus deux [b]),
- les sommets de degré sortant 0 sont appelés *sorties*.

Étant donnée une entrée dans $\{0,1\}^n$ qui assigne un bit à chacun des n sommet d'entrées, le circuit opère un calcul (dont la sémantique est usuelle, voir par exemple [Per14, Définition 5-Q page 135]) assignant un bit à chaque sommet de sortie. Nous aurons en général un seul sommet de sortie, donc un circuit encodera une fonction de $\{0,1\}^n$ dans $\{0,1\}$. La *taille* d'un circuit est le nombre de sommets du graphe, et sa *profondeur* est la longueur maximale d'un chemin d'une entrée à une sortie. Une *formule propositionnelle* se traduit très naturellement en un circuit, en identifiant les feuilles correspondant à une même variable dans son arbre syntaxique.

Pour passer un circuit en entrée d'un problème (que ce soit pour l'évaluer ou pour décrire une fonction locale de réseau d'automates), il faut l'encoder. Ceci requiert un espace quadratique en le nombre de sommets du graphe (ou linéaire avec des listes d'adjacence de voisins entrants puisque le degré entrant du graphe est borné). Nous aborderons au chapitre 5 les encodages succincts (pour atteindre NEXPTIME), dans lesquels un circuit encode un objet comme une formule propositionnelle.

Les circuits booléens sont également un modèle de calcul permettant de résoudre des problèmes de décision et plus généralement de calculer des fonctions, dont les entrées et sorties sont encodées en binaire. Dans ce cadre on parle de *non uniformité*,

a. L'acyclicité offre une notion claire de déroulement du calcul, depuis les entrées (sources) vers les sorties (puits). Un circuit qui n'est pas acyclique est un réseau d'automates, l'objet central du présent document (définition au chapitre 3).

b. Autoriser des portes, comme le « xor » ou « ou exclusif » noté $\oplus$, d'arité arbitraire pourrait induire des écueils car leur traduction en termes de portes $\neg, \wedge, \vee$ peut nécessiter une taille exponentielle.

car pour résoudre *un* problème ou calculer *une* fonction, on doit donner une *famille* de circuits : un circuit pour chaque taille d'entrée (ainsi le problème de l'arrêt est résolu par une famille non uniforme de circuits, car pour chaque taille d'entrée il existe un circuit correct, qui est bien sûr impossible à produire algorithmiquement). Afin d'obtenir effectivement, étant donnée une taille d'entrée n, le circuit correspondant, il est fait appel à un modèle de calcul séquentiel : une famille de circuits $(C_n)_{n\in\mathbb{N}}$ est dite *uniforme* ou P*-uniforme* (*resp.* L-uniforme) lorsque le code du circuit C_n peut être produit en temps polynomial (*resp.* espace logarithmique) en n (encodé en unaire) par une **MT** ou **RAM**. Les liens entre le modèle de calcul des circuits booléens et les **PRAM** (*Parallel Random Access Machines*) sont discutés en annexe B.2.

2.2. Complexité algorithmique

Nous présentons brièvement le bestiaire qui permettra de classifier les problèmes abordés. Pour des définitions précises, nous redirigeons le lecteur vers l'excellent ouvrage de Perifel [Per14] (en français et accès libre depuis la page personnelle de l'auteur), et pour de plus amples informations vers l'incontournable *Complexity Zoo* [Zoo].

Nous aborderons principalement des *problèmes de décision* (sections 2.2.1 et 2.2.2), mais également quelques *problèmes fonctionnels* (section 2.2.3) et des *problèmes de comptage* (sections 2.2.4 et 2.2.5). Dans cette section, la lettre n désignera comme il est d'usage la taille de l'entrée (attention, à partir du chapitre 3 la lettre n désignera le nombre de composantes de nos réseaux d'automates). Nous suivrons les conventions du Perifel : les problèmes de décision sont des langages définis comme l'ensemble des instances positives, et les classes de complexité sont des ensembles de problèmes. Aussi, la vérification de la validité du codage de l'entrée sera implicite dans nos algorithmes ; dans le cas où cette vérification n'est pas triviale, nous l'expliciterons, ou bien nous ferons une promesse (section 2.2.6).

2.2.1. Problèmes de décision et classes de complexité

Les *problèmes de décision* ont une réponse en 1 bit d'information : « oui » ou « non », « accepte » ou « rejette ». La classe P est charnière dans la théorie de la complexité. Selon la *thèse de Cobham-Edmonds*, elle caractérise les problèmes « faciles » à résoudre [Cob65]. Au dessus, nous avons les classes de problèmes « difficiles » (en anglais, on utilise l'adjectif *intractable* [GJ79]).

Les classes sont présentées dans la table 2.1, avec la notation usuelle des oracles en exposant. Les classes $\#\mathsf{P}$ et $\#\cdot\mathsf{OptP}[\log n]$ sont définies en section 2.2.4, la classe US est discutée en section 2.2.6, l'intuition derrière la classe DP sera exposée au chapitre 4, et le modèle **CREW PRAM** de calcul parallèle est présenté en annexe B. Nous considérerons trois notions de *réduction* usuelles entre problèmes de décision : « *many-one* » où une instance est transformée en *une* instance de même décision, « *truth-table* » où une instance est transformée en *un nombre fixé* d'instances dont

Classe	Description (algorithme, pour $k \in \mathbb{N}$ fixé)
NEXPTIME	Séquentiel non déterministe, en temps $\mathcal{O}(k^n)$
EXPTIME	Séquentiel déterministe, en temps $\mathcal{O}(k^n)$
PSPACE	Séquentiel déterministe, en espace $\mathcal{O}(n^k)$
$\mathsf{NP}^{\#\mathsf{P}}$	Séquentiel non déterministe avec oracle #P, en temps $\mathcal{O}(n^k)$
PH	Hiérarchie polynomiale, $\mathsf{PH} = \bigcup_{i\in\mathbb{N}} \Sigma_i^{\mathsf{P}}$
Σ_i^{P} avec $i \in \mathbb{N}$	Niveau i de la hiérarchie polynomiale, défini inductivement par $\Sigma_0^{\mathsf{P}} = \mathsf{P}$ et $\Sigma_{i+1}^{\mathsf{P}} = \mathsf{NP}^{\Sigma_i^{\mathsf{P}}}$ (algorithme NP avec oracle Σ_i^{P})
Π_i^{P} avec $i \in \mathbb{N}$	$\Pi_0^{\mathsf{P}} = \mathsf{P}$ et $\Pi_{i+1}^{\mathsf{P}} = \mathsf{coNP}^{\Pi_i^{\mathsf{P}}} = \mathsf{coNP}^{\Sigma_i^{\mathsf{P}}}$ (complémentaires des problèmes dans Σ_i^{P})
Δ_i^{P} avec $i \in \mathbb{N}$	$\Delta_1^{\mathsf{P}} = \mathsf{P}$ et $\Delta_{i+1}^{\mathsf{P}} = \mathsf{P}^{\Sigma_i^{\mathsf{P}}} = \mathsf{P}^{\Pi_i^{\mathsf{P}}}$
$\mathsf{NP}^{\mathsf{NP}} = \Sigma_2^{\mathsf{P}}$	Séquentiel non déterministe avec oracle NP, en temps $\mathcal{O}(n^k)$
$\mathsf{P}^{\mathsf{NP}} = \Delta_2^{\mathsf{P}}$	Séquentiel déterministe avec oracle NP, en temps $\mathcal{O}(n^k)$
DP	L'ensemble des $L = L_1 \cap L_2$ avec $L_1 \in \mathsf{NP}$ et $L_2 \in \mathsf{coNP}$
US	Séquentiel non déterministe qui accepte ssi le calcul a exactement une branche acceptante, en temps $\mathcal{O}(n^k)$
$\mathsf{coNP} = \Pi_1^{\mathsf{P}}$	Problèmes dont le complémentaire est dans NP
$\mathsf{NP} = \Sigma_1^{\mathsf{P}}$	Séquentiel non déterministe, en temps $\mathcal{O}(n^k)$
$\mathsf{P} = \Sigma_0^{\mathsf{P}} = \Pi_0^{\mathsf{P}}$	Séquentiel déterministe, en temps $\mathcal{O}(n^k)$
NC	Parallèle, en temps $\mathcal{O}((\log n)^k)$
NL	Séquentiel non déterministe, en espace $\mathcal{O}(\log n)$
L	Séquentiel déterministe, en espace $\mathcal{O}(\log n)$
$\#\cdot\mathsf{OptP}[\log n]$	Comptage du nombre de solutions minimum d'un problème dans NP
#P	Comptage du nombre de solutions (ou certificats ou branches acceptantes) d'un problème dans NP

Table 2.1. – Classes de complexité, et leur définition par les modèles de calcul et algorithmes résolvant les problèmes appartenant à chaque classe. Il s'agit de complexités *dans le pire cas* pour la résolution *exacte* des problèmes.

on prend la combinaison des décisions selon une table de vérité, et « *Turing* » où le problème d'arrivée est un oracle pour résoudre le problème de départ. Nous noterons :

$\leq_{\mathrm{m}}^{\mathsf{C}}$ les réductions « *many-one* » [c] dans FC,
$\leq_{\mathrm{tt}}^{\mathsf{C}}$ les réductions « *truth-table* » dans FC,
$\leq_{\mathrm{T}}^{\mathsf{C}}$ les réductions « *Turing* » dans C avec oracle,

où FC désigne la version fonctionnelle de la classe C, c'est-à-dire avec potentiellement plus d'un bit de sortie (voir la section 2.2.3). Les notions de D-*difficulté* et D-*complétude* prendront sens avec :

$\mathsf{C} = \mathsf{P}$	pour D au dessus de P (« exclue »),
$\mathsf{C} = \mathsf{NC}$ ou $\mathsf{C} = \mathsf{L}$	pour $\mathsf{D} = \mathsf{P}$.

On a bien sûr $A \leq_{\mathrm{m}}^{\mathsf{P}} B \implies A \leq_{\mathrm{tt}}^{\mathsf{P}} B \implies A \leq_{\mathrm{T}}^{\mathsf{P}} B$ pour tous problèmes A et B, mais pour un alphabet fini Σ, on a $\Sigma^* \not\leq_{\mathrm{m}}^{\mathsf{P}} \emptyset$ alors que $\Sigma^* \leq_{\mathrm{tt}}^{\mathsf{P}} \emptyset$ et $\Sigma^* \leq_{\mathrm{T}}^{\mathsf{P}} \emptyset$. La distinction des notions de NP-difficulté (et C-difficulté pour d'autres classes C) selon ces trois notions de réduction est ouverte [GHP10 ; MPV14], bien que sans restriction sur le temps de calcul (c'est-à-dire du point de vue de la calculabilité), leurs propriétés diffèrent [Odi92, théorème VI.5.7 page 500]. Pour la NP-difficulté selon chacune des trois réductions $\leq_{\mathrm{m}}^{\mathsf{P}}, \leq_{\mathrm{tt}}^{\mathsf{P}}, \leq_{\mathrm{T}}^{\mathsf{P}}$, nous avons bien l'observation fondamentale suivante : si un problème NP-difficile est dans P, alors P = NP.

2.2.2. Éléments de littérature

La figure 2.1 présente les inclusions connues entre les classes de la table 2.1 [d] (voir l'annexe B pour AC^0, NC^i et NC). Très peu de distinctions entre ces classes sont connues (la plupart sont conjecturées), en voici quelques unes :

- $\mathsf{P} \subsetneq \mathsf{EXPTIME}$ et $\mathsf{NP} \subsetneq \mathsf{NEXPTIME}$ par les théorèmes de hiérarchie en temps [Per14, théorèmes 2-J page 36 et 2-AI page 52],
- $\mathsf{AC}^0 \subsetneq \mathsf{NC}^1$ par le calcul de la parité d'une entrée binaire qui est dans NC^1 mais pas dans AC^0 [AB09, théorème 13.1 page 235] (AC^0 contient l'addition),
- $\mathsf{NC}^1 \subsetneq \mathsf{PSPACE}$ par le théorème de hiérarchie en espace (distinguant $\mathsf{NL} \subsetneq \mathsf{NPSPACE}$ [Per14, théorème 4-Q page 104]) combiné au théorème de Savitch (ayant pour conséquence $\mathsf{PSPACE} = \mathsf{NPSPACE}$ [Per14, corollaire 4-AU page 120]) et à l'observation que $\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2$.

Ainsi, les effondrements des hiérarchies Σ_i^{P} et NC^i à partir d'un niveau $i \in \mathbb{N}$ (c'est-à-dire respectivement $\Sigma_i^{\mathsf{P}} = \Sigma_j^{\mathsf{P}}$ et $\mathsf{NC}^i = \mathsf{NC}^j$, pour tout $j \geq i$) sont ouvertes, tout comme la question suivante qui me fait parfois rougir de tant aimer toute cette théorie...

c. Ce sera la notion de réduction utilisée par défaut, sauf mention contraire.

d. Au sujet de $\mathsf{NP}^{\#\mathsf{P}}$, le *théorème de Toda* [Per14, théorème 9-AI page 226] établit que $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$, et l'on a bien sûr $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{NP}^{\#\mathsf{P}}$, d'où $\mathsf{PH} \subseteq \mathsf{NP}^{\#\mathsf{P}}$. Puisque $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{PSPACE}$ (folklore) et $\mathsf{PSPACE} = \mathsf{NPSPACE}$ (conséquence du théorème de Savitch [Per14, corollaire 4-AU page 120]), nous avons $\mathsf{NP}^{\#\mathsf{P}} \subseteq \mathsf{PSPACE}$. Lorsque cette classe interviendra au chapitre 7, la formulation d'un problème complet fera intervenir la classe PP, car $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}}$ (folklore) d'où l'on déduit $\mathsf{NP}^{\#\mathsf{P}} = \mathsf{NP}^{\mathsf{PP}}$.

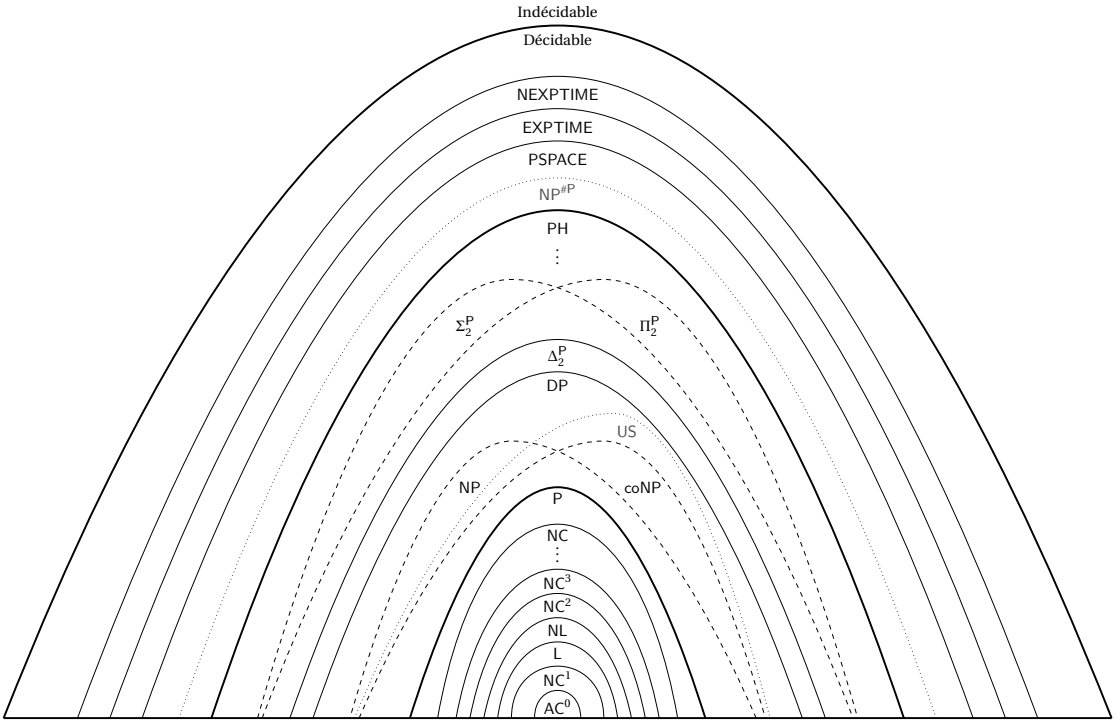


FIGURE 2.1. – Diagramme de Venn (inclusions ensemblistes) des classes de complexité de la table 2.1, *aka* un morceau du « *complexity cake* ».

Question ouverte 2.1 ([AB09, section 6.5.2])**.** *Est-ce que* $\mathsf{PH} = \mathsf{NC}^1$ *?*

En revanche, plusieurs résultats pertinents reposent sur des énoncés hypothétiques, dont voici une sélection :

- $\mathsf{P} = \mathsf{NP}$ si et seulement si $\mathsf{P} = \mathsf{PH}$ [Per14, proposition 8-C page 195],
- si $\mathsf{NP} = \mathsf{coNP}$, alors $\mathsf{NP} = \mathsf{PH}$ [Per14, proposition 8-D page 195],
- si $\mathsf{P} \neq \mathsf{NP}$, alors il existe des problèmes de complexités intermédiaires entre P et NP (dans NP mais ni NP-complets ni dans P), c'est le *théorème de Ladner* [Per14, théorème 3-AK page 86],
- si $\mathsf{P} \neq \mathsf{NP}$, alors $\mathsf{NP} \setminus \mathsf{P}$ n'est pas récursivement présentable [Sch82, corollaire 3], c'est-à-dire qu'il n'existe pas d'énumération calculable de machines de Turing $M_1, M_2, \ldots$ telles que $\{L(M_i) \mid i \in \mathbb{N}\} = \mathsf{NP} \setminus \mathsf{P}$,
- si $\mathsf{NC} \neq \mathsf{P}$, alors il existe des problèmes de complexités intermédiaires entre NC et P (dans P mais ni P-complets ni dans NC), c'est l'équivalent du théorème de Ladner pour NC versus P [Vol91, corollaire 3.2],
- Si PH possède un problème complet, alors elle s'effondre ($\mathsf{PH} = \Sigma_i^{\mathsf{P}}$ pour un certain $i \in \mathbb{N}$) [Per14, lemme 8-K page 199].
- Si $\mathsf{PH} = \mathsf{PSPACE}$, alors PH s'effondre (c'est-à-dire $\mathsf{PH} = \Sigma_i^{\mathsf{P}}$ pour un certain $i \in \mathbb{N}$) [Per14, lemme 8-L page 200]. La contraposée est tout aussi pertinente.

Si l'on croît que $\mathsf{NC} \subsetneq \mathsf{P} \subsetneq \mathsf{NP}$ (ce qui est l'opinion très majoritaire dans la communauté [Gas19]), alors le théorème de Ladner et son équivalent pour NC versus P nous

disent une chose fondamentale : il existe des problèmes intermédiaires que nous *ne savons pas* classifier avec précision dans les classes présentées ici. En effet, c'est la notion de complétude qui permet de cerner dans un premier temps la complexité d'un problème (ensuite peuvent venir de nombreuses précisions, par exemple d'approximation ou dans le domaine de la complexité paramétrée [DF13]).

2.2.3. Problèmes fonctionnels

Les problèmes fonctionnels consistent à calculer plus d'un bit d'information en sortie, c'est-à-dire plus généralement une fonction $f : \Sigma^* \to \Gamma^*$. Par exemple, étant donné un graphe, quelle est la taille de sa plus grande clique ? Les classes de complexité C déterministes ont un équivalent clair en termes de problèmes fonctionnels, et on note FC la classe de complexité associée. Pour les classes de complexité non déterministes, prenons l'exemple de NP : la classe FNP est l'ensemble des fonctions de la forme suivante : étant donnée une entrée x et une relation polynomiale $R(x, y)$, s'il existe un y satisfaisant $R(x, y)$, alors donner y en sortie, sinon rejeter. La classe FNP généralise NP, mais le terme « fonction » est trompeur, car il peut y avoir plusieurs sorties y valides. On a néanmoins le résultat suivant : $\mathsf{FP} = \mathsf{FNP} \iff \mathsf{P} = \mathsf{NP}$.

Au regard de la complétude pour les problèmes fonctionnels, on peut définir une notion de réduction utilisant deux transformations : la première pour traduire dans un sens (encoder) les instances, la seconde pour traduire dans l'autre sens (décoder) les sorties. On se heurte cependant à un obstacle pour trouver des problèmes complets, car certaines fonctions (pas toutes) sont totales : toute instance admet (au moins) une sortie. Donc il n'y a pas d'instance négative à certains problème. La classe des problèmes totaux est TFNP, mais la recherche d'un problème complet pour ces classes est toujours l'objet de recherches actives (voir [GP18] et ses références).

En ce qui nous concerne, nous éviterons autant que possible de manipuler les classes fonctionnelles, et aborderons les problèmes fonctionnels de la forme :

$$\text{étant donné } x, \text{ calculer } f(x)$$

à travers la traduction décisionnelle :

$$\text{étant donnés } x \text{ et } y, \text{ est-ce que } f(x) = y \text{ ?}$$

Notons que nos réductions sont définies avec les classes fonctionnelles FL, FNC et FP (algorithmes déterministes qui donnent une instance en sortie), mais n'utilisent pas la notion de difficulté pour ces classes (simplement l'appartenance).

2.2.4. Problèmes de comptage

Les problèmes de comptage sont une sous-classe de problèmes fonctionnels, ils correspondent au comptage du nombre de solutions et sont de la forme $f : \Sigma^* \to \mathbb{N}$. Pour une introduction plus précise, nous nous permettons de proposer notre

contribution sur l'étude des *feeback arc sets* [Per19] qui exemplifie les notions de base du domaine, et dont nous reprendrons en section 2.2.5 des éléments qui seront utiles au chapitre 8. Pour une vue plus complète, on pourra consulter [Per14, chapitre 9].

La classe #P compte le nombre de solutions (ou branches acceptantes) d'un algorithme séquentiel non déterministe en temps polynomial ou, de façon équivalente, le nombre de certificats d'un problème de décision dans NP. Les réductions sont many-one *parcimonieuses* en temps polynomial et notées $\leq^{\mathsf{P}}_{\mathtt{parci}}$. Elles doivent préserver le nombre de solutions au problème (le nombre de certificats). On peut également considérer la #P-difficulté pour les réductions Turing $\leq^{\mathsf{P}}_{\mathtt{T}}$, nous le préciserons dans les énoncés. Le problème #P-complet canonique pour les réductions $\leq^{\mathsf{P}}_{\mathtt{parci}}$ est **#3-SAT**.

Comptage 3-SAT (**#3-SAT**) *Entrée :* une formule propositionnelle φ. *Sortie :* nombre de valuations satisfaisant φ.

Pour obtenir la #P-complétude pour les réductions parcimonieuses $\leq^{\mathsf{P}}_{\mathtt{parci}}$ sans réduire directement depuis **#3-SAT**, il faut que *toutes* les réductions impliquées soient parcimonieuses. Compter dans un graphe le nombre de cliques de taille k, de *vertex covers*, de *feedback arc sets*, de *feedback vertex sets*, sont également des problèmes #P-complets pour les réductions parcimonieuses $\leq^{\mathsf{P}}_{\mathtt{parci}}$. Notons toutefois que le comptage du *nombre de solutions* peut être « plus difficile » que de décider l'existence d'*une solution*, par exemple **2-SAT** est dans P mais **#2-SAT** est #P-complet.

À propos des réductions many-one Turing $\leq^{\mathsf{P}}_{\mathtt{T}}$ pour la #P-difficulté, remarquons que la classe #P est close pour les réductions $\leq^{\mathsf{P}}_{\mathtt{parci}}$, alors que sa clôture pour les réductions $\leq^{\mathsf{P}}_{\mathtt{T}}$ est ouverte et équivalente à P = NP [HP09]. Cela pose en particulier une difficulté pour la classification des problèmes de comptage du nombre de solutions minimums d'un problème NP-complet, dont nous discuterons dans l'idée de preuve du théorème 2.2. Intuitivement, il n'est pas « facile » de vérifier en temps polynomial si une solution est minimum (plus petite parmi toutes les solutions), car cela requiert de calculer la valeur du minimum (ce qui est NP-difficile) [e]. Heureusement, les auteurs de [HP09] ont défini la classe $\#\cdot\mathsf{OptP}[\log n]$ qui capture de façon naturelle la complexité des problèmes de comptage du nombre de solutions minimums. Elle est définie à partir d'un modèle séquentiel non déterministe où chaque branche accepte ou rejette et en même temps donne en sortie un entier encodé en binaire, que nous appellerons *transducteurs non déterministes*. La classe $\#\cdot\mathsf{OptP}[\log n]$ correspond à compter le nombre de branches acceptantes ayant donné la plus petite valeur parmi toutes les branches acceptantes, pour un transducteur non déterministe en temps polynomial produisant des sorties de taille $\mathcal{O}(\log n)$. La $\#\cdot\mathsf{OptP}[\log n]$-complétude sera considérée ici uniquement pour les réductions parcimonieuses $\leq^{\mathsf{P}}_{\mathtt{parci}}$.

e. Notons qu'en revanche, vérifier si une solution est minimale (ne contient pas une solution strictement plus petite) peut en général être réalisé en temps polynomial.

2.2.5. Feedback arc sets et $\#\mathsf{P}, \#\cdot\mathsf{OptP}[\log n]$

Nous sommes prêts pour énoncer les problèmes de comptage relatifs au *feedback arc sets* (FAS, ensembles d'arcs intersectant tous les cycles d'un graphe orienté) et leur classification, en notant $\mathtt{FAS}(G) \subseteq \mathcal{P}(A)$ l'ensemble des FAS d'un graphe orienté $G = (V, A)$. Les FAS sont des objets qui apparaissent dans l'étude des réseaux d'automates (voir chapitre 3.6), et nous utiliserons les résultats du théorème 2.2 ci-dessous au chapitre 8.2.2 sur les modes de mises à jour.

Cardinalité Feedback Arc Set (**#Card-FAS**) *Entrée :* un graphe orienté G et un entier $k \in \mathbb{N}$. *Sortie :* $\lvert\{F \in \mathtt{FAS}(G) \mid \lvert F\rvert = k\}\rvert$.

Comptage Feedback Arc Set (**#FAS**) *Entrée :* un graphe orienté G. *Sortie :* $\lvert\mathtt{FAS}(G)\rvert$.

Comptage Minimum Feedback Arc Set (**#Minimum-FAS**) *Entrée :* un graphe orienté G. *Sortie :* $\lvert\{F \in \mathtt{FAS}(G) \mid \lvert F\rvert = m\}\rvert$ avec $m = \min\{\lvert F\rvert \mid F \in \mathtt{FAS}(G)\}$.

Théorème 2.2 ([Per19])**.** *Sur le comptage du nombre de feedback arc sets.*

(i) **#Card-FAS** *est* $\#\mathsf{P}$*-complet pour les réductions* $\leq_T^{\mathsf{P}}$.
(ii) **#FAS** *est* $\#\mathsf{P}$*-complet pour les réductions* $\leq_T^{\mathsf{P}}$.
(iii) **#Minimum-FAS** *est* $\#\mathsf{P}$*-difficile et* $\#\mathsf{P}$*-facile pour les réductions* $\leq_T^{\mathsf{P}}$.
(iv) **#Minimum-FAS** *est* $\#\cdot\mathsf{OptP}[\log n]$*-complet pour les réductions* $\leq_{parci}^{\mathsf{P}}$.
(v) Si **#Minimum-FAS** *est dans* $\#\mathsf{P}$*, alors* $\mathsf{P} = \mathsf{NP}$.

Idée de la démonstration. (i) La preuve d'appartenance à $\#\mathsf{P}$ est évidente (deviner $F \subseteq A$ et vérifier qu'il s'agit d'un FAS de taille k). Concernant la $\#\mathsf{P}$-difficulté, on adapte la construction de Karp [Kar72] pour obtenir une réduction $\leq_{\mathtt{T}}^{\mathsf{P}}$ depuis le problème de comptage des *vertex covers* de taille k dans un graphe non orienté (qui est $\#\mathsf{P}$-complet [GJ79, page 169]). Il faut plusieurs appels à l'oracle **#Card-FAS** et quelques calculs de coefficients binomiaux pour retrouver le nombre de *vertex covers* de taille k.

(ii) La preuve d'appartenance à $\#\mathsf{P}$ est évidente (deviner $F \subseteq A$ et vérifier qu'il s'agit d'un FAS). On réalise une réduction $\leq_{\mathtt{T}}^{\mathsf{P}}$ de **#Card-FAS** à **#FAS** avec la méthode des matrices de Vandermonde [PB83 ; Val79] : on remplace les arcs par des chemins de longueurs ℓ qui font grandir artificiellement le nombre de FAS selon la formule $(2^\ell - 1)^{|F|}$, puis des appels à l'oracle pour $\ell \in \{0, \ldots, |A|\}$ permettent de calculer le nombre de FAS de taille k dans le graphe de départ (cette dernière partie de la méthode est donnée gratuitement par un lemme technique [PB83, lemme page 781]).

(iii) La $\#\mathsf{P}$-difficulté est obtenue avec une réduction $\leq_{\mathtt{parci}}^{\mathsf{P}}$ depuis le problème de comptage des *vertex covers* minimums (qui est complet pour les réductions $\leq_{\mathtt{T}}^{\mathsf{P}}$ [PB83,

problème 4]), dont la construction est un cas particulier de la réduction du point (i). La #P-facilité[f] s'obtient simplement depuis **#Card-FAS** en appelant l'oracle pour $k = 0, 1, \ldots$ jusqu'à la première réponse strictement positive.

(iv) L'appartenance à $\#\cdot\mathsf{OptP}[\log n]$ est évidente : chaque branche donne en sortie la taille du FAS deviné. Pour la $\#\cdot\mathsf{OptP}[\log n]$-difficulté, on réutilise notre réduction parcimonieuse du point (iii), car le comptage des *vertex covers* minimums est $\#\cdot\mathsf{OptP}[\log n]$-complet pour les réductions $\leq^{\mathsf{P}}_{\mathtt{parci}}$ [HP09, théorème 11].

(v) Si **#Minimum-FAS** est dans #P, alors, d'après les points (iii) et (iv), nous avons $\#\cdot\mathsf{OptP}[\log n] = \#\mathsf{P}$ (l'autre inclusion est évidente). La conclusion est obtenue par application de [HP09, théorème 9] pour $k = 0$. □

Notons que compter le nombre de FAS minimaux (pour l'inclusion) était déjà connu pour être #P-complet [SS02], avec une réduction parcimonieuse depuis le problème de comptage du nombre d'orientations acycliques d'un graphe non orienté [Lin86]. Vérifier qu'un FAS F est minimal se fait facilement en temps polynomial : pour tout $a \in F$, vérifier que le graphe orienté $(V, (A \setminus F) \cup \{a\})$ comporte un cycle.

Remarquons pour conclure que tous les points du théorème 2.2 s'appliquent identiquement aux problèmes de comptage des *feedback vertex sets* (FVS). En effet, considérer le *line graph orienté* $L(G)$ de G (où les arcs deviennent les sommets, et sont adjacents lorsqu'ils forment un chemin de longueur deux) est une réduction parcimonieuse, car les FAS de G sont en bijection avec les FVS de $L(G)$ [SS02].

2.2.6. Problèmes avec promesse et solution unique

Un problème avec promesse offre la garantie que l'entrée appartient à un certain sous-ensemble de toutes les entrées possibles (il n'est donc pas nécessaire de vérifier cette propriété pour éliminer les instances invalides). Pour être pertinentes, les promesses doivent donc être des propriétés C-difficiles pour la classe de complexité C considérée. En voici un exemple classique.

Unambiguous-SAT (**Unamb-SAT**) *Entrée :* une formule propositionnelle φ. *Promesse :* φ est satisfaite par au plus une valuation. *Question :* φ est-elle satisfaisable ?

RP est la classe P randomisée : refuse toutes les instances négatives, et accepte les instances positives avec probabilité supérieure à $\frac{1}{2}$. Il est conjecturé que RP = P.

Théorème 2.3 ([VV86])**.** **Unamb-SAT** *est* NP*-difficile pour les réductions many-one polynomiales randomisées. Autrement dit, si* **Unamb-SAT** *est dans* P *alors* NP = RP.

f. Un problème est C-facile lorsqu'il peut être réduit à un problème dans C.

Un problème sans promesse proche de *Unambiguous-SAT* est *Unique-SAT*.

Unique-SAT (**U-SAT** ou ∃!-**SAT**) *Entrée :* une formule propositionnelle φ. *Question :* φ est-elle satisfaite par exactement une valuation ?

Pour classifier ce problème, on change le modèle de calcul. Plus précisément, on modifie la définition de calcul *acceptant* d'une machine non déterministe, qui est originellement donnée par l'existence d'*au moins une* branche qui accepte. La classe US contient les problèmes de décision pour lesquels il existe un algorithme non déterministe en temps polynomial, qui accepte si et seulement si son calcul possède *exactement une* branche acceptante [BG82] [g].

Théorème 2.4 ([BG82])**.** coNP ⊆ US, *et* **U-SAT** *est* US*-complet pour les réductions* $\leq_m^{\mathsf{P}}$.

Les classes US (solution unique) et DP (intersection de langages dans NP et coNP) se situent toutes deux entre les niveaux 1 et 2 de la hiérachie polynomiale [h] :

$$\Pi_1^{\mathsf{P}} \subseteq \mathsf{US} \subseteq \Delta_2^{\mathsf{P}} \quad \text{et} \quad \Sigma_1^{\mathsf{P}}, \Pi_1^{\mathsf{P}} \subseteq \mathsf{DP} \subseteq \Delta_2^{\mathsf{P}}.$$

Comme le remarquent les auteurs de [BG82], on a :

$$\mathsf{DP} = \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP} \text{ et } L_2 \in \mathsf{coNP}\} = \{L_1 \setminus L_2 \mid L_1, L_2 \in \mathsf{NP}\}$$

et le problème **U-SAT** (*Unique-SAT*) peut être formulé comme **SAT** auquel on enlève l'ensemble des instances φ qui ont au moins deux valuations satisfaisantes, ce qui correspond à soustraire un problème dans NP à un problème dans NP, donc **U-SAT** est dans DP. Puisque **U-SAT** est US-complet et que les classes US et DP sont closes par $\leq_{\mathtt{m}}^{\mathsf{P}}$, on en déduit :

$$\mathsf{US} \subseteq \mathsf{DP}.$$

En annexe A, nous proposons une discussion sur l'intégration du quantificateur ∃! (« il existe un·e unique ») à la *hiérarchie polynomiale*, en plus des quantificateurs ∃ et ∀ auxquels elle est classiquement associée. Au chapitre 3.4, nous rencontrerons le problème ∀∃!-**SAT** qui est dans Π_2^{P} (et ∃∃!-**SAT** est Σ_2^{P}-complet).

g. Ne pas confondre avec UP [Val76] : ici un algorithme peut, sur une entrée u, avoir plusieurs branches qui acceptent, et alors u est rejetée selon la définition du modèle de calcul pour US.

h. On rappelle les notations $\Sigma_1^{\mathsf{P}} = \mathsf{NP}$, $\Pi_1^{\mathsf{P}} = \mathsf{coNP}$, $\Delta_2^{\mathsf{P}} = \mathsf{P}^{\mathsf{NP}}$, $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$, $\Pi_2^{\mathsf{P}} = \mathsf{coNP}^{\mathsf{NP}}$.

3. Modèles des réseaux d'automates

Nous commencerons par définir toutes les notions (graphe d'interaction, modes de mise à jour, dynamique) sur les réseaux d'automates booléens déterministes en sections 3.1 et 3.2, puis discuterons en section 3.3 des extensions non déterministes, et de celles où les alphabets varient. La section 3.4 présentera les encodages utilisés pour représenter les fonctions et relations locales, et des exemples suivront en section 3.5.

Nous notons $e_i \in \{0,1\}^n$ le i-ème vecteur de base (c'est-à-dire $e_i(j) = 1 \iff j = i$). Pour $x, y \in \{0,1\}^n$, l'addition bit-à-bit modulo deux est simplement notée $x + y$. Étant donné $I \subseteq [n]$, la restriction de $x \in \{0,1\}^n$ à l'ensemble de composantes I est notée x_I.

3.1. Réseau d'automates booléens, graphe d'interaction

Réseau d'automates. Un *réseau d'automates booléens* (RAB) de taille n est une fonction $f : \{0,1\}^n \to \{0,1\}^n$ dans l'ensemble des vecteurs booléens de taille n. L'ensemble des *automates* est $[n] = \{1,\dots,n\}$, et parfois $[\![n]\!] = \{0,\dots,n-1\}$ par commodité (par exemple pour les cycles où l'on utilise des modulos). L'ensemble des *configurations* est $\{0,1\}^n$ (dont les indices seront pris dans $[n]$ ou $[\![n]\!]$ suivant le contexte). Par convention, la i-ème composante d'une configuration $x \in \{0,1\}^n$ est notée x_i, il s'agit de l'*état* de l'automate $i \in [n]$. On note 0^n et 1^n les configurations de taille n dont toutes les composantes sont dans l'état 0 et 1 respectivement. Pour chaque $i \in [n]$, on note $f_i : \{0,1\}^n \to \{0,1\}$ la i-ème composante de la fonction f, que l'on appelle *fonction locale* de l'automate i, avec $f(x)_i = f_i(x)$ pour tous $i \in [n]$ et $x \in \{0,1\}^n$.

La définition d'un réseau d'automates booléens f donne une vision « statique » du système. Sa dynamique sera relative à un *mode de mise à jour*, indiquant l'ordre dans lequel les automates du réseau appliquent leur fonction locale pour mettre à jour leur état (section 3.2).

Graphe d'interaction (signé). À un RAB f, on associe le *graphe d'interaction signé* $G_f^{\pm} = ([n], A^{\pm})$ sur l'ensemble des automates, avec $A^{\pm} \subseteq [n] \times \{-,+\} \times [n]$ tel que :

$$(i,+,j) \in A^{\pm} \iff \exists x \in \{0,1\}^n : x_i = 0 \text{ et } 0 = f_j(x) < f_j(x + e_i) = 1$$
$$(i,-,j) \in A^{\pm} \iff \exists x \in \{0,1\}^n : x_i = 1 \text{ et } 0 = f_j(x) < f_j(x + e_i) = 1.$$

Le graphe d'interaction signé capture les dépendances effectives entre automates : il y a un arc de i vers j lorsque l'état de l'automate i a une influence sur l'état de l'automate j, et le signe est positif (*resp.* négatif) lorsque l'automate j tend à prendre l'état de (*resp.* l'état opposé à) l'automate i. Remarquons que certaines dépendances peuvent être à la fois positives et négatives. On parle alors de *non monotonie*. On considère également que le graphe d'interaction est un graphe orienté simple dont les arcs sont étiquetés parmi $\{+,-,\pm\}$, avec $\pm$ pour les dépendances non monotones.

En projetant $A^{\pm}$ sur $[n]\times[n]$ (c'est-à-dire en oubliant les signes), on obtient le *graphe d'interaction non signé* $G_f = ([n], A)$, dont les arcs sont plus simplement donnés par :

$$(i,j) \in A \iff \exists x \in \{0,1\}^n : f_j(x) \neq f_j(x + e_i).$$

Signe d'un cycle de $G_f^{\pm}$. Dans un graphe d'interaction signé, on dira qu'un cycle, vu comme un ensemble d'arcs, est *positif* lorsqu'il contient un nombre pair d'arcs négatifs, et *négatif* lorsqu'il contient un nombre impair d'arcs négatifs. Intuitivement, en parcourant le cycle l'état d'un automate lui revient à l'identique dans le cas d'un cycle positif (ce qui est stable et « bon » pour obtenir des points fixes), alors qu'il lui revient inversé dans le cas d'un cycle négatif (ce qui est instable et « mauvais » pour obtenir des points fixes).

3.2. Modes de mise à jour, dynamique

Mode de mise à jour. Une *mise à jour* ou *bloc* sur le RAB $f : \{0,1\}^n \to \{0,1\}^n$ est un sous-ensemble d'automates $B \subseteq [n]$ qui sont mis à jour simultanément au cours d'une étape discrète. Étant donnée une configuration $x \in \{0,1\}^n$, on note $f^{[(B)]}$ la configuration obtenue, définie par :

$$\forall i \in [n] : f^{[(B)]}(x)_i = \begin{cases} f_i(x) & \text{si } i \in B \\ x_i & \text{sinon.} \end{cases}$$

Un *mode de mise à jour* est une suite de blocs $\beta = (B_1, \ldots, B_p)$ appliqués séquentiellement. Étant donnée une configuration $x \in \{0,1\}^n$, on note $f^{[\beta]}$ le résultat, défini inductivement par :

$$\forall i \in [p] : f^{[(B_1,\ldots,B_i)]} = f^{[(B_i)]} \circ f^{[(B_1,\ldots,B_{i-1})]} \quad \text{avec } f^{[()]} \text{ l'identité.}$$

Remarquons que l'on se restreint ici aux modes de mise à jour *déterministes périodiques* : les itérations de $f^{[\beta]}$ appliquent la même séquence de blocs β à chaque itération. On parlera d'une *étape* de calcul ou *itération* pour la mise à jour selon β, et de *sous-étapes* de calcul ou *sous-itération* pour la mise à jour d'un bloc B_i avec $i \in [p]$. Le système dynamique discret qui nous intéresse est $f^{[\beta]}$ sur $\{0,1\}^n$. Notons que l'on pourra trouver une liste plus complète de modes de mises à jour dans [PS21 ; PS].

- **Équilibré.** Un mode de mise à jour $\beta = (B_1, \dots, B_p)$ est *équilibré* lorsque tous les automates sont mis à jour au moins une fois au cours d'une itération, c'est-à-dire $\bigcup_{i\in[p]} B_i = [n]$.
- **Bloc-séquentiel.** Un mode de mise à jour $\beta = (B_1, \dots, B_p)$ est *bloc-séquentiel* lorsque chaque automate est mis à jour exactement une fois au cours d'une itération, c'est-à-dire $B_1, \dots, B_p$ est une partition ordonnée de $[n]$ (cela ajoute la condition $\forall i \neq j \in [p] : B_i \cap B_j = \emptyset$ à l'équilibre).
- **Atomique.** Un mode de mise à jour $\beta = (B_1, \dots, B_p)$ est *atomique* lorsque chaque bloc comporte un unique automate.
- **Séquentiel.** Un mode de mise à jour $\beta = (B_1, \dots, B_p)$ est *séquentiel* lorsqu'il est bloc-séquentiel et atomique (cela ajoute la condition $\forall i \in [p] : |B_i| = 1$ au bloc-séquentiel, et donc $p = n$). Il s'agit d'une permutation des automates.
- **Parallèle.** Le mode de mise à jour *parallèle* est $\beta^{\mathtt{par}} = ([n])$, il correspond à la mise à jour synchrone de tous les automates à chaque itération. Bien sûr, on a $f^{[\beta^{\mathtt{par}}]} = f$, et $\beta^{\mathtt{par}}$ est bloc-séquentiel.

Dynamique. Étant donnée $f : \{0,1\}^n \to \{0,1\}^n$ déterministe (qui peut correspondre à $g^{[\beta]}$ pour un RAB $g : \{0,1\}^n \to \{0,1\}^n$ selon un mode de mise à jour β), on appelle *graphe de la dynamique* ou *graphe de transition* ou simplement *dynamique*, le graphe de la fonction f, défini comme $\mathscr{G}_f = (\{0,1\}^n, A)$ de degré sortant 1, avec :

$$(x, y) \in A \iff f(x) = y.$$

Dans la suite du document, nous nous attacherons particulièrement au mode de mise à jour parallèle, qui sera, sauf mention contraire, le mode de mise à jour appliqué. Nous considérerons ainsi directement la fonction $f : \{0,1\}^n \to \{0,1\}^n$. Le chapitre 8 portera quant à lui exclusivement sur les modes de mise à jour.

Dynamique asymptotique. Étant donnée $f : \{0,1\}^n \to \{0,1\}^n$ déterministe, on appelle *point fixe* une configuration $x \in \{0,1\}^n$ telle que $f(x) = x$. On notera $\mathfrak{P}_f$ l'ensemble des points fixes de f.

Un *cycle limite de taille k* est une séquence de configurations $(x^0, x^1, \dots, x^{k-1})$ telles que $f(x^i) = x^{i+1}$ pour tout $i \in [\![k]\!]$ (avec l'addition $i+1$ prise modulo k) et $x^i \neq x^j$ pour tous $i \neq j \in [\![k]\!]$. On assimile également un cycle limite à son ensemble de configurations. Les points fixes sont des cycles limite de taille 1. On notera $\mathfrak{C}_f^k$ l'ensemble des cycles limites de taille k dans f. On nommera k la *période* d'une configuration de $\mathfrak{C}_f^k$.

On définit l'*ensemble limite* de f comme :

$$\Omega_f = \bigcap_{t\in\mathbb{N}} f^t(\{0,1\}^n),$$

avec $f(X) = \bigcup_{x\in X} f(x)$. Les configurations de Ω_f sont appelées *configurations limites*, ce sont celles qui sont observées arbitrairement loin (temporellement) dans la dyna-

mique. Dans le cas déterministe, on a $\Omega_f = \bigcup_{k\in\mathbb{N}_+} \mathfrak{C}_f^k$, car toute configuration a un comportement ultimement périodique (l'espace est fini).

Étant donnée une configuration $x \in \{0,1\}^n$, son *orbite* est

$$\mathfrak{O}_f(x) = \{f^t(x) \mid t \in \mathbb{N}\},$$

il s'agit des configurations qui sont visitées à partir de x (notons que $x \in \mathfrak{O}_f(x)$ à $t = 0$). On dira qu'une configuration y est *atteignable* depuis x lorsque $y \in \mathfrak{O}_f(x)$. Une configuration y telle que $f(y) = x$ est appelée *antécédent* de x. Le *bassin d'attraction* d'un point fixe x est l'ensemble des configurations qui atteignent x, noté

$$\mathfrak{B}_f(x) = \{y \in \{0,1\}^n \mid x \in \mathfrak{O}_f(y)\}.$$

Remarquons que la définition de Ω_f fonctionne identiquement dans le cas non déterministe que nous verrons en section 3.3, mais ce n'est pas le cas de $\mathfrak{O}_f(x)$ qui sera adaptée en prenant l'union des futurs possibles au chapitre 5.2.

3.3. Uniformité, non déterminisme, asynchronisme

Les relations entre variantes des réseaux d'automates booléens déterministes présentées ci-après sont illustrées sur la figure 3.1. Notons qu'en général, un résultat de C-difficulté s'étendra à tous les sur-ensembles de modèles, alors qu'un résultat d'appartenance à C s'étendra à tous les sous-ensembles de modèles. Il faut néanmoins rester vigilant sur les détails, tels que ceux discutés en section 3.4. Le passage du cas déterministe au cas non déterministe, pour les bornes inférieures de complexité sur des problèmes portant sur le graphe de la dynamique, sera explicité en fin de chapitre 5.1 sur les points fixes (remarque 5.4).

Uniformité. On peut naturellement étendre ces définitions à des ensemble d'états plus généraux que $\{0,1\}$. On parlera de *réseau d'automates* (RA) lorsque chaque automate $i \in [n]$ possède son propre alphabet fini A_i, sans perte de généralité avec $A_i = [\![q_i]\!]$ pour $q_i \in \mathbb{N}$, et de *réseau d'automates q-uniforme* (RAU) lorsque $q_i = q$ pour tout $i \in [n]$. Les réseaux d'automates booléens sont donc 2-uniformes.

Dans ce contexte plus général, on aura l'ensemble des configurations $X = \prod_{i\in[n]} A_i$, des fonctions locales $f_i : X \to A_i$ pour chaque automate $i \in [n]$, et $f : X \to X$. Les arcs du graphe d'interaction non signé [a] $G_f = ([n], A)$ seront encore une fois donnés par :

$$(i,j) \in A \iff \exists x \in X : f_j(x) \neq f_j(x + e_i)$$

où l'addition $x + e_i$ est prise modulo q_i sur la composante i.

a. Bien entendu, on peut imaginer ajouter des signes.

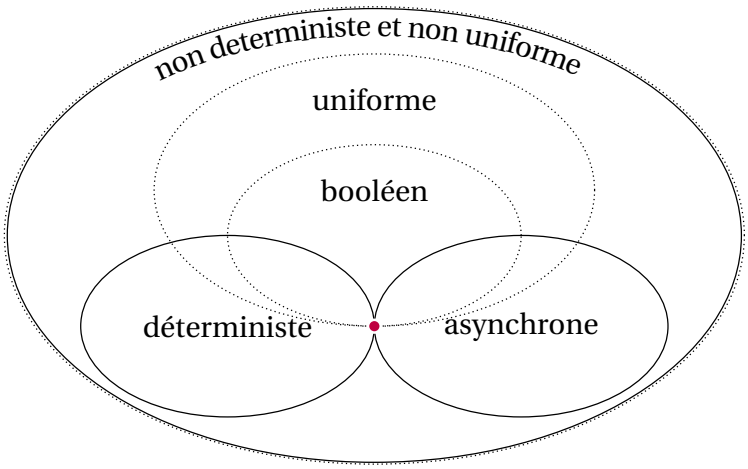


FIGURE 3.1. – Relations d'inclusion entre les modèles de réseaux d'automates présentés en section 3.3, du point de vue des dynamiques. Notons que les réseaux identité définis par $f_i(x) = x_i$ pour tous i et x constituent l'intersection des modes déterministes et asynchrones (point coloré).

Non déterminisme. Afin de définir un *réseau d'automates non déterministe* sur $X = \prod_{i\in[n]} A_i$, les fonctions locales sont étendues en des *relations locales*[b] $r_i : X \to \mathcal{P}(A_i)$ pour chaque automate $i \in [n]$. Ces relations décrivent, pour chaque automate, l'ensemble de ses états possibles lorsqu'il est mis à jour.

La dynamique est donnée par $f : X \to \mathcal{P}(X)$ avec $f(x) = \prod_{i\in[n]} r_i(x)$, c'est-à-dire :

$$\forall x, y \in X : \qquad y \in f(x) \iff \forall i \in [n] : y_i \in r_i(x)$$

et donc le graphe de la dynamique est $\mathcal{G}_f$ sur $\{0,1\}^n$ avec les arcs $\{(x,y) \mid y \in f(x)\}$. Remarquons que l'ensemble vide $\emptyset$ est absorbant pour le produit cartésien, donc si $r_i(x) = \emptyset$ pour $x \in X$ et au moins un $i \in [n]$, alors $f(x) = \emptyset$. Ainsi la dynamique sans aucune transition, c'est-à-dire telle que $f(x) = \emptyset$ pour tout $x \in X$, est la dynamique d'un réseau non déterministe (peu intéressant, mais le théorème 3.3 ci-dessous indique que cela est difficile à vérifier). On dira que $x \in X$ est un *point fixe* lorsque $x \in f(x)$, et un *point fixe fort* lorsque $f(x) = \{x\}$.

Sauf mention contraire, nos réseaux d'automates seront déterministes.

Asynchronisme (parfait). Il s'agit d'un cas particulier de non déterminisme. Dans un *réseau d'automates booléens asynchrone* $\hat{f} : \{0,1\}^n \to \mathcal{P}(\{0,1\}^n)$, une seule composante de $f : \{0,1\}^n \to \{0,1\}^n$ est mise à jour à chaque itération non déterministe.

b. Pour garder une symétrie avec le cas déterministe, on voit une relation $\tilde{r} \subseteq X \times Y$ comme la fonction $r : X \to \mathcal{P}(Y)$ avec $r(x) = \{y \in Y \mid (x,y) \in \tilde{r}\}$ pour tout $x \in X$.

Réseaux d'automates	**Alphabet**	**Temps polynomial**	NP-**difficile**
déterministes	tout	$f(x)$	
non déterministe	fixé	$y \in f(x)$, $r_i(x)$, $f(x) \neq \emptyset$	$f(x)$
	en entrée	$y \in f(x)$	$f(x)$, $r_i(x)$, $f(x) \neq \emptyset$
asynchrone	booléen	$f(x)$	

Table 3.1. – Complexité de calcul des images (dynamique « en avant »), selon la famille de réseaux considérée. Chaque élément correspond à un problème de calcul d'une valeur (par exemple $f(x)$ déterministe) ou d'un ensemble de valeurs (par exemple $f(x)$ non déterministe), ou à un problème de décision (par exemple $y \in f(x)$), qui prend en entrée tous les éléments qui composent la question (par exemple f, x, y pour la question $y \in f(x)$).

Formellement :

$$\forall x, y \in \{0,1\}^n : y \in \hat{f}(x) \iff \begin{cases} y = x + e_i \text{ avec } f_i(x) \neq x_i \text{ (c'est-à-dire } f_i(x) = x_i + 1\text{)} \\ \text{ou } y = x \text{ si } \exists i \in [n] : f_i(x) = x_i \end{cases}$$

que l'on peut également formuler algébriquement comme :

$$\forall x \in \{0,1\}^n : \hat{f}(x) = \{x + (f_i(x) + x_i) \cdot e_i \mid i \in [n]\}$$

avec les additions modulo deux, $0 \cdot e_i = 0^n$ et $1 \cdot e_i = e_i$. Dans le mode asynchrone, la dynamique évolue le long des arêtes de l'hypercube à n dimensions.

3.4. Encodages

Puisque nous traitons de complexité, il faut encoder les objets pour les donner en entrée à des algorithmes. Dans cette section, nous présentons les encodages de réseaux d'automates, et discutons de la complexité de calculer les images (dynamique « en avant »). Le calcul des préimages (dynamique « en arrière ») sera traité au chapitre 5.2. Les principales considérations sont résumées dans la table 3.1.

Déterministe booléen. Lorsqu'un problème prendra en entrée un réseau d'automates booléen déterministe, il s'agira (nous l'indiquerons explicitement) de n circuits booléens (chacun à n entrées et une sortie) : un pour chaque fonction locale $\{0,1\}^n \to \{0,1\}$ du réseau. La taille de chaque circuit sera supposée être au plus 2^{2^n}, qui est la taille d'une table de vérité [c]. Étant donnés $x \in \{0,1\}^n$ et $i \in [n]$, calculer le bit

c. On pourrait imaginer avoir des circuits inutilement plus gros, mais il sera pratique et naturel de considérer que le nombre de réseaux d'une taille donnée est fini.

$f_i(x)$ est un problème de décision P-complet (*Circuit Value Problem* [GHR95]), donc on peut calculer $f(x)$ en temps polynomial.

Déterministe non uniforme. Dans le cas d'un réseau déterministe (potentiellement non) uniforme sur $X = \prod_{i\in[n]} [\![q_i]\!]$, chaque état est encodé en binaire sur $\lceil \log q_i \rceil$ bits pour $i \in [n]$. Le circuit d'une fonction locale f_j pour $j \in [n]$ aura ainsi $\sum_{i\in[n]} \lceil \log q_i \rceil$ bits d'entrées et $\lceil \log q_j \rceil$ bits de sortie. On peut calculer $f(x)$ en temps polynomial.

Nous n'avons pas besoin de considérer l'évaluation de tels circuits pour les entrées $x \in \{0,1\}^n$ *invalides*, c'est-à-dire dont l'état d'au moins un $i \in [n]$ encodé en binaire sur $\lceil \log q_i \rceil$ bits correspond à un entier entre q_i et $2^{\lceil \log q_i \rceil}$, car de telles entrées sont simplement ignorées. En revanche, il faut préciser que les sorties du circuit de la fonction f_i pour $i \in [n]$ sont, par convention, interprétées modulo q_i. En effet, sinon l'étape implicite de vérification qu'un circuit donné en entrée d'un problème est *valide* (voir introduction du chapitre 2.2) serait coNP-complète à réaliser. Une alternative à cette convention consiste à avoir la promesse que pour tous $i \in [n]$ et $x \in X$, le circuit de la fonction locale f_i ne produit pas de sortie supérieure ou égale à q_i.

Non déterministe. Pour les réseaux booléens non déterministes, chaque relation locale r_i est encodée par un circuit booléen prenant en entrée une configuration $x \in \{0,1\}^n$ et un bit $b \in \{0,1\}$, et répondant en 1 bit si $b \in r_i(x)$ ou non. Pour le cas non uniforme sur $X = \prod_{i\in[n]} [\![q_i]\!]$, les relations locales sont encodées par des circuits booléens prenant en entrée une configuration $x \in X$ et un état $a \in A_i$ sur $\lceil \log q_i \rceil$ bits, et répondant en 1 bit de sortie si $a \in r_i(x)$ ou non. Les tailles $q_1, \dots, q_n$ des alphabets sont données en binaire dans l'entrée. Étant donnés un réseau non déterministe f et x, y deux configurations, on peut décider en temps polynomial si $y \in f(x)$.

Le cas non déterministe et non uniforme est affranchi de la convention du paragraphe précédent sur la non uniformité, car les encodages *invalides* (entre q_i et $2^{\lceil \log q_i \rceil}$) sont restreints aux entrées des circuits et donc simplement ignorés pour le calcul de la dynamique.

L'ensemble $f(x)$ des images de x est donné par le produit cartésien des possibilités locales : $f(x) = \prod_{i\in[n]} r_i(x)$. Cet ensemble peut être de taille exponentielle, mais l'on peut calculer chaque élément du produit cartésien en temps polynomial lorsque l'alphabet est fixé (pour x donné on teste pour chaque $i \in [n]$ toutes les lettres de q_i, ce qui donne $r_i(x)$). Remarquons qu'il est en revanche difficile de décider, dans le cas non déterministe à alphabet uniforme mais non fixé, si une configuration x donnée possède une image.

Image non vide non déterministe (**Image-nonvide-nondet**) *Entrée :* un RA f non déterministe q-uniforme à n automates, et $x \in [\![q]\!]^n$. *Question :* est-ce que $f(x) \neq \emptyset$?

Théorème 3.1. *Si l'alphabet est fixé, alors* **Image-nonvide-nondet** *est dans* P*, sinon* **Image-nonvide-nondet** *est* NP*-complet.*

Démonstration. Dans le cas à alphabet fixé, le problème est dans P par notre remarque précédente : les éléments du produit cartésien correspondant à $f(x)$ peuvent être calculés en temps polynomial.

À alphabet non fixé, la taille q de l'alphabet fait partie de l'entrée, et est encodée en binaire. L'appartenance à NP est évidente : deviner $y \in [\![q]\!]^n$ et vérifier $y \in f(x)$ (la valeur de n est donnée en unaire par le nombre de circuits des relations locales de f). La difficulté est donnée par une réduction many-one polynomiale depuis **SAT** : étant donnée une formule φ à n variables, on choisit $q = 2^n$ et l'on construit le réseau à 1 automate dont la relation locale est :

$$r_1(x) = \{y \in [\![q]\!]^1 \mid y \models \varphi\} \qquad \text{où } y \models \varphi \text{ signifie que la valuation [d] } y \text{ satisfait } \varphi.$$

Cette relation est encodée par le circuit sur $2n$ entrées et une sortie, qui ignore les n premiers bits et recopie le circuit de φ sur les n derniers bits, avec l'évaluation (la racine du circuit) de φ en sortie. Pour tout $x \in [\![q]\!]^n$, on a alors $f(x) \neq \emptyset$ si et seulement φ est satisfaisable (toutes les configurations ont les mêmes images). On peut alors terminer de former l'instance de **Image-nonvide-nondet** avec $x = 0^n$. □

Une question difficile à alphabet fixé, même booléen, consiste à décider si un réseau non déterministe donné possède au moins une image (la question ne porte plus sur une configuration particulière, mais sur l'ensemble des configurations du réseau).

Non vide non déterministe (**Nonvide-nondet**) *Entrée :* un RA f non déterministe q-uniforme à n automates. *Question :* est-ce que $\exists x \in [\![q]\!]^n : f(x) \neq \emptyset$?

Théorème 3.2. **Nonvide-nondet** *est* NP*-complet, même à alphabet fixé booléen.*

Démonstration. L'entrée f nous est donnée par des circuits pour chaque relation locale. L'appartenance à NP est évidente : deviner $x, y \in [\![q]\!]^n$ (de tailles polynomiales) et vérifier si $y \in f(x)$ en temps polynomial.

La difficulté dans le cas à alphabet fixé booléen est obtenue avec une réduction many-one polynomiale depuis **SAT**. Étant donnée une formule φ à n variables, on construit le RAB non déterministe à n automates dont les relations locales sont :

- pour $i \in [n-1]$, $\forall x \in \{0,1\}^n : r_i(x) = \{0,1\}$ (circuit trivial à $n+1$ entrées booléennes, qui répond toujours 1),
- r_n est le circuit qui, sur l'entrée $x \in \{0,1\}^n$ et $b \in \{0,1\}$, répond 1 si et seulement si $x \models \varphi$ (c'est-à-dire $\varphi(x) = \top$) : on recopie le circuit de φ en ignorant b.

Alors $f(x) = \{0,1\}^n$ lorsque $x \models \varphi$, et $f(x) = \emptyset$ sinon (car alors $r_n(x) = \emptyset$). □

d. Puisque $q = 2^n$ on identifie $y \in [\![q]\!]$ à un vecteur booléen de taille n, et on identifie un vecteur booléen de taille n à une valuation sur n variables.

On peut remarquer dans les réductions précédentes que nous n'avons pas utilisé tous les degrés de liberté des alphabets uniformes : au théorème 3.1, la valeur de x n'est pas utilisée dans la définition de $r_1(x)$ et, au théorème 3.2, la preuve de difficulté utilise seulement un alphabet booléen. Nous présentons maintenant un problème simple qui exploite pleinement ces possibilités.

Vide non déterministe (**Vide-nondet**) *Entrée :* un RA f non déterministe q-uniforme à n automates. *Question :* est-ce que $\exists x \in [\![q]\!]^n : f(x) = \emptyset$?

Dans cette formulation, la proposition $f(x) = \emptyset$ correspond à une quantification universelle : $\forall y \in [\![q]\!]^n : y \notin f(x)$. On atteint ainsi le niveau suivant de la hiérarchie polynomiale. Nous en sommes venus à jouer un peu dans la définition de ce problème qui semble moins naturel, bien qu'il puisse être pertinent de se demander s'il existe une configuration « puits » [e], ou de façon complémentaire si toute configuration d'un réseau non déterministe possède au moins une image. Toutefois, cela a pour but de montrer que les questions d'encodage et de calcul des images ne sont pas anodines.

Théorème 3.3. *Si l'alphabet est fixé, alors* **Vide-nondet** *est* NP*-complet, sinon* **Vide-nondet** *est* $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$*-complet.*

Démonstration. À alphabet fixé, le problème est dans NP : deviner $x \in [\![q]\!]^n$ et vérifier $f(x) = \emptyset$ en temps polynomial (théorème 3.1). La NP-difficulté est obtenue dans le cas booléen avec une réduction depuis **SAT**, en réutilisant la construction du théorème 3.2 et en ajoutant à r_n une porte de négation à la fin du calcul réalisé par le circuit. On a alors $f(x) = \emptyset$ si et seulement si φ est satisfaisable.

À alphabet non fixé, le problème est dans $\mathsf{NP}^{\mathsf{NP}}$: deviner $x \in [\![q]\!]^n$ et vérifier $f(x) = \emptyset$ grâce à un appel à l'oracle pour **Image-nonvide-nondet** qui est dans NP (on inverse ensuite la réponse de l'oracle, ce qui correspond à utiliser $\mathsf{NP}^{\mathsf{NP}} = \mathsf{NP}^{\mathsf{coNP}}$). Pour la $\mathsf{NP}^{\mathsf{NP}}$-difficulté, on réduit depuis $\exists\forall$**-SAT** : étant donnée une formule propositionnelle φ à n variables et un entier $s \in [n]$, il s'agit de décider si $\exists \tilde{x} \in \{0,1\}^s : \forall \tilde{x}' \in \{0,1\}^{n-s} : \tilde{x}\tilde{x}' \models \varphi$, avec $\tilde{x}\tilde{x}' \in \{0,1\}^n$ la concaténation de $\tilde{x}$ et $\tilde{x}'$. Ce problème est $\mathsf{NP}^{\mathsf{NP}}$-complet [Pap94, théorème 17.10]. On construit le réseau non déterministe uniforme à 1 automate d'alphabet $q = 2^{\max\{s,n-s\}}$, dont la relation locale est donnée par le circuit qui prend en entrée $x, y \in [\![q]\!]^1$ et reproduit le circuit de φ sur les s premiers bits de x et les $n-s$ premiers bits de y (pour un total de n bits d'entrées utilisés), et on ajoute une porte négation pour inverser le bit de sortie. On a ainsi :

$$\forall x, y \in [\![q]\!]^1 : \quad y \notin f(x) \iff x_{[s]}\, y_{[n-s]} \models \varphi,$$

avec $x_{[s]}\, y_{[n-s]}$ la concaténation des s premiers bits de x et des $n-s$ premiers bits de y, ce qui conclut la démonstration. Dans le problème **Vide-nondet**, la partie existentielle de l'instance φ, s de $\exists\forall$**-SAT** porte sur x, et la partie universelle sur y. □

e. Une configuration x telle que $f(x) = \emptyset$ est différente d'un point fixe non déterministe, que l'on définit par $x \in f(x)$ ou bien $f(x) = \{x\}$, voir le chapitre 5.1 sur les points fixes.

De n à 1 automate. On peut remarquer que la réduction du théorème 3.3 utilise encore une fois un seul automate. Il se trouve que dans la cas d'un alphabet non fixé, déterministe ou non déterministe, on peut transformer tout réseau f en un réseau « équivalent » g sur un seul automate. Cette équivalence est formalisée comme suit : étant donné f un réseau d'automates (déterministe ou non déterministe) sur l'espace de configurations X (uniforme ou non uniforme), il existe g un réseau $|X|$-uniforme à 1 automate tel que leurs dynamiques sont reliées par $\mathscr{G}_f = \theta(\mathscr{G}_g)$, où θ est un renommage des sommets selon une bijection triviale entre $[|X|]$ et X. Cette transformation (entre circuits) est de plus réalisable en temps polynomial car les alphabets sont encodés en binaire. On sent ici que l'essence des réseaux d'automates, fondés sur les interactions locales, est perdue. Il sera ainsi préférable, autant que possible, de travailler à alphabet fixé.

Alphabet en entrée et graphes succincts. Les considérations précédentes nous mènent à discuter des encodages succincts de graphes. Un *graphe orienté succinct* $G = (V, A)$ est encodé par un circuit booléen à $2 \cdot \lceil \log |V| \rceil$ entrées et une sortie, qui indique pour chaque paire de sommets $x, y \in V$ si $(x, y) \in A$ ou non [GW83]. Bien que ce ne soit pas le cas pour tout graphe, le plus petit circuit qui encode G peut être beaucoup plus petit que sa matrice d'adjacence. Dans les réseaux d'automates, lorsque l'alphabet n'est pas fixé ou que sa taille n'est pas bornée, pour tout graphe succinct $G = (V, A)$ représenté par un circuit C, on a le réseau d'automates f non déterministe $|V|$-uniforme à 1 automate, donc le circuit de la relation locale est C, et qui est tel que sa dynamique $\mathscr{G}_f$ est exactement le graphe G. Encore une fois, c'est la localité des interactions qui fait la particularité des réseaux d'automates. Par exemple, un réseau à n automates a pour dynamique un graphe dont le nombre de sommets peut être décomposé comme le produit de n nombres entiers positifs (les tailles des alphabets de chaque automate). Ce point sera rediscuté au chapitre 6.3.

Réseau non déterministe « déterministe ». Comme illustré sur la figure 3.1, pour tout réseau déterministe f, il existe un réseau non déterministe g sur le même ensemble d'automates et les mêmes alphabets, tel que $\mathscr{G}_g = \mathscr{G}_f$ (on construit les circuits des relations locales $r_i(x) = \{f_i(x)\}$ à partir des fonctions locales de f, en calculant sur l'entrée x, a la sortie booléenne correspondant à $f_i(x) = a$). Réciproquement, on dira qu'un réseau d'automates non déterministe g est *semi-déterministe* lorsqu'il existe f déterministe sur le même ensemble d'automates et les mêmes alphabets, tel que $\mathscr{G}_f = \mathscr{G}_g$. On remarque aisément que g est semi-déterministe si et seulement si $\mathscr{G}_g$ a degré sortant 1, ce qui est le cas si et seulement si $|r_i(x)| = 1$ pour tout automate i et toute configuration x. Cependant, cette propriété est difficile à tester.

Semi-déterministe (**Semi-det**) *Entrée :* un RA f non déterministe (circuits des relations locales). *Question :* f est-il semi-déterministe ?

À alphabet non fixé, on reformule le problème sous une forme épurée nommée ∀∃!-**SAT**, présentée ci-dessous. On note $A \equiv_{\mathtt{m}}^{\mathsf{P}} B$ lorsque $A \leq_{\mathtt{m}}^{\mathsf{P}} B$ et $B \leq_{\mathtt{m}}^{\mathsf{P}} A$, c'est-à-dire lorsque les problèmes A et B ont la même complexité pour les réduction polynomiales.

Pour tout il existe un unique-**SAT** (∀∃!-**SAT**) *Entrée :* une formule propositionnelle φ à n variables, et $s \in [n]$. *Question :* est-ce que $\forall x \in \{0,1\}^s : \exists! y \in \{0,1\}^{n-s} : xy \models \varphi$? où $xy \in \{0,1\}^n$.

Théorème 3.4. *Si l'alphabet est fixé, alors* **Semi-det** *est* coNP*-complet, sinon* **Semi-det** $\equiv_m^{\mathsf{P}}$ ∀∃!-**SAT** *est dans* $\Pi_2^{\mathsf{P}} = \mathsf{coNP}^{\mathsf{NP}}$.

Démonstration. La borne supérieure de complexité à alphabet fixé $[\![q]\!]$ pour $q \in \mathbb{N}_+$ est donnée par l'algorithme naïf : vérifier non déterministiquement pour tous $i \in [n]$ et $x \in [\![q]\!]^n$ si on a $|r_i(x)| = 1$ (on branche sur les couples (i,x), puis la taille de $r_i(x)$ est calculée en évaluant le circuit de la relation locale pour tous les éléments de $[\![q]\!]$).

Pour la borne inférieure de complexité à alphabet fixé, on réduit depuis une instance φ à n variables du problème **UNSAT**, en construisant le RA non déterministe booléen f à n automates dont les relations locales sont :

- pour $i \in [n-1]$, $\forall x \in \{0,1\}^n : r_i(x) = \{1\}$ (circuit à $n+1$ entrées booléennes, $x \in \{0,1\}^n$ et $b \in \{0,1\}$, qui recopie le dernier bit en sortie),
- r_n est le circuit qui, sur l'entrée $x \in \{0,1\}^n$ et $b \in \{0,1\}$, calcule le bit de sortie $\varphi(x) \vee b$.

On a alors $|r_n(x)| = 2$ lorsque $x \models \varphi$, et $|r_n(x)| = 1$ sinon : le réseau non déterministe f est semi-déterministe si et seulement si φ n'est satisfaite par aucun $x \in \{0,1\}^n$.

On montre l'équivalence avec ∀∃!-**SAT** par deux réductions. Étant donnée une formule φ à n variables et $s \in [n]$, on construit le réseau d'automates non déterministe uniforme à 1 automate d'alphabet $q = 2^{\max\{s,n-s\}}$, dont la relation locale est donnée par le circuit qui prend en entrée $x, y \in [\![q]\!]$ et reproduit le circuit de φ sur les s premiers bits de x, et les $n-s$ premiers bits de y (pour un total de n bits d'entrée utilisés). On a alors une unique lettre $a \in [\![q]\!]$ dans l'image de toute configuration $x \in [\![q]\!]^1$ (c'est-à-dire f est semi-déterministe) si et seulement si $\forall x \in \{0,1\}^s : \exists! a \in \{0,1\}^{n-s} : xa \models \varphi$. Réciproquement, soit un réseau d'automates non déterministe q-uniforme à n automates dont les relations locales sont r_i pour $i \in [n]$, et dont les circuits ont $(n+1) \cdot \lceil \log q \rceil$ bits d'entrée et un bit de sortie. On notera $\tilde{r}_i(x, a_i) \in \{0,1\}$ le résultat du circuit de r_i pour l'entrée $x \in \{0,1\}^{n \cdot \lceil \log q \rceil}$ et $a_i \in \{0,1\}^{\lceil \log q \rceil}$. On construit la formule φ à $2n \cdot \lceil \log q \rceil$ variables booléennes, définie par :

$$\varphi(x, a_1, a_2, \ldots, a_n) = \bigwedge_{i \in [n]} \tilde{r}(x, a_i) \quad \wedge \quad \bigwedge_{i \in [n]} (a_i < q)$$

avec la quantification universelle qui porte sur x, c'est-à-dire $s = n \cdot \lceil \log q \rceil$. La seconde partie de la formule s'assure que les entrées ignorées par les circuits (entre q et $2^{\lceil \log q \rceil}$) ne satisfont pas la formule φ. On a alors un réseau avec une unique image pour toute

configuration si et seulement si, pour toute valuation partielle $x \in \{0,1\}^s$, il existe un unique n-uplet $(a_1, a_2, \ldots, a_n)$ avec $a_i \in \{0,1\}^{\lceil \log q \rceil}$ tel que $x a_1 a_2 \ldots a_n \models \varphi$.

L'appartenance à $\mathsf{coNP}^{\mathsf{NP}}$ est donnée par le lemme A.1, qui utilise deux appels à l'oracle NP pour décider si il existe un unique y (pour x fixé non déterministiquement, est-ce qu'il existe au moins une solution ? puis est-ce qu'il existe au moins deux solutions ? la première réponse doit être « oui », la seconde « non »). □

Dans l'énoncé du théorème 3.4, à alphabet non fixé, on ne caractérise pas la complexité de **Semi-det** par une classe, mais par l'équivalence avec le problème ∀∃!-**SAT**. On pose alors la question suivante, discutée en annexe A.

Question ouverte 3.5. *Est-ce que* ∀∃!-**SAT** *est* $\mathsf{coNP}^{\mathsf{NP}}$*-complet ?*

Asynchrone. Pour les réseaux booléens asynchrones, on encode les fonctions locales déterministes $\{0,1\}^n \to \{0,1\}$ par des circuits[f], et l'on teste encore une fois en temps polynomial si $y \in \hat{f}(x)$. Notons cependant une distinction avec le cas non déterministe : en asynchrone, on peut calculer l'ensemble des images d'une configuration $x \in \{0,1\}^n$ en temps polynomial, car il y en a au plus n et elles se trouvent toutes dans l'ensemble $\{x\} \cup \{x + e_i \mid i \in [n]\}$ qui contient $n+1$ configurations.

Graphe d'interaction (signé). Lorsqu'un problème prendra en entrée un graphe d'interaction non signé, il s'agira simplement de la matrice d'adjacence du graphe orienté, de taille $n \times n$. Dans le cas d'un graphe d'interaction signé, chaque élément de la matrice sera pris dans $\{0, +, -, \pm\}$, suivant si l'interaction est respectivement inexistante, positive, négative, ou non monotone.

3.5. Exemples

- Figure 3.2 page 41 : réseau d'automates déterministe booléen.
- Figure 3.3 page 42 : réseau d'automates déterministe uniforme.
- Figure 3.4 page 43 : réseau d'automates non déterministe booléen.
- Figure 3.5 page 44 : réseau d'automates asynchrone booléen.
- Figure 3.6 page 45 : réseau d'automates avec différents modes de mise à jour.

f. L'encodage d'un réseau d'automates asynchrone $\hat{f}$ est identique à celui du réseau déterministe f : la dynamique de chacun est définie à partir les fonctions locales f_i.

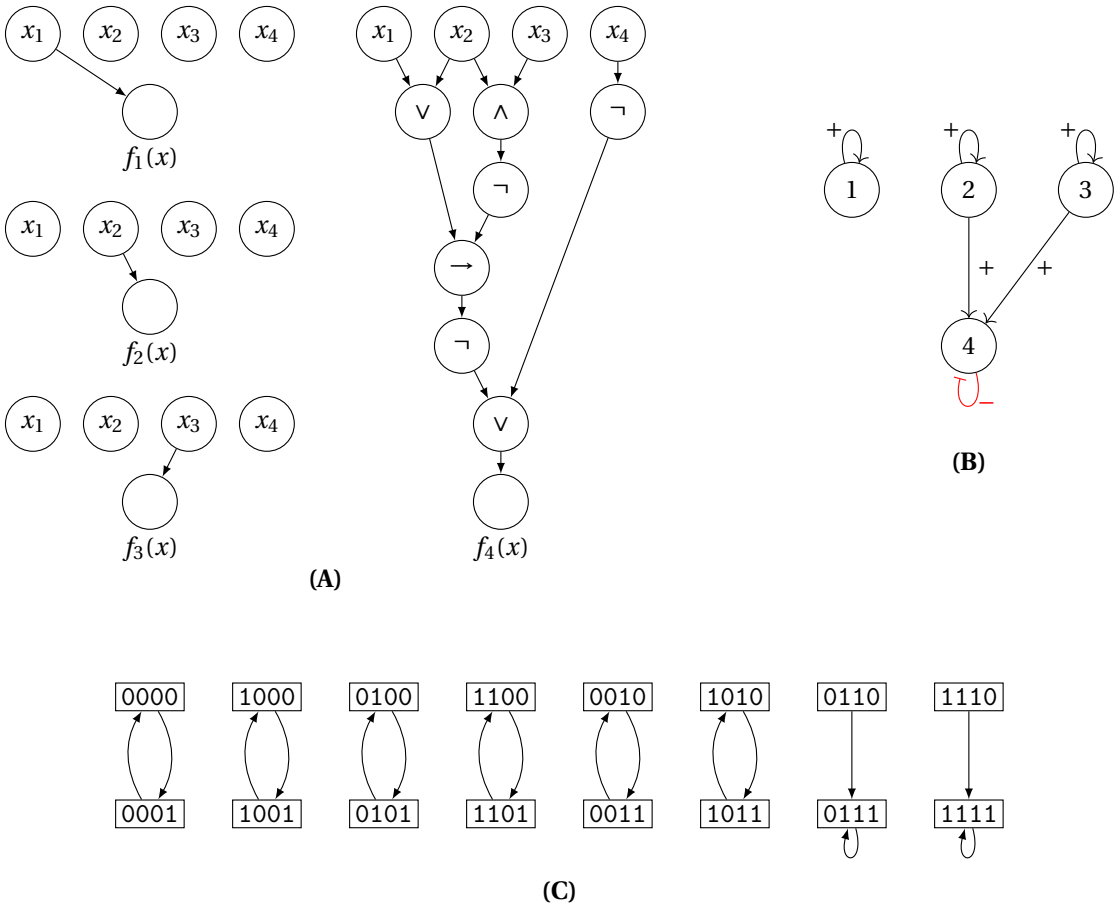


FIGURE 3.2. – Soit φ la formule propositionnelle sur trois variables :

$$\varphi(x_1, x_2, x_3) = \neg[(x_1 \vee x_2) \to \neg(x_2 \wedge x_3)]$$

qui se trouve être logiquement équivalente à $x_2 \wedge x_3$. On définit le réseau d'automates booléen $f : \{0,1\}^4 \to \{0,1\}^4$ à $n = 4$ automates, dont les fonctions locales $f_i : \{0,1\}^4 \to \{0,1\}$ sont :

$$\begin{aligned} f_1(x_1, x_2, x_3, x_4) &= x_1 \\ f_2(x_1, x_2, x_3, x_4) &= x_2 \\ f_3(x_1, x_2, x_3, x_4) &= x_3 \\ f_4(x_1, x_2, x_3, x_4) &= \varphi(x_1, x_2, x_3) \vee \neg x_4 \end{aligned}$$

données par les circuits **(A)**, dont le graphe d'interaction signé $G_f^{\pm}$ est **(B)** où l'on remarque que la présence et le signe des arcs depuis les automates $1,2,3$ vers l'automate 4 dépendent de φ, et le graphe de la dynamique $\mathscr{G}_f$ est **(C)** où l'on remarque la correspondance entre les valuations x_1, x_2, x_3 qui satisfont φ et les points fixes de f qui ont $x_4 = 1$.

On a les points fixes $\mathfrak{P}_f = \{0111, 1111\}$, six cycles limites de longueur 2, l'ensemble limite $\Omega_f = \{0,1\}^4 \setminus \{0110, 1110\}$, et les exemples d'orbite et bassin $\mathfrak{O}_f(0110) = \mathfrak{B}_f(0111) = \{0110, 0111\}$.

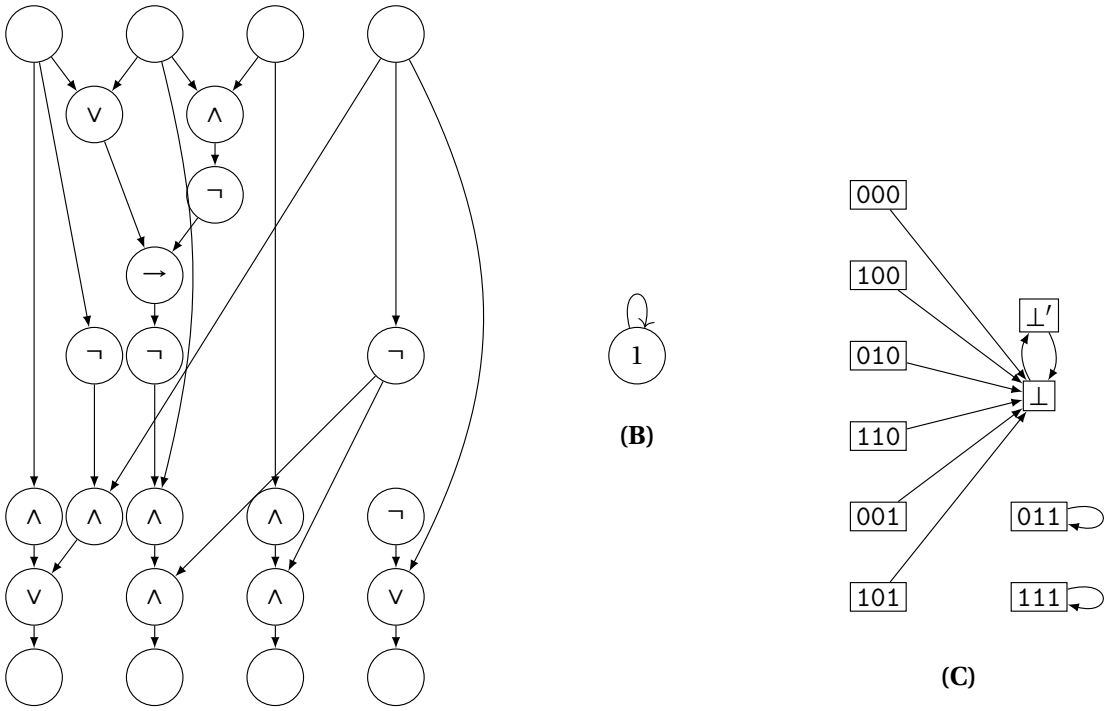


FIGURE 3.3. – Pour la même formule $\varphi(x_1,x_2,x_3) = \neg[(x_1 \vee x_2) \to \neg(x_2 \wedge x_3)] \equiv x_2 \wedge x_3$ qu'en figure 3.2, on définit le réseau d'automates 10-uniforme $g : [10]^1 \to [10]^1$ à $n = 1$ automate d'alphabet $[10]$ que l'on met en bijection avec :

$$\{000, 100, 010, 110, 001, 101, 011, 111, \bot, \bot'\},$$

dont la fonction locale $g_1 : [10]^1 \to [10]$ est :

$$g_1(a) = \begin{cases} \bot' & \text{si } a = \bot \\ \bot & \text{si } a = \bot' \\ a & \text{si } a = x_1x_2x_3 \text{ avec } \varphi(x_1,x_2,x_3) = 1 \\ \bot & \text{si } a = x_1x_2x_3 \text{ avec } \varphi(x_1,x_2,x_3) = 0 \end{cases}$$

donnée par le circuit **(A)** où les entrées 0001 et 1001 encodent respectivement les lettres $\bot$ et $\bot'$ et où les entrées plus grandes (bit de poids fort à droite) sont ignorées, dont le graphe d'interaction non signé G_g est **(B)**, et le graphe de la dynamique $\mathscr{G}_g$ est **(C)** où l'on constate comme en figure 3.2 une correspondance entre les valuations qui satisfont φ et les points fixes de g.

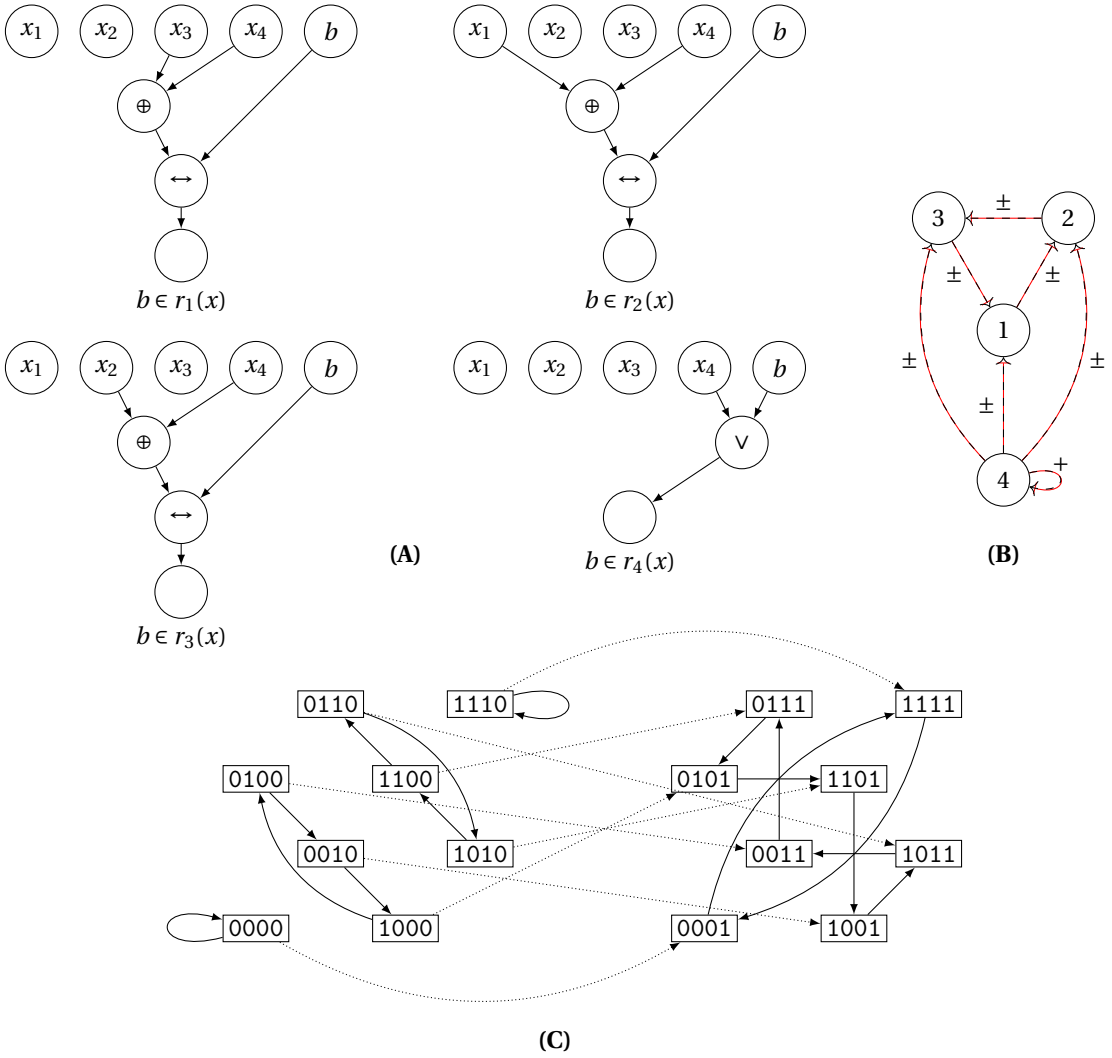


FIGURE 3.4. – On définit le réseau d'automates non déterministe $f : \{0,1\}^4 \to \mathcal{P}(\{0,1\}^4)$ à $n = 4$ automates, dont l'automate 4 contrôle le passage non déterministe d'un cycle positif à un cycle négatif sur les automates $1,2,3$. Ses relations locales $r_i : \{0,1\}^4 \to \mathcal{P}(\{0,1\})$ sont :

$$\forall x \in \{0,1\}^4 : \quad r_1(x) = \{x_3 \oplus x_4\} \quad r_3(x) = \{x_2 \oplus x_4\}$$
$$r_2(x) = \{x_1 \oplus x_4\} \quad r_4(x) = \{1, x_4\}$$

données par les circuits **(A)**, son graphe d'interaction signé $G_f^{\pm}$ est **(B)**, et le graphe de sa dynamique $\mathscr{G}_f$ est **(C)** où les arcs depuis l'hyperplan $x_4 = 0$ vers l'hyperplan $x_4 = 1$ sont dessinés en pointillé.

On a les points fixes $\mathfrak{P}_f = \{0000, 1110\}$ (configuration qui *peut* rester sur elle-même, $x \in f(x)$), aucun point fixe fort (configuration qui *doit* rester sur elle-même, $f(x) = \{x\}$), et l'ensemble limite $\Omega_f = \{0,1\}^4$ qui contient toutes les configurations.

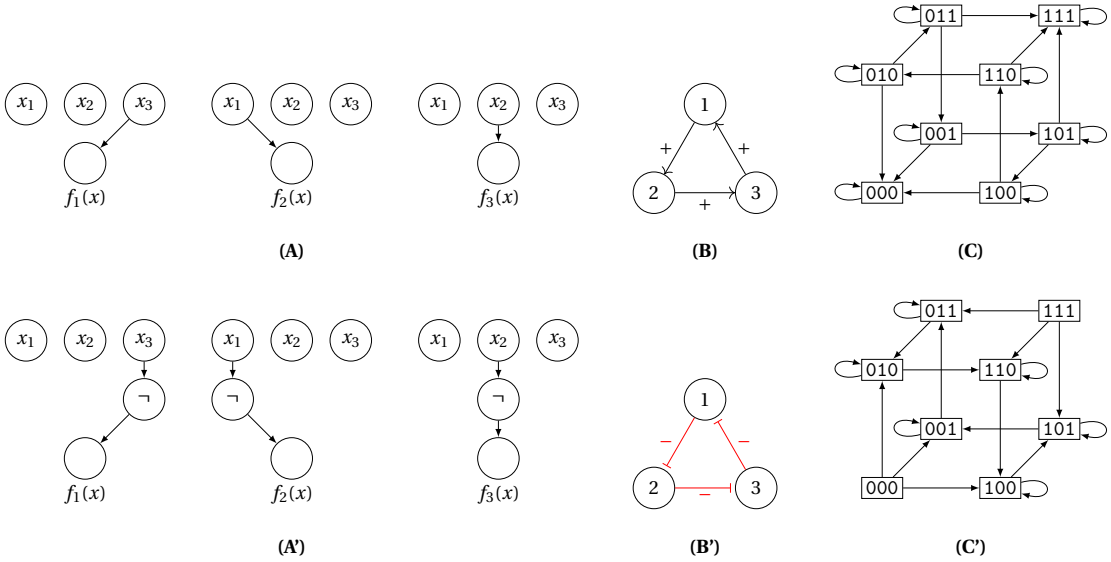


FIGURE 3.5. – On définit deux réseaux d'automates asynchrones booléens $\hat{f}, \hat{f}' : \{0,1\}^3 \to \mathcal{P}(\{0,1\}^3)$ à chacun $n = 3$ automates, qui correspondent respectivement à un cycle positif et à un cycle négatif. Leurs fonctions locales $f_i, f_i' : \{0,1\}^3 \to \{0,1\}$ sont :

$$\begin{aligned} f_1(x) &= x_3 & f_1'(x) &= \neg x_3 \\ f_2(x) &= x_1 & f_2'(x) &= \neg x_1 \\ f_3(x) &= x_2 & f_3'(x) &= \neg x_2 \end{aligned}$$

données par les circuits **(A)** et **(A')**, leurs graphes d'interactions signés $G^{\pm}_f, G^{\pm}_{f'}$ sont **(B)** et**(B')**, et les graphes de leurs dynamiques $\mathscr{G}_{\hat{f}}, \mathscr{G}_{\hat{f}'}$ sont **(C)** et **(C')**.

On a les points fixes $\mathfrak{P}_{\hat{f}} = \{0,1\}^3$ et $\mathfrak{P}_{\hat{f}'} = \{0,1\}^3 \setminus \{000, 111\}$ (configuration qui *peut* rester sur elle-même, $x \in \hat{f}(x)$), $\hat{f}$ possède deux points fixes forts $000, 111$ (configuration qui *doit* rester sur elle-même, $\hat{f}(x) = \{x\}$), et les ensembles limites $\Omega_{\hat{f}} = \{0,1\}^3$ et $\Omega_{\hat{f}'} = \{0,1\}^3 \setminus \{000, 111\}$.

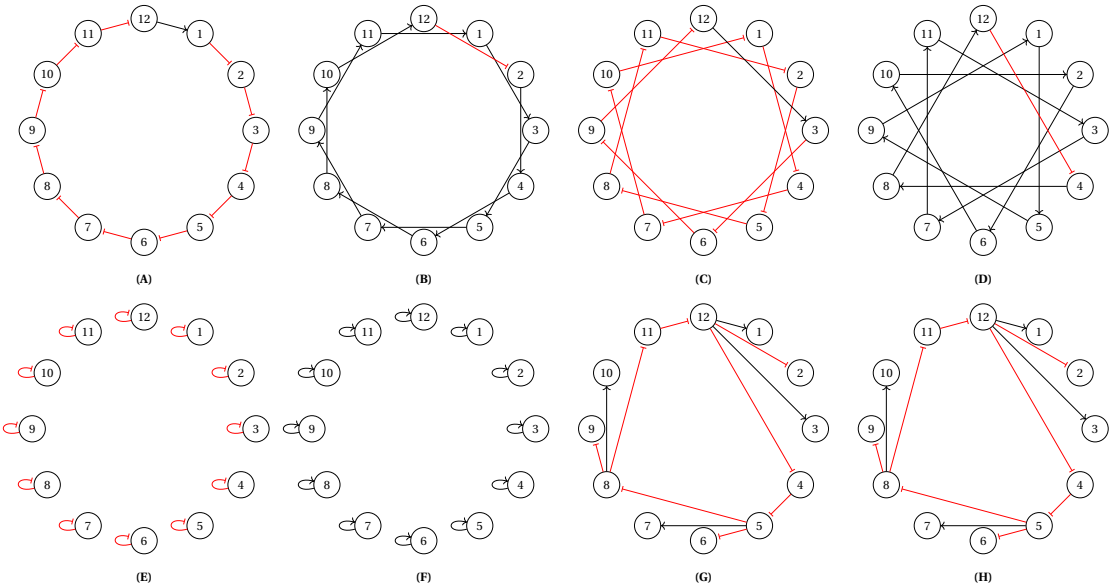


FIGURE 3.6. – On définit le RAB déterministe $f : \{0,1\}^{12} \to \{0,1\}^{12}$ à $n = 12$ automates correspondant à un cycle négatif dont tous les arcs sont négatifs sauf $(12,1)$. Ses fonction locales $f_i : \{0,1\}^{12} \to \{0,1\}$ sont :

$$f_1(x) = x_{12}, \text{ et } f_{i+1}(x) = \neg x_i \text{ pour } i \in [11].$$

Son graphe d'interaction signé $G^{\pm}_f$ est **(A)**, il a degré entrant 1 pour tout sommet. On dessine, pour chacun des huit modes de mise à jour β_j pour $j \in [8]$ ci-dessous, le graphe d'interaction signé de $f^{[\beta_j]}$ qui a également degré entrant 1 pour tout sommet (il s'agit d'un invariant pour l'application de tout mode de mise à jour).

(A) $G^{\pm}_{f^{[\beta_1]}}$ pour $\beta_1 = ([12])$

(B) $G^{\pm}_{f^{[\beta_2]}}$ pour $\beta_2 = ([12],[12])$

(C) $G^{\pm}_{f^{[\beta_3]}}$ pour $\beta_3 = ([12],[12],[12])$

(D) $G^{\pm}_{f^{[\beta_4]}}$ pour $\beta_4 = ([12],[12],[12],[12])$

(E) $G^{\pm}_{f^{[\beta_5]}}$ pour $\beta_5 = ([12],\dots,[12])$ avec 12 fois le bloc $[12]$

(F) $G^{\pm}_{f^{[\beta_6]}}$ pour $\beta_6 = ([12],\dots,[12])$ avec 24 fois le bloc $[12]$

(G) $G^{\pm}_{f^{[\beta_7]}}$ pour $\beta_7 = (\{1,12\},\{2,5,6,9\},\{3,7\},\{4,8,10\},\{11\})$

(H) $G^{\pm}_{f^{[\beta_8]}}$ pour $\beta_8 = (\{1,12\},\{2,5,6,9\},\{3,7,10\},\{4,8,11\})$

Les modes de mise à jour β_j sont équilibrés pour tout $j \in [8]$, et $\beta_1, \beta_7, \beta_8$ sont bloc-séquentiels. On remarque que $\beta_1 = \beta^{\mathtt{par}}$ correspond au mode parallèle avec $G^{\pm}_f = G^{\pm}_{f^{[\beta^{\mathtt{par}}]}}$, que $G^{\pm}_{f^{[\beta_2]}}$ a deux composantes fortement connexes, que $G^{\pm}_{f^{[\beta_3]}}$ a trois composantes fortement connexes, que $G^{\pm}_{f^{[\beta_4]}}$ a quatre composantes fortement connexes, que $f^{[\beta_5]}$ est la négation, que $f^{[\beta_6]}$ est l'identité, et que $G^{\pm}_{f^{[\beta_7]}} = G^{\pm}_{f^{[\beta_8]}}$ (pour les modes bloc-séquentiels on obtient la dynamique d'un cycle de même signe et plus petite taille, comme démontré dans [GN10]).

3.6. Éléments de littérature

La littérature « traditionnelle » sur les réseaux d'automates se concentre majoritairement sur les relations structurelles entre le graphe d'interaction et la dynamique, avec des outils de théorie des graphes. On peut voir un RA f comme la description succincte du graphe $\mathscr{G}_f$ de sa dynamique par les circuits des fonctions locales, ou par son graphe d'interaction G_f (dans ce dernier cas on peut obtenir plusieurs dynamiques, et possiblement un très grand nombre, auxquelles nous nous intéresserons au chapitre 7). Plusieurs de nos travaux abordent directement cette vision, notamment la formulation générale de problèmes en logiques FO et MSO du chapitre 6, ainsi que la preuve de NEXPTIME-difficulté en fin de chapitre 7.3. Par *relation structurelle* on entend des énoncés de la forme :

$$\#P(G_f) \leq \#Q(\mathscr{G}_f) \leq \#R(G_f)$$

avec $\#P, \#Q, \#R$ des comptages d'objets combinatoires graphiques, ou :

$$P(G_f) \implies Q(\mathscr{G}_f)$$

avec P, Q des propriétés graphiques. On observe le lien suivant avec la théorie de la complexité algorithmique : si la transformation de G_f aux objets graphiques considérés est calculable en temps polynomial, et que les bornes sont suffisamment fines, alors il peut s'agir d'une réduction Turing $\leq_{\mathtt{T}}^{\mathsf{P}}$ ou many-one $\leq_{\mathtt{m}}^{\mathsf{P}}$. Donc, si les propriétés des objets graphiques sont connues pour être difficiles à compter ou décider, on peut en déduire que le problème associé sur les réseaux d'automates est tout aussi difficile. Cependant ce n'est en général pas l'angle adopté, l'objectif de ces travaux étant d'apporter des éléments de compréhension via des liens avec des objets « classiques » de théorie des graphes. En voici un exemple auquel j'ai contribué, qui relie la simulation d'un RAB parallèle par un RAB séquentiel, à la coloration de graphes [Bri+17] : étant donné un RAB synchrone $f : \{0,1\}^n \to \{0,1\}^n$ et un mode de mise à jour séquentiel β, le plus petit nombre m d'automates d'un RAB g tel que $\pi \circ g^{[\beta]} = f \circ \pi$ avec $\pi : \{0,1\}^m \to \{0,1\}^n$ une simple projection, est $n + \chi(C(f,\beta))$ avec $\chi(C(f,\beta))$ le nombre chromatique du *graphe de confusion*[g] $C(f,\beta)$ associé à f et β.

Le plus fameux résultat en ce sens est sans conteste le suivant (la démonstration est assez élémentaire, en considérant un ordre topologique sur G_f).

Théorème 3.6 ([Rob80])**.** *Pour tout RA f de taille n, si G_f est acyclique alors $\mathscr{G}_f$ converge vers un unique point fixe en au plus n étapes ($\exists x : \forall y : f^n(y) = x$, et donc $\Omega_f = \{x\}$).*

g. Le graphe de confusion $C(f,\beta)$ pour $f : \{0,1\}^n \to \{0,1\}^n$ et un mode de mise à jour séquentiel $\beta = (B_1, \ldots, B_n)$ (qui est une permutation de $[n]$ pour laquelle on notera $B_i = \{\beta_i\}$), a pour ensemble de sommets $\{0,1\}^n$, et l'arête $\{x,y\}$ lorsqu'il existe $i \in [n]$ tel que $f^{[(\{\beta_1,\ldots,\beta_i\})]}(x) = f^{[(\{\beta_1,\ldots,\beta_i\})]}(y)$. Intuitivement, les arêtes correspondent à des situations nécessitant un bit de mémoire auxiliaire (comme pour l'échange de deux bits, qui requiert séquentiellement un troisième bit), et une coloration permet de répartir l'utilisation des bits supplémentaires. Le nombre chromatique correspond à l'optimal.

On peut également mentionner les bornes suivantes, qui s'intéressent à des quantités qui nous occuperons au chapitre 7.3, mais sous l'œil structurel.

Théorème 3.7 ([Rii07 ; Ara08 ; ARS17])**.** *Le nombre maximum de points fixes possible pour un RAB f dont le graphe d'interaction est G, est borné supérieurement par :*

- $2^{\tau(G)}$ *dans le cas non signé,*
- $2^{\tau^+(G)}$ *dans le cas signé,*

et est borné inférieurement par :

- $\nu(G)+1$ *dans le cas non signé monotone,*
- $2^{\nu^*(G)}$ *dans le cas signé monotone,*

avec $\tau(G)$ le transversal number *(plus petit feedback vertex set), $\tau^+(G)$ le* positive transversal number *(plus petit feedback vertex set positif), $\nu(G)$ le* packing number *(nombre maximum de cycles disjoints), et $\nu^*(G)$ le* special packing number*. Notons que $\nu^*(G) \leq \nu(G) \leq \tau(G)$ et $\tau^+(G) \leq \tau(G)$ pour tout graphe orienté G.*

Les deux résultats suivants s'expriment quant à eux dans l'autre sens, de la dynamique $\mathscr{G}_f$ vers le graphe d'interaction G_f ou $G_f^{\pm}$.

Théorème 3.8 ([Tho81 ; RRT08 ; RC07 ; Ric10 ; Nou12 ; Sen12])**.** *Pour tout RA f, si $\mathscr{G}_f$ a au moins deux points fixes, alors $G_f^{\pm}$ contient un cycle positif.*

Les deux résultats précédents sont vrais dans le cas asynchrone et pour tout mode de mise à jour bloc-séquentiel, mais le second est faux pour les modes de mise à jour déterministes équilibrés où un automate peut être mis à jour plus d'une fois au cours d'une itération (voir le contre-exemple en remarque 8.2 et la figure 3.6 **(F)**).

Théorème 3.9 ([Tho81 ; Ric10])**.** *Pour tout RA f asynchrone, si $\mathscr{G}_f$ possède au moins un cycle limite de taille au moins deux, alors $G_f^{\pm}$ contient un cycle négatif.*

On peut également décrire des conditions nécessaires et suffisantes sur le graphe d'interaction pour obtenir certaines propriétés dynamiques, comme la bijectivité (qui équivaut la réversibilité dans le cas fini).

Théorème 3.10 ([Gad18])**.** *Il existe un RA f bijectif avec $G_f = G$ si et seulement si G peut être couvert par des cycles sommet-disjoints.*

Les notations que nous emploierons au chapitre 7.1 exprimeront notamment l'ensemble des dynamiques que l'on peut obtenir pour un graphe d'interaction G donné. Dans les autres chapitres, l'entrée des problèmes sera un réseau d'automates f donné par des circuits encodant ses fonctions ou relations locales.

Ces résultats soulignent le paradigme suivant, qui traverse toute la littérature sur les réseaux d'automates et plus généralement les « systèmes complexes » : les cycles de rétroactions sont moteurs de la richesse comportementale.

4. Calculer le graphe d'interaction G_f

Les liens entre la dynamique d'un réseau d'automates et son graphe d'interaction sont au cœur du domaine, comme nous l'avons discuté au chapitre 3.6. Au sujet de la définition même de problèmes de décision, il faudra choisir si l'entrée du problème est un réseau d'automates (fonctions locales encodées par des circuits), ou bien un graphe d'interaction, ou bien une paire contenant ces deux objets. Ces trois possibilités ne sont en général pas équivalentes. En particulier, deux réseaux différents peuvent avoir le même graphe d'interaction (des détails et notations seront donnés au chapitre 7.1). Le graphe d'interaction donne donc « moins » d'information que les fonctions locales du réseau. Cependant nous allons voir ici que, étant donné les circuits, calculer les interactions n'est pas pour autant « facile ».

4.1. Définitions et intuition

Intuitivement, décider s'il existe une dépendance effective entre deux automates, c'est-à-dire décider si l'arc (i, j) appartient au graphe d'interaction non signé, est dans NP, car la question s'exprime existentiellement par $\exists x \in \{0,1\}^n : f_j(x) \neq f_j(x + e_i)$. Symétriquement, décider s'il n'existe pas de dépendance effective entre deux automates, c'est-à-dire décider si l'arc (i, j) n'appartient pas au graphe d'interaction non signé, est dans coNP, car la question s'exprime universellement par $\forall x \in \{0,1\}^n : f_j(x) = f_j(x + e_i)$. Nous allons montrer que ces problèmes sont respectivement NP-complet et coNP-complet, et que leur conjonction (qui correspond à la version décisionnelle de calculer le graphe d'interaction) est complète pour DP, qui est une des plus petites classes au dessus de NP et coNP.

Graphe d'interaction arc (**Inter-arc**) *Entrée :* un RAB f de taille n et deux automates $i, j \in [n]$. *Question :* est-ce que (i, j) est un arc de G_f ?
Graphe d'interaction non arc (¬**Inter-arc**) *Entrée :* un RAB f de taille n et deux automates $i, j \in [n]$. *Question :* est-ce que (i, j) n'est pas un arc de G_f ?
Graphe d'interaction (**Inter**) *Entrée :* un RAB f de taille n et un graphe orienté G. *Question :* est-ce que $G_f = G$?

Les théorèmes 4.2, 4.3 et 4.4 ci-dessous nous donnent le résultat suivant.

Théorème 4.1. *On a :*
- **Inter-arc** *est* NP*-complet,*
- ¬**Inter-arc** *est* coNP*-complet,*
- **Inter** *est* DP*-complet.*

On peut intuitivement interpréter le dernier point du théorème 4.1 de la façon suivante : le problème fonctionnel qui consiste, étant donné un RAB f (circuits des fonctions locales), à calculer son graphe d'interaction G_f, est à la fois NP-difficile et coNP-difficile, mais « pas plus difficile que cela ». Remarquons également que l'ajout des signes au graphe d'interaction ne modifie pas la complexité de ces problèmes.

4.2. Variables essentielles (Crama-Hammer)

Dans la terminologie du livre *Boolean Functions* de Crama et Hammer [CH11], une fonction locale $f_j : \{0,1\}^n \to \{0,1\}$ est une *fonction booléenne*, et un arc (i,j) dans G_f correspond au fait que x_i est une *variable essentielle* de f_j. On peut reformuler les problèmes précédents en ces termes. Afin d'éviter les confusions, nous garderons l'indice j dans f_j, bien qu'il soit inutile. Une fonction booléenne est toujours encodée par un circuit booléen.

Problème de variable essentielle (**Ess-var**) *Entrée :* une fonction booléenne $f_j : \{0,1\}^n \to \{0,1\}$, et $i \in [n]$. *Question :* est-ce que x_i est une variable essentielle de f_j ?

Problème de variable inessentielle (¬**Ess-var**) *Entrée :* une fonction booléenne $f_j : \{0,1\}^n \to \{0,1\}$, et $i \in [n]$. *Question :* est-ce que x_i est une variable inessentielle de f_j ?

Problème de l'ensemble des variables essentielles (**Ess-set**) *Entrée :* une fonction booléenne $f_j : \{0,1\}^n \to \{0,1\}$, et $S \subseteq [n]$. *Question :* est-ce que S est exactement l'ensemble des variables essentielles de f_j ?

Théorème 4.2. *En notant $A \equiv_m^{\mathsf{P}} B$ lorsque $A \leq_m^{\mathsf{P}} B$ et $B \leq_m^{\mathsf{P}} A$, on a :*
- **Inter-arc** $\equiv_m^{\mathsf{P}}$ **Ess-var**,
- ¬**Inter-arc** $\equiv_m^{\mathsf{P}}$ ¬**Ess-var**,
- **Inter** $\equiv_m^{\mathsf{P}}$ **Ess-set**.

Démonstration. Pour les deux premiers points, les réductions de **Inter-arc**, ¬**Inter-arc** vers **Ess-var**, ¬**Ess-var** sont évidentes en prenant simplement le circuit de f_j parmi f. Les réductions réciproques sont obtenues, à partir d'une instance (f_j, i) sur n variables, en complétant le réseau d'automates avec des fonctions locales $f_{j'}$ arbitraires (la constante 0) pour $j' \neq j$, et en posant la question sur l'arc (i,j).

La réduction **Inter** $\leq_{\mathrm{m}}^{\mathsf{P}}$ **Ess-set** demande à combiner les questions sur tout le graphe d'interaction en une question sur les variables essentielles d'une seule formule. Étant donné un RAB f de taille n et un graphe orienté G à n sommets, on construit la fonction booléenne $g : \{0,1\}^{n^2} \to \{0,1\}$ comme :

$$g(x^1, \dots, x^n) = \bigoplus_{j \in [n]} f_j(x^j),$$

avec $\bigoplus$ le xor, $x^j \in \{0,1\}^n$ pour tout $j \in [n]$, et $g(x^1, \dots, x^n)$ un raccourci pour $g(\tilde{x})$ avec $\tilde{x}$ la concaténation de $x^1, \dots, x^n$. En prenant :

$$S = \left\{ x_i^j \;\middle|\; (i,j) \text{ est un arc de } G \right\},$$

on obtient bien pour chaque $i, j \in [n]$ que (i,j) est un arc de G_f (ou de façon équivalente x_i est une variable essentielle de f_j) si et seulement si x_i^j est une variable essentielle de g. En effet, en fixant arbitrairement l'état des composantes des autres fonctions locales, on obtient l'implication directe. L'implication réciproque est obtenue car chaque variable x_i^j apparaît uniquement dans la fonction locale f_j.

Enfin, pour **Ess-set** $\leq_{\mathrm{m}}^{\mathsf{P}}$ **Inter**, on complète le réseau d'automates avec des fonctions locales constantes $f_{j'}(x) = 0$ pour chaque $j' \neq j$, et on construit le graphe sur n sommets avec l'ensemble d'arcs $\{(i,j) \mid i \in S\}$. Les fonctions locales ajoutées correspondent bien au graphe (aucun arc ne pointe vers $j' \neq j$) et donc la question posée est restreinte à f_j et à l'ensemble de variables S. □

4.2.1. Une variable (Ess-var et ¬Ess-var)

Les classifications de la complexité des problèmes **Ess-var** et **¬Ess-var** sont des cas d'école, dont nous incluons tout de même une démonstration (la preuve de [CH11] présente une réduction Turing).

Théorème 4.3 ([CH11, théorème 1.32 page 43])**.** *On a :*

- **Ess-var** *est* NP*-complet,*
- **¬Ess-var** *est* coNP*-complet.*

Démonstration. Nous prouvons le résultat de NP-complétude. L'autre problème étant complémentaire, la seconde partie de l'énoncé suit immédiatement. Le problème **Ess-var** est dans NP : deviner $x \in \{0,1\}^n$ et vérifier $f_j(x) \neq f_j(x + e_i)$.

La NP-difficulté est obtenue avec une réduction many-one polynomiale depuis **SAT**. Soit φ une instance de **SAT** sur $n-1$ variables (encodée par un circuit). On construit le circuit de la fonction booléenne $f_j(x) = \varphi(x_{[n-1]}) \wedge x_n$ sur n variables, en prenant $i = n$. Étant donnés $\tilde{x} \in \{0,1\}^{n-1}$ et $b \in \{0,1\}$, notons $\tilde{x}b \in \{0,1\}^n$ la configuration telle que $(\tilde{x}b)_{[n-1]} = \tilde{x}_{[n-1]}$ et $(\tilde{x}b)_n = b$. Si φ est satisfaite par $\tilde{x} \in \{0,1\}^{n-1}$ alors on a $f_j(\tilde{x}0) = 0 \neq 1 = f_j(\tilde{x}1) = f_j(\tilde{x}0 + e_n)$, donc x_n est essentielle. Si x_n est essentielle, alors on doit avoir $f_j(\tilde{x}1) = 1$ pour un $\tilde{x} \in \{0,1\}^{n-1}$, qui satisfait φ par définition de f_j. □

4.2.2. Un ensemble de variables (Ess-set)

Rappelons que la classe DP est l'ensemble des langages $L = L_1 \cap L_2$ avec $L_1 \in \mathsf{NP}$ et $L_2 \in \mathsf{coNP}$. Elle a été introduite par Papadimitriou et Yannakakis dans [PY84]. Intuitivement, DP correspond à la plus simple classe incluant NP et coNP (autre que $\mathsf{NP} \cup \mathsf{coNP}$) : il s'agit de la conjonction de questions dans NP et de questions dans coNP. La complexité de **Ess-set** n'atteint pas le niveau suivant Δ_2^{P} de la hiérarchie polynomiale, parce que les questions sur l'essentialité des variables sont « indépendantes » les unes des autres.

Théorème 4.4. **Ess-set** *est* DP*-complet.*

Démonstration. Le problème est dans DP : il correspond à une intersection de **Ess-var** (dans NP) et **Ess-¬var** (dans coNP). Plus formellement, soit **Ess-var-S** le problème de décider, sur l'entrée (f_j, S), si x_i est une variable essentielle de f_j pour tout $i \in S$. Son appartenance à NP est toujours évidente avec une formulation existentielle. Et soit **¬Ess-var-S** le problème de décider, sur l'entrée (f_j, S), si x_i est une variable inessentielle de f_j pour tout $i \in [n] \setminus S$. De même, son appartenance à coNP est évidente avec une formulation universelle. Alors **Ess-set** = **Ess-var-S** $\cap$ **¬Ess-var-S**.

Nous présentons une réduction many-one en temps polynomial depuis le problème **SAT-UNSAT** qui est DP-complet [PY84] : étant données deux formules booléennes φ, φ', est-il vrai que φ est satisfaisable et φ' n'est pas satisfaisable ? Remarquons que la réduction n'est pas complètement triviale d'après nos considérations sur **Ess-var** dans le théorème 4.3, car maintenant *toutes* les variables sont impliquées dans la décision pour **Ess-set** : ce problème demande si toutes les variables de S sont essentielles *et* toutes les variables de $[n] \setminus S$ sont inessentielles.

Étant données φ sur les variables $x_1, \dots, x_n$ et φ' sur les variables $x_{2n+1}, \dots, x_{3n}$, on construit le circuit de la fonction booléenne f_j sur $4n$ variables $x_1, \dots, x_{4n}$, définie par :

$$f_j(x) = f_j^+(x) \wedge f_j^-(x) \quad \text{avec} \quad f_j^+(x) = \varphi(x_{[n]}) \wedge \bigwedge_{i \in [n]} (x_i \leftrightarrow x_{n+i})$$
$$f_j^-(x) = \neg\Big(\varphi'(x_{\{2n+1,\dots,3n\}}) \wedge \bigwedge_{i \in [n]} (x_{2n+i} \leftrightarrow x_{3n+i})\Big),$$

où $b \leftrightarrow b'$ correspond comme attendu à $(b \vee \neg b') \wedge (\neg b \vee b')$. Et nous prenons $S = [2n]$. Voir la figure 4.1 pour une illustration.

On argumente d'abord qu'une instance positive de **SAT-UNSAT** est envoyée sur une instance positive de **Ess-set**. Si φ est satisfaite par $\tilde{x} \in \{0,1\}^n$, alors on étend $\tilde{x}$ en une configuration sur les composantes $[2n]$ avec $\tilde{x}_{n+i} = \tilde{x}_i$ pour tout $i \in [n]$, de façon à ce que $\tilde{x} \in \{0,1\}^{2n}$ soit un certificat que toutes les variables de S sont essentielles dans f_j^+ (peu importe les $2n$ autres composantes qui sont inessentielles dans f_j^+, grâce aux $\leftrightarrow$ l'ajout de e_i pour $i \in [2n]$ flip le résultat de 1 à 0). Si φ' n'est pas satisfaisable, alors, pour tout x, on a $f_j^-(x) = 1$. Celà mène par définition de f_j à la conclusion que son ensemble de variables essentielles est exactement S.

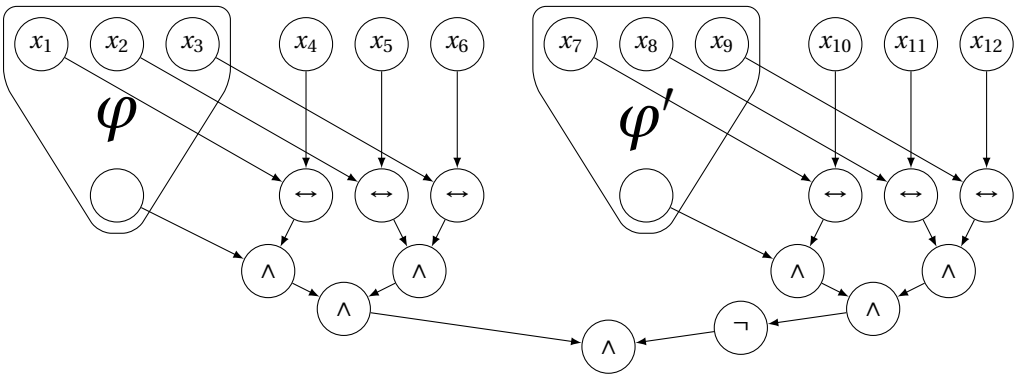


FIGURE 4.1. – Réduction de **SAT-UNSAT** à **Ess-set**, avec $n = 3$ et $S = \{1, \dots, 6\}$. Les entrées de f_j sont en haut, sa sortie en bas.

Réciproquement, supposons que $S = [2n]$ est exactement l'ensemble des variables essentielles de f_j. Premièrement, l'existence d'au moins une variable essentielle force que φ est satisfaisable, parce que $f_j(\tilde{x}) = 1$ pour un certain $\tilde{x}$ et par les conjonctions cela implique $\varphi(\tilde{x}_{[n]}) = 1$. Deuxièmement, le fait que $\{2n+1, \dots, 4n\}$ sont des variables inessentielles force que φ' n'est pas satisfaisable : par l'absurde on aurait $\tilde{x}_{[n]}$ satisfaisant φ et $\tilde{x}_{\{2n+1,\dots,3n\}}$ satisfaisant φ', et l'extension de $\tilde{x}$ en une configuration sur les composantes $[4n]$ avec $\tilde{x}_{n+i} = \tilde{x}_i$ et $\tilde{x}_{3n+i} = \tilde{x}_{2n+i}$ pour tout $i \in [n]$ serait un certificat que toutes les variables de $[4n] \setminus [2n]$ sont également essentielles (par le même argument que précédemment, grâce aux $\leftrightarrow$). □

4.3. Perspectives

D'après le théorème 4.1, si l'on donne f et G son graphe d'interaction en entrée d'un problème, il faut alors soit offrir la promesse que $G = G_f$, soit effectuer cette vérification qui a un coût « élevé » : $\mathsf{DP} \supseteq \mathsf{NP} \cup \mathsf{coNP}$. La notion de *graphe de communication* a également été introduite (voir par exemple [Río21, définition 1.4 page 9]), il s'agit d'un sur-ensemble du graphe d'interaction que l'on peut penser comme une description de l'ensemble des variables pouvant apparaître dans l'expression de chaque fonction locale (l'ensemble des voisins entrants de l'automate) sans nécessairement être essentielles (effectives).

On peut cependant remarquer que certaines promesses sur le graphe d'interaction facilitent cette vérification. On a par exemple le résultat suivant dont Guillaume Theyssier et Martín Ríos Wilson m'ont partagé l'observation, avec $\Delta(G)$ le degré entrant maximum d'un graphe orienté G.

Théorème 4.5. *Avec la promesse $\Delta(G_f) \leq d$ pour $d \in \mathbb{N}$ fixé, étant donné f (circuits des fonctions locales), on peut calculer G_f en temps $n^{\mathcal{O}(d)}$*, i.e. *en temps polynomial.*

Démonstration. Il suffit de tester naïvement, pour chaque fonction locale f_j et chaque sous-ensemble $S \subseteq [n]$ d'au plus d automates (c'est là que se trouve l'astuce car

$\binom{n}{d} < n^d$), si toutes les variables de S sont essentielles (en commençant par $|S| = d$ puis $d-1$, $d-2$, *etc*). Pour tester si toutes les variables de S sont essentielles, on peut fixer toutes les autres composantes à l'état 0 et essayer seulement $2^{|S|}$ configurations (et ajouter chaque e_i pour $i \in S$), car on sait que les autres variables devraient être inessentielles quand on trouve le bon ensemble S (c'est pour cela que l'on prend les sous-ensembles S par taille décroissante). La complexité totale est grossièrement $\mathcal{O}(n^{d+2} \cdot 2^d \cdot c)$ avec c la taille du plus grand circuit (bornée par la taille de l'entrée) à évaluer pour comparer $f_j(x)$ et $f_j(x+e_i)$. □

Quelles autres promesses sur le graphe d'interaction G_f pourraient abaisser la complexité de son calcul à partir de f ? Étant donné que la simple évaluation d'un circuit sur une entrée est P-difficile (il s'agit exactement de **CVP**, voir l'annexe B.2), il paraît hasardeux d'espérer descendre au dessous de la classe P.

Question ouverte 4.6. *Pour quelles propriétés φ la promesse que $G_f \models \varphi$ permet-elle de calculer G_f en temps polynomial ?*

Entre DP et P il y a NP et coNP : peut-on avoir des promesses qui brisent la symétrie entre décider l'existence ou la non existence d'une interaction ? On formulerait plus précisément cette question ainsi (car $\mathsf{P} \subseteq \mathsf{NP} \cap \mathsf{coNP}$).

Question ouverte 4.7. *Pour quelles propriétés φ la promesse $G_f \models \varphi$ donne-t-elle* **Inter** NP*-complet ou* coNP*-complet ?*

Peut-être n'est-il pas nécessaire de calculer tout le graphe d'interaction G_f, mais uniquement de décider s'il possède une certaine propriété. Au regard des travaux présentés au chapitre 6, on pourrait réfléchir à un résultat de la forme suivante (où les expressions entre guillemets sont à préciser).

Question ouverte 4.8. *Pour toute propriété « non triviale » φ exprimable en logique FO ou MSO, décider si un RA f donné vérifie $G_f \models \varphi$ est-il « difficile » ?*

En prenant du recul, le graphe d'interaction donne une information grossière sur les dépendances effectives entre automates : soit l'arc existe soit il n'existe pas, il s'agit d'un seul bit d'information (ou deux bits si l'on ajoute les signes $\{0,+,-,\pm\}$). Cependant, si l'on considère par exemple les fonctions $f_j(x) = x_i \oplus x_{i'} \oplus x_{i''}$ et $f_{j'}(x) = x_i \wedge x_{i'} \wedge x_{i''}$, alors on voudrait dire que j dépend « davantage » de i, i', i'' que j' car, pour tout x, changer n'importe lequel de ces états aura une influence effective sur l'état de j, ce qui n'est pas le cas pour j'. Cela correspond aux notions quantitatives d'*influence*, de *sensibilité* et de *sensibilité moyenne* exposées dans [Juk12, section 2.8]. L'influence de la i-ième variable de f_j, est définie comme :

$$\mathtt{Inf}_i(f_j) = \frac{\left|\left\{x \in \{0,1\}^n \;\middle|\; f(x+e_i) \neq f(x)\right\}\right|}{2^n}.$$

Étant donnée une fonction $f_j : \{0,1\}^n \to \{0,1\}$, les autres notions peuvent être exprimées sur le graphe non orienté bipartie $\mathfrak{G}_{f_j}$ dont les parties sont $f_j^{-1}(0)$ et $f_j^{-1}(1)$, et qui contient l'arc $(x,y) \in f_j^{-1}(0) \times f_j^{-1}(1)$ si et seulement si $\exists i \in [n] : x + e_i = y$.

- La sensibilité de f_j sur x, notée $\mathtt{s}(f_j, x)$, est le degré du sommet x dans $\mathfrak{G}_{f_j}$.
- La sensibilité de f_j, notée $\mathtt{s}(f_j)$, est le degré maximum d'un sommet de $\mathfrak{G}_{f_j}$.
- La sensibilité moyenne de f_j, notée $\mathtt{as}(f_j)$, est le degré moyen de $\mathfrak{G}_{f_j}$.

Des résultats intuitifs bornent inférieurement la sensibilité d'une fonction qui dépend effectivement de ses n variables (vérifie $\mathtt{s}(f_j) \geq \frac{1}{2}\log n - \frac{1}{2}\log\log n$ [Juk12, corollaire 2.16 page 71]), et montrent que les fonctions booléennes très sensibles nécessitent de grand circuits (un circuit de profondeur d calculant f_j a une taille exponentielle en $\mathtt{as}(f_j)^{1/(d-1)}$ [Juk12, théorème 12.13 page 351]). Ces perspectives sur une généralisation des graphes d'interaction capturant une information plus fine que l'existence ou la non existence de dépendance effective font écho à la relative pauvreté du graphe d'interaction dans le cas des réseaux d'automates non booléens. Terminons cette section sur une question plus large au sujet du graphe d'interaction.

Question ouverte 4.9. *Quelles étiquettes faut-il ajouter au graphe d'interaction d'un réseau d'automates pour en déduire des informations plus précises sur sa dynamique ?*

Au regard du fait que le graphe d'interaction complet permet d'engendrer toutes les dynamiques sauf deux (voir le théorème 7.2 du chapitre 7.1), c'est-à-dire qu'il ne donne quasiment aucune information, cette direction de recherche semble pertinente.

5. Complexité de la dynamique asymptotique I : f en entrée

Rappelons que les réseaux d'automates booléens sont des systèmes dynamiques finis sur l'espace $\{0,1\}^n$ ou plus généralement $\prod_{i\in[n]} A_i$, donc toutes les questions raisonnables[a] sont décidables, y compris concernant la dynamique asymptotique. Suivant comment elle est formellement posée, nous allons voir que la question des points fixes à elle seule peut largement traverser le spectre des classes de complexité, de P à NEXPTIME en passant par NP, $\mathsf{NP}^{\mathsf{NP}}$ et $\mathsf{NP}^{\#\mathsf{P}}$ (chapitres 5 et 7).

Nous distinguons deux parties concernant la complexité de décider des propriétés de la dynamique asymptotique d'un RAB (existence de point fixe, nombre de points fixes, cycles limites...), suivant l'entrée du problème. Dans le présent chapitre 5, l'entrée est le réseau (circuits des fonctions locales), et au chapitre 7, l'entrée sera le graphe d'interaction. Dans ces deux chapitres, nous allons considérer le mode de mise à jour parallèle, et simplement noter f pour $f^{[\beta^{\mathrm{par}}]}$.

Les questions les plus naturelles concernent l'existence de points fixes et de cycles limites d'un réseau d'automates booléens donné en entrée. Elles seront abordées dans la section 5.1, où nous verrons que les circuits des fonctions et relations locales permettent directement l'encodage de formules propositionnelles (nous ferons le lien avec les problèmes canoniques de la hiérarchie polynomiale PH). Dans la section 5.2, nous étudierons la complexité de calculer les préimages, qui correspondent à la dynamique « en arrière », dont les comptages sont difficiles. L'atteignabilité et la considération plus générale de l'ensemble limite pour les réseaux d'automates q-uniformes nous feront sauter à la PSPACE-complétude, comme nous le verrons dans la section 5.2. On remarquera entre autres que vérifier si l'ensemble limite est de taille minimale, c'est-à-dire un, est PSPACE-complet, alors que vérifier s'il est de taille maximale, c'est-à-dire 2^n dans le cas booléen, est seulement coNP-complet.

Mentionnons qu'au chapitre 6, nous présenterons un « méta-théorème » de NP- et coNP- difficulté, qui inclut les problèmes abordés ci-après portant sur la dynamique $\mathscr{G}_f$ et exprimables en logique du premier ordre (comme l'existence d'un point fixe, $\exists x : f(x) = x$). Ce méta-théorème s'appliquera dans le cas déterministe à alphabet non fixé, sauf pour les questions sur la dynamique limite où il s'appliquera aussi à alphabet uniforme fixé.

a. On pourrait imaginer des choses farfelues, comme par exemple : est-ce qu'il existe un point fixe qui encode le numéro d'une machine de Turing qui s'arrête sur l'entrée vide ?

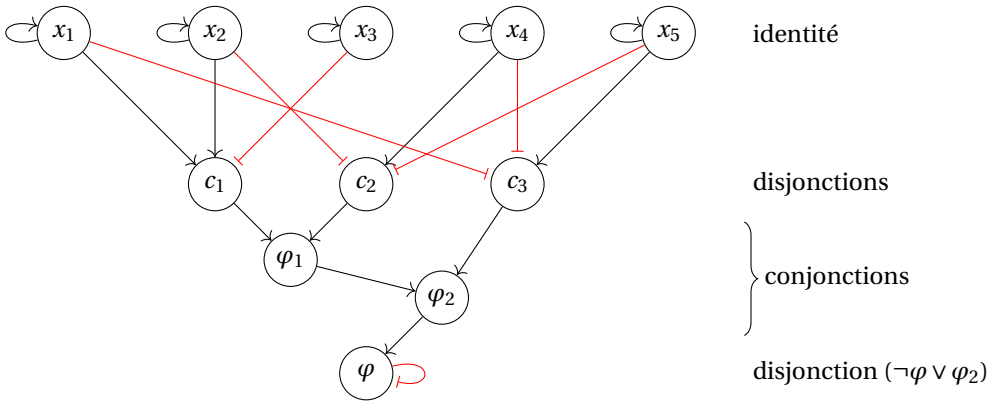


FIGURE 5.1. – Graphe d'interaction signé (arcs négatifs en rouge avec pointe plate) obtenu dans la réduction de **3-SAT** à **PF**, pour la formule $\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_4 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_4 \vee x_5)$. En notant abusivement x_i pour l'état de l'automate x_i, et en voyant une clause comme un ensemble de littéraux (ici par exemple $c_1 = \{x_1, x_2, \neg x_3\}$), les automates correspondant aux variables sont l'identité $f_{x_i}(x) = x_i$, les automates correspondant aux clauses sont des disjonctions $f_{c_j}(x) = \bigvee_{\ell_i \in c_j} \ell_i$, il y a ensuite un arbre binaire de conjonctions qui mène à une racine (ici φ_2), et enfin l'automate φ dont la fonction locale est une disjonction entre sa négation et l'automate racine de l'arbre.

5.1. Points fixes, bassins et cycles limites

Points fixes. L'existence d'un point fixe dans un réseau d'automates est un problème standard de la littérature, dont nous donnons une preuve avec notre formalisme, en commençant par le cas déterministe booléen. Rappelons que l'ensemble des points fixes d'un RA booléen déterministe est noté $\mathfrak{P}_f = \{x \in \{0,1\}^n \mid f(x) = x\}$.

Point fixe (**PF**) *Entrée :* un RAB déterministe $f : \{0,1\}^n \to \{0,1\}^n$ (circuits des fonctions locales). *Question :* est-ce que $\mathfrak{P}_f \neq \emptyset$?

Théorème 5.1 ([Alo85])**.** **PF** *est* NP*-complet, même avec la promesse* $\Delta(G_f) \leq 2$.

Idée de démonstration. L'appartenance à NP est évidente : deviner $x \in \{0,1\}^n$ et vérifier $f(x) = x$. On montre **3-SAT** $\leq_{\mathrm{m}}^{\mathrm{P}}$ **PF**. Étant donnée une formule propositionnelle φ sur les variables $x_1, \ldots, x_n$ avec m clauses, on construit un réseau d'automates booléens f de taille $n + 2m$ tel que présenté sur la figure 5.1.

Si φ est satisfaite par $x \in \{0,1\}^n$, alors on l'étend en un point fixe du réseau f en attribuant l'état 1 à tous les autres automates (parcourir f du haut vers le bas sur la figure 5.1). Si f possède un point fixe, alors tous les automates autres que $x_1, \ldots, x_n$

doivent être dans l'état 1, et la projection de ce point fixe sur $x_1, \ldots, x_n$ est une valuation qui satisfait φ (parcourir f du bas vers le haut sur la figure 5.1).

On peut abaisser le degré maximum $\Delta(G_f)$ du graphe d'interaction à 2 en décomposant les automates des clauses $c_j = \{\ell_1, \ell_2, \ell_3\}$ qui ont degré entrant 3 en deux automates c_j, c'_j avec $f_{c'_j}(x) = \ell_1 \vee \ell_2$ et $f_{c_j}(x) = c'_j \vee \ell_3$. □

Ce résultat de NP-difficulté est toujours vrai pour plusieurs restrictions sur l'expressivité des fonctions locales. Par exemple, la construction proposée donne un réseau ET/OU (la fonction locale de chaque automate est une conjonction ou une disjonction, possiblement avec des négations). Floréen et Orponen ont montré dans [FO89] la NP-complétude du problème d'existence d'un point fixe pour les *Hopfield nets* qui correspondent à des réseaux booléens dont les fonctions locales réalisent une somme pondérée comparée à un seuil (selon la terminologie contemporaine on parlerait de *réseaux de neurones*, ils incluent les réseaux ET/OU).

Kosub propose dans [Kos08] des théorèmes de dichotomie sur le problème d'existence d'un point fixe, dans lequel l'entrée du problème est un réseau d'automates booléen et son graphe d'interaction avec la promesse (implicite) qu'il s'agit bien du graphe d'interaction du réseau. Suivant l'encodage du réseau (table dé vérité, formules, circuits), et des restrictions sur la famille de fonctions locales (toujours closes par composition) et sur la famille de graphes d'interaction (toujours close par mineur), le problème d'existence d'un point fixe est dans P ou NP-complet.

Notre construction dans la preuve du théorème 5.1 possède un corollaire déjà observé sur le comptage du nombre de points fixes.

Comptage points fixes (**#PF**) *Entrée :* un RAB $f : \{0,1\}^n \to \{0,1\}^n$ (circuits des fonctions locales). *Sortie :* $\lvert\mathfrak{P}_f\rvert$.

Corollaire 5.2 ([Orp92])**.** **#PF** *est* #P*-complet, même avec la promesse* $\Delta(G_f) \leq 2$.

Idée de démonstration. L'appartenance à #P est toujours évidente avec le même algorithme, et la complétude est obtenue en remarquant que la réduction du théorème 5.1 est parcimonieuse (car dans tout point fixe de f, les automates autres que $x_1, \ldots, x_n$ sont nécessairement dans l'état 1). □

La réduction parcimonieuse de **3-SAT** à **PF** donnée en figure 5.1 est suffisamment élémentaire pour être réutilisée à l'identique pour les problèmes de satisfiabilité canoniques à tous les étages de la hiérarchie polynomiale.

Corollaire 5.3. *Étant donné un RAB f déterministe donné par ses circuits, décider*

$$Q_1 x_1 \in \{0,1\}^{s_1} : Q_2 x_2 \in \{0,1\}^{s_2} : \ldots : Q_k x_k \in \{0,1\}^{s_k} : f(x_1 x_2 \ldots x_k) = x_1 x_2 \ldots x_k$$

avec $Q_i \in \{\exists, \forall\}$ qui alternent, est complet pour Σ_k^{P} si $Q_1 = \exists$ et pour Π_k^{P} si $Q_1 = \forall$.

Démonstration. L'énoncé suppose que le réseau f a $n = \sum_{i\in[k]} s_i$ automates booléens. Dans la construction en figure 5.1, on a la correspondance $f(x) = x$ si et seulement si, avec V les automates correspondant aux variables, on a $\varphi(x_V) = \top$ et $x_i = 1$ pour tout $i \in [n] \setminus V$. On obtient ainsi le résultat lorsque la question comporte au moins une quantification universelle (sous laquelle on place les x_i pour $i \in [n] \setminus V$), et la construction vérifie encore la promesse $\Delta(G_f) \leq 2$. Pour $k = 1$ et $Q_1 = \forall$, on obtient le résultat énoncé (coNP-complétude, en réduisant depuis φ pour **UNSAT**) avec f sur n automates, dont les fonctions locales sont $f_i(x) = x_i$ pour $i \in [n-1]$, et $f_n(x) = x_n \oplus \neg\varphi(x)$. Ici seulement, on perd la promesse sur le degré du graphe d'interaction. □

Remarque 5.4. Les théorème 5.1 et corollaires 5.2 et 5.3 ci-dessus s'appliquent à l'identique aux réseaux non déterministes, lorsqu'on définit un point fixe comme une configuration qui *peut* ne pas bouger, c'est-à-dire $x \in \{0,1\}^n$ telle que $x \in f(x)$. Par exemple, pour la théorème 5.1 on adapte :

- la borne supérieure de complexité en testant $x \in f(x)$ plutôt que $f(x) = x$,
- la borne inférieure de complexité en définissant pour tout $i \in [n]$ la relation locale r_i à partir de la fonction locale f_i de façon à avoir $r_i(x) = \{f_i(x)\}$ pour tout $x \in \{0,1\}^n$: on obtient alors la même dynamique.

On peut également définir un point fixe non déterministe comme une configuration qui *doit* ne pas bouger, c'est-à-dire $x \in \{0,1\}^n$ telle que $f(x) = \{x\}$. Nous appelons ces configurations des *points fixes forts*, ils sont testés par une quantification universelle :

$$f(x) = \{x\} \iff \big[x \in f(x)\big] \wedge \big[\forall y \in \{0,1\}^n : (x \neq y) \to (y \notin f(x))\big]$$

et l'on obtient les résultats de complexité suivants.

Point fixe fort x (**PFFx-nondet**) *Entrée :* un réseau non déterministe f (circuits des relations locales) sur X, et $x \in X$. *Question :* est-ce que $f(x) = \{x\}$?

Point fixe fort (**PFF-nondet**) *Entrée :* un réseau non déterministe f (circuits des relations locales) sur X. *Question :* est-ce que $\exists x \in X : f(x) = \{x\}$?

Théorème 5.5. *Si l'alphabet est fixé, alors* **PFFx-nondet** *est dans* P *et* **PFF-nondet** *est* NP*-complet, sinon ils sont respectivement* coNP*-complet et* $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$*-complet.*

Démonstration. Pour un réseau non déterministe f à alphabet fixé, étant donné x on peut tester en temps polynomial si $f(x) = \{x\}$, car on peut calculer chacun des $r_i(x)$ dont c'est le produit cartésien en temps polynomial. Cela donne les deux bornes supérieures de complexité.

Étant donnée une formule φ à n variables, on construit les circuits des relations locales de f non déterministe booléen à n automates, avec $r_i(x) = \{x_i \oplus \neg\varphi(x)\}$ pour

tout $i \in [n]$. Si $\varphi(x) = \top$ alors $r_i(x) = \{x_i\}$ et x est un point fixe fort, sinon $r_i(x) = \{\neg x_i\}$ et x n'est pas un point fixe fort. Ceci conclut notre réduction **SAT** $\leq_{\mathrm{m}}^{\mathsf{P}}$ **PFF-nondet**.

À alphabet uniforme non fixé, les bornes supérieures de complexité sont obtenues directement par la formulation universelle du test $f(x) = \{x\}$.

On obtient la première borne inférieure (pour **PFFx-nondet**) depuis φ à n variables pour **UNSAT**, que l'on prend telle que $\varphi(0^n) = \bot$. On construit le réseau non déterministe f sur un automate d'alphabet $[2^n]$ (on voit les états comme des valuations à n variables booléennes), dont le circuit prend en entrée x, y et calcul le bit de sortie $(x = y) \vee \varphi(y)$. On a $f(0^n) = \{0^n\} \cup \{x \in [2^n] \mid \varphi(x) = \top\}$, donc φ n'est pas satisfaisable si et seulement si 0^n est un point fixe fort.

Pour le dernier résultat on part d'une instance $\varphi(x, y)$ de $\exists\forall$-**SAT**, qui pose la question $\exists x \in \{0,1\}^s : \forall y \in \{0,1\}^{n-s} : \varphi(x, y) = \top$? On construit le réseau non déterministe f sur un automate d'alphabet $[2^{\max\{s,n-s\}}]$, dont le circuit prend en entrée x, y et calcul le bit de sortie :

$$(x < 2^s) \wedge (y < 2^{n-s}) \wedge ((x = y) \oplus \neg\varphi(x, y)),$$

où $x, y \in [2^{\max\{s,n-s\}}]$ sont vus comme des valuations à s et $n - s$ variables booléennes. S'il existe $\tilde{x}$ tel que $\forall y : \varphi(\tilde{x}, y) = \top$ alors $\tilde{x}$ est un point fixe fort (les deux premières comparaisons sont utiles ici). Sinon, pour toute configuration x :

- si $\varphi(x, x) = \bot$ alors $x \notin f(x)$,
- si $\varphi(x, x) = \top$ alors il existe $\tilde{y} \neq x$ tel que $\varphi(x, \tilde{y}) = \bot$, et donc $\tilde{y} \in f(x)$.

Dans les deux cas x n'est pas un point fixe fort. □

Pour les réseaux d'automates booléens asynchrones, on obtient les mêmes complexités que dans le cas non déterministe à alphabet fixé.

Théorème 5.6. *Dans le cas booléen asynchrone,* **PF** *est* NP*-complet,* **PFF**x *est dans* P*, et* **PFF** *est* NP*-complet.*

Démonstration. Le problème **PFF**x appartient à P avec l'algorithme qui teste $f_i(x) = \{x_i\}$ pour tout $i \in [n]$ (voir table 3.1), et les bornes supérieures NP de façon analogue. Pour les bornes inférieures on réduit depuis une instance φ à n variables de **SAT**, en construisant le réseau booléen asynchrone $\hat{f}$ dont les fonctions locales sont $f_i(x) = x_i \oplus \neg\varphi(x)$ pour chaque automate $i \in [n]$. Si $\varphi(\tilde{x}) = \top$ alors $f_i(\tilde{x}) = \tilde{x}_i$ pour tout i et $\tilde{x}$ est un point fixe fort, sinon $f_i(\tilde{x}) = \neg\tilde{x}_i$ pour tout i et $\tilde{x}$ n'est pas un point fixe. □

Bassins. Dans la suite de cette section, on s'intéresse aux bassins d'attraction et aux cycles limites dans le cas déterministe.

Des résultats de Floréen et Orponen caractérisent avec des techniques similaires la complexité de questions liées au bassin d'attraction $\mathfrak{B}_f(x)$ d'un point fixe x donné (on peut commencer par vérifier en temps polynomial si x est bien un point fixe).

Théorème 5.7 ([FO89])**.** *Étant donné un RAB f déterministe, et :*

- *un point fixe x, décider si $\mathfrak{B}_f(x) \neq \{x\}$ est* NP*-complet,*
- *un point fixe x, compter $|\mathfrak{B}_f(x)|$ est* #P*-complet.*

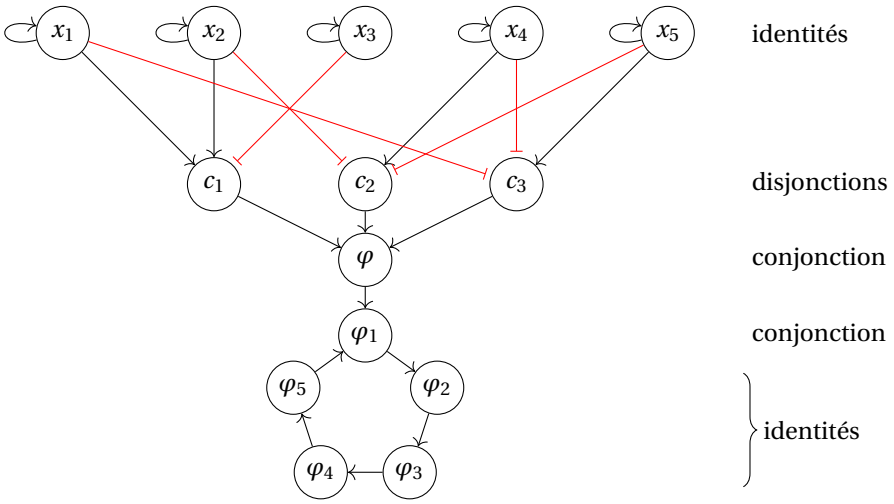


FIGURE 5.2. – Graphe d'interaction signé (arcs négatifs en rouge avec pointe plate) obtenu dans la réduction de **3-SAT** à **CL**, pour $k = 5$ et la formule $\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_4 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_4 \vee x_5)$. Les automates correspondant aux variables sont l'identité, les automates correspondant aux clauses sont des disjonctions, les automates φ et φ_1 sont des conjonctions et les automates φ_2 à φ_5 sont l'identité.

Cycles limites. En adaptant notre construction du théorème 5.1 sur les points fixes, on obtient un résultat de NP-difficulté analogue pour les cycles limites. Rappelons que l'ensemble des cycles limites de taille k est noté $\mathfrak{C}_f^k$, et que tout réseau possède au moins un cycle limite (l'espace est fini donc la dynamique est ultimement périodique, et les points fixes sont des cycles limites de taille 1).

Cycle limite de taille k (k-**CL**) *Entrée :* un RAB $f : \{0,1\}^n \to \{0,1\}^n$ (circuits des fonctions locales). *Question :* est-ce que $\mathfrak{C}_f^k \neq \emptyset$?

Soulignons que la taille k du cycle limite est fixée dans la définition de cette famille de problèmes de décision.

Théorème 5.8 ([Bri+21])**.** k-**CL** *est* NP*-complet pour tout* $k \in \mathbb{N}_+$*, même avec la promesse* $\Delta(G_f) \leq 2$.

Idée de démonstration. Pour $k = 1$, il s'agit du théorème 5.1. Nous allons donc considérer uniquement le cas $k \geq 2$. L'appartenance à NP est toujours donnée par un algorithme évident : deviner une configuration $x \in \{0,1\}^n$ et vérifier si elle appartient à un cycle limite de taille k (x est différente de $f(x), f^2(x), \ldots, f^{k-1}(x)$, et égale à $f^k(x)$).

On effectue à nouveau une réduction depuis une instance φ du problème **3-SAT**, en remplaçant le bas de la construction par un cycle de k automates qui auront pour

fonction de créer un cycle limite de taille k lorsque la formule est satisfaite, et de converger vers un point fixe sinon. Voir la figure 5.2.

Si φ est satisfaite par $x \in \{0,1\}^n$, alors on l'étend en une configuration d'un cycle limite de taille k en attribuant l'état 1 aux automates des clauses ainsi qu'à φ et φ_1, et l'état 0 aux automates $\varphi_2, \dots, \varphi_k$. On peut alors observer que l'état 1 va tourner dans le cycle sur un arrière-plan d'états 0 (avec l'automate φ fixé dans l'état 1, les automates φ_1 à φ_k forment un cycle positif simple dont la dynamique est bien caractérisée, voir par exemple [Sen12, lemme 6 page 75]). Réciproquement, dans toute configuration de l'ensemble limite, l'état des automates correspondant aux variables et clauses ne varie plus. Cependant, si l'une des clauses est dans l'état 0, alors cet état 0 se propage à φ puis à tous les automates du cycle et l'on obtient un point fixe. On en conclut qu'un cycle limite de longueur $k \geq 2$ doit correspondre sur les automates $x_1, \dots, x_n$ à une valuation satisfaisant toutes les clauses de la formule φ.

On peut abaisser à 2 le degré entrant des automates des clauses en les dédoublant comme dans la preuve du théorème 5.1, et également le degré entrant de φ en intercalant un arbre binaire comme sur la figure 5.1. □

La construction du théorème 5.8 est encore une fois un réseau ET/OU. Le problème k-**CL** porte sur les cycles limites de taille exactement k. On énonce ci-après un corollaire pour des variantes de cette question (taille au moins k, taille au plus k).

Corollaire 5.9. *Pour k fixé, étant donné un RAB $f : \{0,1\}^n \to \{0,1\}^n$ (circuits), décider :*

(i) $\bigcup_{k' \in [2^n] \setminus [k]} \mathfrak{C}_f^{k'} \neq \emptyset$*, il existe un cycle limite de taille au moins k, est* NP*-complet,*

(ii) $\bigcup_{k' \in [k]} \mathfrak{C}_f^{k'} \neq \emptyset$*, il existe un cycle limite de taille au plus k, est* NP*-complet,*

(iii) $\bigcup_{k' \in [2^n] \setminus \{1\}} \mathfrak{C}_f^{k'} \neq \emptyset$*, il existe un cycle limite non trivial, est* NP*-complet,*

même avec la promesse $\Delta(G_f) \leq 2$.

Démonstration. Pour (i) et (iii) on observe que la construction du théorème 5.8 possède uniquement des points fixes dans le cas où φ n'est pas satisfaisable.

Pour (ii) on doit faire en sorte que les configurations $x \in \Omega_f$ de l'ensemble limite pour lesquelles $x_\varphi = 0$ fassent partie de cycles limites de longueur strictement supérieure à k. Pour cela, on ajoute $k-1$ automates $\varphi'_1, \varphi'_2, \dots, \varphi'_{k-1}$ à la construction en figure 5.2, dont les fonctions locales sont :

- $f_{\varphi'_1}(x) = \neg x_\varphi \wedge \neg x_{\varphi'_{k-1}}$,
- $f_{\varphi'_{i+1}}(x) = x_{\varphi'_1} \wedge x_{\varphi'_i}$ pour $i \in [k-2]$.

Pour tout $x \in \Omega_f$ dans l'ensemble limite, on considère deux cas. Lorsque $x_\varphi = 1$, on a $x_{\varphi'_i} = 0$ pour tout $i \in [k-1]$. Ainsi, x a même période que dans la preuve du théorème 5.8 (c'est-à-dire qu'on peut étendre tout cycle limite de la preuve du théorème 5.8 en un cycle limite pour le RAB obtenu par l'ajout des $k-1$ automates). Donc il existe un cycle limite de longueur k sur l'ensemble limite. Lorsque $x_\varphi = 0$, on a symétriquement une unique possibilité pour tous les automates de la preuve du théorème 5.8, que

l'on peut étendre en un unique cycle limite sur les automates $x_{\varphi'_i}$ pour $i \in [k-1]$, de longueur $k+1$ (avec l'état de φ'_1 à gauche) :

$$\curvearrowright \texttt{000...0} \longrightarrow \texttt{100...0} \longrightarrow \texttt{110...0} \longrightarrow \texttt{111...0} \longrightarrow \texttt{111...1} \longrightarrow \texttt{011...1} \curvearrowright.$$

Sur l'unicité de ce cycle limite, on remarque que φ'_1 passe nécessairement par l'état 0, ce qui conduit en une itération à l'une des deux premières configurations ci-dessus. Ainsi, une valuation qui ne satisfait pas φ donnera un unique cycle limite de longueur $k+1$, et une valuation qui satisfait φ pourra donner un cycle limite de longueur k. □

Remarquons enfin, pour les énoncés où elle s'applique, que la promesse $\Delta(G_f) \leq 2$ est optimale, dans le sens où $\Delta(G_f) \leq 1$ correspond à une collection de cycles isolés auxquels sont branchés des arbres dont les arcs pointent vers les feuilles. Alors [Sen12, section 4.2.1] caractérise la dynamique asymptotique, et permet de résoudre en temps polynomial les problèmes **PF**, **#PF**, k-**CL** et ses variantes (en temps linéaire on découvre les longueurs de tous les cycles du graphe d'interaction[b]).

5.2. Préimages, atteignabilité et ensemble limite Ω_f

Préimages. De façon très élémentaire comme nos considérations sur le calcul des images en table 3.1, on peut se demander si une configuration x donnée possède un antécédent (une préimage), compter son nombre d'antécédents, et compter le nombre de configurations qui possèdent un antécédent, dans les cas déterministe, non déterministe et asynchrone.

$$\begin{aligned} \text{Déterministe :} \quad & f^{-1}(x) = \{y \mid x = f(y)\} \text{ et } f(X) = \{f(x) \mid x \in X\} \\ \text{Non déterministe :} \quad & f^{-1}(x) = \{y \mid x \in f(y)\} \text{ et } f(X) = \bigcup_{x \in X} f(x) \end{aligned}$$

Préimage (**Préimage**) *Entrée :* un RA f et une configuration x. *Question :* est-ce que $f^{-1}(x) \neq \emptyset$?

Comptage des préimages (**#Préimages**) *Entrée :* un RA f et une configuration x. *Sortie :* $\lvert f^{-1}(x) \rvert$.

Comptage des points images (**#Images**) *Entrée :* un RA f sur l'espace de configurations X. *Sortie :* $\lvert f(X) \rvert$.

b. Notons toutefois que la promesse $\Delta(G_f) \leq 1$ ne veut pas dire que le graphe d'interaction nous est donné, mais avec cette promesse on peut le calculer en temps polynomial grâce au théorème 4.5.

La complexité de **Préimage** pour les réseaux d'automates booléens déterministes a déjà été donnée dans [FO89]. Les *points images* sont les configurations qui sont l'image d'au moins une configuration (qui possèdent au moins une préimage). La classe $\#\mathsf{P}^{\mathsf{NP}}$ est introduite dans [Wag86]. Il s'agit de l'ensemble des problèmes de comptage dont la réponse correspond au nombre de branches acceptantes d'une machine de Turing en temps polynomial avec oracle dans NP. Pour être incluse dans la hiérarchie polynomiale de comptage, elle est en général définie comme une classe de problèmes de décision où l'on demande si le nombre de certificats est au moins k, et contient alors Σ_2^{P} et Π_2^{P} [Tor91].

Théorème 5.10. *Dans les cas déterministe et non déterministe,* **Préimage** *est* NP*-complet,* **#Préimages** *est* #P*-complet et* **#Images** *est* $\#\mathsf{P}^{\mathsf{NP}}$*-complet. Dans le cas asynchrone ces trois problèmes sont respectivement dans* P*,* FP *et* #P*-complet.*

Démonstration. Les bornes supérieures de complexité NP et #P dans le cas déterministe (*resp.* non déterministe) proviennent de l'algorithme naïf qui devine y et teste $x = f(y)$ (*resp.* $x \in f(y)$). Les bornes inférieures de NP-difficulté et #P-difficulté pour **Préimage** et **#Préimages** sont données par une réduction parcimonieuse depuis **SAT** : étant donnée une formule φ à n variables, on construit un réseau déterministe (*resp.* non déterministe) booléen f à n variables dont les fonctions (*resp.* relations) locales sont données par $f_i(x) = \varphi(x)$ (*resp.* $r_i(x) = \{\varphi(x)\}$) pour tout $i \in [n]$. Si $\varphi(x) = 1$ alors $f_i(x) = 1^n$ (*resp.* $\{1^n\}$), et sinon $f_i(x) = 0^n$ (*resp.* $\{0^n\}$). L'ensemble des antécédents de 1^n est donc l'ensemble des valuations qui satisfont φ, et les autres configurations ont pour image 0^n :

$$f^{-1}(1^n) = \{x \in \{0,1\}^n \mid \varphi(x) = 1\} \quad \text{et} \quad f^{-1}(0^n) = \{x \in \{0,1\}^n \mid \varphi(x) = 0\}.$$

Le problème **#Images** est dans $\#\mathsf{P}^{\mathsf{NP}}$ dans les cas déterministe et non déterministe d'après ce qui précède : deviner une configuration x et appeler l'oracle **Préimage** pour décider si x est un point image. Pour la $\#\mathsf{P}^{\mathsf{NP}}$-difficulté on donne une réduction faiblement parcimonieuse depuis une instance $\varphi(x, y)$ de **#∃-SAT** : étant donnée une formule φ à n variables et $s \in [n]$, il s'agit de compter :

$$|\{x \in \{0,1\}^s \mid \exists y \in \{0,1\}^{n-s} : \varphi(x, y) = 1\}|.$$

On construit le réseau déterministe booléen f à $n+1$ variables tel que :

$$\forall x \in \{0,1\}^s, y \in \{0,1\}^{n-s}, b \in \{0,1\} : f(xyb) = \begin{cases} x0^{n-s}1 & \text{si } \varphi(x, y) = 1 \text{ et } b = 1 \\ 0^{n+1} & \text{sinon} \end{cases},$$

avec les fonctions locales :

- $f_i(xyb) = x_i \wedge \varphi(x, y) \wedge b$ pour $i \in [s]$,
- $f_{s+i}(xyb) = 0$ pour $i \in [n-s]$,
- $f_{n+1}(xyb) = \varphi(x, y) \wedge b$.

On a alors 0^{n+1} qui possède toujours au moins un antécédent (elle-même), $x0^{n-s}1$ qui possède un antécédent si et seulement si $\exists y : \varphi(x,y) = 1$, et toutes les autres configurations n'ont pas d'antécédent. Ainsi, la réponse à **#Images** vaut une unité de plus que la réponse à **#∃-SAT**. Il ne semble pas possible de donner une réduction parcimonieuse car, si la formule de **#∃-SAT** est toujours fausse et que son comptage donne 0, tout réseau déterministe vers lequel on réduit aura au moins une image et donc le comptage de **#Images** sera d'au moins une unité. Dans le cas non déterministe, on peut en revanche adapter la construction pour que la réduction soit parcimonieuse, en prenant le réseau booléen f sur n automates tel que :

$$\forall x \in \{0,1\}^s, y \in \{0,1\}^{n-s} : f(xy) = \begin{cases} \{x0^{n-s}\} & \text{si } \varphi(x,y) = 1 \\ \emptyset & \text{sinon.} \end{cases}$$

Pour le cas asynchrone, toute configuration $x \in \{0,1\}^n$ possède au plus $n+1$ antécédents, qui sont parmi x et $x + e_i$ pour $i \in [n]$. On peut calculer les images de chacune (table 3.1), et décider en temps polynomial lesquelles sont dans $f^{-1}(x)$, ce qui répond à **Préimage** et **#Préimage**. Pour **#Images**, on présente une réduction faiblement parcimonieuse depuis **#SAT**, car tout réseau booléen asynchrone $\hat{f}$ à n automates possède au minimum 2^{n-1} points images : une configuration $\tilde{x}$ qui n'est pas un point image vérifie en particulier $\tilde{x} \notin \hat{f}(\tilde{x})$, donc $f_i(\tilde{x}) = \neg \tilde{x}_i$ pour tout $i \in [n]$, ce qui consomme n arêtes de l'hypercube indépendamment des autres configurations qui ne sont pas des points images, or il y a $n2^{n-1}$ arêtes dans l'hypercube. Étant donnée une formule φ à n variables, on construit $\hat{f}$ à $n+1$ automates dont les fonctions locales sont définies pour tout $x \in \{0,1\}^{n+1}$ par

$$f_i(x_1,\dots,x_{n+1}) = x_i \oplus \left(\neg\varphi(x_1,\dots,x_n) \wedge \bigoplus_{j\in[n+1]} x_j \right) \text{ pour tout } i \in [n+1].$$

On a alors :

- la moitié des $x \in \{0,1\}^{n+1}$ qui vérifient $\bigoplus_{j\in[n+1]} x_j = 0$ ont pour unique image x,
- parmi l'autre moitié des $x \in \{0,1\}^{n+1}$ qui vérifient $\bigoplus_{j\in[n+1]} x_j = 1$:
  - celles qui vérifient $\varphi(x_1,\dots,x_n) = 1$ ont également pour unique image x,
  - les autres ont pour images $\{x + e_i \mid i \in [n+1]\}$, qui ne contient aucune des configurations $y \in \{0,1\}^{n+1}$ qui vérifient $\bigoplus_{j\in[n+1]} y_j = 1$.

On en déduit que le nombre de points images de $\hat{f}$ est égal au nombre de valuations qui satisfont φ (parmi les configurations dont la somme des bits vaut 1 modulo 2), plus 2^n (toutes les configurations dont la somme des bits vaut 0 modulo 2). □

Atteignabilité. Un autre problème naturel est l'atteignabilité. Dans le cas non déterministe, l'orbite de x dans f est formellement définie en prenant l'union de tous les futurs possibles, $\mathcal{O}_f(x) = \bigcup_{t\in\mathbb{N}} f^t(x)$. Les réseaux d'automates seront encore encodés par des circuits pour les fonctions locales.

Atteignabilité (**At**) *Entrée :* un RA f de taille n et deux configurations x, y. *Question :* est-ce que $x \in \mathfrak{O}_f(y)$?

L'atteignabilité se rapproche des problèmes sur les bassins d'attraction du théorème 5.7, mais désormais la configuration de départ y dont il est question de décider si elle possède la configuration d'arrivée x dans son orbite, est explicitement nommée. Dans le cas ou la configuration d'arrivée x est un point fixe (ce qui sera le cas dans nos démonstrations), alors **At** revient exactement à demander : est-ce que $y \in \mathfrak{B}_f(x)$?

Théorème 5.11 ([Cha+20])**.** **At** *est* PSPACE*-complet pour les RA booléens asynchrones.*

Idée de démonstration. Les auteurs de [Cha+20] développent des simulations entre RAB et *safe Read Petri nets* (RPN), dont ce résultat est un corollaire (par application d'un résultat de PSPACE-difficulté connu pour les RPN). On peut intuitivement comprendre ce résultat au travers d'une réduction depuis le problème suivant : étant donnée une machine de Turing linéairement bornée (MTLB) M et une entrée u, est-ce que M accepte u ? Il s'agit d'un problème PSPACE-complet canonique [GJ79, page 175]. Une MTLB peut être définie comme non déterministe (ce qui correspondra à l'asynchronisme du RAB), avec son entrée délimitée par deux marqueurs qu'elle n'est pas autorisée à dépasser (ce qui rendra la réduction polynomiale en $|M|$ et $|u|$). Il est alors assez naturel (et fastidieux) de construire un RA uniforme asynchrone sur $|u|$ automates qui simule la machine M, et atteint un point fixe particulier (qui sera notre x) si et seulement si la machine accepte son entrée (qui sera notre y). Le passage à un alphabet booléen ne devrait ensuite pas poser de difficulté. □

Il se trouve que le problème **At** est également PSPACE-complet dans le cas uniforme déterministe. Nous donnons le lemme technique suivant, qui explicite les propriétés de la simulation de machine de Turing réalisée pour cette réduction. Ce lemme nous servira à déduire d'autres corollaires, sur la taille de l'ensemble limite.

Lemme 5.12 ([Gam+21])**.** *Étant donnés une* ***MT*** M*, deux entiers $k, n \in \mathbb{N}$ et une entrée u de taille au plus n, il existe $q \in \mathbb{N}$ dépendant uniquement de M et k et un RA q-uniforme déterministe $f : [\![q]\!]^n \to [\![q]\!]^n$ constructible en temps polynomial tel que :*

- *si M accepte u en au plus $k^n - 1$ étapes avec espace au plus n, alors $\Omega_f = \{0^n\}$,*
- *sinon il existe une configuration $\tilde{x} \in \Omega_f$ appartenant à un cycle limite de longueur k^n dans lequel l'état 0 n'apparaît pas : $\forall t \in \mathbb{N} : \forall i \in [n] : f^t(\tilde{x})_i \neq 0$.*

Idée de démonstration. Soit Σ l'alphabet, S les états de M, et $\bot, 0$ de nouveaux symboles. On prend l'alphabet $Q = (\Sigma \times H \times [k]) \sqcup \{0\}$ avec $H = (S \sqcup \{\bot\})$, que l'on met en bijection avec $[\![q]\!]$ pour $q = |Q|$. Une configuration $x \in Q^n$ ne contenant pas l'état 0 peut être vue comme un triplet (x^Σ, x^H, x^k) où :

- x^Σ encode le contenu du ruban (limité à n cases),
- x^H encode l'état et la position de la tête (avec exactement une position $x^h_i \in S$),
- x^k encode un compteur de 0 à $k^n - 1$ en base k.

On construit le RA q-uniforme qui, partant de la configuration $\tilde{x}$ correspondant à l'état initial de la machine M sur l'entrée u, simule la machine de Turing en incrémentant le compteur à chaque étape de calcul. On s'assure ensuite que l'acceptation en au plus $k^n - 1$ étapes mène au point fixe 0^n, et que l'épuisement du compteur x^k réinitialise le calcul à $\tilde{x}$. Les configurations invalides sont envoyées sur 0^n ou $\tilde{x}$ pour pallier aux détails techniques, et l'on obtient le comportement énoncé. □

Corollaire 5.13. At *est* PSPACE*-complet pour les RA uniformes déterministes.*

Démonstration. Pour l'appartenance à PSPACE, on peut simplement dérouler le calcul de l'orbite $\mathcal{O}_f(x)$ durant q^n étapes en cherchant y, sans se souvenir de l'historique des configurations visitées. Pour la PSPACE-complétude, soit $L \in$ PSPACE décidé par une **MT** M fonctionnant en espace polynomial. Étant donnée une entrée u de taille n pour cette machine, on applique le lemme 5.12 avec k suffisamment grand pour que M fonctionne en moins de k^n étapes de calcul sur toute instance de taille au plus n (la valeur de k est polynomiale). On obtient un RA f uniforme déterministe tel que $0^n \in \mathcal{O}_f(\tilde{x})$ si et seulement si M accepte u, ce qui donne $L \leq_{\mathrm{m}}^{\mathsf{P}}$ **At**. □

De façon assez proche de l'atteignabilité, on peut demander si une configuration donnée fait partie de l'ensemble limite Ω_f. Dans [Pau+20], Paulevé et. al en caractérisent la complexité.

Ensemble limite (**Ens. lim.**) *Entrée :* un RA f de taille n et une configuration x. *Question :* est-ce que $x \in \Omega_f$?

Théorème 5.14 ([Pau+20])**. Ens. lim.** *est* PSPACE*-complet pour les RA booléens, asynchrones ou déterministes.*

Notons que [Pau+20] discute d'un mode de mise à jour appelé « most permissive », qui permet aux états booléens d'un réseau d'automates de simuler proprement certains types de boucles de rétroactions (« feed forward loops »), et dont la complexité des problèmes **At** et **Ens. lim.** sont moindres : respectivement complets pour $\mathsf{P}^{\mathsf{NP}} = \Delta_2^{\mathsf{P}}$ et $\mathsf{coNP}^{\mathsf{NP}} = \Pi_2^{\mathsf{P}}$ (et même P et coNP pour les réseaux localement monotones, c'est-à-dire dont les fonctions locales sont monotones : avec $f_i(x) \leq f_i(x + e_j)$ pour tous $i, j \in [n]$ et $x \in \{0,1\}^n$ avec $x_j = 0$).

Ensemble limite Ω_f. Le lemme 5.12 a été conçu comme un outil pour démontrer des résultats de PSPACE-difficulté, tel que notre nouveau corollaire 5.13. Il a été utilisé pour obtenir certains des résultats qui suivent sur l'ensemble limite Ω_f d'un réseau d'automates. On s'intéresse tout d'abord à vérifier si la taille de l'ensemble limite est la plus petite possible (nilpotence), puis la plus grande possible (bijectivité).

Nilpotence (**Nil**) *Entrée :* un RA uniforme déterministe $f : [\![q]\!]^n \to [\![q]\!]^n$. *Question :* est-ce que $\lvert\Omega_f\rvert = 1$?

Corollaire 5.15 ([Gam+21])**.** **Nil** *est* PSPACE*-complet.*

Idée de démonstration. L'appartenance à PSPACE provient d'un algorithme naïf qui teste si $f^{q^n}(x)$ donne la même configuration pour toute $x \in [\![q]\!]^n$ (il suffit de se souvenir de la première configuration de Ω_f ainsi calculée, pour la comparer aux suivantes). La PSPACE-complétude est une application directe du lemme 5.12 à partir de la définition de PSPACE et d'un k suffisamment grand, comme pour la démonstration du corollaire 5.13. □

Un réseau d'automates déterministe est *bijectif* lorsque $\mathcal{G}_f$ a degré sortant *et* entrant exactement 1, c'est-à-dire lorsque $\mathcal{G}_f$ est une union disjointe de cycles. Puisque nos systèmes dynamiques sont finis de $\{0,1\}^n$ dans $\{0,1\}^n$, nous avons le résultat suivant.

Théorème 5.16. *Un RA est bijectif si et seulement s'il est injectif.*

On remarque également, puisqu'il s'agit de réseaux d'automates déterministes, que la bijectivité est équivalente à avoir toutes les configurations dans l'ensemble limite.

Bijectivité (**Bij**) *Entrée :* un RAB $f : \{0,1\}^n \to \{0,1\}^n$ (circuits des fonctions locales). *Question :* est-ce que $\lvert\Omega_f\rvert = 2^n$, c'est-à-dire est-ce que f est bijectif?

Théorème 5.17. **Bij** *est* coNP*-complet.*

Démonstration. D'après le théorème 5.16, il faut et il suffit de tester si f est injective, c'est-à-dire si $\forall x, y \in \{0,1\}^n : x \neq y \implies f(x) \neq f(y)$, ce qui montre l'appartenance à coNP. Pour la difficulté, étant donnée une instance φ à n variables de **UNSAT** dont la valuation 0^n n'est pas satisfaisante, on construit un RAB à $n+1$ automates dont les fonctions locales sont l'identité, sauf celle du dernier automate qui est $f_{n+1}(xb) = b \vee \varphi(x)$ (avec xb la concaténation de $x \in \{0,1\}^n$ et $b \in \{0,1\}$). La dynamique $\mathcal{G}_f$ sera l'identité ($f(x) = x$ pour tout x) lorsque φ n'est pas satisfaisable, et les deux configurations $\tilde{x}0$ et $\tilde{x}1$ auront la même image ($\tilde{x}1$) lorsque $\tilde{x}$ satisfait φ. □

De façon analogue à la bijectivité, on caractérise la complexité de décider si un réseau déterministe donné est l'identité ou une constante.

Théorème 5.18. *Étant donné un RAB $f : \{0,1\}^n \to \{0,1\}^n$ (circuits) :*
(i) décider si f est l'identité ($\forall x \in \{0,1\}^n : f(x) = x$),
(ii) décider si f est constante ($\forall x \in \{0,1\}^n : f(x) = f(0^n)$),
sont des problèmes coNP*-complets.*

Démonstration. On réduit depuis **UNSAT**, en transformant une formule φ sur n variables en un réseau booléen f sur n automates avec : pour (i) les fonctions locales $f_i(x) = x_i \oplus \varphi(x)$, et pour (ii) les fonctions locales $f_i(x) = \varphi(x)$, pour tout $i \in [n]$. □

Toujours dans le cas déterministe, nous pouvons considérer des questions plus larges que la nilpotence ou la bijectivité, sur la taille de l'ensemble limite. Étant donnée une fonction $\lambda : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ telle que $\lambda(q,n) \leq q^n$ (pour tous $q, n \in \mathbb{N}$), on définit le problème suivant.

Taille λ de l'ensemble limite (**TEL**$_\lambda$) *Entrée :* un RA q-uniforme $f : \llbracket q \rrbracket^n \to \llbracket q \rrbracket^n$ déterministe. *Question :* est-ce que $\lvert\Omega_f\rvert \geq \lambda(q,n)$?

Nous allons voir que le problème **TEL**$_\lambda$ est PSPACE-difficile lorsque λ est « petite » (il y a beaucoup de configurations qui ne sont possiblement pas dans l'ensemble limite), mais que la complexité tombe dans la hiérarchie polynomiale lorsque λ est « grande » (dans ce cas peu de configurations ne sont pas dans l'ensemble limite).

Théorème 5.19 ([Gam+21])**.** *Soit $\delta : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ une fonction polynomiale. On a :*
- *si $\forall n \in \mathbb{N} : 2 \leq \lambda(q,n) \leq k^n$ avec $k > 1$, alors* **TEL**$_\lambda$ *est* PSPACE*-difficile,*
- *si $\lambda(q,n) = q^n - \delta(q,n)$, alors* **TEL**$_\lambda$ *est dans Σ_3^{P} et* coNP*-difficile.*

Idée de démonstration. Le premier point est encore une application du lemme 5.12 analogue aux preuves des corollaires 5.13 et 5.15.

La borne supérieure Σ_3^{P} du second point est obtenue en remarquant que dans ce cas où λ est « grande », partant de toute configuration on doit atteindre l'ensemble limite en au plus $\delta(q,n)$ itérations. Ainsi, on doit avoir $f^{\delta(q,n)}(\llbracket q \rrbracket^n) = f^{\delta(q,n)+1}(\llbracket q \rrbracket^n) = \Omega_f$. Ces deux égalités peuvent être exprimées avec respectivement $\forall\exists$ et $\exists\forall\exists$, et des parties propositionnelles vérifiables en temps polynomial grâce à l'hypothèse sur δ.

On démontre la coNP-difficulté avec une réduction depuis **UNSAT**, en fixant $q = 4$ qui donne $\delta(4,n) < 2^n - 1$ pour n suffisamment grand. Étant donnée φ sur n variables, on construit $f : [4]^n \to [4]^n$ en projetant les configurations de $\llbracket 4 \rrbracket^n$ sur les valuations de $\{0,1\}^n$ avec l'opération modulo deux (ainsi chaque valuation est obtenue à partir de 2^n configurations), et en faisant évoluer une configuration sur elle-même si son projeté ne satisfait pas φ, et sur 0^n si son projeté satisfait φ. Ainsi, une seule valuation satisfaisant φ retirera plus de $\delta(4,n)$ configurations à Ω_f. □

Pour aller plus loin dans les problèmes sur l'ensemble limite, nous allons nous intéresser à des « propriétés abstraites », et passer aux RA non déterministes. Cela nous donnera un avant goût des développements exposés au chapitre 6. On dira qu'une propriété P est *effectivement non triviale* pour une famille de RA (déterministe ou non déterministe, uniforme ou non uniforme), lorsqu'il existe un algorithme qui, étant donné n en unaire, produit en temps polynomial deux RA f_1, f_2 de taille n dans la famille, tels que $f_1 \models P$ et $f_2 \not\models P$ (avec $f \models P$ signifiant « f vérifie la propriété P »).

La condition d'effectivité est une demande assez naturelle pour pouvoir réaliser une réduction : il faut être capable de construire une instance positive et une instance négative du problème vers lequel on réduit. Nous allons considérer ici uniquement des *propriétés de l'ensemble limite*, c'est-à-dire telles que $\Omega_f = \Omega_g$ implique $f \models P \iff g \models P$, et restreindre cette étude aux RA uniformes grâce au résultat suivant.

Lemme 5.20 ([Gam+21])**.** *Toute propriété effectivement non triviale de l'ensemble limite des RA non uniformes est également effectivement non triviale pour les RA uniformes (déterministes ou non déterministes).*

Démonstration. Il s'agit, à partir des f_1 et f_2 données par l'effectivité mais possiblement non uniformes, de compléter l'algorithme pour produire f_1' et f_2' uniformes. En prenant l'union des alphabets de chaque automate et en envoyant de façon déterministe toutes les nouvelles configurations sur une même configuration choisie arbitrairement, on a $\Omega_{f_1} = \Omega_{f_1'}$ et $\Omega_{f_2} = \Omega_{f_2'}$, donc la propriété est identique. La construction des nouveaux circuits des fonctions locales est réalisable en temps polynomial, et ne modifie pas la nature déterministe ou non déterministe des RA. □

Propriété P de l'ensemble limite (**PEL**$_P$)
Entrée : un RA q-uniforme $f : [\![q]\!]^n \to \mathcal{P}([\![q]\!]^n)$ non déterministe.
Question : est-ce que $f \models P$?

Théorème 5.21 ([Gam+21])**.** *Si P est une propriété effectivement non triviale de l'ensemble limite, alors* **PEL**$_P$ *est* PSPACE*-difficile pour les réductions* $\leq^{\mathsf{P}}_{tt}$.

Idée de démonstration. L'effectivité nous donne f_1, f_2 uniformes sur l'alphabet Q_f dont on distingue $0 \in Q_f$. Étant donné un RA h uniforme déterministe sur l'alphabet Q_h dont on distingue $1 \in Q_h$, nous construisons deux RA g_1, g_2 uniformes non déterministes sur l'alphabet $((Q_h \times Q_f) \cup Q_f)^n$. Il y a deux types de configurations :

- sur $(x^h, x^f) \in (Q_h \times Q_f)$ les deux RA g_1 et g_2 ont le choix non déterministe :
  - reproduire le comportement de h sur x^h et laisser x^f inchangé, ou
  - aller dans la configuration x^f,
- sur $x^f \in Q_f$ le RA g_i reproduit le comportement de f_i.

On ajoute une condition : si l'état 1 apparaît dans x^h, ou si la configuration mélange des états dans $(Q_h \times Q_f)$ et dans Q_f, alors aller dans 0^n. On obtient que :

- si h possède une orbite qui évite l'état 1, alors $\Omega_{g_1} = \Omega_{g_2}$, car leur comportement est identique sur $(Q_h \times Q_f)$ et le choix non déterministe d'aller dans x^f peut être repoussé arbitrairement loin donc $Q_f^n \subseteq \Omega_{g_i}$,
- si l'état 1 apparaît dans toutes les orbites de h, alors g_i est forcé d'aller dans Q_f et $\Omega_{g_i} = \Omega_{f_i}$, ainsi $\Omega_{g_1} \neq \Omega_{g_1}$ car $f_1 \models P$ alors que $f_2 \not\models P$.

On testera alors la condition par une table de vérité à deux entrées pour **PEL**$_P$, or ce problème (de décider si h possède une orbite qui évite l'état 1) est PSPACE-complet par application du lemme 5.12. □

6. Complexité des questions FO

Nous allons grandement généraliser, dans le cas déterministe à alphabet non fixé, les résultats sur la complexité des problèmes abordés dans le chapitre 5.1. L'existence d'un point fixe ou d'un cycle limite de longueur fixée sont des propriétés de la dynamique, vue comme le graphe orienté $\mathcal{G}_f$ de la fonction $f : \{0,1\}^n \to \{0,1\}^n$ (en l'occurrence de degré sortant 1 pour tout sommet), exprimables en logique du premier ordre. Au théorème 6.1, on énonce que toute propriété du premier ordre est soit triviale, soit NP-difficile, soit coNP-difficile.

Une *formule* désignera une formule close en logique du premier ordre (*first order*, FO) sur la signature $\{=, \to\}$ comportant deux relations binaires. Les variables seront les configurations, avec $x \to y$ qui désignera $f(x) = y$. Nous utiliserons la terminologie de théorie des modèles [CE12 ; DF13 ; Lib04], en parlant de *modèles* et *contre-modèles* d'une formule et en utilisant le symbole « $\models$ ». Notons toutefois que les questions exprimées en logique du premier ordre ne permettent pas de désigner de configuration particulière : la dynamique des réseaux d'automates est considérée à isomorphisme près, c'est-à-dire modulo renommage des étiquettes du graphe de la dynamique.

Les travaux présentés dans cette section visent à obtenir un théorème « de type Rice », en référence au fameux résultat d'indécidabilité de toute question non triviale sur le comportement des machines de Turing, démontré par Rice en 1953 [Ric53]. En section 6.1, nous verrons donc que toute formule (FO) « non triviale » est « difficile » à tester (permettons nous de rappeler que tout est fini donc décidable). Nous commencerons par enlever les guillemets de cette dernière phrase, puis nous présenterons la structure de la démonstration. La section 6.2 donnera des exemples de formules qui atteignent tous les niveaux de la hiérarchie polynomiale PH, ce qui aura pour conséquence que passer la formule dans l'entrée du problème mène à PSPACE. La section 6.3 mentionnera des extensions envisageables du résultat principal de ce chapitre. Enfin, la section 6.4 présentera des perspectives de passage à la logique monadique du second ordre, sur lesquelles nous travaillons actuellement.

6.1. ψ-Dynamique est $\mathcal{O}(1)$ ou NP- ou coNP-difficile

Étant donnée une formule ψ, on dira qu'elle est ω *non triviale* lorsqu'elle possède une infinité de modèles et une infinité de contre-modèles.

ψ-*Dynamique* (ψ-**Dyn**) *Entrée :* un RA f déterministe à n automates (circuits des fonction locales). *Question :* est-ce que $\mathscr{G}_f \models \psi$?

On peut penser ce problème comme si l'encodage de f par des circuits pour les fonctions locales correspond à un encodage succinct du graphe de transition, dans le cas déterministe parallèle qui nous intéresse ici (cette remarque sera discutée en section 6.3). Nous allons présenter les ingrédients conduisant au résultat suivant.

Théorème 6.1 ([Gam+21])**.** *Si ψ est ω non triviale, alors ψ-**Dyn** est soit* NP*-difficile soit* coNP*-difficile, sinon ψ-**Dyn** peut être résolu en temps $\mathcal{O}(1)$, pour les RA déterministes à alphabet non fixé.*

La seconde partie de l'énoncé est évidente, et montre que la condition d'ω non trivialité est optimale. Si une formule est ω triviale, alors il suffit de tester parmi une liste finie d'objets pour décider si notre entrée est un modèle ou un contre-modèle (soit il existe un nombre fini de modèles, soit il existe un nombre fini de contre-modèles). On utilise ici notre remarque de la section 3.4 sur l'encodage des fonctions locales par des circuits : le nombre d'entrées à n automates est fini.

Remarquons la symétrie du résultat vis-à-vis de NP et coNP, qui est nécessaire puisque nous ne savons rien de précis sur ψ : l'existence d'un point fixe et l'absence de point fixe sont deux propriétés exprimables en logique du premier ordre, qui sont respectivement NP-complète et coNP-complète à décider étant donné un RA booléen déterministe (théorème 5.1). De plus, on peut imaginer des formules ψ à la fois NP-difficiles et coNP-difficiles, par exemple plus haut dans PH (section 6.2).

La preuve du théorème 6.1 procède en trois parties :

1. encoder une instance φ de **SAT** dans la dynamique d'un RA (section 6.1.1),
2. préparer des ingrédients graphiques à partir de ψ (section 6.1.2),
3. considérer une disjonction de cas pour conclure (section 6.1.3).

Nous en proposons ici une version allégée, avec uniquement l'union disjointe $\sqcup$ de graphes pour l'assemblage de dynamiques, et où la disjonction de cas finale ne sera pas complète. Ainsi, la démonstration présentée ci-après contient la structure principale du raisonnement menant au théorème 6.1, mais ne traite pas toutes les formules ψ au premier ordre.

6.1.1. Encodage : SAT dans $\mathscr{G}_f$

Dans cette première partie, nous exposons un outil permettant d'obtenir la NP- ou coNP-difficulté à partir d'une construction de pompage dans les modèles ou contre-modèles de ψ. Intuitivement, dans le RA f, chaque valuation d'une instance φ de **SAT** produira, dans la dynamique $\mathscr{G}_f$:

- soit une copie d'un graphe « neutre » si elle ne satisfait pas la formule,
- soit une copie d'un graphe « suffisant » si elle satisfait la formule.

Lesdits graphes sont assemblés dans $\mathscr{G}_f$ par union disjointe (il s'agit d'une simplification, le cas général requiert des assemblages plus complexes).

Étant donnés deux graphes G, G', nous noterons $G \sqcup G'$ l'union disjointe de G et G', et pour tout $k \in \mathbb{N}$ on aura $\bigsqcup^k G = G \sqcup \cdots \sqcup G$ avec k copies de G. Soit maintenant $n \in \mathbb{N}$, un n-uplet de graphes $\Gamma = (G_1, \ldots, G_n)$, et w un mot sur l'alphabet $[n]$. On définit $\mathcal{U}^{G,\Gamma}(w)$ par induction : $\mathcal{U}^{G,\Gamma}(\epsilon) = G$ et $\mathcal{U}^{G,\Gamma}(w_1 \ldots w_k) = \mathcal{U}^{G,\Gamma}(w_1 \ldots w_{k-1}) \sqcup G_{w_k}$ (avec ϵ le mot vide). En partant de G, le mot w indique les graphes à piocher dans Γ.

Lemme 6.2 ([Gam+21])**.** *Soit ψ une formule. S'il existe des graphes non vides B, N, S tels que pour tous $k \in \mathbb{N}$ et $k' \in \mathbb{N}_+$ on a :*

- $B \sqcup (\bigsqcup^k N) \not\models \psi$, *et*
- $B \sqcup (\bigsqcup^k N) \sqcup (\bigsqcup^{k'} S) \models \psi$,

*alors ψ-**Dyn** est* NP*-difficile.*

Idée de démonstration. Comme expliqué précédemment, le graphe N joue le rôle de neutre et le graphe S est suffisant : la présence d'une copie de S suffit à donner un modèle de ψ. Le graphe B est une base qui nous sera imposée dans la partie suivante de la preuve. Soit φ une instance de **SAT** sur s variables. Il s'agit de construire un réseau d'automates f tel que sa dynamique $\mathscr{G}_f$ comprenne, partant de la base B, une copie de N pour chaque valuation ne satisfaisant pas φ, et une copie de S pour chaque valuation satisfaisant φ. Voir figure 6.1.

La construction des circuits des fonctions locales de f ne pose pas de difficulté particulière, car les réseaux d'automates sont très expressifs et φ nous est déjà donnée sous la forme d'un circuit, qu'il s'agit d'évaluer pour créer des transitions qui produisent les copies de B, N et S désirées. La principale considération technique réside dans le choix des alphabets de chaque automate afin d'avoir le bon nombre de sommets dans $\mathscr{G}_f$. Il s'agit précisément de la raison pour laquelle le théorème 6.1 est exprimé sur la famille des réseaux à alphabet non uniforme et non fixé. Pour commencer on peut remplacer N par $\bigsqcup^{|S|} N$, et remplacer S par $\bigsqcup^{|N|} S$, ainsi $|N| = |S|$. Une solution simple consiste alors à prendre un unique automate d'alphabet $[|B| + 2^s \cdot |S|]$, mais on tâche néanmoins de faire un effort en direction d'alphabets plus petits. En jouant avec l'algorithmique des nombres et notamment l'indicatrice d'Euler, le graphe de transition que nous obtenons sera :

$$\mathcal{U}^{B,(N,S)}(\overline{\varphi}) \sqcup (\bigsqcup^k N),$$

où $\overline{\varphi}$ est un mot de longueur 2^s sur l'alphabet $\{1,2\}$ dont la i-ème lettre est 1 si $\varphi(i) = \bot$ et 2 si $\varphi(i) = \top$ (en lisant l'écriture binaire de i comme une valuation), et $k \in \mathbb{N}$ servira à faire du remplissage avec des copies du graphe neutre, N. □

D'après notre discussion au chapitre 3.4, dans la preuve précédente, nous aurions pu plus simplement produire un réseau uniforme à 1 automate, dont l'alphabet est $[|B| + 2^s \cdot |N|]$ (avec $|N| = |S|$). La construction esquissée ci-dessus, que nous avons présentée dans [Gam+21], constitue un effort inabouti pour décomposer le réseau en plusieurs automates d'alphabets plus petits.

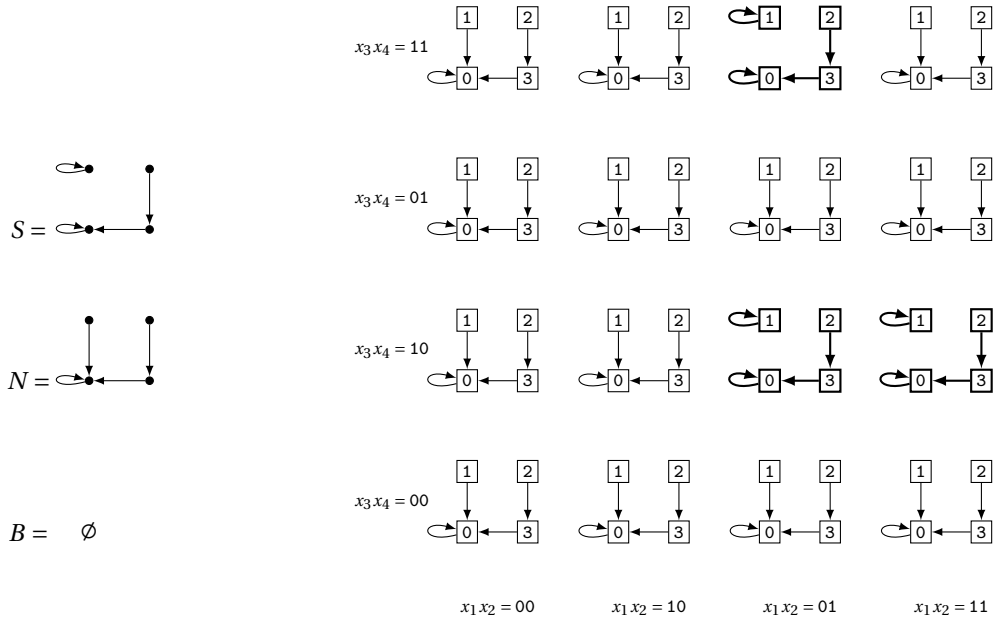


FIGURE 6.1. – Illustration de la construction du lemme 6.2, pour la propriété

$$\psi = \exists x : \big[\exists y : y \to x\big] \wedge \big[\forall y : \neg(y \to x) \vee (\exists z : z \to y)\big]$$

qui est vérifiée par les graphes pour lesquels « il existe une configuration avec au moins une préimage et dont toutes les préimages ont au moins une préimage ». On a les graphes de base B, neutre N et suffisant S (à gauche, avec $|B| = 0$ et $|N| = |S| = 4$). Étant donnée la formule :

$$\varphi(x) = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_4) \wedge (x_2 \vee x_4) \wedge x_3,$$

on construit le réseau déterministe $f : [2]^4 \times [4] \to [2]^4 \times [4]$ sur les automates $1,2,3,4$ booléens et l'automate 5 d'alphabet $[4]$. Les fonctions locales sont $f_i(x) = x_i$ pour $i \in [4]$ (les quatre composantes booléennes sont fixes), et :

$$f_5(x) = \begin{cases} 0 & \text{si } x_5 \in \{0,3\} \\ 0 & \text{si } x_5 = 1 \text{ et } \varphi(x_{[4]}) = \bot \\ 1 & \text{si } x_5 = 1 \text{ et } \varphi(x_{[4]}) = \top \\ 3 & \text{si } x_5 = 2 \end{cases}$$

qui encode S ou N sur la cinquième composante, suivant si $x_{[4]} = x_1x_2x_3x_4$ satisfait φ ou non, respectivement. On obtient une dynamique (à droite, avec l'état de x_5 inscrit dans chaque configuration) qui contient trois copies de S correspondant aux trois valuations qui satisfont φ : l'instance de **SAT** est positive, et le réseau déterministe vérifie $\mathscr{G}_f \models \psi$.

Il s'agit d'un cas de formule au premier ordre ψ simple avec $|B| = 0$, et pour lequel l'union disjointe suffit. En général, la preuve d'existence de B, N, S (lemme 6.7) donnera des graphes potentiellement très grands (mais qui ne dépendent que de ψ, et non de φ), et qu'il faudra assembler en redirigeant quelques arcs entre configurations bien identifiées.

6.1.2. Ingrédients graphiques : DULC

C'est dans cette partie de la preuve que la théorie des modèles intervient. Nous sommes dans le cas déterministe, donc nos graphes de dynamique ont degré sortant exactement 1, et toute composante connexe terminale est un cycle dont les sommets sont racine d'un arbre dont tous les sommets pointent vers cette racine. En notant $\mathcal{T}$ l'ensemble de tels arbres, et $G \equiv_m G'$ lorsque G et G' sont des modèles du même ensemble de formules à rang de quantification [a] m, la théorie des modèles nous donne le résultat classique suivant (voir [Lib04, corollaire 3.16 page 35]).

Lemme 6.3 ([Gam+21])**.** *Le nombre de classes d'équivalence de $\equiv_m$ sur $\mathcal{T}$ est fini.*

Chacun des cycles d'un graphe de dynamique peut donc être vu comme un mot cyclique sur un alphabet fini, dont chaque lettre indique la classe d'équivalence de l'arbre dont il est la racine : nos graphes sont des unions disjointes de cycles étiquetés (*disjoint unions of labeled cycles*, DULC).

Notation. Le passage d'un graphe G a un DULC est noté $\mathcal{E}_m(G)$.

Les graphes B, N, S que nous allons obtenir au lemme 6.7 seront des DULC. La finitude du nombre d'étiquettes pour une formule ψ donnée (dépendant même uniquement de son rang de quantification m) nous permettra ci-après de travailler sur une notion de *profil* d'un DULC, qui compte le nombre d'occurrences des différentes boules jusqu'à un rayon donné (ne dépendant là aussi que de m), et suffit à déterminer si un graphe de dynamique satisfait ψ ou non.

Les *jeux d'Ehrenfeucht-Fraïssé* pour FO constituent une technique prolifique de théorie des modèles [Lib04, chapitre 3] pour démontrer $G \equiv_m G'$: Spoiler tente de montrer que les deux graphes ne sont pas équivalents en désignant un sommet de l'un des graphes à chaque tour, et Duplicator doit répondre en désignant un sommet de l'autre graphe de façon à ce que les deux ensembles de sommets obtenus (l'un pour G et l'autre pour G') ne permettent pas de distinguer les deux graphes au regard des formules de rang de quantification m. Si après m tours (chaque tour correspond au choix d'une variable quantifiée) Spoiler ne peut pas piéger Duplicator, ce dernier a une stratégie gagnante et l'on en déduit $G \equiv_m G'$. On obtient ainsi le résultat suivant, qui nous permet de passer des DULC aux graphes.

Lemme 6.4 ([Gam+21])**.** *Si $\mathcal{E}_m(G) \equiv_m \mathcal{E}_m(G')$ alors $G \equiv_m G'$.*

Le *lemme de localité de Hanf* donne un rayon $e(m) \in \mathbb{N}$ (en termes de distance dans le graphe ; il dépend du rang de quantification) permettant de définir le *profil* $\pi_{G,m}$ d'un graphe G comme une fonction qui, à chaque possible boule de rayon 3^m, associe son nombre d'occurrences dans G, avec $+\infty$ qui remplace tout comptage excédant $e(m)$ (les trop nombreuses occurrences deviennent indistingables). Si $\pi_{G,m} = \pi_{G',m}$ alors $G \equiv_m G'$ [Han63, lemme 2.3]. On adapte ces raisonnements aux DULC.

a. Le *rang de quantification* d'une formule ψ est le nombre de quantificateurs $\exists$ et $\forall$ (pas le nombre d'alternances, simplement le nombre). Par exemple la formule $\exists x : \forall y : y \to x$, dont les modèles sont les graphes de fonctions constantes, a rang de quantification 2.

Lemme 6.5 ([Gam+21]). *Pour tous DULC H et H', si $\pi_{H,m} = \pi_{H',m}$ alors $H \equiv_m H'$.*

Ce sont les deux énoncés suivants qui seront utilisés dans la dernière partie du raisonnement menant au théorème 6.1, afin d'obtenir les graphes B, N, S pour appliquer le lemme 6.2. Le lemme 6.6 permettra de passer par les DULC puis revenir aux graphes.

Lemme 6.6 ([Gam+21]). *Il existe $\mathcal{E}(\psi)$ telle que $G \models \psi$ si et seulement si $\mathcal{E}_m(G) \models \mathcal{E}(\psi)$.*

Idée de démonstration. Étant donnée ψ de rang de quantification m, on veut exhiber $\mathcal{E}(\psi)$. Pour un m donné, il y a un nombre fini de profils de DULC, et il existe donc une formule finie $\mathcal{E}(\psi)$ qui exprime « le DULC a l'un des profils de $\mathcal{E}_m(G)$ avec $G \models \psi$ ». Si $G \models \psi$, alors $\mathcal{E}_m(G) \models \mathcal{E}(\psi)$ par définition de $\mathcal{E}(\psi)$. Et si $\mathcal{E}_m(G) \models \mathcal{E}(\psi)$, alors, par définition de $\mathcal{E}(\psi)$, le DULC $\mathcal{E}_m(G)$ a le même profil $\pi_{\mathcal{E}_m(G),m} = \pi_{\mathcal{E}_m(G'),m}$ qu'un G' avec $G' \models \psi$. Par le lemme 6.5, on obtient $\mathcal{E}_m(G) \equiv_m \mathcal{E}_m(G')$, donc le lemme 6.4 donne $G \equiv_m G'$, et l'on peut conclure $G \models \psi$. □

Lemme 6.7 ([Gam+21]). *Si ψ est une formule ω non triviale sur les DULC, alors il existe des DULC B, N, S vérifiant les hypothèses du lemme 6.2 pour* NP*, ou bien les hypothèses symétriques pour* coNP.

Idée de démonstration. On manipule les profils et les boules qui sont comptées, afin d'obtenir B, N et S. Soit m le rang de quantification de ψ. L'union disjointe de deux DULC faisant croître les valeurs du profil, on peut considérer le profil maximum ρ où les comptages de toutes les boules valent $+\infty$ (excèdent la valeur maximum $e(m)$ donnée par le lemme de localité de Hanf). Sans perte de généralité, supposons que le profil ρ satisfait ψ (l'autre cas est symétrique). Puisque ψ est ω non triviale, il existe un profil maximal ne satisfaisant pas ψ, appelons θ l'un d'entre eux.

Le profil θ nous permettra de conclure : un graphe B dont c'est le profil sera notre base, une boule N de rayon 3^m telle que $\theta(N) = +\infty$ sera notre neutre (en ajouter des copies ne change pas la valeur du profil), et une boule S de rayon 3^m telle que $\theta(S) < +\infty$ sera notre graphe suffisant, par maximalité de θ (en ajouter au moins une copie nous fera passer dans un profil satisfaisant la formule ψ). □

6.1.3. Conclusion du théorème « de type Rice » et corollaires

La preuve du théorème 6.1 est conclue par une disjonction de cas sur la formule ψ.

Idée de démonstration du théorème 6.1. Le premier cas est : si ψ et $\neg\psi$ possèdent chacune des modèles contenant des cycles de longueur non bornée, alors ψ-**Dyn** est soit NP-difficile soit coNP-difficile. Puisque les cycles sont non bornés, le passage aux DULC, avec la formule $\mathcal{E}(\psi)$ donnée par le lemme 6.6, préserve la ω non trivialité (les DULC remplacent les arbres par des étiquettes, mais gardent les cycles). On peut ensuite appliquer le lemme 6.7 pour obtenir des DULC B, N, S, que l'on peut ramener dans le monde des graphes grâce au lemme 6.6, et il reste enfin simplement à appliquer le lemme 6.2.

Les autres cas sont les formules dont les modèles ont des cycles bornés mais des degrés non bornés, puis les formules dont les modèles ont des cycles bornés et des degrés bornés mais des arbres de profondeur non bornée, et enfin les formules dont les modèles ont des cycles bornés mais des occurrences des composantes connexes non bornées. Ces cas requiert des assemblages de graphes autres que la simple union disjointe considérée dans cette version allégée, avec redirection de certains arcs bien identifiés. À isomorphisme près (les formules FO ne peuvent pas désigner de configuration particulière), la négation de ces quatre cas pour des bornes données n'engendre qu'un nombre fini de graphes, donc une formule ω non triviale possède une infinité de modèles dans un de ces cas, qui nous permet de conclure. □

Les constructions employées sont suffisamment souples pour autoriser l'ajout des restrictions suivantes. Soit $\mathcal{G}_f[\Omega_f]$ le sous-graphe de la dynamique $\mathcal{G}_f$ induit par l'ensemble limite Ω_f, c'est-à-dire la dynamique restreinte au comportement asymptotique, que l'on nommera *dynamique limite*.

ψ-Dynamique limite (ψ-**DynLim**) *Entrée :* un RA f déterministe à n automates (circuits des fonction locales). *Question :* est-ce que $\mathcal{G}_f[\Omega_f] \models \psi$?

Le passage à la dynamique limite nous affranchit de l'obstacle lié au nombre de sommets du graphe considéré (qui nous est imposé par la partie de la preuve utilisant la théorie des modèles), car on peut prendre un $\mathcal{G}_f$ « plus gros » à q^n sommets puis en enlever le nombre de sommets voulu lors du passage à $\mathcal{G}_f[\Omega_f]$. On obtient ainsi le résultat suivant pour un alphabet uniforme fixé.

Corollaire 6.8 ([Gam+21])**.** *Si ψ est ω non triviale sur les dynamiques limites, alors ψ-**DynLim** est* NP- *ou* coNP*-difficile, même restreint aux RA déterministes à alphabet uniforme fixé.*

Le second corollaire concerne les réseaux d'automates bijectifs, qui sont des unions disjointes de cycles. Vérifier qu'un réseau d'automates est bijectif étant coNP-difficile (théorème 5.17), on ajoute une promesse à la formulation du problème.

ψ-Dynamique bijectif (ψ-**DynBij**) *Entrée :* un RA f déterministe à n automates (circuits des fonction locales). *Promesse :* f est bijectif. *Question :* est-ce que $\mathcal{G}_f \models \psi$?

Notre construction du lemme 6.2 (figure 6.1) avec l'union disjointe préserve la bijectivité lorsque B, N, S sont bijectifs, et l'on obtient le corollaire suivant.

Corollaire 6.9 ([Gam+21])**.** *Si ψ est ω non triviale sur les dynamiques bijectives, alors ψ-**DynBij** est* NP- *ou* coNP*-difficile, pour les RA déterministes à alphabet non fixé.*

6.2. Complétude pour tous les niveaux de PH

Le théorème 6.1 donne une bonne inférieure de complexité pour toute formule du premier ordre ω non triviale, mais aucune borne supérieure. La borne supérieure triviale est PH de par la structure d'une formule du premier ordre, et elle est optimale d'après le résultat suivant qui donne des bornes exactes.

Théorème 6.10 ([Gam+21])**.** *Pour tout $N \in \mathbb{N}_+$, il existe une formule du premier ordre ψ_N telle que ψ_N-***Dyn** *est Σ_N^{P}-complet.*

Idée de démonstration. Soit $N \in \mathbb{N}_+$. Étant donnée une formule booléenne φ de la forme $\exists b_1 : \forall b_2 : \ldots : Q b_N : R(b_1, \ldots, b_N)$ dans Σ_N^{P} (avec $Q = \exists$ si N est impair, et $Q = \forall$ si N est pair), nous allons construire en temps polynomial un RAB déterministe f_φ (dépendant de φ) et une formule FO ψ_N (ne dépendant que de N), tels que $\mathscr{G}_{f_\varphi} \models \psi_N$ si et seulement si φ est vraie. Pour simplifier l'exposé, on supposera que b_i est une variable booléenne ; la démonstration s'adapte pour un vecteur booléen.

Le RAB déterministe f_φ est constitué de $N+2$ automates (voir figure 6.2) :

$$\{\top, \bot\} \sqcup \bigsqcup_{i=1}^{N} \{0,1\}^i \sqcup \overline{\{0,1\}^N} \sqcup \overline{\overline{\{0,1\}^N}}$$

de façon à ce que $\mathscr{G}_{f_\varphi}$ soit constitué :

- des points fixes $\top$, $\bot$, et $\overline{\overline{\{0,1\}^N}}$ (ces derniers pour du remplissage),
- d'un arbre binaire complet sur $\bigsqcup_{i=1}^{N} \{0,1\}^i$ qui pointe en direction de la racine $\top$,
- une configuration $\overline{x}$ pour $x \in \{0,1\}^N$ pointe vers la configuration x si le N-uplet x satisfait R, et vers $\bot$ sinon.

Les fonctions locales sont construites en temps polynomial d'après cette description. On remarquera que le nombre de configurations est bien : $2 + \sum_{i=1}^{N} 2^i + 2 \cdot 2^N = 2^{N+2}$.

La formule ψ_N considérée décrit les graphes de la forme de $\mathscr{G}_{f_\varphi}$ dans lesquels on a l'alternance de quantifications souhaitée. Avec $x \to y$ pour $f(x) = y$ dans la signature de notre logique du premier ordre, et $\implies$ l'implication utilisée pour implémenter « pour tout antécédent », ψ_N est :

$$\begin{aligned}&\exists x_0, x_1, y_2 : \forall x_2 : \exists x_3 : \forall x_4 : \ldots : Q x_N : \exists y_N : (x_0 \neq x_1) \wedge (y_2 \to x_1 \to x_0 \to x_0) \wedge \\ &\quad [(x_2 \to x_1) \implies [(x_3 \to x_2) \wedge [(x_4 \to x_3) \implies [\ \ldots\ (y_N \to x_N \to x_{N-1})]]]]\end{aligned}$$

(notons que x_i dans ψ_N est une configuration, alors que b_i dans φ est une variable booléenne, pour $i \in [N]$). La première ligne force la configuration x_0 à être $\top$, et les chemins de longueur $N+1$ ont pour source une configuration feuille $y_N \in \overline{\{0,1\}^N}$ correspondant à un N-uplet satisfait la relation R. Ainsi, la formule et la construction forcent la cohérence des variables x_1 à x_N (dans le sens où, partant de x_N, on égraine les bits pour arriver à x_1), et dans la dernière partie $y_N \to x_N$ force que x_N (et donc les choix de x_1 à x_N par ce qui précède) corresponde à un N-uplet satisfaisant R.

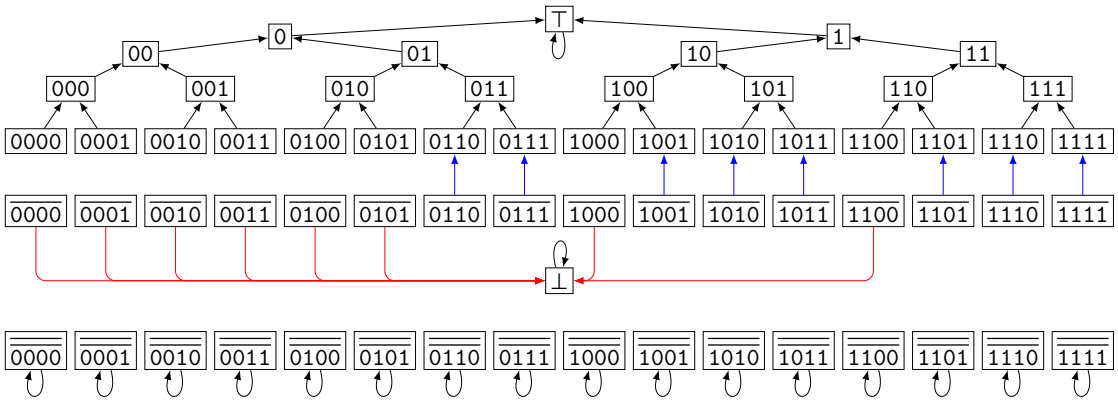


FIGURE 6.2. – Illustration de la construction du théorème 6.10, pour la formule :

$$\varphi = \exists b_1 : \forall b_2 : \exists b_3 : \forall b_4 : (b_1 \wedge b_4) \vee (b_2 \wedge b_3) \vee (b_1 \wedge \neg b_2 \wedge b_3 \wedge \neg b_4)$$

où $N = 4$ est pair. On a l'arbre binaire complet de hauteur 4 relié à $\top$, et les configurations $\overline{(b_1, b_2, b_3, b_4)}$ pointent vers (b_1, b_2, b_3, b_4) lorsqu'elles satisfont la partie propositionnelle de φ, et vers $\bot$ sinon. Il s'agit d'une instance φ positive du problème complet pour Σ_4^{P}, et l'instance f_φ dont c'est le graphe de la dynamique est une instance positive de ψ_4-**Dyn** car il existe le point fixe $x_0 = \top$ et l'antécédent $x_1 = 1$, tel que pour tous ses antécédents $x_2 \in \{10, 11\}$, il existe l'antécédent $x_3 = 101, 111$ respectivement, tel que tous leurs antécédents $x_4 \in \{1010, 1011, 1110, 1111\}$ ont chacun un antécédent.

L'alternance de quantification sur les variables x_1 à x_N dans ψ_N étant identique à celle sur b_1 à b_N dans φ, on obtient le *si et seulement si* démontrant la Σ_N^{P}-difficulté.

Remarquons que ψ_N ne parle pas de R, c'est bien l'instance f_φ de ψ_N-**Dyn** qui est l'image de φ par notre réduction et donne $\mathscr{G}_{f_\varphi} \models \psi_N \iff \varphi$ est vraie.

Pour N impair, l'appartenance à Σ_N^{P} découle immédiatement du fait que $Q = \exists$. Pour N pair, on considère $\neg\varphi$ où l'on passe la négation jusqu'à la relation R (ainsi $Q = \exists$ et la formule est Π_N^{P}), avec la même réduction vers $f_{\neg\varphi}$, en adaptant ψ_N :

$$\forall x_0, x_1, y_2 : \exists x_2 : \forall x_3 : \exists x_4 : \ldots : Q x_N : \exists y_N : [(x_0 \neq x_1) \wedge (y_2 \to x_1 \to x_0 \to x_0)] \implies$$
$$[(x_2 \to x_1) \wedge [(x_3 \to x_2) \implies [(x_4 \to x_3) \wedge [\ \ldots\ (y_N \to x_N \to x_{N-1})]]]].$$

La première ligne permet de restreindre la seconde ligne aux cas où x_0 joue le rôle de $\top$ (sous l'hypothèse que la relation R n'est pas vide, ce qui ne change pas la complexité), et le reste du raisonnement est analogue. Ainsi $\mathscr{G}_{f_{\neg\varphi}} \models \psi_N \iff \neg\varphi$ est vraie, et ψ_N-**Dyn** est dans Π_N^{P} lorsque N est pair. On en déduit que ψ_N-**Dyn** est Π_N^{P}-complet, donc $(\neg\psi_N)$-**Dyn** est Σ_N^{P}-complet lorsque N est pair. □

On obtient immédiatement le corollaire suivant, dans le cas où la formule ψ est donnée en entrée du problème.

FO-Dynamique (**FO-Dyn**) *Entrée :* un RA f et une formule FO ψ. *Question :* est-ce que $\mathscr{G}_f \models \psi$?

Corollaire 6.11 ([Gam+21])**.** **FO-Dyn** *est* PSPACE*-complet pour les RA booléens déterministes.*

6.3. Perspectives sur FO

Alphabet fixé. Nous souhaitons généraliser le théorème 6.1 aux RA déterministes uniformes à alphabet fixé, ce que nous formulons comme la conjecture suivante.

Conjecture 6.12. *Si ψ est ω non triviale, alors ψ-***Dyn** *est soit* NP*-difficile soit* coNP*-difficile pour les RA déterministes à alphabet uniforme fixé (par exemple booléen).*

Il s'agit de retravailler la construction du lemme 6.2 (comme évoqué à sa suite), pour obtenir un réseau d'automates uniforme sur un alphabet de taille q (un réseau q-uniforme avec q fixé). Dans ce cas, la dynamique doit nécessairement comporter exactement q^n sommets où n est le nombre d'automates. Or, nous n'avons pas le choix sur les tailles des graphes B, N, S qui sont donnés par une technique de pompage en théorie des modèles (lemme 6.7) : partant d'une base B, on peut ajouter des copies de N et S. Nous avons vu que l'on peut supposer $|N| = |S| = \alpha$ et en prendre une quantité polynomiale arbitraire, mais en partant de $|B| = \gamma$, nous ne savons pas résoudre l'équation $\gamma + x \cdot \alpha = q^n$ pour α, γ, q fixés et x, n variables. Un angle d'attaque consisterait plutôt à améliorer la technique de pompage, afin d'avoir plus de souplesse sur les graphes B, N, S (en particulier la suppression de B résoudrait l'obstacle).

Représentations succinctes de graphes. Au regard du passage de n à 1 automate lorsque l'alphabet n'est pas fixé (voir chapitre 3.4), le théorème 6.1 donne dans sa forme présente un « méta-théorème » de difficulté sur les représentations succinctes de graphes, tel que suggéré en perspective de [GW83]. Rappelons que sur un automate d'alphabet $[q]$ il n'y a plus d'interactions, l'unique circuit encode la relation binaire $A \subseteq V \times V$ sur $V = [2^{\lceil \log q \rceil}]$. Une revue bibliographique plus approfondie des représentations par circuit de graphes pourrait apporter des bornes de complexité aux réseaux d'automates, sur des propriétés du graphe de la dynamique :

- supérieures dans les cas déterministe ou non, à alphabet fixé ou non,
- inférieures dans le cas non déterministe à alphabet non fixé, avec une réduction triviale vers un unique automate dont le circuit est la représentation succincte du graphe (construction qui n'exploite pas l'essence interactionnelle du modèle).

Voici quelques résultats, pour la représentation succincte d'un graphe orienté $G = (V, A)$ par un circuit à $2 \cdot \lceil \log |V| \rceil$ bits d'entrée et un bit de sortie, encodant sur l'entrée $x, y \in V$ si $(x, y) \in A$. Pour les graphes non orientés, on ne regarde que la moitié de la

matrice d'adjacence ainsi encodée (les entrées avec $i < j$). L'article [GW83] a introduit cet encodage (qui est le sujet de la thèse de doctorat de Galperin, 1982) et considéré les problèmes dans P (pour l'encodage non succinct par matrice d'adjacence) qui deviennent NP-difficiles ; il a rapidement été suivi par [PY86] et [LB89] qui donnent des bornes de difficulté plus élevées pour certains problèmes, notamment des passages de NP à NEXPTIME, et closent la plupart des caractérisations ouvertes dans l'article de Galperin et Wigderson).

Théorème 6.13 ([GW83 ; PY86 ; LB89])**.** *Étant donné un graphe succinct, décider :*

(i) s'il a au moins un arc ou une arête, s'il a un triangle, si son degré maximum est au moins k fixé, sont des problèmes NP*-complets ;*

(ii) si son degré minimum est au plus k fixé, est un problème Σ_2^{P}*-complet ;*

(iii) s'il est acyclique, s'il existe un chemin de x à y donnés, s'il est connexe, s'il est planaire, s'il est bipartie, sont des problèmes PSPACE*-difficiles ;*

(iv) s'il est 3*-colorable, s'il a une clique de taille* $\frac{|V|}{2}$*, s'il a un chemin Hamiltonien, sont des problèmes* NEXPTIME*-complets.*

Notons que les trois articles cités au théorème 6.13 présentent des « méta-théorèmes » qui sont ensuite appliqués aux différents problèmes.

Nommage des configurations. Une autre extension de notre théorème de Rice sur les propriétés exprimables en logique du premier ordre, consiste à permettre aux formules de nommer explicitement des configurations. En effet, nous avons mentionné qu'une formule FO sur la signature $\{=, \to\}$ considère le graphe de la dynamique à isomorphisme près, c'est-à-dire si $\mathscr{G}_g$ est un renommage des sommets de $\mathscr{G}_f$ alors $\mathscr{G}_g \models \varphi \iff \mathscr{G}_f \models \varphi$. Or dans les applications des réseaux d'automates (par exemple à la régulation génétique en biologie), les étiquettes des configurations peuvent porter une information qui n'est pas « interchangeable ». On souhaite ainsi ajouter des relations à la signature, qui parleraient des états des automates dans les configurations, comme par exemple $E_i^q(x)$ qui serait vraie si et seulement si $x_i = q$. De nombreuses questions sont alors P-complètes, comme intuitivement :

$$\exists x, y : (x \to y) \wedge E_0^1(y) \wedge \bigwedge_{i \in [n]} E_i^0(x)$$

dans le cas déterministe booléen, qui demande à évaluer le circuit de la fonction locale f_0 sur l'entrée 0^n pour savoir si la sortie vaut 1 (on réduit depuis *Circuit Value Problem* à entrée fixée, voir annexe B.2). Cette observation laisse penser que les techniques sont largement à retravailler, en commençant par la définition de formule « triviale ».

6.4. Perspectives sur MSO

Cette section présente des travaux en cours avec mes co-auteurs de [Gam+21] : Guilhem Gamard (LIS, Marseille), Pierre Guillon (I2M, Marseille) et Guillaume Theyssier

(I2M, Marseille). Nous sommes en quête d'un équivalent pour les formules en logique monadique du second ordre (*monadic second order*, MSO) [Lib04 ; CE12].

En MSO, on peut quantifier sur les sommets, les arcs, les ensembles de sommets et les ensembles d'arcs. Le reste est identique à FO, et l'on peut remonter toutes les quantifications du second ordre avant toutes les quantifications du premier ordre [Lib04, page 115]. Bien que MSO soit plus expressive que FO (voir par exemple [CE12, section 5.2]), rien ne dit a priori que les nouvelles formules soient toutes difficiles.

Avec une technique analogue au cas du premier ordre, nous arrivons présentement à obtenir le résultat suivant.

Théorème 6.14 (travail en cours)**.** *Si ψ est une formule MSO avec une infinité de modèles de largeur arborescente k_1 et une infinité de contre-modèles de largeur arborescente k_2, alors ψ-***Dyn** *est soit* NP*- soit* coNP*-difficile.*

Ces réflexions nous ont menés à poser la question suivante, répondue par des échanges avec Édouard Bonnet (LIP, Lyon), et d'un intérêt sans doute plus général que nos travaux sur les réseaux d'automates.

Théorème 6.15. *Soit $m \in \mathbb{N}$. Il existe un graphe Λ_m tel que, pour toute formule MSO ψ de rang de quantification m, l'un des deux énoncés suivants soit vérifié :*

- *pour tout graphe G on a $G \sqcup \Lambda_m \models \psi$, ou*
- *pour tout graphe G on a $G \sqcup \Lambda_m \not\models \psi$.*

En mots, tout formule MSO admet soit un graphe *suffisant* (dont l'union disjointe avec n'importe quel G satisfait la formule), soit un graphe *interdit* (dont l'union disjointe avec n'importe quel G ne satisfait pas la formule). Remarquons (!) que le graphe Λ_m dépend uniquement du rang de quantification m, et pas de la formule ψ choisie.

Idée de démonstration, par Édouard Bonnet (LIP, Lyon). La principale difficulté technique consiste à démontrer, en utilisant un jeu de Ehrenfeucht-Fraïssé pour MSO [Lib04, section 7.2], que, pour tout graphe non vide G, il existe un entier $q(G,m)$ tel que $\bigsqcup^{q(G,m)} G \equiv_m \bigsqcup^{q(G,m)+1} G$. L'entier en question est une tour d'exponentielle.

Puisque la relation $\equiv_m$ a un nombre fini $a(m)$ de classes d'équivalences [Lib04, proposition 7.5 page 116], on peut désigner des représentants $A_1, \ldots, A_{a(m)}$ de chaque classe. On définit alors :

$$\Lambda_m = \bigsqcup_{i=1}^{a(m)} \bigsqcup^{q(A_i,m)} A_i.$$

Pour tout G, on a $G \equiv A_i$ pour $i \in [a(m)]$, donc $G \sqcup \Lambda_m \equiv_m A_i \sqcup \Lambda_m$ (cette fois la stratégie de Duplicator est évidente). L'ajout de A_i à Λ_m constitue la « $q(A_i,m)+1$ -ème » copie de A_i, qui donne $A_i \sqcup \Lambda_m \equiv_m \Lambda_m$. On conclut bien $G \sqcup \Lambda_m \equiv_m \Lambda_m$: il s'agit d'un modèle ou d'un contre-modèle indépendamment du graphe G. □

On appelle *arborescente* une formule ψ qui satisfait les conditions du théorème 6.14. Donc, quid des formules non arborescentes ? On sait que l'arborescence joue un

rôle important dans les relations entre MSO et la théorie de la complexité (on pense bien évidemment au théorème de Courcelle [DF13, chapitre 13]). Les formules non arborescentes ne sont pas extravagantes, le fait d'être une clique en est une : pour toute taille n, il n'y a qu'un unique modèle, qui a largeur arborescente $n-1$.

Tout cela est très enthousiasmant! Les énoncés sont très généraux et la théorie des modèles est un domaine d'une grande richesse. Les réseaux d'automates apportent également des questions nouvelles, et nous ne savons pas encore précisément jusqu'où tout cela va nous mener, mais nous prenons bien du plaisir au voyage.

7. Complexité de la dynamique asymptotique II : G_f en entrée

Étudier la dynamique d'un réseau d'automates à partir de son graphe d'interaction est au cœur du domaine, et constitue un intérêt majeur des travaux sur le modèle. Cette aspiration est motivée par les applications à la biologie, où les réseaux d'automates modélisent les phénomènes de régulation génétique : dans le noyau d'une cellule, l'expression d'un gène produit des protéines qui activeront ou inhiberont l'expression d'autres gènes. Les attracteurs de cette dynamique (le comportement asymptotique) sont les états stables et oscillations stables de l'expression génique (points fixes et cycles limites), et peuvent chacun correspondre à un type cellulaire ou plus largement à des phénotypes particuliers [Kau69 ; Rob69 ; Tho73 ; TT95 ; MTA99]. Par des expérimentations en laboratoire « humide » on découvre le graphe d'interaction signé d'un réseau d'automates (ou des corrélations entre les états des automates), dont il s'agit de comprendre la dynamique. C'est précisément le type de questions que nous aborderons dans ce chapitre, qui nous fera visiter un large éventail de classes de complexité, de P à NEXPTIME.

On se restreint dans ce chapitre aux réseaux d'automates booléens déterministes mis à jour en parallèle, et n désignera le nombre d'automates.

7.1. Information du graphe d'interaction

Contrairement aux circuits des fonctions locales, prendre un graphe d'interaction signé G en entrée du problème ne donne pas une unique dynamique, mais définit :

$$\mathcal{F}_G = \left\{ f : \{0,1\}^n \to \{0,1\}^n \;\middle|\; G_f^{\pm} = G \right\}$$

et toutes les dynamiques associées. La taille de $\mathcal{F}_G$ est en général très grande : il y a 2^{n2^n} RAB à n automates et seulement 4^{n^2} graphes d'interaction signés à n sommets. On partitionne naturellement nos considérations à chaque fonction locale : étant donné $i \in [n]$ on définit :

$$\mathcal{F}_G(i) = \left\{ f_i : \{0,1\}^n \to \{0,1\} \;\middle|\; f \in \mathcal{F}_G \right\}.$$

La taille de $\mathcal{F}_G(i)$ est doublement exponentielle en le degré entrant $d \in \mathbb{N}$ de i dans G. La valeur précise de $|\mathcal{F}_G(i)|$ en fonction de d n'est pas triviale et un fameux sujet de

recherche dans le cas où i n'a que des dépendances positives [A00b]. La difficulté est liée à l'essentialité des variables (voir chapitre 4.2).

On a en revanche plusieurs cas extrêmes, à commencer par des graphes signés G tels que $\mathcal{F}_G = \emptyset$. Par exemple, avoir un automate de degré entrant 1 avec une dépendance non monotone est impossible à obtenir. Plus généralement, on a le résultat suivant, que l'on teste en temps linéaire.

Lemme 7.1 ([PR10])**.** *Étant donné un graphe signé G, il existe f tel que $G_f^{\pm} = G$ si et seulement si tout sommet ayant un unique arc entrant non monotone a au moins deux autres arcs entrants monotones.*

Pour un automate de degré entrant 0, il existe deux fonctions locales possibles : les constantes 0 et 1. Pour un automate de degré entrant 1, il existe une unique fonction locale correspondante : l'identité si l'arc est positif, et la négation si l'arc est négatif. Ce sont des observations que nos constructions utiliseront.

Remarquons un travail récent de Picard Marchetto et Richard, démontrant notamment que le graphe d'interaction complet donne une information quasi-nulle sur le graphe de la dynamique déterministe : à isomorphisme près, toutes les dynamiques peuvent être obtenues, sauf deux (la dynamique identité $f(x) = x$ pour tout $x \in \{0,1\}^n$ est uniquement engendrée par le graphe d'interaction munit exclusivement des n boucles, et la dynamique constante avec $\tilde{x} \in \{0,1\}^n$ et $f(x) = \tilde{x}$ pour tout $x \in \{0,1\}^n$ est uniquement engendrée par le graphe d'interaction sans aucun arc ni boucle).

Théorème 7.2 ([PR21])**.** *Pour tout $n \geq 5$ et tout réseau $f : \{0,1\}^n \to \{0,1\}^n$ différent de l'identité et d'une constante, il existe un réseau $g : \{0,1\}^n \to \{0,1\}^n$ de dynamique $\mathscr{G}_g$ isomorphe à $\mathscr{G}_f$, et dont le graphe d'interaction G_g est la clique à n sommets et n^2 arcs.*

7.2. Notations

On notera $\mathcal{N}_G(i)$ les *voisins entrants* de l'automate $i \in [n]$ dans le graphe signé G, que l'on partitionne en :

- $\mathcal{N}_G^+(i)$ les voisins entrants *positifs*,
- $\mathcal{N}_G^-(i)$ les voisins entrants *négatifs*,
- $\mathcal{N}_G^{\pm}(i)$ les voisins entrants *non monotones*.

On dira qu'un graphe signé G est *simple* lorsqu'il ne comporte pas d'arc non monotone (c'est-à-dire $\mathcal{N}_G^{\pm}(i) = \emptyset$ pour tout $i \in [n]$).

Afin de dépendre effectivement de $\mathcal{N}_G(i)$ dans un graphe d'interaction signé simple, on peut attribuer une conjonction (*resp.* une disjonction) à la fonction locale de l'automate $i \in [n]$, tout en respectant les signes avec des négations :

$$f_i(x) = \bigwedge_{j \in \mathcal{N}_G(i)} (x_j + \tilde{\sigma}_{ji}) \qquad (\textit{resp. } f_i(x) = \bigvee_{j \in \mathcal{N}_G(i)} (x_j + \tilde{\sigma}_{ji}))$$

où $\tilde{\sigma}_{ji} = 0$ si l'arc (j,i) est positif, et $\tilde{\sigma}_{ji} = 1$ si l'arc (j,i) est négatif. On parlera de fonction ET (*resp.* fonction OU) associée à l'automate i sur le graphe signé G.

7.3. Nombre maximum de points fixes

Nous allons commencer par nous intéresser aux problèmes de calcul du nombre maximum de points fixes d'un RAB correspondant à un graphe d'interaction donné. Rappelons que $\mathfrak{P}_f$ désigne l'ensemble des points fixes d'un RAB f. La quantité dont nous allons traiter est :

$$\mathfrak{P}^{\max}(G) = \max\left\{|\mathfrak{P}_f| \;\middle|\; f \in \mathcal{F}_G\right\},$$

à travers le premier problème suivant, où la valeur de k est fixée (le lecteur souhaitant une vue d'ensemble des résultats à venir pourra consulter la table 7.1 page 93).

Point fixe maximum k (**PFmax-**k) *Entrée :* un graphe d'interaction signé G. *Question :* est-ce que $\mathfrak{P}^{\max}(G) \geq k$?

Pour $k = 1$, c'est-à-dire décider s'il existe un RAB avec au moins un point fixe, le problème est « facile » (mais l'algorithme n'est pas du tout trivial !).

Théorème 7.3 ([Bri+19])**.** **PFmax-**1 *est dans* P.

Idée de démonstration. Étant donnée une instance G', on commence par se ramener à un graphe d'interaction simple G en supprimant des arcs de G' (incluant tous les arcs non monotones, mais possiblement plus), avec $\mathfrak{P}^{\max}(G') \geq 1 \iff \mathfrak{P}^{\max}(G) \geq 1$.

On montre ensuite le résultat suivant : $\mathfrak{P}^{\max}(G) \geq 1$ si et seulement si toute composante fortement connexe initiale (n'ayant pas d'arc depuis une autre composante vers elle) et non triviale (qui n'est pas un sommet isolé sans arc) de G possède un cycle positif. Pour l'implication directe, la présence d'un cycle positif permet de construire des fonctions ET et OU telles que des points fixes partiels sur les composantes initiales vont se répandre sur les autres automates du réseau. Pour l'implication réciproque (déjà énoncée dans [Ara08, corollaire 3]), si x est un point fixe alors, pour chaque automate $i \in [n]$, le fait que $f_i(x) = x_i$ implique qu'au moins une dépendance de i est « non frustrée », c'est-à-dire qu'il existe un automate $j \in \mathcal{N}_G(i)$ tel que $x_i = x_j + \tilde{\sigma}_{ji}$ (les états de i et j correspondent au signe de l'arc (j,i), cela provient de la monotonie). Par induction, on doit retomber sur un automate ℓ déjà visité et avoir $x_\ell = x_\ell$, ce qui correspond à avoir parcouru un cycle positif dans la composante (par hypothèse sur celle-ci on ne peut pas en sortir).

Décider si un graphe possède un cycle de longueur paire est réalisable en temps polynomial, il s'agit d'une conjecture difficile résolue par [McC04 ; RST99]. On doit donc calculer les composantes fortement connexes initiales (en temps linéaire), et

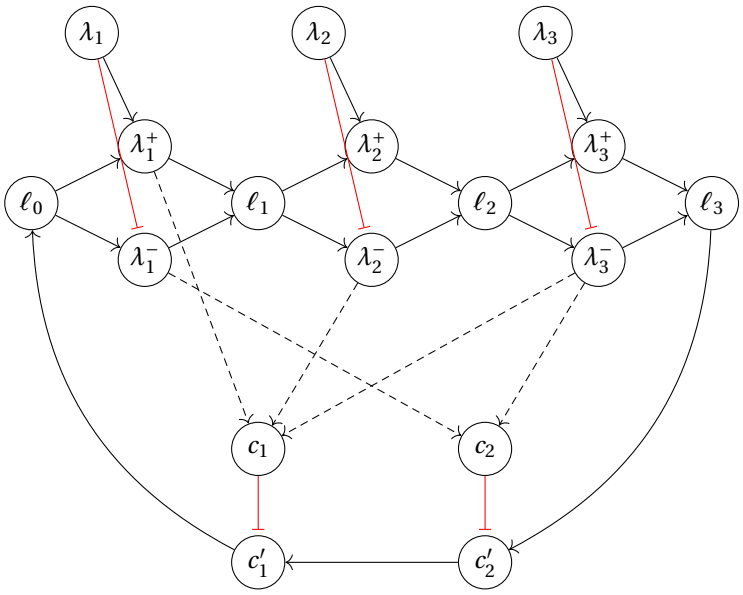


FIGURE 7.1. – Graphe orienté signé (arcs négatifs en rouge avec pointe plate) obtenu dans la réduction de **3-SAT** à **PFmax-k**, pour la formule :
$$\varphi = (\lambda_1 \vee \neg\lambda_2 \vee \neg\lambda_3) \wedge (\neg\lambda_1 \vee \neg\lambda_3).$$

pour chaque composante transformer les arcs positifs en des chemins de longueur deux, puis appeler l'un des algorithmes que nous venons de citer (la complexité annoncée est cubique). La conjonction des réponses correspondra à notre décision. □

Pour $k \geq 2$ le problème gagne en complexité.

Théorème 7.4 ([Bri+19])**.** **PFmax-k** *est* NP*-complet pour tout* $k \geq 2$*, même restreint à* $\Delta(G) \leq 2$.

Idée de démonstration. L'appartenance à NP n'est pas triviale car, comme nous l'avons vu au chapitre 4, vérifier si $G = G_f^{\pm}$ est un problème dans NP si et seulement si NP = coNP. L'astuce consiste à donner un certificat contenant :

- k points fixes,
- une configuration par arc positif ou négatif,
- deux configurations par arc non monotone.

On démontre que si ces informations sont cohérentes avec le graphe G, alors il existe une extension de cette information partielle en un RAB f tel que $G_f^{\pm} = G$. Il s'agit donc bien d'un certificat, et toute instance positive admet un tel f donc un tel certificat.

On commence par montrer la NP-difficulté pour $k = 2$. Étant donnée une instance φ de **3-SAT**, on construit un graphe orienté signé tel qu'illustré en figure 7.1. À chaque variable λ_i de φ, on fait correspondre trois automates, λ_i^+, λ_i^- et λ_i. Les arcs tiretés implémentent les clauses de la formule. Cette construction a toujours un RAB avec au

moins un point fixe, en prenant des fonctions ET et tous les états à 0. La partie plus confortable de la démonstration consiste à exhiber un f avec $G_f^{\pm} = G$, qui a toujours au moins un point fixe, et un second point fixe si φ est satisfaisable (c'est la valuation satisfaisant φ qui permet de construire le second point fixe). Intéressons-nous plutôt à la réciproque.

Que faut-il pour avoir deux points fixes $x \neq y$? Pour commencer, les automates λ_i sont constants, donc $x_{\lambda_i} = y_{\lambda_i}$. Aracena a montré dans [Ara08] que $x_I \neq y_I$ pour tout *feedback vertex set* positif (FVS+, qui coupe tous les cycles positifs) I de G. Nous avons des FVS+ singletons : $x_{\ell_i} \neq y_{\ell_i}$ et $x_{c'_i} \neq y_{c'_i}$ pour tout i, pour lesquels un argument de « non frustration » implique qu'ils sont tous dans l'état 0 dans x et tous dans l'état 1 dans y, ou inversement. En considérant les deux fonctions locales possibles en c'_i, on observe alors par induction de la droite vers la gauche que pour respecter les dépendances données par G on doit avoir $x_{c_i} = y_{c_i}$ pour tout i. Cette dernière déduction requiert à son tour, par un nouvel argument de « non frustration », que chaque c_i ait un antécédent dont les états ne diffèrent pas entre x et y : il s'agit d'une variable qui « satisfait » la clause. Enfin, les couples $\{\lambda_i^+, \lambda_i^-\}$ sont des FVS+ de taille deux, donc si l'un est utilisé pour une clause, l'autre doit avoir des états différents dans x et y (et ne peut donc pas « satisfaire » de clause). On conclut bien que pour avoir deux points fixes, la formule doit est satisfaisable. On peut également voir le choix de la valuation comme, pour un cycle de différences entre x et y passant par ℓ_0 et ℓ_n, le complémentaire des choix réalisés lors de la traversée de la partie haute du graphe.

Augmenter la valeur de k est ensuite évident, en ajoutant à G des boucles isolées multipliant par 2 le nombre de points fixes. On en choisira la bonne quantité ℓ pour que la valeur de k tombe entre 2^ℓ et $2^{\ell+1}$.

On peut enfin abaisser le degré entrant à $\Delta(G) = 2$, en utilisant deux automates par clause (qui sont les seuls automates de degré entrant supérieur à 2 dans notre construction de la figure 7.1). □

On peut ensuite considérer que la valeur de k n'est pas fixée, mais incluse dans l'entrée du problème, encodée en binaire. La complexité du problème augmente brutalement.

Point fixe maximum (**PFmax**) *Entrée :* un graphe d'interaction signé G et un entier binaire $k \in \mathbb{N}_+$. *Question :* est-ce que $\mathfrak{P}^{\max}(G) \geq k$?

Théorème 7.5 ([Bri+19])**.** **PFmax** *est* $\mathsf{NP}^{\#\mathsf{P}}$*-complet restreint à* $\Delta(G) \leq d$ *pour* $d \in \mathbb{N}_+$ *fixé (voir erratum 7.6).*

Idée de démonstration. Pour l'appartenance à $\mathsf{NP}^{\#\mathsf{P}}$, on peut deviner les fonctions locales qui sont en quantité finie pour $\Delta(G) \leq d$ fixé, et appeler l'oracle #P pour compter son nombre de points fixes (corollaire 5.2).

Pour la $\mathsf{NP}^{\#\mathsf{P}}$-difficulté, le problème canonique est ∃-**Maj-3-SAT** [LGM98] : étant donnée φ sur les variables $\lambda_1, \dots, \lambda_n$ et $s \in [n]$, on doit décider s'il existe une valuation sur les s premières variables, telle que la majorité des façon de la compléter sur les $n-s$ variables restantes satisfont φ. On adapte la construction précédente (figure 7.1) en ajoutant une boucle positive sur les $n-s$ automates λ_{s+1} à λ_n. Les fonctions locales de ces automates sont toutes l'identité, et tout élément de $\{0,1\}^{n-s}$ correspondra à un ou deux points fixes suivant si elle complète la partie $\lambda_1, \dots, \lambda_s$ (leur rôle ne change pas : les fonctions locales fixent leur valuation) en une valuation qui satisfait φ. On choisit alors la bonne valeur, $k = 3 \cdot 2^{n-s-1}$. □

Erratum 7.6. Dans un article récemment déposé sur arXiv [AW21], Akmal et Williams montrent un résultat surprenant de complexité : **Maj-k-SAT** est dans P pour tout $k \in \mathbb{N}_+$ fixé, alors que ce problème était conjecturé comme étant PP-complet. En conséquence, ils pointent une confusion dans la littérature : le problème ∃-**Maj-3-SAT** que nous utilisons dans la preuve précedente est en réalité « seulement » NP-complet [AW21, théorème 1.2 page 3]. Il semble néanmoins possible (travail en cours) d'adapter notre construction pour une réduction depuis le problème ∃-**Maj-6-SAT** avec une clause de taille arbitraire, qui est $\mathsf{NP}^{\#\mathsf{P}}$-complet [AW21, théorème D.1 page 54].

Théorème 7.7 ([Bri+19])**. PFmax** *est* NEXPTIME*-complet dans le cas général.*

Idée de démonstration. L'algorithme naïf est dans NEXPTIME (on a de l'ordre de 2^{2^d} fonctions possibles par automate de degré entrant d) : deviner f, vérifier $G_f^{\pm} = G$, calculer $|\mathfrak{P}_f|$.

La difficulté est donnée par une réduction depuis **Succinct-3-SAT** [Pap94, théorème 20.2], qui correspond à **3-SAT** mais où la formule φ est donnée sous la forme d'un circuit booléen C_φ prenant (potentiellement) exponentiellement moins de place que son écriture explicite. Le circuit en question prend en entrée le numéro de la clause en binaire et la position dans la clause parmi $0,1,2$, et donne en sortie le numéro de la variable en binaire et sa polarité (éventuelle négation). La formule φ aura 2^n variables et 2^m clauses, avec n et m le nombre de nœuds correspondant en sortie et entrée dans le circuit. On transforme ce circuit en un graphe orienté signé sur la base de notre construction en figure 7.1, prenant une « variable » λ_i pour chaque nœud du circuit et en ajoutant des « clauses » pour contraindre les points fixes d'un RAB sur G à simuler correctement le calcul implémentées dans C_φ (les guillemets sont utilisés pour faire référence aux éléments de la construction en figure 7.1). On aura $\mathfrak{P}^{\mathtt{max}}(G) = 2^{m+1}$ points fixes si C_φ encode une formule φ satisfaisable, et strictement moins sinon. Il y a des arcs supplémentaires, et de nombreux détails techniques. □

La table 7.1 résume les résultats abordés sur le nombre maximum de points fixes.

Problème	$\Delta(G) \leq d$	$k = 1$	$k \geq 2$	k en entrée
$\mathfrak{P}^{\max}(G) \geq k$	non	P	NP-complet	NEXPTIME-complet
	oui			$\mathsf{NP}^{\#\mathsf{P}}$-complet

Table 7.1. – Complexité des problèmes de décider si, étant donné un graphe orienté signé G, il existe un RAB f vérifiant $G_f^{\pm} = G$ avec au moins k points fixes.

Problème	$\Delta(G) \leq d$	$k \geq 1$	k en entrée
$\mathfrak{P}^{\min}(G) < k$	non	NEXPTIME-complet	
	oui	$\mathsf{NP}^{\mathsf{NP}}$-complet	$\mathsf{NP}^{\#\mathsf{P}}$-complet

Table 7.2. – Complexité des problèmes de décider si, étant donné un graphe orienté signé G, il existe un RAB f vérifiant $G_f^{\pm} = G$ avec moins de k points fixes.

7.4. Nombre minimum de points fixes

On s'intéresse ici aux problèmes duaux, et à la quantité

$$\mathfrak{P}^{\min}(G) = \min\left\{|\mathfrak{P}_f| \;\middle|\; f \in \mathcal{F}_G\right\}.$$

Point fixe minimum k (**PFmin-k**)
Entrée : un graphe d'interaction signé G.
Question : est-ce que $\mathfrak{P}^{\min}(G) < k$?

Point fixe minimum (**PFmin**)
Entrée : un graphe d'interaction signé G et un entier binaire $k \in \mathbb{N}_+$.
Question : est-ce que $\mathfrak{P}^{\min}(G) < k$?

Une nouvelle classe apparaît : le second niveau de la hiérarchie polynomiale pour **PFmin-k** à degré borné (pour $k = 1$: il existe un RAB sur G tel que toute configuration n'est pas un point fixe, $\exists\forall$). La table 7.2 résume ces résultats.

Théorème 7.8 ([Bri+20])**.** *Pour $k \geq 1$ et $d \geq 2$ fixés, on a :*
- **PFmin-k** *et* **PFmin** *sont* NEXPTIME*-complets,*
- **PFmin-k** *est* $\mathsf{NP}^{\mathsf{NP}}$*-complet pour* $\Delta(G) \leq d$*,*
- **PFmin** *est* $\mathsf{NP}^{\#\mathsf{P}}$*-complet pour* $\Delta(G) \leq d$*.*

Idée de démonstration. Les bornes supérieures NEXPTIME et $\mathsf{NP}^{\#\mathsf{P}}$ sont données par des algorithmes analogues aux problèmes duaux **PFmax** et **PFmax-k**. Pour **PFmin-k** avec degré borné, on peut devenir les fonctions locales, appeler l'oracle NP pour savoir s'il existe au moins k points fixes, puis accepter si et seulement si l'oracle répond

« non ». Les démonstrations utilisent la construction présentée en figure 7.1, avec la principale différence suivante : l'arc de c'_1 à ℓ_0 devient négatif. Tout cycle passe par ℓ_0, et donc change de signe. Cela a pour conséquénce de transformer la propriété :

- f tel que $G^{\pm}_f = G$ a au moins un point fixe, et un second si φ est satisfaite

en :

- f tel que $G^{\pm}_f = G$ a au plus un point fixe, et aucun si φ est satisfaite

(rappelons que f est notre choix, c'est-à-dire que ses fonctions locales encodent une valuation de φ). Les démonstrations reprennent et adaptent les éléments du cas maximum, pour le cas minimum. □

7.5. Perspectives

7.5.1. Une question nouvelle

Les travaux présentés dans ce chapitre constituent, à notre connaissance, les premières études de la complexité algorithmique des problèmes prenant en entrée un graphe orienté signé G, et posant une question sur la dynamique des réseaux d'automates qui ont G pour graphe d'interaction [Bri+19 ; Bri+20]. Nous n'avons étudié que les points fixes, et avons déjà rencontré de nombreuses classes de complexité : les perspectives sont nombreuses. Nous souhaitons maintenant étudier les cycles limites (voir chapitre 5.1), différents modes de mises à jour (voir chapitre 8), et des alphabets plus généraux (définition en 3.3). Un pas intéressant pourrait être fait en direction des conditions expérimentales réelles (où la présence et l'absence d'un arc ne sont pas sûres à 100%) par l'ajout d'une notion d'information incertaine sur le graphe d'interaction $G^{\pm}_f$, par exemple : les réseaux d'automates f doivent vérifier $d(G, G^{\pm}_f) \leq \theta$, pour d une notion de distance entre graphes et $\theta \in \mathbb{R}_+$ un seuil.

La première direction que nous souhaitons ouvrir est celle de considérations analogues sur les points fixes, pour le graphe d'interaction dans le cas non signé. Les raisonnements présentés ci-avant reposent grandement sur les signes de l'entrée G, et la suppression des contraintes relatives aux signes ajoutera de larges degrés de liberté dans les réseaux f possibles.

Des restrictions sur l'entrée G sont également envisageables. Nous avons vu qu'avoir degré entrant maximum $\Delta(G)$ borné a une influence sur la complexité, et l'on pourrait considérer d'autres restrictions graphiques. Que se passe-t-il si le graphe n'a que des arcs positifs ? que des arcs négatifs ?

7.5.2. Extension de fonctions booléennes partielles

Nous avons abordé brièvement, dans la preuve d'appartenance à NP du théorème 7.4, la question de l'extension d'une information partielle sur un réseau booléen déterministe. Elle se ramène à l'extension des fonctions locales du réseau, c'est-à-dire à l'extension de fonctions booléennes partielles, dont parlent Crama et Hammer

dans [CH11, chapitre 12]. Étant donnés deux ensembles $V, F \subseteq \{0,1\}^n$ et $M \subseteq [n]$, il s'agit de décider s'il existe $f : \{0,1\}^n \to \{0,1\}$ telle que :

- $\forall x \in V : f(x) = 1$, et
- $\forall x \in F : f(x) = 0$, et
- la dépendance en tout $i \in M$ est monotone effective (positive), et
- la dépendance en tout $i \in [n] \setminus M$ est non monotone effective.

Cette question a suscité notre intérêt avec Adrien Richard (I3S, Nice Sophia Antipolis), car elle nous semble pertinente pour les applications à la régulation génétique, où le graphe d'interaction signé est typiquement bien approximé (conditions sur les dépendances en M et $[n] \setminus M$) mais la dynamique peu connue (fonction partielle donnée par V et F) [TK01 ; Le 15]. Nous avons une série de résultats.

Dans le cas général le problème d'extension, étant donnés V, F et M, est NP-complet. L'appartenance à NP n'est pas triviale car donner naïvement une fonction $f : \{0,1\}^n \to \{0,1\}$ peut être de taille doublement exponentielle, il s'agit donc de démontrer l'existence de certificats plus petits. Pour la NP-difficulté, on remarque que les dépendances monotones positives induisent que, pour tous $x \in V$ et $y \geq x$ (ordre partiel défini par $y_i \geq x_i$ pour tout $i \in [n]$), on doit avoir $f(y) = 1$, et symétriquement $f(y) = 0$ pour tous $x \in F$ et $y \leq x$. On note $V^+ = \{y \in \{0,1\}^n \mid \exists x \in V : y \geq x\}$ et $F^- = \{y \in \{0,1\}^n \mid \exists x \in F : y \leq x\}$. De façon abstraite, le problème d'extension sur les dépendances monotones peut ainsi être exprimé par des antichaînes (ensembles de configurations incomparables), et on démontre le résultat suivant. Étant données deux antichaînes $V, F \subseteq \{0,1\}^n$, décider si $V^+ \cup F^- \neq \{0,1\}^n$ est NP-complet. Il s'agit d'une réduction depuis **Monotone-SAT** où l'on comprend bien la structure des dépendances. Cependant, avec l'hypothèse supplémentaire $V^+ \cap F^- = \emptyset$ qui doit être vérifiée lorsque *toutes* les dépendances sont monotones, le problème est dans P : les éléments de F doivent être exactement les éléments minimaux de $\{0,1\}^n \setminus V$, que l'on peut énumérer par *backtracking* en temps polynomial en $|V|$. Ainsi, lorsque toutes les dépendances sont monotones ($M = [n]$ pour toutes les fonctions locales), on obtient un algorithme polynomial pour l'extension d'un réseau booléen déterministe partiel. Lorsque toutes les dépendances sont non monotones ($M = \emptyset$ pour toutes les fonctions locales), le problème est intuitivement plus simple car il suffit d'ajouter les dépendances manquantes, sans contrainte sur la monotonicité. On observe que si $|V| + |F| \leq 2^{n-1} - 2$, alors il existe toujours au moins une extension non monotone (il faut beaucoup d'images déjà données dans V et F pour empêcher d'ajouter les dépendances manquantes), et sinon un algorithme exponentiel en n sera polynomial en la taille de l'entrée. Pour le cas général qui est NP-complet, nous avons également des algorithmes de complexités temporelles $(|V| + |F|)^{\mathcal{O}(n)}$ et $n^{\mathcal{O}(|V|+|F|)}$.

7.5.3. Décider l'existence de cycles postifs et négatifs

Pour classifier **PFmax**-1 dans P, nous avons abordé le problème de décider si un graphe d'interaction possède un cycle positif, qui se trouve être un problème resté

ouvert quelques dizaines d'années. Nous proposons un rapide survol des questions de ce type, et une conjecture.

Avec la transformation qui remplace tous les arcs positifs par des chemins de longueur deux (opération *split*), on a des équivalences entre :

- cycle positif et pair,
- cycle négatif et impair.

Tout d'abord, imposer le passage par un arc rend le problème difficile.

Existence d'un cycle impair avec arc prescrit (**Cycle**-1-**arc**) *Entrée :* un graphe orienté $G = (V, A)$ et un arc $e \in A$. *Question :* est-ce que G contient un cycle de longueur impaire passant par e ?

Existence d'un cycle pair avec arc prescrit (**Cycle**-0-**arc**) *Entrée :* un graphe orienté $G = (V, A)$ et un arc $e \in A$. *Question :* est-ce que G contient un cycle de longueur paire passant par e ?

Théorème 7.9 ([Tho85])**.** **Cycle**-1-**arc** *et* **Cycle**-0-**arc** *sont* NP*-complets.*

Idée de démonstration. Réduction depuis le problème de trouver deux chemins disjoints de s_1 à t_1, et de s_2 à t_2, dans un graphe orienté G [FHW80]. On splite tous les arcs de G et on ajoute les deux arcs (t_2, s_1) et (t_1, s_2), pour obtenir une réduction vers **Cycle**-0-**arc** avec l'arc (t_2, s_1). On splite en plus l'arc (t_2, s_1) en (t_2, u) et (u, s_1) pour obtenir une réduction vers **Cycle**-1-**arc** avec l'arc (t_2, u). □

Voici les problèmes qui nous ont intéressés plus haut.

Existence d'un cycle impair (**Cycle**-1) *Entrée :* un graphe orienté $G = (V, A)$. *Question :* est-ce que G contient un cycle de longueur impaire ?

Existence d'un cycle pair (**Cycle**-0) *Entrée :* un graphe orienté $G = (V, A)$. *Question :* est-ce que G contient un cycle de longueur paire ?

Théorème 7.10. **Cycle**-1 *est* NL*-complet.*

Démonstration. Pour l'appartenance à NL, il suffit de remarquer que si l'on trouve un pseudo-cycle de longueur impair (avec possiblement des répétitions de sommets), alors il existe un cycle de longueur impaire (sans répétition de sommet).

Pour la NL-difficulté on réduit $\leq_{\mathrm{m}}^{\mathsf{L}}$ depuis le problème d'atteignabilité orienté (*ST-Connectivity* dans [Pap94]). Étant donnés $G = (V, A)$ et $s, t \in V$, on splite tous les arcs de G et on ajoute l'arc (t, s). L'unique façon d'avoir un cycle impair est de traverser (t, s), et donc d'être capable d'aller de s à t (pour que ce soit un cycle). □

Théorème 7.11 ([McC04 ; RST99])**.** **Cycle**-0 *est dans* P.

Nous terminons avec une conjecture qui, de façon surprenante, n'est pas évidente.

Conjecture 7.12. **Cycle**-0 *est* P*-complet.*

8. Complexité des mises à jour

Dans ce chapitre, nous nous intéressons à faire varier le mode de mise à jour. Il s'agit d'une thématique importante du domaine, avec de nombreux travaux (voir [Nou12; Sen12; Fat14] et leurs références). Les questions que l'on peut poser sont très larges, car ces considérations ajoutent une nouvelle dimension au modèle. Afin de faire partie de l'entrée d'un problème, on se restreindra aux modes de mises à jour déterministes périodiques qui ont une représentation finie explicite. Nous considérerons également uniquement des modes de mise à jour équilibrés, car le non équilibre (un ou plusieurs automates qui ne sont jamais mis à jour) se traduit simplement par la substitution de l'identité aux fonctions locales concernées.

8.1. Dynamiques bloc-séquentielles et cycles limites

On notera $\mathtt{BS}_n$ l'ensemble des modes de mise à jour bloc-séquentiels sur n automates. Il s'agit des partitions ordonnées sur n éléments.

Avant de présenter nos résultats et les travaux liés dans la littérature, remarquons que l'ajout d'un mode de mise à jour bloc-séquentiel en entrée d'un problème dans une classe incluant P, ne change intuitivement pas sa complexité. En effet, à partir des circuits de $f : \{0,1\}^n \to \{0,1\}^n$ et $\beta \in \mathtt{BS}_n$ on peut calculer des circuits pour $f^{[\beta]}$ en temps polynomial [Rob86].

Les premières questions que l'on pourrait poser concernant la dynamique portent sur les points fixes et les cycles limites, de façon analogue au chapitre 5.1. On peut remarquer une chose simple : les points fixes sont invariants par l'application de n'importe quel mode de mise à jour bloc-séquentiel (qui inclut le mode parallèle) [GM90].

Théorème 8.1. *Pour tous f, $\beta \in \mathtt{BS}_n$ et x, on a $f(x) = x \iff f^{[\beta]}(x) = x$.*

Cela n'est en revanche pas vrai pour la famille plus large des modes de mise à jour déterministes équilibrés, comme le montre l'exemple suivant.

Remarque 8.2. Soit le RAB f dont le graphe d'interaction est un cycle négatif de taille n, qui n'a aucun point fixe en parallèle. Soit le mode de mise à jour $\beta = ([n], \dots, [n])$ où le bloc $[n]$ est répété $2n$ fois, alors $f^{[\beta]}$ est l'identité avec 2^n points fixes. Un exemple est donné en figure 3.6 **(F)**.

Cela nous mène, dans la présente section, à la considération des cycles limites.

FIGURE 8.1. – Deux RAB et leurs graphes d'interaction respectifs (tous les arcs sont positifs). Gauche : pour $\beta = (\{1\},\{2\}) \in \mathtt{BS}_2$ on a $\mathfrak{C}^2_{f^{[\beta]}} = \emptyset$ alors que pour le mode parallèle on a $|\mathfrak{C}^2_{f^{[\beta^{\mathtt{par}}]}}| = 1$. Droite : pour $\beta' = (\{1\},\{2,3\}) \in \mathtt{BS}_3$ on a $|\mathfrak{C}^2_{f'^{[\beta']}}| = 1$ avec $\mathtt{001} \leftrightarrow \mathtt{110}$ alors que pour le mode parallèle on a $\mathfrak{C}^2_{f'^{[\beta^{\mathtt{par}}]}} = \emptyset$.

Étant donné un RAB en entrée, le théorème 5.8 (chapitre 5.1) nous dit qu'il est NP-complet de décider s'il possède un cycle limite de longueur k (fixé) pour la dynamique parallèle. L'ajout d'une seconde quantification existentielle sur les modes de mises à jour bloc-séquentiels n'augmente pas la complexité du problème : il s'agit toujours de deviner et vérifier des objets polynomiaux.

Bloc-séquentiel cycle limite de taille k (**BS-k-CL**)
Entrée : un RAB $f : \{0,1\}^n \to \{0,1\}^n$ (circuits des fonctions locales).
Question : est-ce que $\exists \beta \in \mathtt{BS}_n : \mathfrak{C}^k_{f^{[\beta]}} \neq \emptyset$?

Théorème 8.3 ([Bri+21 ; Góm15])**.** **BS-k-CL** *est* NP*-complet pour tout* $k \geq 2$.

Idée de démonstration. L'appartenance à NP est obtenue en remarquant que le calcul de $f^{[\beta]}$ est réalisé en temps polynomial à partir de f et β [Rob86]. La NP-difficulté est démontrée avec la même construction qu'au théorème 5.8 : l'implication directe est obtenue avec $\beta = ([n])$, et pour la réciproque on converge encore, pour tout $\beta \in \mathtt{BS}_n$, vers un unique point fixe lorsqu'une clause n'est pas satisfaite. □

Ce résultat est également prouvé dans la thèse de doctorat de Gómez [Góm15] pour les réseaux ET/OU, ce qui est le cas également pour notre construction.

En revanche, si l'on demande à éviter un cycle limite de longueur k (problème coNP-complet pour le mode de mise à jour parallèle), l'alternance de quantification $\exists\forall$ nous mène au niveau suivant dans la hiérarchie polynomiale.

Bloc-séquentiel sans cycle limite de taille k (**BS-$\neg k$-CL**)
Entrée : un RAB $f : \{0,1\}^n \to \{0,1\}^n$ (circuits des fonctions locales).
Question : est-ce que $\exists \beta \in \mathtt{BS}_n : \mathfrak{C}^k_{f^{[\beta]}} = \emptyset$?

Remarquons que les problèmes **BS-k-CL** et **BS-$\neg k$-CL** ne sont pas complémentaires l'un de l'autre : il existe des instances qui sont positives pour les deux (figure 8.1).

Pour les f dont le graphe d'interaction est un cycle (comme sur la figure 8.1), la dynamique en termes de nombre de cycles limites de taille k est entièrement caractérisée pour tout mode de mise à jour bloc-séquentiel dans [DNS12], sur la base de [GN10].

Théorème 8.4 ([Bri+21])**.** **BS-¬k-CL** *est* $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$*-complet pour tout* $k \geq 2$.

Idée de démonstration. Le cas $k = 2$ est traité avec une construction légèrement plus simple [Bri+21, théorème 3 et corollaire 2]. L'appartenance à $\mathsf{NP}^{\mathsf{NP}}$ est évidente, en remarquant que $\mathsf{NP}^{\mathsf{NP}} = \mathsf{NP}^{\mathsf{coNP}}$ car un oracle et son complémentaire sont identiquement utiles : deviner $\beta \in \mathtt{BS}_n$ et vérifier $\mathfrak{C}_f^k = \emptyset$ (qui est dans coNP par le théorème 5.8) grâce à l'oracle.

Pour la $\mathsf{NP}^{\mathsf{NP}}$ difficulté, on procède par réduction depuis **∃∀-3-SAT** : étant donnée une formule propositionnelle φ sur les variables [a] $\lambda_1, \dots, \lambda_n$ et un entier $s \in [n]$, on doit décider si $\exists \tilde{x} \in \{0,1\}^s : \forall \tilde{x}' \in \{0,1\}^{n-s} : \varphi(\tilde{x}\tilde{x}') = \top$, avec $\tilde{x}\tilde{x}' \in \{0,1\}^n$ la concaténation de $\tilde{x}$ et $\tilde{x}'$. Ce problème est $\mathsf{NP}^{\mathsf{NP}}$-complet [Pap94, théorème 17.10].

Pour les RAB, l'idée générale consiste à avoir une horloge à environ k automates avec un état 1 qui tourne sur un arrière-plan d'états 0 (qui donne un cycle limite de longueur k), et à arrêter cette horloge si et seulement le mode de mise à jour bloc-séquentiel encode un $\tilde{x} \in \{0,1\}^s$ satisfaisant la formule pour tout $\tilde{x}' \in \{0,1\}^{n-s}$. C'est donc bien le mode de mise à jour du problème **BS-¬k-CL** qui est quantifié existentiellement. La réduction n'est pas triviale, et est présentée de façon incrémentale dans [Bri+21]. Nous allons présenter ici directement les idées du cas général.

À partir de φ et s, on construit les circuits des fonctions locales du RAB f présenté sur la figure 8.2, qui comporte trois parties principales :

- à gauche, on reproduit notre construction canonique évaluant la formule φ pour la valuation encodée dans les états des automates $\lambda_1, \dots, \lambda_n$, avec une couche supérieure λ'_i pour $i \in [s]$,
- au centre, on a une horloge sur les $k+1$ automates h_i, où l'état 1 sur arrière-plan d'états 0 suivra l'ordre indiqué dans les états des automates b_i,
- à droite, les états des automates b_i devront encoder le mode de mise à jour bloc-séquentiel qui est appliqué à l'horloge (sur $\lceil \log |\mathtt{BS}_{k+1}| \rceil$ automates, qui est grand comme discuté au chapitre 8.2 mais k est fixé dans la définition du problème).

C'est la comparaison entre l'ordre des mises à jour des automates λ_i et λ'_i (pour $1 \leq i \leq s$) et l'automate h_0 qui encodera les variables existentielles de φ, avec :

$$f_{\lambda'_i}(x) = x_{h_0} \quad \text{et} \quad f_{\lambda_i}(x) = \begin{cases} x_{\lambda'_i} & \text{si } x_{h_0} = 1 \\ x_{\lambda_i} & \text{sinon} \end{cases}$$

a. Pour éviter les confusions entre états des automates du RAB et variables de la formule, nous nommons λ_i les variables, et aurons x_{λ_i} pour l'état dans la configuration x de l'automate correspondant à la variable λ_i.

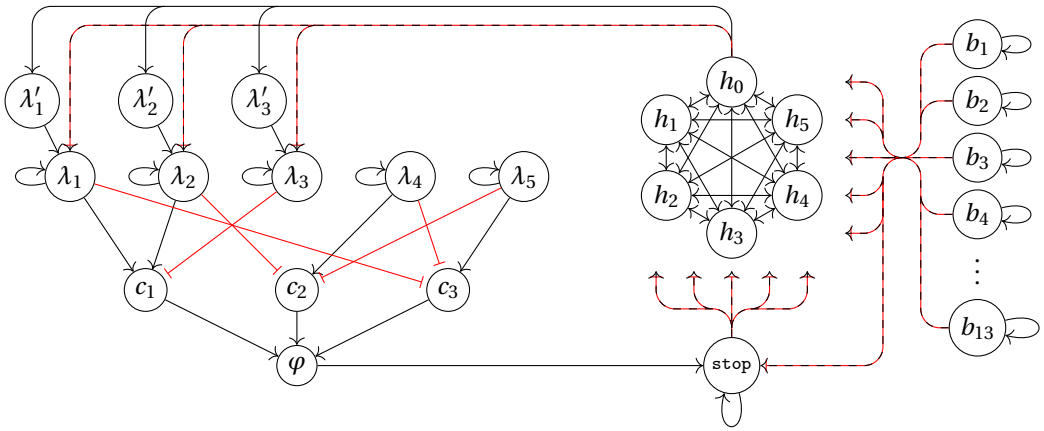


FIGURE 8.2. – Graphe d'interaction signé (arcs négatifs en rouge avec pointe plate, arcs non monotones tiretés) obtenu dans la réduction de $\exists\forall$**-3-SAT** à **BS-**$\neg k$**-CL**, pour $k = 5$ ($|\mathtt{BS}_6| = 4683$), la formule $\varphi = (\lambda_1 \vee \lambda_2 \vee \neg\lambda_3) \wedge (\neg\lambda_2 \vee \lambda_4 \vee \neg\lambda_5) \wedge (\neg\lambda_1 \vee \neg\lambda_4 \vee \lambda_5)$ et $s = 3$. Les automates `stop` et b_i sont reliés à tous les automates h_i par des arcs $\pm$.

(l'automate λ'_i va passer dans l'état `1` toutes les k étapes, et l'état de l'automate λ_i dépendra du moment où il copie l'état de λ'_i, tout cela contrôlé par l'automate h_0). Les boucles positives sur x_i pour $s < i \leq n$ encodent les variables universelles : les deux états fixes `0` et `1` sont possibles dans les configurations limites. De même pour les boucles positives sur b_i, qui encodent un mode de mise à jour bloc-séquentiel. Enfin, lorsque l'automate φ est dans l'état `1` on arrête l'horloge via l'automate `stop`.

Voici la logique de la démonstration : si φ est positive, alors il existe $\beta \in \mathtt{BS}$ qui met à jour les automates de l'horloge en parallèle, et encode dans l'ordre entre λ_i et λ'_i une valuation $\tilde{x} \in \{0,1\}^s$ satisfaisant le reste de la formule. Dans toute configuration limite, l'automate φ sera dans l'état `1` (quels que soient les états de b_i), ce qui arrêtera l'horloge et donnera uniquement des points fixes. Si φ est négative, alors pour tout $\beta \in \mathtt{BS}$ on peut choisir les états de b_i de façon à ce que l'horloge h_i ait période k, et quels que soient les états des variables existentielles (données par β) on peut choisir les états des variables universelles (grâce aux boucles positives) de façon à avoir l'état `0` dans l'automate φ : on obtient un cycle limite car l'horloge n'est pas stoppée.

Permettons-nous de souligner ici l'astuce d'ajouter les automates b_i permettant au réseau de « connaître » le mode de mise à jour (pour tout β on peut choisir les états des b_i qui lui correspondent), qui s'avère très utile d'un point de vue technique. □

Puisque k est fixé, la construction du théorème 8.4 a un graphe d'interaction de degré entrant borné. On peut également remarquer que la dynamique comporte soit un cycle limite de longueur k et des points fixes, soit uniquement des points fixes, donc le résultat peut être étendu à la question : étant donné un RAB f, est-ce qu'il ne possède pas de cycle limite de longueur au moins k? On peut également adapter la

construction (en remplaçant les points fixes par des cycles de longueur plus grande que k) pour traiter la question : étant donné un RAB f, est-ce qu'il ne possède pas de cycle limite de longueur au plus k ?

Nous souhaitons prolonger ces travaux en considérant des questions sur les cycles limites où la taille k du cycle, encodée en binaire, fait partie de l'entrée. Nous conjecturons qu'un saut de complexité est opéré, analogue au chapitre 7.3 (table 7.1).

Bloc-séquentiel cycle limite (**BS-CL**)
Entrée : un RAB $f : \{0,1\}^n \to \{0,1\}^n$ et un entier binaire $k \in \mathbb{N}_+$.
Question : est-ce que $\exists \beta \in \mathtt{BS}_n : \mathfrak{C}^k_{f^{[\beta]}} \neq \emptyset$?

Question ouverte 8.5. *Le problème* **BS-CL** *est dans* NEXPTIME. *Est-il* NEXPTIME*-difficile ?*

On trouve d'autres résultats de complexité très intéressants sur les modes de mise à jour bloc-séquentiels dans la littérature.

Il existe des questions faciles pour le cas parallèle, qui deviennent difficiles en rajoutant la quantification $\exists \beta \in \mathtt{BS}_n$: étant donné un RAB $f : \{0,1\}^n \to \{0,1\}^n$ et deux configurations $x, y \in \{0,1\}^n$, décider si $\exists \beta \in \mathtt{BS}_n : f^{[\beta]}(x) = y$ est NP-complet [Góm15].

Les auteurs de [AGS13] ont également étudié des questions impliquant plus d'un mode de mise à jour bloc-séquentiel. Étant donnés un RAB $f : \{0,1\}^n \to \{0,1\}^n$, un $\beta \in \mathtt{BS}_n$ et un cycle limite $C \in \mathfrak{C}^k_{f^{[\beta]}}$, décider si $\exists \beta' \in \mathtt{BS}_n$ tel que $\beta' \neq \beta$ et $C \in \mathfrak{C}^k_{f^{[\beta']}}$ (c'est-à-dire β' a également le cycle limite C) est NP-complet. Le partage des cycles limites est leur sujet d'étude, et plusieurs variantes sont considérées, comme par exemple décider si les ensembles de cycles limites pour deux $\beta, \beta' \in \mathtt{BS}$ sont égaux, et si les ensembles de cycles limites pour deux $\beta, \beta' \in \mathtt{BS}$ partagent au moins un élément. Ces deux variantes sont également NP-complètes.

Le problème suivant est lui aussi, de façon assez surprenante au premier regard, NP-complet [Ara+13a] : étant donné un RAB $f : \{0,1\}^n \to \{0,1\}^n$, décider si $\exists \beta, \beta' \in \mathtt{BS}_n : f^{[\beta]} \neq f^{[\beta']}$ (c'est-à-dire s'il existe deux modes de mise à jour bloc-séquentiels qui diffèrent dans l'image d'au moins une configuration). La preuve repose en revanche sur une construction assez simple, similaire à notre démonstration du théorème 5.1 sur les points fixes.

Il est démontré dans [Góm15] que, étant donné un RAB $f : \{0,1\}^n \to \{0,1\}^n$, décider s'il existe un $\beta \in \mathtt{BS}_n$ tel que $f^{[\beta]}$ n'a que des points fixes, est NP-difficile. On peut alors observer que notre construction du théorème 8.4 montre que ce même problème est $\mathsf{NP}^{\mathsf{NP}}$-difficile, comme nous l'avons déjà remarqué plus haut. Formalisons cela.

Bloc-séquentiel sans cycle limite de taille $k \geq 2$ (**BS-¬CL**)
Entrée : un RAB $f : \{0,1\}^n \to \{0,1\}^n$ (circuits des fonctions locales).
Question : est-ce que $\exists \beta \in \mathtt{BS}_n : \forall k \geq 2 : \mathfrak{C}^k_{f^{[\beta]}} = \emptyset$?

Corollaire 8.6. **BS-¬CL** *est* $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$*-difficile.*

Démonstration. La $\mathsf{NP}^{\mathsf{NP}}$-difficulté est obtenue avec notre construction du théorème 8.4 pour $k = 3$ (le cas $k = 2$ utilise une construction différente que nous n'avons pas présentée), car elle a soit un cycle limite de taille k et des points fixes, soit uniquement des points fixes. □

Tester l'absence de cycle limite de toute taille $k \geq 2$ n'est pas évident : naïvement il faut essayer, sur toutes les configurations x et pour tous les k de 2 à 2^n, de calculer $f^k(x)$, ce qui prend à chaque fois un temps quadratique en k.

Question ouverte 8.7. *Où se situe précisément* **BS-¬CL**, *entre* Σ_2^{P} *et* PSPACE *?*

8.2. Comptages bloc-séquentiels

Comme nous l'avons répété, $\mathtt{BS}_n$ est l'ensemble des partitions ordonnées d'un ensemble à n éléments. Elles sont comptées par les *nombres de Bell ordonnés* (séquence `A000670` [A00a]) :

$$|\mathtt{BS}_n| = \sum_{i=0}^{n-1} i! \left\{ {n-1 \atop i} \right\} = \sum_{i=0}^{n-1} \sum_{j=0}^{i} (-1)^{i-j} \binom{i}{j} j^{n-1}$$

en utilisant les nombres de Stirling de seconde espèce (notés $\{\}$) qui comptent le nombre d'applications surjectives d'un ensemble à $n-1$ éléments vers un ensemble à i éléments [NS11]. Une formulation récursive est donnée par

$$|\mathtt{BS}_n| = \sum_{i=0}^{n-1} \binom{n}{i} |\mathtt{BS}_i| \quad \text{avec} \quad |\mathtt{BS}_1| = 1$$

(intuitivement, on choisit entre 0 et $n-1$ automates parmi n, que l'on ordonne récursivement, puis on place les automates restants dans un nouveau dernier bloc). Dans cette section nous allons en compter des sous-ensembles particuliers.

Dans une série de travaux sur la sensibilité au synchronisme des automates cellulaires élémentaires (*elementary cellular automata*, ECA) finis [Mon+17 ; Rui+18 ; Bal+20 ; Per+20 ; Rui+20], nous avons compté le nombre d'éléments de $\mathtt{BS}_n$ pour des graphes d'interaction particuliers. En effet, les ECA de taille n avec condition aux bords périodique peuvent être vus comme des réseaux d'automates booléens sur les automates $[\![n]\!]$ (dimension 1) où toutes les fonctions locales sont les mêmes (il s'agit de la *règle locale* de l'ECA), avec le graphe d'interaction comportant les arcs de la forme $(i-1, i)$, (i, i) et $(i+1, i)$ (il s'agit du voisinage de rayon 1). Ces études ont motivé la question plus générale du comptage du nombre de modes de mise à jour bloc-séquentiels, à partir de la relation d'équivalence introduite par Aracena et al. dans [Ara+09 ; Ara+11].

$f_1(x) = 0$ $\quad$ $f_3(x) = x_1 \vee \neg x_4$
$f_2(x) = \neg x_4$ $\quad$ $f_4(x) = x_3$

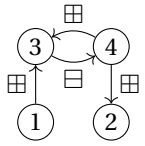


FIGURE 8.3. – Exemple d'un RAB f et le graphe des mises à jour $G_f^{\beta} = G_f^{\beta'}$ pour $\beta = (\{1,2,3\},\{4\})$ et $\beta' = (\{1,3\},\{2,4\})$.

8.2.1. Graphe des mises à jour et relation d'équivalence

On a l'intuition à partir de quelques exemples que, pour un RAB $f : \{0,1\}^n \to \{0,1\}^n$ et un $\beta \in \mathtt{BS}_n$ donnés, l'ordre relatif de mise à jour dans β entre deux automates qui n'ont aucun arc entre eux dans G_f, n'a aucune influence sur la dynamique de $f^{[\beta]}$. Cette idée est formalisée par la notion de graphe des mises à jour, qui est un étiquetage du graphe d'interaction.

Graphe des mises à jour. Le *graphe des mises à jour* $G_f^{\beta} = ([n], A^{\beta})$ d'un RAB $f : \{0,1\}^n \to \{0,1\}^n$ pour un mode de mise à jour bloc-séquentiel $\beta = (B_1, \dots, B_p) \in \mathtt{BS}_n$ est défini avec $A^{\beta} \subseteq [n] \times \{\boxplus, \boxminus\} \times [n]$ par :

$$(i, \boxplus, j) \in A^{\beta} \iff (i,j) \in G_f \text{ et } t_i \geq t_j, \text{ avec } i \in B_{t_i}, j \in B_{t_j}$$
$$(i, \boxminus, j) \in A^{\beta} \iff (i,j) \in G_f \text{ et } t_i < t_j, \text{ avec } i \in B_{t_i}, j \in B_{t_j}$$

où $(i,j) \in G_f$ est un abus de notation pour désigner les arcs du graphe d'interaction (voir figure 8.3). Remarquons que les boucles sont toujours étiquetées $\boxplus$, nous pouvons les ignorer. On peut exprimer le résultat intuitivement annoncé, comme une relation d'équivalence entre modes bloc-séquentiels pour un graphe donné.

Théorème 8.8 ([Ara+09])**.** *Pour tout $f : \{0,1\}^n \to \{0,1\}^n$, soit la relation d'équivalence $\beta \equiv_{G_f} \beta' \iff G_f^{\beta} = G_f^{\beta'}$. Alors $\beta \equiv_{G_f} \beta' \implies f^{[\beta]} = f^{[\beta']}$ (*i.e.* $\mathscr{G}_{f^{[\beta]}} = \mathscr{G}_{f^{[\beta']}}$).*

Soit $G_f = ([n], A)$, il est très important pour notre propos de remarquer que ses $2^{|A|}$ étiquetages par $\{\boxplus, \boxminus\}$ ne correspondent pas tous à un graphe des mises à jour G_f^{β} avec $\beta \in \mathtt{BS}_n$. Par exemple, un cycle dont tous les arcs sont étiquetés $\boxminus$ est contradictoire : en parcourant le cycle on en déduirait que $t_i < t_i$. On parlera d'étiquetages *valides* et *invalides*, ils sont caractérisés par le résultat suivant.

Théorème 8.9 ([Ara+11])**.** *Un graphe orienté étiqueté $\{\boxplus, \boxminus\}$ est valide si et seulement si le multigraphe où les arcs étiquetés $\boxminus$ sont renversés ne comporte aucun cycle comportant un arc étiqueté $\boxminus$ (que nous appellerons* cycle interdit*).*

Intuitivement, dans un graphe des mises à jour G_f^{β}, suivre un arc étiqueté $\boxplus$ donne $\geq$, et suivre à l'envers un arc étiqueté $\boxminus$ donne $>$. On obtient alors une contradiction lorsque qu'on retombe sur un automate en ayant parcouru au moins une inégalité stricte : c'est exactement à cela que correspond un cycle interdit.

8.2.2. Complexité des problèmes liés à G_f^β

Nous sommes intéressés par le problème de comptage du nombre de modes de mises à jour bloc-séquentiels non équivalents, étant donné un graphe d'interaction non signé. Remarquons que la mention à un f devient inutile.

Comptage des graphes des mises à jour bloc-séquentiels (**#GBS**) *Entrée :* un graphe orienté $G = ([n], A)$. *Sortie :* $\#\mathtt{GBS}(G) = \lvert \mathtt{BS}_n / \equiv_G \rvert = \left\lvert \left\{ G^\beta \;\middle\vert\; \beta \in \mathtt{BS}_n \right\} \right\rvert$.

On peut se restreindre aux composantes fortement connexes car, en notant $V_1, \dots, V_k$ les composantes fortement connexes de G et A^* les arcs entre composantes fortement connexes, on a la relation suivante :

$$\#\mathtt{GBS}(G) = 2^{|A^*|} \cdot \prod_{i \in [k]} \#\mathtt{GBS}(G[V_i])$$

(les comptages sur les composantes fortement connexes sont indépendants).

En corollaire de la caractérisation des étiquetages valides du théorème 8.9, le problème de décider si un étiquetage est valide appartient à P [Ara+11]. Nous pouvons classifier **#GBS**.

Théorème 8.10 ([PSA16 ; Noû+20])**.** **#GBS** *est* $\#\mathsf{P}$*-complet pour les réductions* $\leq_T^{\mathsf{P}}$.

Démonstration. L'appartenance à $\#\mathsf{P}$ est évidente par ce qui précède : l'algorithme non déterministe qui devine un étiquetage de G et accepte si et seulement s'il est valide, fonctionne en temps polynomial et a exactement une branche acceptante pour chaque élément de $\#\mathtt{GBS}(G)$.

On présente une réduction parcimonieuse (qui préserve le nombre de solutions) depuis le problème de compter le nombre d'orientations acycliques d'un graphe non orienté G, qui est $\#\mathsf{P}$-complet pour les réductions Turing $\leq_{\mathrm{T}}^{\mathsf{P}}$ [Lin86]. Étant donné un graphe non orienté $G = (V, E)$, soit $\prec$ un ordre total arbitraire sur V. On construit le graphe orienté $G' = (V, A)$ avec A l'orientation de E selon $\prec$:

$$(i, j) \in A \iff \{i, j\} \in E \text{ et } i \prec j$$

(voir figure 8.4). Remarquons que G' est acyclique, par définition.

Pour obtenir une bijection entre les étiquetages $\{\boxplus, \boxminus\}$ valides de G' et les orientations acycliques de G, il suffit de renverser l'orientation dans G' des arcs avec l'étiquette $\boxminus$ (voir figure 8.5). Le théorème 8.9 permet alors de conclure, car tout cycle dans le graphe ainsi obtenu doit comporter au moins un arc renversé (avec étiquette $\boxminus$) puisque G' est acyclique. □

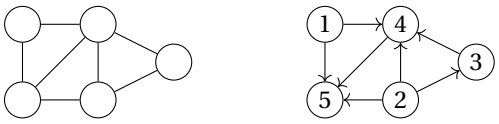


FIGURE 8.4. – Graphe non orienté G (instance du comptage des orientations acycliques), et le graphe orienté G' obtenu (instance du comptage des graphes des mises à jour).

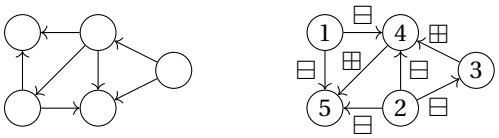


FIGURE 8.5. – Orientation acyclique de G, et étiquetage valide de G' correspondant.

Des bornes graphiques de la littérature permettent de dériver simplement des bornes de complexité pour des variantes de **#GBS**. Les auteurs de [Ara+11] ont mis en avant une relation forte entre les étiquetages valides et les *feedback arc sets* (FAS) d'un graphe orienté. Cette relation est développée dans la preuve de NP-complétude du problème de décision suivant : étant donné un graphe orienté G et un entier $k \in \mathbb{N}$, est-ce qu'il existe un étiquetage valide de G avec au plus k étiquettes ⊞ ? Intuitivement, placer des étiquettes ⊟ est plus difficile que de placer des étiquettes ⊞ (car les étiquettes ⊟ sont requises pour avoir un cycle interdit, en particulier étiqueter tous les arcs avec ⊞ est toujours valide et correspond au mode parallèle), ce qui explique la formulation de ce problème. Nous reproduisons ce lien comme un lemme, car il nous semble intéressant en soi. La *taille d'un étiquetage* est son nombre d'arcs ⊞.

Lemme 8.11 ([Ara+11, dans la preuve du théorème 16])**.** *Pour tout graphe, il existe une bijection entre les étiquetages valides minimaux et les FAS minimaux.*

Idée de démonstration. La bijection consiste simplement à identifier un étiquetage à son ensemble d'arcs étiquetés ⊞. La minimalité est utilisée pour passer d'un FAS minimal F à un étiquetage ⊞ des arcs de F et ⊟ des autres arcs : pour tout arc de F, il existe un cycle dont c'est le seul arc dans F, d'où l'on déduit un moyen de remplacer des arcs étiquetés ⊞ (dans F) par des chemins étiquetés ⊟ (avec le cycle dont c'était le seul arc dans F). Ainsi, si l'étiquetage comporte un cycle interdit, on peut former un cycle comportant uniquement des arcs étiquetés ⊟, ce qui contredit le fait que F est un FAS (car les étiquettes ⊟ sont assignés aux arcs qui ne sont pas dans F). □

Puisque tout étiquetage valide correspond à un FAS (par définition des cycles interdits), et que tout FAS minimal correspond à un étiquetage valide comme nous venons de l'énoncer dans le lemme 8.11 (la réciproque de cette dernière assertion est en revanche fausse, d'où notre construction n'utilisant pas trivialement de FAS dans la réduction parcimonieuse du théorème 8.10), on obtient les bornes suivantes. La

borne inférieure stricte provient du fait que l'étiquetage de tous les arcs avec ⊞ est toujours valide, mais jamais minimal.

Théorème 8.12 ([Ara+11])**.** *Pour tout graphe orienté G on a $\#mFAS(G) < \#GBS(G) \leq \#FAS(G)$, avec $\#FAS(G)$ (*resp. *$\#mFAS(G)$) le nombre de FAS (*resp. *FAS minimaux) de G.*

À partir des résultats sur le comptage des FAS présentés au théorème 2.2 (chapitre préliminaire 2), on obtient les bornes de complexité suivantes. Puisque les FAS minimums sont minimaux, l'identité est une réduction parcimonieuse depuis les problèmes de FAS vers les problèmes de validité des étiquetages correspondants.

Théorème 8.13 ([Noû+20])**.** *Compter le nombre d'étiquetages valides de taille minimale est $\#\mathsf{P}$-complet pour $\leq_T^{\mathsf{P}}$, et compter le nombre d'étiquetages valides de taille minimum est $\# \cdot \mathsf{OptP}[\log n]$-complet pour $\leq_{parci}^{\mathsf{P}}$.*

8.2.3. Deux cas polynomiaux : cactus et série-parallèle

Cactus. La difficulté de compter le nombre de graphes de mises à jour (étiquetages valides du graphe d'interaction) tient sa source dans les interconnexions entre les possibles cycles, comme l'atteste notre réduction parcimonieuse depuis le problème de compter les orientations acycliques. Le comptage pour l'orientation d'un arbre à n sommets est triviale : ses 2^{n-1} étiquetages sont valides. Les cactus sont des graphes non orientés définis par une restriction sur les interconnexions entre les cycles, qui peut être exploitée pour répondre efficacement à **#GBS**.

Pour un graphe orienté $G = (V, A)$, notons $\bar{G} = (V, E)$ le multigraphe non orienté sous-jacent, qui possède une arête $\{i, j\} \in E$ pour tout arc $(i, j) \in A$. Un *cactus* est un graphe connexe non orienté tel que chaque sommet (ou de façon équivalente chaque arête) appartienne à au plus un cycle simple (cycle sans répétition de sommet). Un *cactus orienté* est un graphe orienté G tel que $\bar{G}$ est un cactus. Intuitivement, les cactus sont des graphes dans lesquels certains sommets sont remplacés par des cycles. En utilisant le *squelette* d'un cactus qui capture cette décomposition en cycles (voir figure 8.6), on obtient le résultat suivant.

Théorème 8.14 ([Noû+20])**.** **#GSB** *est calculable en temps $\mathcal{O}(n^2 \log n \log\log n)$ pour les cactus orientés.*

Idée de démonstration. À partir du squelette d'un cactus illustré en figure 8.6, on peut décomposer le calcul en sous-problèmes de comptage indépendants, et obtenir le résultat grâce à la formule :

$$\#\mathrm{GBS}(G) = \prod_{H \in A^g} 2^{|H|} \prod_{H \in A^c} (2^{|H|} - 1) \prod_{H \in A^h} (2^{|H|} - 2),$$

où A^g, A^c, A^h partitionnent les arcs de G en sous-ensembles correspondant à chaque type. Un groupe H de type arborescent aura tous ses $2^{|H|}$ étiquetages valides, pour

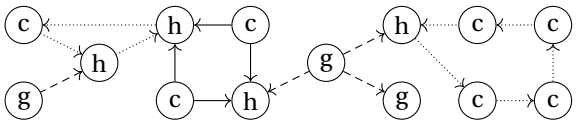


FIGURE 8.6. – Un cactus orienté G et son squelette décomposé en groupes de sommets et arcs de type c, g et h. Les sommets de type g appartiennent à des parties arborescentes, de type c à des cycles orientés, et de type h à des cycles ne formant pas de cycle orienté. Sur cet exemple le théorème 8.14 compte $\#\mathtt{GBS}(G) = 2^1 2^3 (2^3-1)(2^5-1)(2^4-2) = 48\,608$.

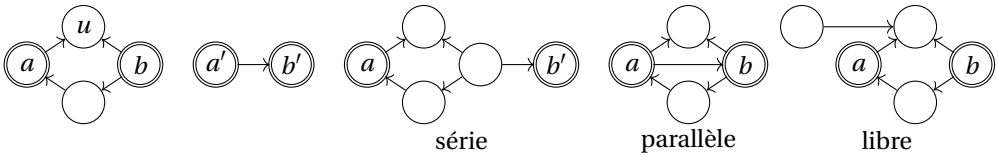


FIGURE 8.7. – Deux graphes série-parallèles, et leur composition série, parallèle, et libre en u, b.

le type cycle orienté il faut retirer l'unique étiquetage interdit avec $\boxminus$ sur tous les arcs, et pour le type cycle qui n'est pas un cycle orienté on peut créer deux cycles interdits (un pour chaque orientation du cycle). Le squelette peut être calculé en temps linéaire [BK98], et le produit de nombre entiers binaires donne la borne énoncées avec l'algorithme de Schönhage-Strassen [SS71]. □

Remarquons que l'on pourrait utiliser l'algorithme de multiplication de Fürer en $\mathcal{O}(n \log n 2^{2\log^* n})$ pour améliorer la borne [Für07], ou peut-être l'algorithme conjecturé asymptotiquement optimal déposé sur HAL par Harvey et Van Der Hoeven en 2019 [HV19], en $\mathcal{O}(n \log n)$.

Série-parallèle. Les graphes non orientés $G = (V, E)$ *série-parallèles* sont définis par induction comme (G, a, b) avec deux sommets $a \neq b \in V$ pointés, respectivement *source* et *puits*,. Partant du graphe de base possédant uniquement deux sommets a, b et une arête $\{a, b\}$, trois opérations de recollement sont autorisées [Duf65 ; VTL79] (voir figure 8.7) :

- la composition *série* de (G, a, b) et (G', a', b') identifie les sommets b, a',
- la composition *parallèle* de (G, a, b) et (G', a', b') identifie a, a' et b, b'.
- la composition *libre* en v, v' de (G, a, b) et (G', a', b'), qui identifie le sommet v de G avec le sommet v' de G'.

Un graphe orienté G tel que $\bar{G}$ est série-parallèle est appelé *série-parallèle orienté*. Comme dans le cas des cactus, $\bar{G}$ doit donc être sans multiarête. On peut également définir un série-parallèle orienté comme l'orientation d'un série-parallèle.

Grâce à la décomposition structurelle à partir de laquelle ils sont définis, on peut calculer efficacement le nombre de graphes des mises à jour possibles sur un graphe

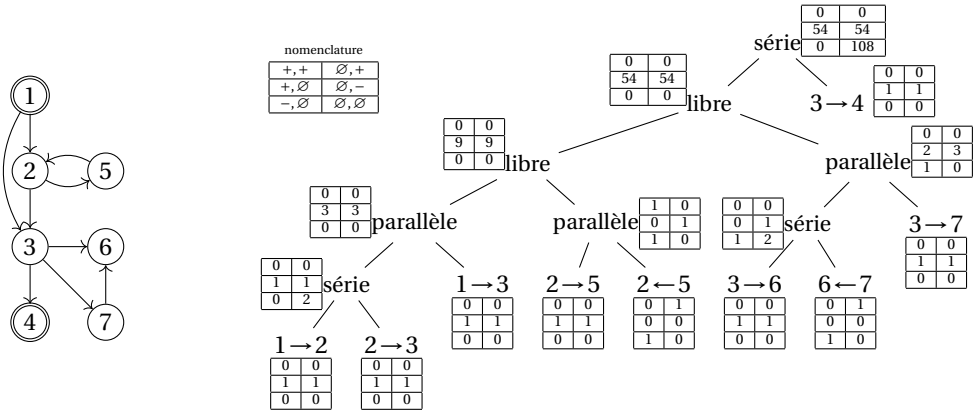


FIGURE 8.8. – Un graphe série-parallèle orienté, et l'application de la méthode diviser pour régner comptant le nombre d'étiquetages valides, $\#\mathtt{GBS}(G) = 216$.

série-parallèle orienté donné.

Théorème 8.15 ([Noû+20])**.** **#GSB** *est calculable en temps* $\mathcal{O}(n^2(\log n)^2 \log\log n)$ *pour les graphes série-parallèle orientés (sans promesse).*

Idée de démonstration. Tout d'abord, il n'est pas nécessaire d'avoir la promesse que le graphe est bien série-parallèle, car cette propriété peut-être vérifiée, et la décomposition calculée, en temps linéaire [VTL79].

On partitionne les étiquetages valides d'un graphe série-parallèle orienté (G, a, b), suivant l'existence de chemin, et de chemin négatif (qui comporte une étiquette $\boxminus$), de a vers b, et de b vers a. Il y a trois possibilités pour a vers b, et trois possibilités pour b vers a :

1. les étiquetages pour lesquels il n'y a pas de chemin,
2. les étiquetages pour lesquels il existe un chemin et aucun chemin n'est négatif,
3. les étiquetages pour lesquels il existe un chemin négatif,

qui donnent six parties (car trois des neufs possibilités ont des cycles interdits, par exemple s'il existe un chemin positif de a vers b et un chemin négatif de b vers a). La structure des compositions série et parallèle nous permet alors de calculer récursivement le nombre d'étiquetages valides dans chacune de ces six parties, en utilisant le même genre de raisonnement élémentaire (et en multipliant les possibilités qui ne créent pas de cycle interdit). On étudie exhaustivement toutes les combinaisons possibles (36 pour chaque composition). Remarquons que la composition série de deux étiquetages valides ne créée pas de cycle interdit, alors que cela est possible pour la composition parallèle. On inclut également les compositions libres (qui ne créent jamais de cycles interdits), et le cas de base est donné par les graphes à deux sommets et une arête. Une fois la racine de l'arbre de décomposition atteinte, on somme les nombres d'étiquetages valides dans chaque partie pour obtenir $\#\mathtt{GBS}(G)$. Un exemple est proposé en figure 8.8.

Bien que la valeur de $\#\mathtt{GBS}(G)$ puisse être exponentielle en la taille du graphe, toutes les quantités sont écrites en binaire et leur valeur est bornée par le nombre d'étiquetages 2^m (valides et invalides) pour m arcs. Le calcul des quantités pour le cas de base est trivial, et l'on applique $\mathcal{O}(n \log n)$ compositions, chacune en temps $\mathcal{O}(n \log n \log\log n)$ (nous passons les détails). □

On peut ici aussi améliorer la complexité avec un meilleur algorithme de multiplication d'entiers.

8.3. Perspectives

Les graphes série-parallèle sont clos par mineur [RS04], ils peuvent être définis comme les graphes sans mineur K_4 (la clique à 4 sommets) [Duf65]. Il s'agit également de la famille des graphes de largeur arborescente au plus 2, ce qui est très lié à leur décomposition. Se pourrait-il que le problème soit FPT (*fixed parameter tractable*) pour le paramètre largeur arborescente ? Il s'agirait d'étudier comment une décomposition arborescente de G peut nous aider à calculer $\#\mathtt{GBS}(G)$.

On connaît des formules closes pour certaines familles de graphes orientés (un graphe orienté est dit *symétrique* lorsque sa matrice d'adjacences l'est, c'est-à-dire lorsque (i, j) est un arc si et seulement si (j, i) est également un arc) :

- $\#\mathtt{GBS}(G) = |\mathtt{BS}_n|$ (sequence `A000670` [A00a]) pour le graphe symétrique complet (sans boucle) [NS11],
- $\#\mathtt{GBS}(G) = n!$ si et seulement si G est un tournoi à n sommets [Ara+13b],
- $\#\mathtt{GBS}(G) = 3^n - 2^{n+1} + 2$ pour les cycles symétriques à n sommets, qui correspondent au graphe d'interaction des automates cellulaires élémentaires finis périodiques (de rayon 1) [Per+20].

De la réduction parcimonieuse du théorème 8.10, on peut également dériver :

- $\#\mathtt{GBS}(G) = \mathcal{T}_{\bar{G}}(2,0)$ lorsque G est acyclique, avec $\mathcal{T}_{\bar{G}}$ le polynôme de Tutte sur $\bar{G}$.

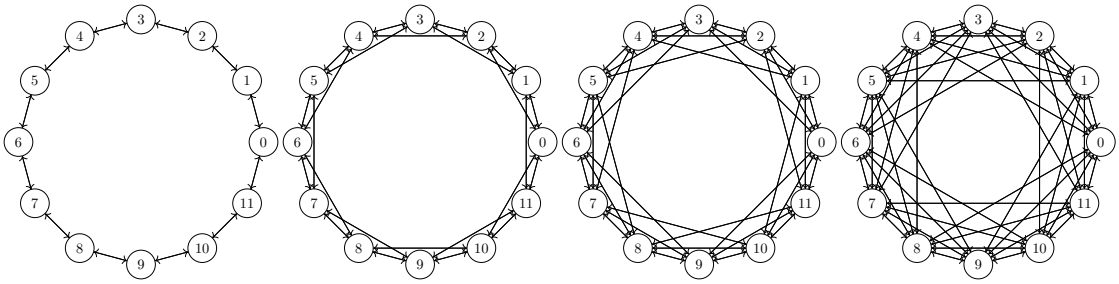
Pour quelles autres familles de graphes orientés peut-on obtenir une formule close ? La combinatoire ainsi étudiée pourrait inspirer des algorithmes plus généraux. Voici une famille qui nous intéresse.

Question ouverte 8.16. *Combien vaut $\#\mathtt{GBS}(G)$ pour le graphe d'interaction des automates cellulaires finis périodiques en une dimension, à n automates et voisinage de rayon r ? Voir figure 8.9.*

Plus naïvement, on peut poser la question suivante sur l'ensemble des entiers naturels qui peuvent être le résultat d'un comptage.

Question ouverte 8.17. *Est-ce que pour tout $n \in \mathbb{N}_+$ il existe un graphe G tel que $\#\mathtt{GBS}(G) = n$? Sinon, que peut-on dire de $\#\mathtt{GBS} = \{\#\mathtt{GBS}(G) \mid G \text{ est un graphe orienté}\}$?*

Remarquons qu'il suffit d'engendrer les nombres premiers pour engendrer tout $\mathbb{N}_+$, car on a $\#\mathtt{GBS}(G \sqcup G') = \#\mathtt{GBS}(G) \cdot \#\mathtt{GBS}(G')$. Les valeurs possibles de $\#\mathtt{GBS}(G)$ pour tous les graphes à $n = 1, 2, 3, 4, 5$ sommets sont présentées en figure 8.10.



$n=$	2	3	4	5	6	7	8	9	10
$r=1$	3	13	51	181	603	1933	6051	18661	57003
$r=2$	3	13	75	541	3147	18509	98443	?	?
$r=3$	3	13	75	541	4683	47293	?	?	?
$r=+\infty$	3	13	75	541	4683	47293	545835	7087261	102247563

FIGURE 8.9. – Graphe d'interaction (sans boucle) pour les automates cellulaires élémentaires finis périodiques de rayon $r = 1,2,3,4$ et $n = 12$ automates, et quelques comptages de $\#\mathtt{GBS}(G)$ réalisés par force brut. Pour $n \leq 2r+1$ on obtient le graphe complet (cases grisées). On remarque que le nombre d'étiquetages $\{\boxplus, \boxminus\}$ grossit beaucoup trop vite pour mon algorithme qui teste naïvement l'ensemble des sous-ensemble d'arcs (script `python3` utilisant le package `multiprocessing`, sans élimination des symétries, lancé sur une machine à 40 processeurs; comptage pour rayon 2 à 8 sommets réalisé en environ 2 heures).

$\in \#\mathtt{GBS}$	$\notin \#\mathtt{GBS}$
1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 18, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100,	5, 10, 17, 19, 25, 34, 50, 85,

FIGURE 8.10. – Énumération naïve des valeurs de $\#\mathtt{GBS}(G)$ obtenues pour tous les graphes G à $n = 1,2,3,4,5$ sommets (adaptation du script `python3` utilisé en figure 8.9, lancé sans multiprocessing, résultat obtenu en environ 13 heures). À gauche les entiers entre 1 et 100 pour lesquels il existe un graphe G à $n \leq 5$ sommets tel que $\#\mathtt{GBS}(G) = n$, à droite les autres (la valeur maximale obtenue étant $|\mathtt{BS}_5| = 541$, pour le graphe complet à 5 sommets).

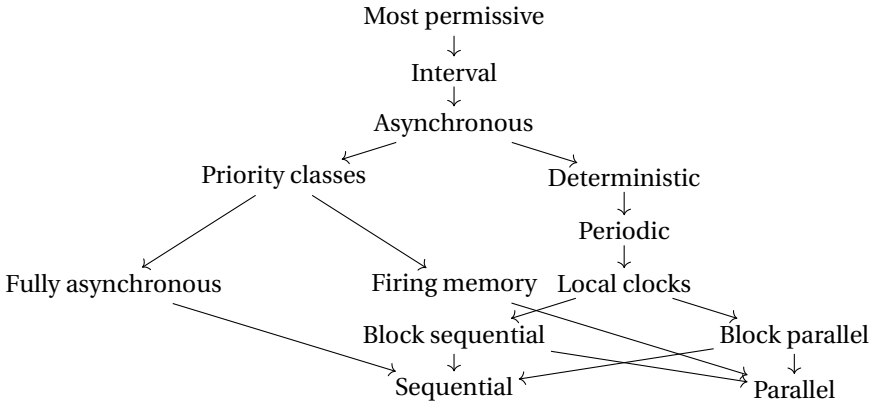


FIGURE 8.11. – Modes de mise à jour (noms en anglais) et relation de simulation entre eux. Diagramme de Loïc Paulevé et Sylvain Sené, reproduit avec leur autorisation (et leur code `TikZ`).

Nous nous intéressons, avec Damien Regnault (IBISC, Evry), Sylvain Sené (LIS, Marseille) et Lucas Venturini (ÉNS Lyon), aux modes de mise à jour déterministes périodiques, qui autorisent plusieurs mises à jour d'un même automate au cours d'une itération. Le champ est très vaste et peu exploré, nous souhaitons comprendre les liens structurels entre le graphe d'interaction des fonctions locales G_f, le mode β, et le graphe d'interaction obtenu par l'application de β sur f, c'est-à-dire $G_{f^{[\beta]}}$. Dans le cas bloc-séquentiel sur les réseaux conjonctifs (dont le graphe d'interaction G définit un unique réseau d'automates $f^{\wedge}_G$), Lilian Salinas a présenté à WAN'2021 des travaux en cours sur la complexité (NP-complétude) de décider :

- étant donnés un graphe G' à n sommets et un mode de mise à jour bloc-séquentiel $\beta \in \mathtt{BS}_n$, s'il existe un graphe d'interaction G tel que $G_{f^{\wedge}_G{}^{[\beta]}} = G'$,
- étant donnés deux graphes G, G' à n sommets, s'il existe un mode de mise à jour bloc-séquentiel $\beta \in \mathtt{BS}_n$ tel que $G_{f^{\wedge}_G{}^{[\beta]}} = G'$.

Plus largement, il y a un riche zoo de modes de mise à jour, de « *most permissive* » à parallèle et séquentiel. Loïc Paulevé et Sylvain Sené en ont donné une illustration à AUTOMATA'2021, reproduite en figure 8.11 [PS21 ; PS]. Les arcs donnent une relation de simulation très forte (même nombre d'automates, avec les mêmes fonctions locales), définie comme : un mode de mise à jour M' en simule un autre M lorsque :

$$\forall \beta \in M : \quad \exists \beta' \in M' : \quad \forall n \in \mathbb{N}_+, f : \{0,1\}^n \to \{0,1\}^n, x \in \{0,1\}^n, y \in \{0,1\}^n, t \in \mathbb{N} :$$
$$\exists t' \in \mathbb{N} : \quad y \in f^{[\beta]^t}(x) \implies y \in f^{[\beta']^{t'}}(x).$$

Autrement dit, pour tout mode de mise à jour β dans M on doit pouvoir donner un mode de mise à jour β' dans M', tel pour tout réseau d'automates f, la dynamique $f^{[\beta']}$ contient la relation d'atteignabilité de $f^{[\beta]}$ (les premières quantifications sont $\forall \beta \in M : \exists \beta' \in M'$, la quantification universelle sur f se trouve après).

9. Conclusion

En prenant du recul sur les objets considérés, commençons par formuler trois observations naïves concernent peut-être autant la théorie de la complexité algorithmique que les réseaux d'automates. Premièrement, la caractérisation précise (par un résultat de complétude) de certains problèmes qui semblent à première vue anodins peut nous faire découvrir des classes modérément connues (US, DP, et $\mathsf{NP}^{\#\mathsf{P}}$), voire exotiques ou inconnues ($\#\cdot\mathsf{OptP}[\log n]$, et le problème $\forall\exists!$**-SAT** rencontré au chapitre 3.4 et discuté en annexe A). Deuxièmement, la complexité algorithmique est un domaine jeune et dynamique qui, malgré les questions ouvertes liées à ses fondements, progresse (par exemple la classe $\#\cdot\mathsf{OptP}[\log n]$ a été introduite en 2009 [HP09], et le résultat **Maj-**k**-SAT**$\in \mathsf{P}$ en erratum 7.6 est en cours de publication [AW21]). Enfin, constatant le large éventail de classes de complexité rencontrées dans nos études, il semble inévitable d'être menés à la conclusion ultime que les subtilités sont nombreuses.

9.1. De la « complexité » des réseaux d'automates

En introduction, nous avons suggéré qu'un panorama de résultats de complexité algorithmique sur les réseaux d'automates nous donnerait des éléments d'une certaine compréhension globale du modèle. Sans obtenir de réponse triviale au déchiffrement de cette question complexe, il nous semble possible de dégager plusieurs éléments.

Graphe d'interaction. Commençons par rappeler qu'il s'agit d'un objet contenant une information partielle sur les fonctions locales du réseau, mais qu'il n'est pour autant pas facile de le calculer (au dessus de NP et coNP, chapitre 4). Dit autrement, étant donné un réseau d'automates, il est difficile de cerner les arcs à ajouter partant du graphe vide, ou les arcs à retirer partant du graphe complet, afin d'identifier exactement les dépendences effectives. Une alternative consiste à considérer un graphe de communication qui donne, pour chaque automate, l'ensemble des automates dont il *peut* dépendre (il peut ne pas dépendre de tous ses voisins entrants du graphe de communication, mais il ne peut en revanche pas dépendre d'un automate qui n'est pas dans ses voisins entrants du graphe de communication). La promesse d'un degré entrant borné, sur laquelle nous reviendrons plus loin, est un moyen alternatif de réduire la combinatoire des possibilités et ainsi ramener le problème dans P.

Au chapitre 7, nous avons considéré des problèmes prenant en entrée un graphe d'interaction, et posant des questions sur les réseaux d'automates possibles. Les constructions proposent un regard nouveau, utilisant à la fois des propriétés globales

(liens entre points fixes et feedback vertex sets) et locales (petits degrés pour raisonner sur la combinatoire des possibilités). Nous avons ainsi considéré frontalement des questionnements relatifs à l'information sur les dynamiques $f : \{0,1\}^n \to \{0,1\}^n$ que l'on peut déduire de la seule connaissance du graphe d'interaction $G_f^{\pm}$ (signé). Dans la recherche d'une construction réalisant une réduction, on se demande : « comment construire une famille de graphes dont on contrôle les dynamiques possibles, afin d'encoder l'instance d'un autre problème de façon à préserver la décision ? » Nos réductions depuis **3-SAT** concernent les points fixes, et ne semblent pas immédiatement adaptables à d'autres éléments des dynamiques possibles sur un graphe d'interaction donné. Les algorithmes atteignant les bornes de complexité optimales sont quant à eux relativement naïfs, mais ont initié une problématique plus générale sur l'extension d'une connaissance partielle d'un réseau (graphe d'interaction et quelques images). La question peut être posée indépendamment pour chaque fonction locale. Ces travaux ont joué sur les liens entre monotonie des dépendances (arcs + et −) et antichaînes (les configurations comparables doivent respecter la monotonie, sur un ordre transitif donc globalement), et mis en évidence les degrés de liberté offerts par la non monotonie (arcs ±) car dans ce cas il n'y a pas de contrainte globale forte à respecter (deux paires de configurations comparables suffisent à obtenir une dépendance non monotone). Il est intéressant de noter que, pour le problème d'existence d'un réseau étendant une connaissance partielle donnée, les cas exclusivement monotone et exclusivement non monotone peuvent être résolus efficacement (dans P), mais leur combinaison rend le problème difficile (NP-complet).

Enfin, les travaux sur la dynamique à isomorphisme près (quand on oublie les étiquettes de $\mathscr{G}_f$ pour $f : \{0,1\}^n \to \{0,1\}^n$), initiés par Picard Marchetto et Richard [PR21], révèlent une autre propriété importante des graphes d'interaction. Il existe deux cas particuliers : la dynamique identité qui correspond au graphe d'interaction avec uniquement n boucles, et la dynamique constante qui correspond au graphe d'interaction sans arc ni boucle. Le cas général est ensuite le suivant : toute dynamique peut être obtenue sur le graphe d'interaction complet (avec boucles). Ainsi, le graphe d'interaction complet ne donne quasiment aucune information sur toute propriété de la dynamique à isomorphisme près (comme le nombre de points fixes, cycles limites, plus généralement sur les occurrences de tout sous-graphe induit dans la dynamique, et encore plus généralement sur toute question exprimable en une logique quelconque sur la signature $\{=, \to\}$). Par les symétries du graphe complet, on pourrait étudier dans quelle mesure il est possible de relâcher la condition d'isomorphisme.

Localité. Dans les réseaux d'automates, les relations de causalité sont implémentées dans les arcs du graphe d'interaction. Ainsi, avoir un degré borné (entrant et/ou sortant, toutes les combinaisons ont leur pertinence) représente une forme de localité des échanges. Cette notion est ici d'une grande souplesse (comparée aux automates cellulaires, dont la structure est parfaitement uniforme sur une grille ou un graphe de Cayley). Il semble particulièrement intéressant d'explorer comment les restrictions

de la topologie (au sens de la structure du graphe d'interaction) peuvent limiter les capacités du modèle à calculer, et donc diminuer la complexité algorithmique des problèmes. Certains deviennent plus faciles, comme le calcul du graphe d'interaction, mais pour d'autres problèmes, la complexité est inchangée, comme les résultats sur la dynamique asymptotique du chapitre 5 qui sont obtenus avec des constructions de degré entrant maximum $\Delta(G_f) \leq 2$. Il se trouve que la rigidité du degré borné peut être contrebalancée par la souplesse d'autres aspects...

Dynamique limite et alphabets. Se restreindre à la dynamique limite (c'est-à-dire ne pas en considérer la partie transitoire), comme nous l'avons fait aux chapitres 5 et 6, simplifie les constructions. En effet, une contrainte importante est ainsi relâchée : *toute* la dynamique n'est pas prise en compte. Il s'ensuit que les configurations transitoires peuvent intuitivement être utilisées comme des variables temporaires. Dans nos résultats sur la complexité algorithmique de décider si l'ensemble limite a une certaine taille (chapitre 5.2), on observe plus précisément que lorsque celles-ci sont en grande quantité (exponentielle) le problème est PSPACE-difficile, alors que lorsqu'elles sont en petite quantité (polynomiale) le problème est coNP-difficile et dans Σ_3^{P} [Gam+21]. On en déduit l'intuition qu'en petite quantité, les variables temporaires de la dynamique transitoire peuvent ne pas suffire à implémenter certains calculs complexes. Peut-être qu'un lien plus formel avec l'espace pourrait être établi.

Des rigidités structurelles apparaissent également lorsque l'alphabet est fixé, par exemple dans les réseaux q-uniformes. Dans ce cas, le nombre de configurations *doit* être de la forme q^n avec n le nombre d'automates, et pour obtenir une certaine taille de la dynamique $\mathscr{G}_f$, il est *nécessaire* d'utiliser plusieurs automates. Du point de vue inverse, laisser libre l'alphabet ramène nos considérations au simple encodage succinct du graphe de la dynamique par un unique circuit (réseau sur 1 automate), comme nous l'avons discuté à propos des encodages au chapitre 3.4 et en perspectives sur FO au chapitre 6.3. L'essence interactionnelle du modèle se retrouve donc à alphabet fixé, ou dont la taille est bornée. En outre, Guillaume Theyssier (I2M, Marseille) me fait régulièrement la remarque suivante : l'alphabet booléen $\{0,1\}$ est un cas particulier [a]. Cette observation est ettayée par plusieurs éléments, à commencer par le graphe d'interaction auquel on aurait envie d'ajouter des étiquettes plus précises que $\{+,-,\pm\}$ lorsque l'alphabet n'est pas booléen. De plus, certains résultats et constructions ne fonctionnent pas dans le cas booléen, ou seulement dans le cas booléen [b].

a. Même s'il permet de simuler tout autre alphabet de façon efficace, en encodant les lettres en binaire (un nombre logarithmique d'automates booléens simule un automate d'alphabet plus grand).

b. Par exemple, pour tout réseau booléen $f : \{0,1\}^n \to \{0,1\}^n$ mis à jour en parallèle, il existe un réseaux booléen $g : \{0,1\}^n \to \{0,1\}^n$ (sur n automates également) et un mode de mise à jour atomique β (une séquence d'ensembles atomiques) tels que $f^{[\beta^{\mathtt{par}}]} = g^{[\beta]}$. Pour l'échange de deux bits $f : x_1x_2 \mapsto x_2x_1$ avec les fonctions locales $f_1 : x_1x_2 \mapsto x_2$ et $f_2 : x_1x_2 \mapsto x_1$ en parallèle, on a les fonctions locales $g_1 : x_1x_2 \mapsto x_1 \oplus x_2$ et $g_2 : x_1x_2 \mapsto x_1$ telles que $f^{[\beta^{\mathtt{par}}]} = f = g^{[(\{1\},\{2\},\{1\})]}$. Ce résultat n'est plus vrai pour les réseaux q-uniformes à partir de $q = 3$ [BGT20b]. On trouvera d'autres considérations qui dépendent des alphabets dans [Bri18 ; Bri19 ; BGT20a].

Encodages et comparaisons entre modèles. Le point de vue bas niveau des encodages permet de passer d'une variante du modèle à une autre. Par exemple, nous avons vu au chapitre 5.1 (remarque 5.4) que tout réseau déterministe peut être transformé efficacement en un réseau non déterministe dont le graphe de la dynamique est identique. La complexité algorithmique d'un même problème n'est en revanche pas égale dans toutes les variantes du modèle. Il serait intéressant de donner un tableau exhaustif des résultats de complexité (et plus généralement des propriétés dynamiques, structurelles et combinatoires) pour toutes les combinaisons parmi :

- déterministe, non déterministe et asynchrone (et pour d'autres modes de mise à jour, mais la notion de « même problème » doit être précisée),
- alphabet booléen, q-uniforme pour q fixé, borné par q fixé (c'est-à-dire où chaque automate a un alphabet $[\![q']\!]$ pour $q' \leq q$), uniforme et libre,
- degré entrant et/ou sortant borné et non borné du graphe d'interaction.

Le dernier point pourra être élargi à différentes familles de graphes. Le travail semble colossal, mais nous avons maintenant à notre disposition de nombreuses constructions pour nous inspirer. Une telle étude systématique et une présentation adéquate des résultats (autre défi) pourrait offrir un recul pertinent sur les réseaux d'automates en tant que modèles de calcul. Des lignes de démarcation seraient sans doute visibles et difficilement contredites, telles que les supériorités (en termes de complexités algorithmiques, c'est-à-dire de capacités à encoder du calcul) :

- du cas non déterministe comparé au cas déterministe,
- de l'alphabet libre comparé à uniforme comparé à borné comparé à q-uniforme comparé à booléen,
- de toute restriction structurelle sur le graphe d'interaction exprimée par une famille F, comparée à une famille $F' \subseteq F$.

On souhaiterait ici reprendre les éléments de la remarque 5.4, et donner un méta-théorème de simulation d'une variante par une autre (réduction d'une variante vers une autre), préservant les bornes inférieures de complexité. C'est précisément ce à quoi nous travaillons avec les membres du projet ANR-18-CE40-0002 *Foundations of Automata Networks* (FANs) porté par Sylvain Sené (LIS, Marseille). Dans un cadre dépassant les réseaux d'automates, nous rédigeons un document de synthèse reprenant les différents modèles de calcul qui animent nos communautés, et les notions de simulation afférentes. Nous souhaitons établir un lien avec la complexité au sens large, à travers un résultat de la forme intuitive :

> si le système dynamique X simule le système dynamique Y grâce à $\Pi : X \to Y$,
> alors pour toute notion de complexité C on aura $C(Y) \leq C(X) + C(\Pi)$.

La majeure partie du travail consiste à établir un cadre unifié où de nombreux modèles se déclinent en raffinements d'une définition commune, c'est-à-dire à définir formellement les termes de cet énoncé. Pacôme Perrotin en développe une réflexion dans ses travaux de thèse, basée sur les diagrammes espace-temps [Per21].

Si le sens de progression de la complexité des réseaux d'automates selon ces trois dimensions (mode de mise à jour, alphabet, graphe d'interaction) semble clair, il serait

intéressant d'observer plus finement les courbes de niveau pour chaque problème. C'est par exemple ce qu'ont réalisé Ríos Wilson et Theyssier dans [RT21], en se focalisant sur les liens entre symétries des interactions locales et asynchronisme (dans le sens de « non synchronisme »). Plus précisément, ils établissent une hiérarchie de symétries locales (sur des graphes d'interaction non orientés, et des fonctions locales ne distinguant les voisins que par le signe des arrêtes), une hiérarchie de modes de mise à jour (parallèle, bloc-séquentiel, horloges locales, général périodique), et analysent comment les propriétés dynamiques et la complexité algorithmique (au travers de plusieurs variantes du problème de prédiction de l'état futur d'un automate donné) varient selon ces deux dimensions. Les résultats montrent que les modes de mise à jour peuvent compenser les symétries locales, et précisent les transferts possibles entre ces deux aspects des modèles de réseaux d'automates.

9.2. Perspectives à long terme

Des perspectives ont été proposées en clôture des chapitres. Nous en reprenons et commentons ici quelques éléments offrant de larges directions de recherche.

Extension d'une information partielle. Au chapitre 7 et ses perspectives, nous avons initié des travaux sur l'existence de réseaux d'automates satisfaisant une ou plusieurs des contraintes suivantes, données en entrée de problèmes de décision :

- dépendances effectives entre automates données par un graphe d'interaction,
- images données par des couples de configurations (x, y) telles que $f(x) = y$, ou par (x, a) tels que $f_j(x) = a$ au niveau des fonctions locales,
- nombre de points fixes donné par une borne inférieure ou supérieure $k \in \mathbb{N}$.

Ces contraintes peuvent être vues comme une information partielle sur un réseau, dont il s'agit d'étudier les extensions possibles. Lorsqu'on s'est intéressé au nombre de points fixes maximum (*resp.* minimum) des réseaux d'automates booléens sur un graphe d'interaction signé donné, les démonstrations de nos réductions se sont attaquées à la question suivante : pour le graphe G_φ construit à partir d'une formule propositionnelle φ, *existe-t-il* un réseau avec au moins k (*resp.* au plus k) points fixes dont c'est le graphe d'interaction ? (La réponse étant « oui » si et seulement si φ est une instance positive.) Ainsi, la propriété visée (ici le nombre de points fixes) peut être formulée comme contrainte d'un problème de décision.

Quelles informations partielles peut-on plus largement envisager ?
Nous proposons quatre classes :

1. liées au graphe d'interaction : existence ou absence d'arcs (avec ou sans signes), propriétés structurelles (par exemple degré borné),
2. liées aux images : globales $f(x) = y$ ou locales $f_j(x) = a$,
3. liées à la dynamique : bornes sur le nombre de points fixes, de cycles limites, propriétés structurelles (par exemple la bijectivité),

4. liées aux circuits des fonctions locales : dont la taille est polynomiale [c].

Les combinaisons sont nombreuses (avec les cas déterministe et non déterministe, pour des alphabets variés, avec un nombre d'automates fixé ou non...), et des considérations applicatives pourraient guider les directions de recherche.

On pense évidement aux mécanismes biologiques dont les réseaux d'automates sont un modèle. À cet effet, les considérations d'informations partielles potentiellement erronées, ou accompagnées de probabilités, seraient pertinentes. De plus, les problèmes de comptage du nombre de réseaux d'automates possibles (plutôt que la simple existence, ≥ 1) pourraient indiquer quelles informations supplémentaires font le plus diminuer le nombre de possibilités, proposant ainsi une orientation aux expérimentations biologiques. Je dois avouer être à peu près ignorant dans ce domaine, ce que je me permets de formuler plus positivement comme un appel à collaboration.

Une source d'inspiration supplémentaire est donnée dans les travaux présentés par Sergiu Ivanov à WAN'2021, sur la « reprogrammation cellulaire ». Quel médicament donner, et quand le donner, pour obtenir un comportement souhaité ? Le problème est modélisé par des entrées additionnelles dans les circuits des fonctions locales, dont l'état est « contrôlé » dans l'objectif de passer d'un attracteur à un autre. Il s'agit d'une généralisation de l'atteignabilité, qui est PSPACE-difficile [PID21].

Théorèmes « de type Rice » et théorie des modèles. Le chapitre 6 a présenté des résultats sur « toute propriété de la dynamique formulable en logique du premier order (FO) ou monadique du second ordre (MSO) ». La généralité de tels énoncés suscite naturellement l'envie de réutiliser les techniques. Il s'agit dans un premier temps d'identifier les éléments requis par une idée de construction implémentant une réduction, comme par exemple nos graphes « neutre » et « suffisant ». Puis il faut montrer que toute propriété ou formule soit possède de tels objets, soit est « triviale » en un certain sens. Dans notre étude, le « suffisant » est obtenu par un argument de maximalité qui nous donne la symétrie entre NP et coNP (on ne sait pas si le maximum est un modèle ou contre-modèle), et le « neutre » est obtenu grâce au lemme de localité de Hanf en logique du premier ordre, qui permet de réaliser un « pompage ».

Comme il a été discuté en perspectives du chapitre 6 (voir 6.3 et 6.4), notre résultat sur FO (théorème 6.1) ne nous donne pas encore entière satisfaction. Une piste consisterait à produire un graphe « neutre » universel (pour toute formule de rang de quantification m), comme il a été fait au théorème 6.15 pour le graphe « suffisant/interdit » Λ_m en logique MSO. Il est cependant nécessaire de considérer des assemblages plus complexes que l'union disjointe. En effet, pour la propriété d'avoir une dynamique connexe (exprimable en MSO), tout graphe est « interdit » (l'union disjointe casse la connexité, excepté avec le graphe vide...). De plus, au premier ordre on a par exemple la formule $\exists x : \forall y : y \to x$ (c'est-à-dire, le réseau est une fonction

c. Pour n automates booléens, on a 2^{2^n} fonctions locales distinctes possibles, ainsi presque toutes ne sont implémentées que par des circuits de taille exponentielle.

constante) qui est bien ω non triviale, mais pour laquelle tout graphe est encore une fois « interdit » (pour l'union disjointe).

La théorie des modèles opère comme un bulldozer qui ratisse large dans les questions qu'il est possible de poser. Les bornes inférieures de complexité sont parfois faibles (NP- ou coNP-difficulté alors que tous les niveaux de PH sont atteints, théorème 6.10), mais séparent tout de même le trivial ($\mathcal{O}(1)$) du difficile (à partir de NP ou coNP). Il est assez instinctif de vouloir passer au niveau « méta » pour d'autres problèmes que les propriétés de la dynamique étant donné un réseau d'automates.

- Sur le calcul du graphe d'interaction d'un réseau d'automates f donné (chapitre 4) : que peut-on dire de la complexité des propriétés FO et MSO du graphe d'interaction G_f ?
- Avec le graphe d'interaction G en entrée (chapitre 7) : que peut-on dire de la complexité des propriétés FO et MSO des dynamiques $\mathscr{G}_f$ de réseaux f ayant pour graphe d'interaction $G_f = G$?

Notons toutefois qu'il est possible d'énoncer des résultats généraux sans faire appel à la théorie des modèles. Par exemple, au théorème 5.21 (chapitre 5 page 71), nous avons uniquement utilisé la non trivialité effective (exitence d'un algorithme produisant un modèle et un contre-modèle pour toute taille n de réseau d'automates), sans mention à une logique dans laquelle est exprimée la propriété.

Complexités algorithmiques. La théorie de la complexité algorithmique est un domaine extraordinairement vaste, et la complexité *dans le pire cas* en est la facette la plus étudiée. Cette facette de la (en anglais) *worst case complexity* prévaut, car en pratique elle offre une précieuse *garantie*, et en théorie elle pose des défis fondamentaux à la science informatique. Toutes les questions posées dans ce document se déclinent pour d'autres notions de complexité algorithmique, par exemple pour la complexité paramétrée (qui identifie plus finement les éléments *moteurs* de la complexité), la complexité d'approximation (pertinente d'un point de vue pragmatique et applicatif), la complexité moyenne (qui évite de donner trop d'importance aux cas extrêmes), et la complexité de communication (qui se concentre sur les ressources en communication nécessaires pour la résolution de problèmes sur des modèles de calcul distribué). Chacune pourra apporter un nouveau point de vue sur les réseaux d'automates.

9.3. Ouvertures sur un monde gelé et sabloneux

L'expressivité des réseaux d'automates est une grande force pour modéliser et analyser de nombreux phénomènes naturels. Les résultats énoncés dans ce document ont ainsi une portée générale, dont le revers de la médaille est une complexité algorithmique élevée : typiquement, avec des temps de calcul au mieux polynomiaux, et des bornes inférieures caractérisées par des classes dont les algorithmes sont en pratique (sur nos ordinateurs séquentiels) exponentiels. Nous ouvrons ici sur des

restrictions de cette expressivité, qui intersectent mes intérêts pour les modèles de piles de sable. Ces restrictions feront diminuer la complexité des problèmes liés à la prédiction (calculer l'état d'un automate après $t \in \mathbb{N}$ (unaire) itérations), posant la question de leur P-complétude (intrinsèquement séquentiel) ou de leur appartenance à la classe NC (efficacement parallélisable). L'annexe B présente ces notions.

Sans définir précisément le terme « simulation » auquel fait référence le symbole $\succeq$, il est assez intuitif [d] d'énoncer les relations suivantes entre les modèles des réseaux d'automates (RA), des automates cellulaires (AC) et des piles de sable (SPM, de l'anglais *Sand Pile Models*) : RA $\succeq$ AC $\succeq$ SPM. Dans les automates cellulaires, la topologie du réseau (son graphe d'interaction) est une grille $\mathbb{Z}^d$ (infinie), et les fonctions locales sont uniformes (avec le même voisinage relatif pour tous les automates). Dans les piles de sable, on a de plus une notion de conservation de la masse (avec la somme des états, vus comme des entiers, constante). On peut ainsi définir formellement une famille de réseaux d'automates $(\rho_n)_{n\in\mathbb{N}}$ correspondant aux piles de sable, dont les éléments seront notés ρ pour un n fixé, par simplicité. Par exemple, en prenant n^2 automates nommés $(i,j) \in [n]^2$, sur l'alphabet $[\![8]\!]$ (les états représenteront la quantité de grains dans chaque cellule), dont les fonctions locales sont :

$$\rho_{(i,j)}(x) = \begin{cases} x_{(i,j)} - 4 \cdot \mathtt{H}(x_{(i,j)} - 4) + \sum\limits_{(i',j')\in \mathtt{N}(i,j)} \mathtt{H}(x_{(i',j')} - 4) & \text{si } i \notin \{1,n\} \text{ et } j \notin \{1,n\} \\ 0 & \text{sinon} \end{cases}$$

avec $\mathtt{H}(a) = 1$ si $a > 0$, $\mathtt{H}(a) = 0$ si $a \leq 0$, et $\mathtt{N}(i,j) = \{(i,j+1),(i+1,j),(i,j-1),(i-1,j)\}$.

Pour le mode de mise à jour parallèle, on obtiendra une « simulation pas à pas » du modèle traditionnel de piles de sable défini par Bak, Tang et Wiesenfeld en 1987 [BTW87], en se concentrant sur les éléments suivants. On ignore les automates (i,j) pour $i \in \{1,n\}$ ou $j \in \{1,n\}$, qui agissent comme des puits recevant des grains mais n'en donnant jamais [e]. On se restreint aux orbites partant des configurations de la forme $x + e_{(\alpha,\beta)}$, avec $x_{(i,j)} < 4$ pour tous $(i,j) \in [n]^2$, et $e_{(\alpha,\beta)}$ la base contenant l'état 1 sur la cellule $(\alpha,\beta) \in [n]^2$ et l'état 0 partout ailleurs. On observe alors que les configurations x avec $x_{(i,j)} < 4$ pour tous $(i,j) \in [n]^2$ sont des points fixes (configurations stables), et que suite à l'ajout d'une base (un grain sur une cellule) la dynamique vérifie l'invariant $y_{(i,j)} \in [\![8]\!]$ pour tous $y \in \mathfrak{O}_\rho(x + e_{(\alpha,\beta)})$ et $(i,j) \in [n]^2$. Voir figure 9.1.

Entre les modèles les plus généraux de réseaux d'automates et la famille restreinte des piles de sable, il y a un monde foisonnant. Nous proposons un rapide survol de contributions animant la communauté, pour des restrictions portant sur les fonc-

d. Même si techniquement faux, à cause de l'infinité de l'espace pour les automates cellulaires, et de l'infinité de l'ensemble des états pour les piles de sable. On peut néanmoins naturellement considérer des réseaux d'automates sur un nombre infini dénombrable d'automates, et des piles de sable sur un ensemble fini d'états, comme nous le discutons ci-après.

e. Il s'agit d'une convention standard pour définir des supports de piles de sable finis, qui soient intuitivement « dissipatifs », et formellement tels que toute configuration converge vers un point fixe. On obtient ainsi une belle structure algébrique : le fameux groupe des piles de sable [Dha90; Dha+95].

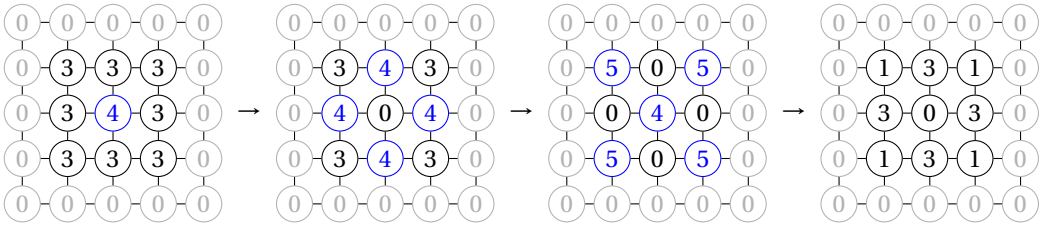


FIGURE 9.1. – Simulation d'une pile de sable par un réseau sur les automates $[5]^2$ (cellules instables en bleu). La configuration avec 3 grains partout plus un grain à la position (3,3) atteint une configuration stable en 3 itérations.

tions locales autorisées, la topologie du graphe d'interaction, et la considération d'un monde gelé (en anglais, *freezing*).

- Pour démontrer la P-difficulté de problèmes de prédiction de l'état futur d'un automate, on réduit depuis le *Circuit Value Problem* (**CVP**) (voir annexe B.2). La technique, appelée approche de Banks [Ban71], consiste à construire une configuration telle que la dynamique du système « simule pas à pas » le calcul d'un circuit booléen donné, via des *signaux*. L'information sur l'état des portes se propage des entrées vers les sorties (pour les piles de sable, cette propagation est comparée à une *avalanche*). C'est par exemple l'approche suivie par Cook pour démontrer l'universalité Turing de la règle 110 [Coo04]. Dans [Gol+17b], nous donnons des méta-résultats de simulation et complexité sur cette approche, qui révèlent une distinction importante entre les modèles pour lesquels les canaux de transmission d'information (*signaux*, ou *wires* en anglais) peuvent être réutilisés, et ceux pour lesquels ils sont à usage unique.
- Dans le premier chapitre de sa thèse de doctorat en 1991 [Fed91, pages 2-3], Feder propose une caractérisation systématique de six problèmes sur les réseaux d'automates booléens (évaluation, stabilité, convergence, énumération, comptage, optimisation), lorsque la base de portes utilisées pour les circuits des fonctions locales est restreinte ($\{\wedge,\neg\}$,$\{\wedge,\vee\}$, $\{\wedge,\neg\}$, $\{\vee,\neg\}$, $\{\oplus\}$, $\{\neg\}$). Par exemple, pour les réseaux booléens (mis à jour en parallèle) dont toutes les portes sont des *ou exclusifs* ($\{\oplus\}$), le calcul des images et l'existence d'un point fixe sont dans NC, et la convergence vers un point fixe d'une configuration donnée est dans P.
- Dans le cas booléen, les fonctions locales à seuil sont de la forme :

$$f_i(x) = \mathrm{H}(-\theta_i + \sum_{j \in \mathrm{N}(i)} w_{i,j} \cdot x_j),$$

pour un voisinage $\mathrm{N}(i)$, une matrice de poids w, et un vecteur de seuils θ. Elles sont l'objet d'une attention particulière, et incluent la *majorité*, où chaque automate prend l'état majoritaire parmi ses voisins [GO80]. La prédiction est P-complète sur les grilles en dimension $d \geq 3$ [Moo97], et sur les graphes symétriques planaires de degré au plus 5 [GM15]. Les équilibres sont ténus : suivant le comportement défini en cas d'égalité (majorité stricte), d'autres contraintes sur

le graphe, et pour la variante gelée, la complexité de prédiction oscille entre NC et la P-complétude [GMT13 ; GM14 ; Gol+17a]. En quantifiant existentiellement sur certains modes de mise à jour, on atteint la NP-complétude [Gol+21b]. Les systèmes *totalistiques* (dont la règle locale dépend d'un comptage des voisins à 1, sur l'alphabet {0,1}) et *ensemblistes* (dont la règle locale dépend de l'ensemble des états des voisins, ce qui uniformise complètement le voisinage) ont également été décortiqués [Río21 ; Gol+21c ; Gol+21a].

- Le cadre général des automates cellulaires gelés a été introduit dans [GOT15] (bien qu'ils aient déjà été étudiés, par exemple au travers de *bootstrap percolation* [GMT13] et de *life without death* [GM96]). Il consiste à n'autoriser que les changements d'états croissants selon un ordre donné, ou à limiter le nombre de changements d'état possibles (*bounded changes*), ou à imposer la convergence vers un point fixe [OT19]. Ces définitions peuvent être appliquées à tout système dynamique, et plusieurs des travaux déjà cités considèrent des modèles *freezing*. Les mondes froids donnent du relief à un obstacle observé et étudié dans la conception de réductions depuis **CVP** pour les preuves de P-difficulté : le *croisement* d'information. Nous examinons cet aspect ci-après.

Les questions évoquées dans cette liste gravitent autour d'une question ouverte, concernant le modèle de piles de sable en deux dimensions défini par ρ.

Prédiction des piles de sable (**PRED-SPM**)
Entrée : une configuration $x : [n]^2 \to [\![4]\!]$, et deux cellules $(\alpha,\beta),(i,j) \in [n]^2$.
Question : est-ce que qu'il existe $y \in \mathfrak{S}_\rho(x + e_{(\alpha,\beta)})$, telle que $y_{(i,j)} \geq 4$?

Question ouverte 9.1. *Est-ce que* **PRED-SPM** *est* P*-complet, ou bien dans* NC *?*

La variante gélée du problème **PRED-SPM** (où chaque cellule est *éboulée*[f] au plus une fois) est également ouverte. On peut commencer par remarquer qu'un algorithme naïf, qui consiste à calculer les configurations successives (partant de $x + e_{(\alpha,\beta)}$, jusqu'à atteindre une configuration stable), fonctionne en temps polynomial [FP19]. En une dimension ce problème est dans NC, et en dimension trois ou plus il est P-complet (voir l'article [MN99] qui a lancé la question, et notre revue [FP19] pour une formulation plus générale de cette sensibilité à la dimension). En suivant l'approche de Banks, on se heurte à la conception d'une porte de croisement. Ou, de façon symétrique, la commutativité des piles de sable[g] fait obstacle à la conception de portes non monotones. Or, comme discuté en annexe B.2, la non monotonie ou la non planarité (croisement d'information) est requise pour que le problème **CVP** soit P-difficile (le cas monotone planaire est dans NC [Yan91]). Goles et Gajardo ont démontré qu'il est impossible de réaliser des croisements, pour des formes élémentaires de *signaux* transportant l'information [GG06]. Face à cet achoppement, différentes relaxations

f. L'éboulement d'une cellule consiste, lorsque son état contient au moins 4 grains, à diminuer son contenu de 4 unités, et d'ajouter 1 unité à chacune de ses quatre voisines (sur $[n]^2$).

g. On atteint la même configuration stable, quel que soit l'ordre dans lequel on éboule les cellules.

ont été considérées, dans le but d'en cerner les contours. Tout d'abord, l'obstacle au croisement se trouve dans la planarité du graphe d'interaction (ou du *support* de la dynamique, dans la terminologie des piles de sable) [NP18 ; FP19 ; FNP20]. En élargissant le voisinage, on observe que des modèles de piles de sable en deux dimensions atteignent la P-complétude [FGM12 ; GG06]. Les croisements sont de même réalisables pour toute « forme » de voisinage, y compris convexe qui complique l'entrelacement de deux signaux [NP18]. Dans [GMP21], nous avons mis en avant des relations de simulation entre piles de sable et automates booléens dans un monde gelé :

état de la cellule dans x (nombre de grains)	0	1	2	3	4
fonction locale	porte $\wedge$	majorité stricte	majorité	porte $\vee$	constante 1

Cela nous a permis de faire appel à la riche littérature présentée ci-avant, pour étudier les restrictions du problème **PRED-SPM** suivant les états autorisés dans la configuration x, parmi $[\![4]\!]$. Il en ressort que la variante gelée du problème **PRED-SPM** peut être restreinte aux configurations dont les états sont choisis parmi $\{0,2,4\}$, sans changer la complexité du problème (dans cette formulation avec l'état 4, aucun grain n'est ensuite ajouté en (α,β), c'est-à-dire que l'on considère directement l'orbite de x).

Dans une perspective plus générale sur les croisements, pour les automates cellulaires *freezing* sur une grille en deux dimensions avec voisinage de von Neumann, deux changements d'état sont nécessaires et suffisants pour obtenir l'universalité intrinsèque [Bec+18]. Ces questionnements se rapprochent de l'essence des processus calculatoires dans les systèmes « complexes » et, dans certains cas, de notre incapacité à démontrer formellement qu'il est impossible de réaliser une réduction depuis **CVP**. La notion de *signal* pour transporter et manipuler l'information est très lâche, et seules des formes élémentaires sont en général considérées [Gol+17b ; FP19]. Typiquement, dans les piles de sable, on utilise une suite de cellules qui sont éboulées pour représenter l'état 1 se diffusant dans un circuit. Les portes $\wedge$ et $\vee$ sont extrêmement simples à construire, et la difficulté réside dans l'implémentation des croisements (ou, de façon équivalence, dans l'implémentation de la négation). Comme l'ont souligné Delorme et Mazoyer dans [DM02], quantifier sur les possibles encodages de la notion de *signal* afin de donner des résultats d'impossibilité généraux, est un obstacle majeur.

Deux autres angles d'attaque de la question 9.1 restent ouverts. Il se pourrait que le problème **PRED-SPM** soit dans NC, mais probablement résolu seulement par des algorithmes compliqués (comme l'existence de cycle pair mentionné au chapite 7.5.3, dans P mais dont les seuls algorithmes connus sont très conséquents). Il se pourrait enfin que ce problème soit dans P, mais ne soit ni P-difficile ni dans NC, car des classes intermédiaires existent sous l'hypothèse $\mathsf{NC} \neq \mathsf{P}$ (voir le chapitre préliminaire 2.2.2). Dans ce dernier cas, un savoir-faire reste à inventer.

Bibliographie personnelle depuis le doctorat (2013)

[APS15] Aurore ALCOLEI, Kévin PERROT et Sylvain SENÉ. « On the flora of asynchronous locally non-monotonic Boolean automata networks ». In : *Proceedings of SASB'2015*. T. 326. ENTCS. 2015, p. 3-25. DOI : 10.1016/j.entcs.2016.09.016. ARXIV : 1510.05452.

[Bal+20] Pedro Paulo BALBI, Enrico FORMENTI, Kévin PERROT, Sara RIVA et Eurico L. P. RUIVO. « Non-maximal Sensitivity to Synchronism in Periodic Elementary Cellular Automata : Exact Asymptotic Measures ». In : *Proceedings of AUTOMATA'2020*. T. 12286. LNCS. 2020, p. 14-28. DOI : 10.1007/978-3-030-61588-8_2. ARXIV : 2004.07128 (cf. p. 102).

[BNP18] Cédric BERENGER, Peter NIEBERT et Kévin PERROT. « Balanced Connected Partitioning of Unweighted Grid Graphs ». In : *Proceedings of MFCS'2018*. T. 117. LIPIcs. 2018, 39 :1-39 :18. DOI : 10.4230/LIPIcs.MFCS.2018.39.

[Bri+19] Florian BRIDOUX, Amélia DURBEC, Kévin PERROT et Adrien RICHARD. « Complexity of Maximum Fixed Point Problem in Boolean Networks ». In : *Proceedings of CiE'2019*. T. 11558. LNCS. 2019, p. 132-143. DOI : 10.1007/978-3-030-22996-2_12. HAL : hal-02403974 (cf. p. 89-92, 94).

[Bri+20] Florian BRIDOUX, Amélia DURBEC, Kévin PERROT et Adrien RICHARD. *Complexity of fixed point counting problems in Boolean Networks*. Soumis à un journal. 2020. ARXIV : 2012.02513 (cf. p. 93, 94).

[Bri+21] Florian BRIDOUX, Caroline GAZE-MAILLOT, Kévin PERROT et Sylvain SENÉ. « Complexity of Limit-Cycle Problems in Boolean Networks ». In : *Proceedings of SOFSEM'2021*. T. 12607. LNCS. 2021, p. 135-146. DOI : 10.1007/978-3-030-67731-2_10. ARXIV : 2001.07391 (cf. p. 62, 98, 99).

[Bri+17] Florian BRIDOUX, Pierre GUILLON, Kévin PERROT, Sylvain SENÉ et Guillaume THEYSSIER. « On the Cost of Simulating a Parallel Boolean Automata Network by a Block-Sequential One ». In : *Proceedings of TAMC'2017*. T. 10185. LNCS. 2017, p. 112-128. DOI : 10.1007/978-3-319-55911-7_9. HAL : hal-01479439. ARXIV : 1702.03101 (cf. p. 46).

[Cre+15] Christophe CRESPELLE, Tien-Nam LE, Kévin PERROT et Thi Ha Duong PHAN. « Linearity is strictly more powerful than contiguity for encoding graphs ». In : *Proceedings of WADS'2015*. T. 9214. LNCS. 2015, p. 212-223.

[Cre+16] Christophe CRESPELLE, Tien-Nam LE, Kévin PERROT et Thi Ha Duong PHAN. « Linearity is strictly more powerful than contiguity for encoding graphs ». In : *Discrete Mathematics* 339.8 (2016), p. 2168-2177. ARXIV : 1803.05414.

[FNP20] Jérémy FERSULA, Camille NOÛS et Kévin PERROT. *Sandpile toppling on Penrose tilings : identity and isotropic dynamics*. Soumis à un journal. 2020. ARXIV : 2006.06254 (cf. p. 123).

[FP19] Enrico FORMENTI et Kévin PERROT. « How Hard is it to Predict Sandpiles on Lattices? A Survey ». In : *Fondamenta Informaticae* 171.1-4 (2019), p. 189-219. DOI : 10.3233/FI-2020-1879. ARXIV : 1909.12150 (cf. p. 122, 123).

[FPR14] Enrico FORMENTI, Kévin PERROT et Eric RÉMILA. « Computational complexity of the avalanche problem on one dimensional Kadanoff sandpiles ». In : *Proceedings of AUTOMATA'2014*. T. 8996. LNCS. 2014, p. 21-30. ARXIV : 1803.05498.

[FPR18] Enrico FORMENTI, Kévin PERROT et Eric RÉMILA. « Computational complexity of the avalanche problem for one dimensional decreasing sandpiles ». In : *Journal of Cellular Automata* 13.3 (2018), p. 215-228. HAL : halshs-01417248.

[Gam+21] Guilhem GAMARD, Pierre GUILLON, Kevin PERROT et Guillaume THEYSSIER. « Rice-Like Theorems for Automata Networks ». In : *Proceedings of STACS'2021*. T. 187. LIPIcs. Schloss Dagstuhl, 2021, 32 :1-32 :17. DOI : 10.4230/LIPIcs.STACS.2021.32 (cf. p. 67, 69-71, 74, 75, 77-80, 82, 83, 115).

[GMP21] Eric GOLES, Pedro MONTEALEGRE et Kévin PERROT. « Freezing sandpiles and Boolean threshold networks : Equivalence and complexity ». In : *Advances in Applied Mathematics* 125 (2021), p. 102161. DOI : 10.1016/j.aam.2020.102161. ARXIV : 2101.04204 (cf. p. 123).

[Gol+17b] Eric GOLES, Pedro MONTEALEGRE, Kévin PERROT et Guillaume THEYSSIER. « On the complexity of two-dimensional signed majority cellular automata ». In : *Journal of Computer and System Sciences* 91 (2017), p. 1-32. DOI : 10.1016/j.jcss.2017.07.010. HAL : hal-01472161 (cf. p. 121, 123).

[Mon+17] Marco MONTALVA MEDEL, Kévin PERROT, Pedro de OLIVEIRA et Eurico RUIVO. « Sensitivity to synchronism in some boolean automata networks ». In : *Proceedings of AUTOMATA'2017 exploratory papers* (2017), p. 69-76. HAL : hal-01785462 (cf. p. 102).

[NP18] Viet-Ha NGUYEN et Kévin PERROT. « Any shape can ultimately cross information on two-dimensional abelian sandpile models ». In : *Proceedings of AUTOMATA'2018*. T. 10875. LNCS. 2018, p. 127-142. DOI : 10.1007/978-3-319-92675-9_10. HAL : hal-01824872. ARXIV : 1709.00464 (cf. p. 123).

[NP21] Viet-Ha NGUYEN et Kévin PERROT. *Rikudo is NP-complete*. Soumis à un journal. 2021. ARXIV : 2101.09332.

[NPV20] Viet-Ha NGUYEN, Kévin PERROT et Mathieu VALLET. « NP-completeness of the game Kingdomino ». In : *Theoretical Computer Science* 822 (2020), p. 23-35. DOI : 10.1016/j.tcs.2020.04.007. HAL : hal-03121418. ARXIV : 1909.02849.

[Noû+20] Camille NOÛS, Kévin PERROT, Sylvain SENÉ et Lucas VENTURINI. « #P-completeness of counting update digraphs, cacti, and series-parallel decomposition method ». In : *Proceedings of CiE'2020*. T. 12098. LNCS. 2020, p. 326-338. DOI : 10.1007/978-3-030-51466-2_30. ARXIV : 2004.02129 (cf. p. 104, 106, 108).

[Per19] Kévin PERROT. *On the complexity of counting feedback arc sets*. Soumis à un journal. 2019. ARXIV : 1909.03339 (cf. p. 24, 25).

[Per+20] Kévin PERROT, Marco MONTALVA-MEDEL, Pedro P. B. de OLIVEIRA et Eurico L. P. RUIVO. « Maximum sensitivity to update schedule of elementary cellular automata over periodic configurations ». In : *Natural Computing* 19.1 (2020), p. 51-90. DOI : 10.1007/s11047-019-09743-9. HAL : hal-02179732v1 (cf. p. 102, 109).

[PNP13] Kévin PERROT, Doanh NGUYEN-NGOC et Ha Duong PHAN. « Effects of Migration of Three Competing Species on Their Distributions in Multizone Environment ». In : *Proceedings of RIVF'2013*. IEEE. 2013, p. 227-232.

[PPS18] Kévin PERROT, Pacôme PERROTIN et Sylvain SENÉ. « A framework for (de)composing with Boolean automata networks ». In : *Proceedings of MCU'2018*. T. 10881. LNCS. 2018, p. 121-136. DOI : 10.1007/978-3-319-92402-1_7. HAL : hal-01654221. ARXIV : 1802.10400.

[PPS20] Kévin PERROT, Pacôme PERROTIN et Sylvain SENÉ. « On the Complexity of Acyclic Modules in Automata Networks ». In : *Proceedings of TAMC'2020*. T. 12337. LNCS. 2020, p. 168-180. DOI : 10.1007/978-3-030-59267-7_15. ARXIV : 1910.07299.

[PPS21a] Kévin PERROT, Pacôme PERROTIN et Sylvain SENÉ. « On Boolean automata networks (de)composition ». In : *Fundamenta Informaticae* 181.2-3 (2021), p. 163-188. DOI : 10.3233/FI-2021-2055. ARXIV : 1802.10400.

[PPS21b] Kévin PERROT, Pacôme PERROTIN et Sylvain SENÉ. « Optimising attractor computation in Boolean automata networks ». In : *Proceedings of LATA'2021*. T. 12638. LNCS. 2021, p. 68-80. ARXIV : 2005.14531.

[PR14] Kévin PERROT et Éric RÉMILA. « Emergence of regularities on decreasing sandpile models ». In : *15th Mons Theoretical Computer Science Days, Nancy, France* (2014).

[PR15a] Kévin PERROT et Éric RÉMILA. « Emergence on decreasing sandpile models ». In : *Proceedings of MFCS'2015*. T. 9234. LNCS. 2015, p. 419-431. HAL : halshs-01212069.

[PR15b] Kévin PERROT et Éric RÉMILA. « Piles de sable décroissantes 1D : classification expérimentale d'émergences ». In : *Technique et Science Informatiques – Automates cellulaires et réseaux d'automates : le rôle central de l'irrégularité* 34.4 (2015), p. 377-400. HAL : hal-01738027.

[PR20] Kévin PERROT et Éric RÉMILA. « On the emergence of regularities on one-dimensional decreasing sandpiles ». In : *Theoretical Computer Science* 843 (2020), p. 1-24. DOI : 10.1016/j.tcs.2020.06.018. HAL : halshs-02884875.

[PR17] Kévin PERROT et Eric RÉMILA. « Strong emergence of wave patterns on Kadanoff sandpiles ». In : *Electronic Journal of Combinatorics* 24.2 (2017). HAL : halshs-01417254.

[PV15] Kévin PERROT et Trung VAN PHAM. « Feedback arc set problem and NP-hardness of minimum recurrent configuration problem of Chip-firing game on directed graphs ». In : *Annals of Combinatorics* 19 (2015), p. 373-396. ARXIV : 1303.3708.

[PV16] Kévin PERROT et Trung VAN PHAM. « Chip-firing game and partial Tutte polynomial for Eulerian digraphs ». In : *Electronic Journal of Combinatorics* 23.1 (2016). ARXIV : 1306.0294.

[Rui+20] Eurico L. P. RUIVO, Pedro Paulo BALBI, Marco MONTALVA-MEDEL et Kévin PERROT. « Maximum sensitivity to update schedules of elementary cellular automata over infinite configurations ». In : *Information and Computation* (2020), p. 104538. DOI : 10.1016/j.ic.2020.104538 (cf. p. 102).

[RBP20] Eurico L. P. RUIVO, Pedro Paulo BALBI et Kévin PERROT. « An asynchronous solution to the synchronisation problem for binary one-dimensional cellular automata ». In : *Physica D : Nonlinear Phenomena* 413 (2020), p. 132554. DOI : 10.1016/j.physd.2020.132554.

[Rui+18] Eurico L. P. RUIVO, Marco MONTALVA-MEDEL, Pedro P. B. de OLIVEIRA et Kévin PERROT. « Characterisation of the elementary cellular automata in terms of their maximum sensitivity to all possible asynchronous updates ». In : *Chaos, Solitons & Fractals* 113 (2018), p. 209-220. DOI : 10.1016/j.chaos.2018.06.004 (cf. p. 102).

Bibliographie générale

[A00a] OEIS A000679. « The Online Encyclopedia of Integer Sequences ». https://oeis.org/A000670 (cf. p. 102, 109).

[A00b] OEIS A006126. « The Online Encyclopedia of Integer Sequences ». https://oeis.org/A006126 (cf. p. 88).

[AW21] S. AKMAL et R. WILLIAMS. *MAJORITY-3SAT (and Related Problems) in Polynomial Time*. 2021. ARXIV : 2107.02748 (cf. p. 92, 113).

[AW93] E. W. ALLENDER et K. W. WAGNER. « Counting hierarchies : polynomial time and constant depth circuits ». In : *Current Trends in Theoretical Computer Science* (1993), p. 469-483. DOI : 10.1142/9789812794499_0035 (cf. p. 144).

[Alo85] N. ALON. « Asynchronous threshold networks ». In : *Graphs and Combinatorics* 1 (1985), p. 305-310. DOI : 10.1007/BF02582959 (cf. p. 58).

[Ara08] J. ARACENA. « Maximum Number of Fixed Points in Regulatory Boolean Networks ». In : *Bulletin of Mathematical Biology* 70.5 (2008), p. 1398-1409. DOI : 10.1007/s11538-008-9304-7 (cf. p. 47, 89, 91).

[Ara+13a] J. ARACENA, J. DEMONGEOT, É. FANCHON et M. MONTALVA. « On the number of different dynamics in Boolean networks with deterministic update schedules ». In : *Mathematical Biosciences* 242 (2013), p. 188-194. DOI : 10.1016/j.mbs.2013.01.007 (cf. p. 101).

[Ara+13b] J. ARACENA, J. DEMONGEOT, É. FANCHON et M. MONTALVA. « On the number of update digraphs and its relation with the feedback arc sets and tournaments ». In : *Discrete Applied Mathematics* 161 (2013), p. 1345-1355. DOI : 10.1016/j.dam.2012.12.018 (cf. p. 109).

[Ara+11] J. ARACENA, É. FANCHON, M. MONTALVA et M. NOUAL. « Combinatorics on update digraphs in Boolean networks ». In : *Discrete Applied Mathematics* 159 (2011), p. 401-409 (cf. p. 102-106).

[Ara+09] J. ARACENA, E. GOLES, A. MOREIRA et L. SALINAS. « On the robustness of update schedules in Boolean networks ». In : *Biosystems* 97 (2009), p. 1-8. DOI : 10.1016/j.biosystems.2009.03.006 (cf. p. 102, 103).

[AGS13] J. ARACENA, L. GÓMEZ et L. SALINAS. « Limit cycles and update digraphs in Boolean networks ». In : *Discrete Applied Mathematics* 161 (2013), p. 1-12. DOI : 10.1016/j.dam.2012.07.003 (cf. p. 101).

[ARS17] J. ARACENA, A. RICHARD et L. SALINAS. « Number of Fixed Points and Disjoint Cycles in Monotone Boolean Networks ». In : *SIAM Journal on Discrete Mathematics* 31.3 (2017), p. 1702-1725. DOI : 10.1137/16M1060868 (cf. p. 47).

[AB09] S. ARORA et B. BARAK. *Computational Complexity*. Version préliminaire libre d'accès (janvier 2007). Cambridge University Press, 2009. ISBN : 9780521424264 (cf. p. 21, 22).

[BTW87] P. BAK, C. TANG et K. WIESENFELD. « Self-organized criticality : An explanation of the $1/f$ noise ». In : *Physical Review Letters* 59.4 (1987), p. 381-384. DOI : 10.1103/PhysRevLett.59.381 (cf. p. 120).

[Ban71] E. R. BANKS. « Information processing and transmission in cellular automata ». URL : mit:1721.1/6891. Thèse de doct. Massachusetts Institute of Technology, 1971 (cf. p. 121).

[Bec+18] F. BECKER, D. MALDONADO, N. OLLINGER et G. THEYSSIER. « Universality in Freezing Cellular Automata ». In : *Proceedings of CiE'2018*. T. 10936. LNCS. 2018, p. 50-59. DOI : 10.1007/978-3-319-94418-0_5 (cf. p. 123).

[Bei91] R. BEIGEL. « Bounded queries to SAT and the Boolean hierarchy ». In : *Theoretical Computer Science* 84.2 (1991), p. 199-223. DOI : 10.1016/0304-3975(91)90160-4 (cf. p. 143).

[BBF98] R. BEIGEL, H. BUHRMAN et L. FORTNOW. « NP Might Not Be as Easy as Detecting Unique Solutions ». In : *Proceedings of STOC'98*. 1998, p. 203-208. DOI : 10.1145/276698.276737 (cf. p. 143).

[BG82] A. BLASS et Y. GUREVICH. « On the unique satisfiability problem ». In : *Information and Control* 55.1 (1982), p. 80-88. DOI : 10.1016/S0019-9958(82)90439-9 (cf. p. 27, 142, 143).

[Bri18] F. BRIDOUX. *Sequentialization and Procedural Complexity in Automata Networks*. 2018. ARXIV : 1803.00438 (cf. p. 115).

[Bri19] F. BRIDOUX. « Simulations intrinsèques et complexités dans les réseaux d'automates ». TEL : tel-02436228. Thèse de doct. Aix-Marseille Université, 2019 (cf. p. 115).

[BGT20a] F. BRIDOUX, M. GADOULEAU et G. THEYSSIER. « Expansive automata networks ». In : *Theoretical Computer Science* 843 (2020), p. 25-44. DOI : 10.1016/j.tcs.2020.06.019 (cf. p. 115).

[BGT20b] F. BRIDOUX, M. GADOULEAU et G. THEYSSIER. « On Simulation in Automata Networks ». In : *Proceedings of CiE'2020*. T. 12098. LNCS. 2020, p. 227-288. DOI : 10.1007/978-3-030-51466-2_24 (cf. p. 115).

[BK98] R. E. BURKARD et J. KRARUP. « A linear algorithm for the pos/neg-weighted 1-median problem on a cactus ». In : *Computing* 60 (1998), p. 193-215. DOI : 10.1007/BF02684332 (cf. p. 107).

[Cha+20] T. CHATAIN, S. HAAR, J. KOLČÁK, L. PAULEVÉ et A. THAKKAR. « Concurrency in Boolean networks ». In : *Natural Computing* 19.1 (2020), p. 91-109. DOI : 10.1007/s11047-019-09748-4 (cf. p. 67).

[Cob65] A. COBHAM. « The intrinsic computational difficulty of functions ». In : *Logic, Methodology and Philosophy of Science : Proceedings of the 1964 International Congress (Studies in Logic and the Foundations of Mathematics)*. 1965, p. 24-30 (cf. p. 10, 19).

[Coo04] M. COOK. « Universality in Elementary Cellular Automata ». In : *Complex Systems* 15 (2004). URL : complex-systems:v15_i01_a01, p. 1-40 (cf. p. 121).

[CE12] B. COURCELLE et J. ENGELFRIET. *Graph structure and monadic second-order logic. A language-theoretic approach*. Cambridge University Press, 2012. ISBN : 9780521898331. DOI : 10.1017/CBO9780511977619. HAL : hal-00646514 (cf. p. 73, 84).

[CH11] Y. CRAMA et P. L. HAMMER. *Boolean Functions : Theory, Algorithms, and Applications*. Cambridge University Press, 2011. ISBN : 9780521847513. DOI : 10.1017/CBO9780511852008 (cf. p. 50, 51, 95).

[DM02] M. DELORME et J. MAZOYER. « Signals on Cellular Automata ». In : *Collision-based Computing*. Springer, 2002, p. 231-275. DOI : 10.1007/978-1-4471-0129-1_9 (cf. p. 123).

[DNS12] J. DEMONGEOT, M. NOUAL et S. SENÉ. « Combinatorics of Boolean automata circuits dynamics ». In : *Discrete Applied Mathematics* 160 (2012), p. 398-415. DOI : 10.1016/j.dam.2011.11.005 (cf. p. 99).

[Dha90] D. DHAR. « Self-organized critical state of sandpile automaton models ». In : *Physical Review Letters* 64.14 (1990), p. 1613-1616. DOI : 10.1103/PhysRevLett.64.1613 (cf. p. 120).

[Dha+95] D. DHAR, P. RUELLE, S. SEN et D. N. VERMA. « Algebraic aspects of Abelian sandpile models ». In : *Journal of Physics A* 28.4 (1995), p. 805-831. DOI : 10.1088/0305-4470/28/4/009 (cf. p. 120).

[DF13] R. DOWNEY et M. R. FELLOWS. *Fundamentals of Parameterized Complexity*. Springer-Verlag, 2013. ISBN : 9781447155584. DOI : 10.1007/978-1-4471-5559-1 (cf. p. 23, 73, 85).

[Dox+10] A. K. DOXIÀDIS, C. PAPADIMITRIOU, A. PAPADATOS et A. DI DONNA. *Logicomix*. Vuibert (édition française), 2010. ISBN : 978-2311102321 (cf. p. 10).

[Duf65] R. J. DUFFIN. « Topology of series-parallel networks ». In : *Journal of Mathematical Analysis and Applications* 10 (1965), p. 303-318. DOI : 10.1016/0022-247X(65)90125-3 (cf. p. 107, 109).

[Emd90] P. van EMDE BOAS. « Machine Models and Simulations ». In : *Algorithms and Complexity*. Handbook of Theoretical Computer Science. 1990, p. 1-66. DOI : 10.1016/B978-0-444-88071-0.50006-0 (cf. p. 17).

[Fat14] N. FATÈS. « A guided tour of asynchronous cellular automata ». In : *Journal of Cellular Automata* 9 (2014), p. 387-416. HAL : hal-00908373 (cf. p. 97).

[Fed91] T. FEDER. « Stable Networks and Product Graphs ». Thèse de doct. Stanford University, 1991 (cf. p. 121).

[FO89] P. FLOREEN et P. ORPONEN. « On the computational complexity of analyzing Hopfield nets ». In : *Complex Systems* 3 (1989). URL : complex-systems:v03_i06_a02, p. 577-587 (cf. p. 59, 61, 65).

[FGM12] E. FORMENTI, E. GOLES et B. MARTIN. « Computational Complexity of Avalanches in the Kadanoff Sandpile Model ». In : *Fundamenta Informaticae* 115.1 (2012), p. 107-124. DOI : 10.3233/FI-2012-643 (cf. p. 123).

[FHW80] S. FORTUNE, J. HOPCROFT et J. WYLLIE. « The directed subgraph homeomorphism problem ». In : *Theoretical Computer Science* 10.2 (1980), p. 111-121. DOI : 10.1016/0304-3975(80)90009-2 (cf. p. 96).

[Für07] M. FÜRER. « Faster Integer Multiplication ». In : *Proceedings of STOC'07*. 2007, p. 57-66. DOI : 10.1145/1250790.1250800 (cf. p. 107).

[Gad18] M. GADOULEAU. « On the Rank and Periodic Rank of Finite Dynamical Systems ». In : *Electronic Journal of Combinatorics* 25.3 (2018), P3.48. DOI : 10.37236/7017 (cf. p. 47).

[GG06] A. GAJARDO et E. GOLES. « Crossing information in two-dimensional Sandpiles ». In : *Theoretical Computer Science* 369.1-3 (2006), p. 463-469. DOI : 10.1016/j.tcs.2006.09.022 (cf. p. 122, 123).

[GW83] H. GALPERIN et A. WIGDERSON. « Succinct representations of graphs ». In : *Information and Control* 56.3 (1983), p. 183-198. DOI : 10.1016/S0019-9958(83)80004-7 (cf. p. 38, 82, 83).

[GJ79] M. R. GAREY et D. S. JOHNSON. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Première édition. W. H. Freeman, 1979. ISBN : 0716710455 (cf. p. 11, 19, 25, 67).

[Gas19] W. I. GASARCH. « Guest Column : The Third P =? NP Poll ». In : *ACM SIGACT News* 50.1 (2019). DOI : 10.1145/3319627.3319636 (cf. p. 22).

[Göd36] K. GÖDEL. « Über die Länge von Beweisen (On the length of proofs) ». In : *Ergebnisse eines Mathematischen Kolloquiums* p. 23-24 (1936). Traduction anglaise dans *Kurt Gödel : Collected Works* I, p. 396-399, Oxford University Press, 1986 (cf. p. 9).

[GP18] P. W. GOLDBERG et C. H. PAPADIMITRIOU. « Towards a unified complexity theory of total functions ». In : *Journal of Computer and System Sciences* 94 (2018), p. 167-192. DOI : 10.1016/j.jcss.2017.12.003 (cf. p. 23).

[Gol+21a] E. GOLES, A. ADAMATZKY, P. MONTEALEGRE et M. RÍOS WILSON. « Generating Boolean Functions on Totalistic Automata Networks ». In : *International Journal of Unconventional Computing* 16.4 (2021), p. 343-391. ARXIV : 2107.14799 (cf. p. 122).

[Gol+17a] E. GOLES, D. MALDONADO, P. MONTEALEGRE et N. OLLINGER. « On the Computational Complexity of the Freezing Non-strict Majority Automata ». In : *Proceedings of AUTOMATA'2017*. T. 10248. LNCS. 2017, p. 109-119. DOI : 10.1007/978-3-319-58631-1_9 (cf. p. 122).

[Gol+21b] E. GOLES, D. MALDONADO, P. MONTEALEGRE et M. RÍOS WILSON. « On the Complexity of Asynchronous Freezing Cellular Automata ». In : *Information and Computation* (2021). In press, p. 104764. DOI : 10.1016/j.ic.2021.104764 (cf. p. 122).

[GM90] E. GOLES et S. MARTÍNEZ. *Neural and automata networks : dynamical behavior and applications*. Kluwer Academic Publishers, 1990. ISBN : 9789400905290. DOI : 10.1007/978-94-009-0529-0 (cf. p. 97).

[GM14] E. GOLES et P. MONTEALEGRE. « Computational complexity of threshold automata networks under different updating schemes ». In : *Theoretical Computer Science* 559 (2014), p. 3-19. DOI : 10.1016/j.tcs.2014.09.010 (cf. p. 122).

[GM15] E. GOLES et P. MONTEALEGRE. « The complexity of the majority rule on planar graphs ». In : *Advances in Applied Mathematics* 64 (2015), p. 111-123. DOI : 10.1016/j.aam.2014.11.005 (cf. p. 121).

[Gol+21c] E. GOLES, P. MONTEALEGRE, M. RÍOS WILSON et G. THEYSSIER. « On the Impact of Treewidth in the Computational Complexity of Freezing Dynamics ». In : *Proceedings of CiE'2021*. T. 12813. LNCS. 2021, p. 260-272. DOI : 10.1007/978-3-030-80049-9_24 (cf. p. 122).

[GMT13] E. GOLES, P. MONTEALEGRE-BARBA et I. TODINCA. « The complexity of the bootstraping percolation and other problems ». In : *Theoretical Computer Science* 504 (2013), p. 73-82. DOI : 10.1016/j.tcs.2012.08.001 (cf. p. 122).

[GN10] E. GOLES et M. NOUAL. « Block-sequential update schedules and Boolean automata circuits ». In : *Proceedings of AUTOMATA'2010*. DMTCS. 2010, p. 41-50. DOI : 10.46298/dmtcs.2762 (cf. p. 45, 99).

[GO80] E. GOLES et J. OLIVOS. « Periodic behaviour of generalized threshold functions ». In : *Discrete Mathematics* 30.2 (1980), p. 187-189. DOI : 10.1016/0012-365X(80)90121-1 (cf. p. 121).

[GOT15] E. GOLES, N. OLLINGER et G. THEYSSIER. « Introducing Freezing Cellular Automata ». In : *Proceedings of AUTOMATA'15*. T. 24. TUCS Lecture Notes. 2015, p. 65-73. HAL : hal-01294144 (cf. p. 122).

[Góm15] L. GÓMEZ. « Dynamics of discrete networks with deterministic updates schedules. Application to genetic regulatory networks ». Thèse de doct. Universidad de Concepción, 2015 (cf. p. 98, 101).

[GHR95] R. GREENLAW, H. J. HOOVER et W. L. RUZZO. *Limits to Parallel Computation : P-Completeness Theory*. Oxford University Press, 1995. ISBN : 0195085914 (cf. p. 35, 147, 148).

[GM96] D. GRIFFEATH et C. MOORE. « Life Without Death is P-complete ». In : *Complex Systems* 10.6 (1996). URL : complex-systems:v10_i06_a03, p. 437-447 (cf. p. 122).

[GHP10] X. GU, J. M. HITCHCOCK et A. PAVAN. « Collapsing and Separating Completeness Notions under Average-Case and Worst-Case Hypotheses ». In : *Proceedings of STACS'2010*. T. 5. LIPIcs. 2010, p. 429-440. DOI : 10.4230/LIPIcs.STACS.2010.2462 (cf. p. 21).

[Han63] W. HANF. « Model-theoretic methods in the study of elementary logic ». In : *The Theory of Models*. North-Holland, 1963, p. 132-145. DOI : 10.1016/B978-0-7204-2233-7.50020-4 (cf. p. 77).

[HV19] D. HARVEY et J. VAN DER HOEVEN. « Integer multiplication in time O(n log n) ». HAL : hal-02070778. 2019 (cf. p. 107).

[HP09] M. HERMANN et R. PICHLER. « Complexity of counting the optimal solutions ». In : *Theoretical Computer Science* 410.38 (2009), p. 3814-3825. DOI : 10.1016/j.tcs.2009.05.025 (cf. p. 24, 26, 113).

[JáJ92] J. JÁJÁ. *An Introduction to Parallel Algorithms*. Addison-Wesley, 1992. ISBN : 0201548569 (cf. p. 145, 147).

[Juk12] S. JUKNA. *Boolean Function Complexity*. Springer-Verlag, 2012. ISBN : 9783642245084. DOI : 10.1007/978-3-642-24508-4 (cf. p. 54, 55).

[Kar72] R. M. KARP. « Reducibility among Combinatorial Problems ». In : *Complexity of Computer Computations*. 1972, p. 85-103. DOI : 10.1007/978-1-4684-2001-2_9 (cf. p. 25).

[Kau69] S. A. KAUFFMAN. « Metabolic stability and epigenesis in randomly constructed genetic nets ». In : *Journal of Theoretical Biology* 22 (1969), p. 437-467. DOI : 10.1016/0022-5193(69)90015-0 (cf. p. 87).

[Kau90] S. A. KAUFFMAN. « Requirments for evolvability in complex systems : orderly dynamics and frozen components ». In : *Physica D : Nonlinear Phenomena* 42 (1990), p. 135-152. DOI : 10.1016/0167-2789(90)90071-V (cf. p. 10).

[Kos08] S. KOSUB. « Dichotomy Results for Fixed-Point Existence Problems for Boolean Dynamical Systems ». In : *Mathematics in Computer Science* 1 (2008), p. 487-505. DOI : 10.1007/s11786-007-0038-y (cf. p. 59).

[Lad75] R. E. LADNER. « The Circuit Value Problem is Log Space Complete for P ». In : *ACM SIGACT News* 7.1 (1975), p. 18-20. DOI : 10.1145/990518.990519 (cf. p. 148).

[Lan90] C. G. LANGTON. « Computation at the edge of chaos : Phase transitions and emergent computation ». In : *Physica D : Nonlinear Phenomena* 42.1 (1990), p. 12-37. DOI : 10.1016/0167-2789(90)90064-V (cf. p. 10).

[Le 15] N. LE NOVÈRE. « Quantitative and logic modelling of molecular and gene networks ». In : *Nature Reviews Genetics* 16 (2015), p. 146-158. DOI : 10.1038/nrg3885 (cf. p. 95).

[Lib04] L. LIBKIN. *Elements of Finite Model Theory*. Springer-Verlag, 2004. ISBN : 9783540212027. DOI : 10.1007/978-3-662-07003-1 (cf. p. 73, 77, 84).

[Lin86] N. LINIAL. « Hard Enumeration Problems in Geometry and Combinatorics ». In : *SIAM Journal on Algebraic Discrete Methods* 7.2 (1986), p. 331-335. DOI : 10.1137/0607036 (cf. p. 26, 104).

[LGM98] M. L. LITTMAN, J. GOLDSMITH et M. MUNDHENK. « The Computational Complexity of Probabilistic Planning ». In : *Journal of Artificial Intelligence Research* 9 (1998), p. 1-36. DOI : 10.1613/jair.505 (cf. p. 92).

[LB89] A. LOZANO et J. L. BALCÁZAR. « The Complexity of Graph Problems Fore Succinctly Represented Graphs ». In : *Proceedings of WG'89*. T. 411. LNCS. 1989, p. 277-286. DOI : 10.1007/3-540-52292-1_20 (cf. p. 83).

[MPV14] D. MANDAL, A. PAVAN et R. VENUGOPALAN. « Separating Cook Completeness from Karp-Levin Completeness Under a Worst-Case Hardness Hypothesis ». In : *Proceedings of FSTTCS'2014*. T. 29. LIPIcs. 2014, p. 445-456. DOI : 10.4230/LIPIcs.FSTTCS.2014.445 (cf. p. 21).

[McC04] W. MCCUAIG. « Pólya's permanent problem ». In : *The Electronic Journal of Combinatorics* 11.1 (2004), p. 79. DOI : 10.37236/1832 (cf. p. 89, 96).

[MTA99] L. MENDOZA, D. THIEFFRY et E. R. ALVAREZ-BUYLLA. « Genetic control of flower morphogenesis in Arabidopsis thaliana : a logical analysis ». In : *Bioinformatics* 15.7 (1999), p. 593-606. DOI : 10.1093/bioinformatics/15.7.593 (cf. p. 87).

[MS72] A. R. MEYER et L. J. STOCKMEYER. « The equivalence problem for regular expressions with squaring requires exponential space ». In : *Proceedings of SWAT'72*. 1972, p. 125-129. DOI : 10.1109/SWAT.1972.29 (cf. p. 143).

[Moo90] C. MOORE. « Unpredictability and undecidability in dynamical systems ». In : *Physical Review Letters* 64.20 (1990), p. 2354-2357. DOI : 10.1103/PhysRevLett.64.2354 (cf. p. 10).

[Moo97] C. MOORE. « Majority-Vote Cellular Automata, Ising Dynamics, and P-Completeness ». In : *Journal of Statistical Physics* 88.3 (1997), p. 795-805. DOI : 10.1023/B:JOSS.0000015172.31951.7b (cf. p. 121).

[MM11] C. MOORE et S. MERTENS. *The Nature of Computation*. Oxford University Press, 2011. ISBN : 9780199233212. DOI : 10.1093/acprof:oso/9780199233212.001.0001 (cf. p. 10).

[MN99] C. MOORE et M. NILSSON. « The Computational Complexity of Sandpiles ». In : *Journal of Statistical Physics* 96 (1999), p. 205-224. DOI : 10.1023/A:1004524500416 (cf. p. 122).

[Nou12] M. NOUAL. « Updating Automata Networks ». TEL : tel-00726560. Thèse de doct. École normale supérieure de Lyon, 2012 (cf. p. 47, 97).

[NS11] M. NOUAL et S. SENÉ. *Towards a theory of modelling with Boolean automata networks - I. Theorisation and observations*. 2011. ARXIV : 1111.2077 (cf. p. 102, 109).

[Odi92] P. ODIFREDDI. *Classical Recursion Theory : The Theory of Functions and Sets of Natural Numbers*. Elsevier Science, 1992. ISBN : 9780080886596. DOI : 10.1016/S0049-237X(08)70011-9 (cf. p. 21).

[OT19] N. OLLINGER et G. THEYSSIER. *Freezing, Bounded-Change and Convergent Cellular Automata*. 2019. ARXIV : 1908.06751 (cf. p. 122).

[Orp92] P. ORPONEN. « Neural networks and complexity theory ». In : *Proceedings of MFCS'1992*. T. 629. LNCS. 1992, p. 50-61. DOI : 10.1007/3-540-55808-X_5 (cf. p. 59).

[PSA16] E. PALMA, L. SALINAS et J. ARACENA. « Enumeration and extension of non-equivalent deterministic update schedules in Boolean networks ». In : *Bioinformatics* 32.5 (2016), p. 722-729. DOI : 10.1093/bioinformatics/btv628 (cf. p. 104).

[Pap94] C. H. PAPADIMITRIOU. *Computational complexity*. Addison-Wesley, 1994. ISBN : 0201530821 (cf. p. 37, 92, 96, 99).

[PY84] C. H. PAPADIMITRIOU et M. YANNAKAKIS. « The complexity of facets (and some facets of complexity) ». In : *Journal of Computer and System Sciences* 28.2 (1984), p. 244-259. DOI : 10.1016/0022-0000(84)90068-0 (cf. p. 52).

[PY86] C. H. PAPADIMITRIOU et M. YANNAKAKIS. « A note on succinct representations of graphs ». In : *Information and Control* 71.3 (1986), p. 181-185. DOI : 10.1016/S0019-9958(86)80009-2 (cf. p. 83).

[PID21] J. PARDO, S. IVANOV et F. DELAPLACE. « Sequential reprogramming of biological network fate ». In : *Theoretical Computer Science* 872 (2021), p. 97-116. DOI : 10.1016/j.tcs.2021.03.013 (cf. p. 118).

[Pau+20] L. PAULEVÉ, J. KOLČÁK, T. CHATAIN et S. HAAR. « Reconciling qualitative, abstract, and scalable modeling of biological networks ». In : *Nature Communications* 11 (2020), p. 4256. DOI : 10.1038/s41467-020-18112-5 (cf. p. 12, 68).

[PR10] L. PAULEVÉ et A. RICHARD. « Topological Fixed Points in Boolean Networks ». In : *Comptes Rendus Mathematique* 348.15 (2010), p. 825-828. DOI : 10.1016/j.crma.2010.07.014 (cf. p. 88).

[PS21] L. PAULEVÉ et S. SENÉ. « Non-Deterministic Updates of Boolean Networks ». In : *Proceedings of AUTOMATA'2021*. T. 90. OASIcs. 2021, 10 :1-10 :16. DOI : 10.4230/OASIcs.AUTOMATA.2021.10 (cf. p. 30, 111).

[PS] L. PAULEVÉ et S. SENÉ. « Boolean networks and their dynamics : the impact of updates ». In : *Systems biology modelling and analysis : formal bioinformatics methods and tools*. À paraître (cf. p. 30, 111).

[Per14] S. PERIFEL. *Complexité algorithmique*. Ellipses, 2014. ISBN : 9782729886929 (cf. p. 11, 18, 19, 21, 22, 24, 143, 147).

[Per21] P. PERROTIN. « Simulation entre modèles de calcul naturel et modularité des réseaux d'automate ». TEL : tel-03188307. Aix-Marseille Université, 2021 (cf. p. 116).

[PR21] A. PICARD MARCHETTO et A. RICHARD. « Isomorphic Boolean networks and dense interaction graphs ». In : *Proceedings of AUTOMATA'2021 exploratory papers*. 2021. HAL : hal-03235769 (cf. p. 88, 114).

[Por10] A. E. PORRECA. *On the length of proofs (episode II)*. https://aeporreca.org/blog/length-of-proofs-2, consulté en juin 2021. 2010 (cf. p. 9).

[PB83] J. S. PROVAN et M. O. BALL. « The complexity of counting cuts and of computing the probability that a graph is connected ». In : *SIAM Journal on Computing* 12.4 (1983), p. 777-788. DOI : 10.1137/0212053 (cf. p. 25).

[RRT08] É. REMY, P. RUET et D. THIEFFRY. « Graphic requirements for multistability and attractive cycles in a Boolean dynamical framework ». In : *Advances in Applied Mathematics* 41.3 (2008), p. 335-350. DOI : 10.1016/j.aam.2007.11.003 (cf. p. 47).

[Ric53] H. G. RICE. « Classes of recursively enumerable sets and their decision problems ». In : *Transactions of the AMS* 74.2 (1953), p. 358-366. DOI : 10.1090/s0002-9947-1953-0053041-6 (cf. p. 73).

[Ric10] A. RICHARD. « Negative circuits and sustained oscillations in asynchronous automata networks ». In : *Advances in Applied Mathematics* 44.4 (2010), p. 378-392. DOI : 10.1016/j.aam.2009.11.011 (cf. p. 47).

[RC07] A. RICHARD et J.-P. COMET. « Necessary conditions for multistationarity in discrete dynamical systems ». In : *Discrete Applied Mathematics* 155.18 (2007), p. 2403-2413. DOI : 10.1016/j.dam.2007.04.019 (cf. p. 47).

[Rii07] S. RIIS. « Information Flows, Graphs and their Guessing Numbers ». In : *Electronic Journal of Combinatorics* 14.R44 (2007). DOI : 10.37236/962 (cf. p. 47).

[Río21] M. RÍOS WILSON. « On automata networks dynamics : an approach based on computational complexity theory ». TEL : tel-03264167. Thèse de doct. Aix-Marseille Université, 2021 (cf. p. 53, 122).

[RT21] M. RÍOS WILSON et G. THEYSSIER. *On Symmetry versus Asynchronism : at the Edge of Universality in Automata Networks*. 2021. ARXIV : 2105.08356 (cf. p. 117).

[Rob69] F. ROBERT. « Blocs-H-matrices et convergence des methodes iteratives classiques par blocs ». In : *Linear Algebra and its Applications* 2.2 (1969), p. 223-265. DOI : 10.1016/0024-3795(69)90029-9 (cf. p. 87).

[Rob80] F. ROBERT. « Itérations sur des ensembles finis et automates cellulaires contractants ». In : *Linear Algebra and its Applications* 29 (1980), p. 393-412. DOI : 10.1016/0024-3795(80)90251-7 (cf. p. 46).

[Rob86] F. ROBERT. *Discrete iterations : a metric study*. Springer-Verlag, 1986. ISBN : 9783642648823. DOI : 10.1007/978-3-642-61607-5 (cf. p. 97, 98).

[RST99] N. ROBERTSON, P. SEYMOUR et R. THOMAS. « Permanents, Pfaffian orientations, and even directed circuits ». In : *Annals of Mathematics* 150.3 (1999), p. 929-975. DOI : 10.2307/121059 (cf. p. 89, 96).

[RS04] N. ROBERTSON et P. D. SEYMOUR. « Graph Minors. XX. Wagner's conjecture ». In : *Journal of Combinatorial Theory, Series B* 92.2 (2004), p. 325-357. DOI : 10.1016/j.jctb.2004.08.001 (cf. p. 109).

[SS71] A. SCHÖNHAGE et V. STRASSEN. « Schnelle Multiplikation großer Zahlen ». In : *Computing* 7 (1971), p. 281-292. DOI : 10.1007/BF02242355 (cf. p. 107).

[Sch82] U. SCHÖNING. « A uniform approach to obtain diagonal sets in complexity classes ». In : *Theoretical Computer Science* 18.1 (1982), p. 95-103. DOI : 10.1016/0304-3975(82)90114-1 (cf. p. 22).

[SS02] B. SCHWIKOWSKI et E. SPECKENMEYER. « On enumerating all minimal solutions of feedback problems ». In : *Discrete Applied Mathematics* 117.1 (2002), p. 253-265. DOI : 10.1016/S0166-218X(00)00339-5 (cf. p. 26).

[Sen12] S. SENÉ. « Sur la bio-informatique des réseaux d'automates ». TEL : tel-00759287. Université d'Évry Val d'Essonne, 2012 (cf. p. 47, 63, 64, 97).

[Sip05] M. SIPSER. *Introduction to the Theory of Computation*. Seconde édition. Course Technology Inc, 2005. ISBN : 9780619217648 (cf. p. 17).

[Sto76] L. J. STOCKMEYER. « The polynomial-time hierarchy ». In : *Theoretical Computer Science* 3.1 (1976), p. 1-22. DOI : 10.1016/0304-3975(76)90061-X (cf. p. 143, 144).

[TT95] D. THIEFFRY et R. THOMAS. « Dynamical behaviour of biological regulatory networks—II. Immunity control in bacteriophage lambda. » In : *Bulletin of Mathematical Biology* 57 (1995), p. 277-297. DOI : 10.1007/BF02460619 (cf. p. 87).

[Tho73] R. THOMAS. « Boolean formalization of genetic control circuits ». In : *Journal of Theoretical Biology* 42.3 (1973), p. 563-585. DOI : 10.1016/0022-5193(73)90247-6 (cf. p. 87).

[Tho81] R. THOMAS. « On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations ». In : *Numerical methods in the study of critical phenomena*. 1981, p. 180-193. DOI : 10.1007/978-3-642-81703-8_24 (cf. p. 47).

[TK01] R. THOMAS et M. KAUFMAN. « Multistationarity, the basis of cell differentiation and memory. II. Logical analysis of regulatory networks in terms of feedback circuits ». In : *Chaos : An Interdisciplinary Journal of Nonlinear Science* 11.1 (2001), p. 180-195. DOI : 10.1063/1.1349893 (cf. p. 95).

[Tho85] C. THOMASSEN. « Even Cycles in Directed Graphs ». In : *European Journal of Combinatorics* 6.1 (1985), p. 85-89. DOI : 10.1016/S0195-6698(85)80025-1 (cf. p. 96).

[Tor91] J. TORÁN. « Complexity Classes Defined by Counting Quantifiers ». In : *Journale of the ACM* 38.3 (1991), p. 752-773. DOI : 10.1145/116825.116858 (cf. p. 65, 144).

[VTL79] J. VALDES, R. E. TARJAN et E. L. LAWLER. « The recognition of series parallel digraphs ». In : *Proceedings of STOC'79*. 1979, p. 1-12. DOI : 10.1145/800135.804393 (cf. p. 107, 108).

[Val76] L. G. VALIANT. « Relative complexity of checking and evaluating ». In : *Information Processing Letters* 5.1 (1976), p. 20-23. DOI : 10.1016/0020-0190(76)90097-1 (cf. p. 27).

[Val79] L. G. VALIANT. « The Complexity of Enumeration and Reliability Problems ». In : *SIAM Journal on Computing* 8.3 (1979), p. 410-421. DOI : 10.1137/0208032 (cf. p. 25).

[VV86] L. G. VALIANT et V. V. VAZIRANI. « NP is as easy as detecting unique solutions ». In : *Theoretical Computer Science* 47 (1986), p. 85-93. DOI : 10.1016/0304-3975(86)90135-0 (cf. p. 26).

[Vol91] H. VOLLMER. « The gap-language-technique revisited ». In : *Proceedings of CSL'90*. T. 533. LNCS. 1991, p. 389-399. DOI : 10.1007/3-540-54487-9_72 (cf. p. 22).

[Wag86] K. W. WAGNER. « The complexity of combinatorial problems with succinct input representation ». In : *Acta Informatica* 23 (1986), p. 325-356. DOI : 10.1007/BF00289117 (cf. p. 65, 144).

[Wag90] K. W. WAGNER. « Bounded Query Classes ». In : *SIAM Journal on Computing* 19.5 (1990), p. 833-846. DOI : 10.1137/0219058 (cf. p. 143).

[Wra76] C. WRATHALL. « Complete sets and the polynomial-time hierarchy ». In : *Theoretical Computer Science* 3.1 (1976), p. 23-33. DOI : 10.1016/0304-3975(76)90062-1 (cf. p. 143, 144).

[Yan91] H. YANG. « An NC algorithm for the general planar monotone circuit value problem ». In : *Proceedings of SPDP'91*. 1991, p. 196-203. DOI : 10.1109/SPDP.1991.218279 (cf. p. 122, 148).

[Zoo] Complexity ZOO. https://complexityzoo.net (cf. p. 19).

A. Solution unique ∃! et PH

La classe US est définie comme l'ensemble des problèmes que l'on peut résoudre en temps polynomial, sur une machine non déterministe qui accepte si et seulement si *exactement une* branche d'exécution accepte. Il s'agit d'une modification du modèle de calcul non déterministe, qui sert intuitivement à décider l'existence d'une solution (certificat) unique à un problème dans NP.

Au chapitre 3.4 sur l'encodage des réseaux d'automates, nous avons vu le problème de décider si un réseau non déterministe, donné par les circuits de ses relations locales, est semi-déterministe (chaque configuration possède exactement une image). Le théorème 3.4 montre l'équivalence (pour les réductions $\leq_{\mathrm{m}}^{\mathsf{P}}$) avec le problème suivant. Nous noterons $\varphi(x) = 1$ lorsque la formule φ est satisfaite par la valuation x, et $\varphi(x) = 0$ sinon.

Pour tout il existe un unique-SAT (∀∃!-**SAT**) *Entrée :* une formule propositionnelle φ à n variables, et $s \in [n]$. *Question :* est-ce que $\forall x \in \{0,1\}^s : \exists! y \in \{0,1\}^{n-s} : \varphi(x,y) = 1$?

Ce problème est intuitivement dans $\mathsf{coNP}^{\mathsf{US}}$: pour tout x, appeler l'oracle pour vérifier qu'il existe un unique y tel que $\varphi(x,y) = 1$. Dans cette annexe, nous allons remarquer qu'il est également dans $\Pi_2^{\mathsf{P}} = \mathsf{coNP}^{\mathsf{NP}}$, et définir le problème symétrique ∃∃!-**SAT**. En utilisant le fait que $\mathsf{coNP} \subseteq \mathsf{US}$, nous allons démontrer que ce dernier problème est complet pour $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$. Nous discuterons ensuite de la $\mathsf{coNP}^{\mathsf{NP}}$-complétude de ∀∃!-**SAT** : on ne sait pas si $\mathsf{NP} \subseteq \mathsf{US}$ ou $\mathsf{NP} \not\subseteq \mathsf{US}$, d'où l'asymétrie des déductions que nous exposons. Toutes les notions de difficulté sont relatives aux réductions many-one polynomiales $\leq_{\mathrm{m}}^{\mathsf{P}}$.

A.1. $\mathsf{NP}^{\mathsf{NP}} = \mathsf{NP}^{\mathsf{US}}$ et ∃∃!-SAT est $\mathsf{NP}^{\mathsf{NP}}$-complet

On commence par définir le problème symétrique de ∀∃!-**SAT**.

il existe il existe un unique-SAT (∃∃!-**SAT**) *Entrée :* une formule propositionnelle φ à n variables, et $s \in [n]$. *Question :* est-ce que $\exists x \in \{0,1\}^s : \exists! y \in \{0,1\}^{n-s} : \varphi(x,y) = 1$?

Les problèmes ∀∃!-**SAT** et ∃∃!-**SAT** ne sont pas complémentaires l'un de l'autre, en revanche toute instance positive de ∀∃!-**SAT** est une instance positive de ∃∃!-**SAT**.

Le problème ∀∃!-**SAT** serait complémentaire de ∃¬∃!-**SAT**, avec ¬∃! le quantificateur « il n'existe pas un·e unique », et le problème ∃∃!-**SAT** complémentaire de ∀¬∃!-**SAT**. Néanmoins, on peut exprimer ∃! comme une conjonction de la forme ∃ ∧ ∀ :

$$\exists! y : \psi(y) \quad \equiv \quad \big[\exists y : \psi(y)\big] \wedge \big[\forall y', y'' : \neg\psi(y') \vee \neg\psi(y'') \vee (y' = y'')\big],$$

dont on déduit les bornes supérieures de complexité suivantes.

Lemme A.1. ∀∃!-**SAT** $\in \mathsf{coNP}^{\mathsf{NP}}$ *et* ∃∃!-**SAT** $\in \mathsf{NP}^{\mathsf{NP}}$.

Démonstration. On reformule $\forall x : \exists! y : \varphi(x, y) = 1$ en :

$$\forall x, y', y'' : \exists y : \big[\varphi(x, y) = 1\big] \wedge \big[\neg\varphi(x, y') \vee \neg\varphi(x, y'') \vee (y' = y'')\big],$$

et $\exists x : \exists! y : \varphi(x, y) = 1$ en :

$$\exists x, y : \forall y', y'' : \big[\varphi(x, y) = 1\big] \wedge \big[\neg\varphi(x, y') \vee \neg\varphi(x, y'') \vee (y' = y'')\big],$$

qui constituent des réductions vers ∀∃-**SAT** et ∃∀-**SAT**, respectivement. □

Nadia Creignou (LIS, Marseille) m'a fait remarquer que ces reformulations mènent au résultat plus général suivant.

Théorème A.2. $\mathsf{NP}^{\mathsf{NP}} = \mathsf{NP}^{\mathsf{US}}$, *et donc* $\mathsf{coNP}^{\mathsf{NP}} = \mathsf{coNP}^{\mathsf{US}}$.

Démonstration. Pour $\mathsf{NP}^{\mathsf{US}} \subseteq \mathsf{NP}^{\mathsf{NP}}$, on transforme chaque appel à l'oracle US pour ∃!-**SAT** (problème US-complet, théorème 2.4) en deux appels à l'oracle NP pour **SAT** :

1. est-ce que φ possède au moins une solution ?
2. est-ce que φ possède au moins deux solutions ? avec $\exists x y : \varphi(x) \wedge \varphi(y) \wedge (x \neq y)$.

La réponse de l'appel à l'oracle US sur φ est « oui », si et seulement si les réponses des deux appels à l'oracle NP sont respectivement « oui » et « non ». L'inclusion réciproque est une conséquence de $\mathsf{coNP} \subseteq \mathsf{US}$ (théorème 2.4 [a]). □

On peut utiliser $\mathsf{coNP} \subseteq \mathsf{US}$ pour montrer la $\mathsf{NP}^{\mathsf{NP}}$-difficulté de ∃∃!-**SAT**.

Théorème A.3. ∃∃!-**SAT** *est* $\mathsf{NP}^{\mathsf{NP}}$*-complet, et donc* $\mathsf{NP}^{\mathsf{US}}$*-complet.*

Démonstration. La borne supérieure est donnée par le lemme A.1, et pour la borne inférieure on réduit depuis ∃∀-**SAT**. Pour séparer l'ensemble des variables, on aura $s = n$ dans les deux formules. Soit φ une instance sur les variables $x = x_1, \ldots, x_n$ et $y = y_1, \ldots, y_m$, on construit ψ en utilisant la réflexion donnée en introduction de [BG82], avec une variable booléenne y_0 supplémentaire sous le quantificateur ∃! :

$$\psi(x, y_0, y) = (y_0 \wedge y_1 \wedge \cdots \wedge y_m) \vee (\neg y_0 \wedge \neg\varphi(x, y)).$$

a. Étant donnée φ sur n variables pour **UNSAT**, on ajoute une variable x_{n+1} et l'on construit la formule $\varphi(x) \vee (x_1 = x_2 = \cdots = x_{n+1} = 0)$. Si φ n'est pas satisfaisable, alors on a une unique solution 0^{n+1}. Si φ est satisfaite par y, alors en choisissant $y_{n+1} = 0$ ou 1 on obtient deux solutions.

Si $\exists\tilde{x} : \forall y : \varphi(\tilde{x}, y) = 1$;alors $\psi(\tilde{x}, y_0, y) \equiv (y_0 \wedge y_1 \wedge \cdots \wedge y_m)$ est satisfaite uniquement par $y_0 = 1$ et $y = 1^m$ (c'est-à-dire $y_0 = y_1 = \cdots = y_m = 1$). Si $\forall x : \exists\tilde{y} : \varphi(x, \tilde{y}) = 0$, alors, pour tout x, on a $\psi(x, 0, \tilde{y}) = \psi(x, 1, 1^m) = 1$ donc la solution sous le quantificateur $\exists!$ n'est pas unique. □

On a bien sûr que l'hypothèse $\mathsf{NP} \subseteq \mathsf{US}$ (question ouverte depuis [BG82]) impliquerait que $\forall\exists!$**-SAT** est $\mathsf{coNP}^{\mathsf{NP}}$-complet, avec une construction analogue à la preuve du théorème A.3. La réciproque de la phrase qui précède n'est pas évidente.

Question ouverte A.4. *Est-ce que* $\forall\exists!$**-SAT** *est* $\mathsf{coNP}^{\mathsf{NP}}$*-complet ?*

Il n'est pas aisé d'en énoncer des conséquences. Les résultats sur les machines de Turing avec oracles sont parfois d'une subtilité surprenante, car il s'agit de considérations qui touchent aux rouages internes du modèle de calcul. On pense bien sûr au théorème de Baker-Gill-Solovay [Per14, théorème 7-AA page 183] (il existe deux oracles A et B tels que $\mathsf{P}^A = \mathsf{NP}^A$ et $\mathsf{P}^B \neq \mathsf{NP}^B$), mais également aux résultats de Blass et Gurevich sur US [BG82] (il existe deux oracles A et B tels que $\mathsf{NP}^A \not\subseteq \mathsf{US}^A$, et $\mathsf{NP}^B \subseteq \mathsf{US}^B$ mais $\mathsf{NP}^B \neq \mathsf{coNP}^B$). Soulignons toutefois que ces subtilité (auxquelles ont peut adjoindre les résultats de Beigel, Burhman et Fortnow sur DP [BBF98]) apparaissent pour des oracles « exotiques », qui sont construits pour avoir ces propriétés, alors que la question ouverte A.4 ne concerne qu'un problème particulier.

Pour démonter la $\mathsf{coNP}^{\mathsf{NP}}$-difficulté de $\forall\exists!$**-SAT**, il n'est pas non plus évident d'adapter la relativisation de la construction du théorème de Cook-Levin comme dans [MS72] qui démontre la $\mathsf{coNP}^{\mathsf{NP}}$-difficulté de $\forall\exists$**-SAT**. Cette dernière argumentation repose sur le fait qu'une réponse négative de l'oracle existentiel ($\neg\exists$) peut être décidée par le quantificateur $\forall$ de la machine non déterministe elle-même (ceci est explicité par le choix des variables quantifiées existentiellement et universellement dans la formule issue de la réduction vers $\forall\exists$**-SAT**). Or, dans le cas présent, nous n'avons pas de correspondance immédiate entre $\neg\exists!$ et $\forall$. Les équivalences vues précédemment offrent toutefois un espoir car le $\exists!$ peut être transformé en $\exists \wedge \forall$, mais cela requiert d'« ouvrir » les formules et de jouer avec les détails d'une preuve déjà très technique. Une alternative serait d'adapter d'autres formulations des résultats de complétude des problèmes de satisfaction canoniques pour les niveaux de PH, comme dans [Sto76; Wra76] et [Per14, Propositions 8-F et 8-M pages 196 et 200].

Au théorème A.2, nous avons plus précisément démontré que $\mathsf{NP}^{\mathsf{US}[k]} \subseteq \mathsf{NP}^{\mathsf{NP}[2k]}$, car k appels à l'oracle US sont transformés en $2k$ appels à l'oracle NP. Le nombre d'appels à l'oracle pourrait être une clé pour cerner la difficulté de $\forall\exists!$**-SAT**, et les classes définies avec un nombre borné de requêtes à l'oracle (en anglais, *bounded query classes*) une source d'inspiration [Wag90 ; Bei91]. D'après la $\mathsf{NP}^{\mathsf{US}}$-complétude de $\exists\exists!$**-SAT**, on a l'intuition qu'il suffit d'un seul appel à l'oracle US (correspondant à la quantification $\exists!$) par branche non déterministe (correspondant à la quantification $\exists$). Pour autant, peut-on construire Z solution de l'équation suivante pour X, Y données ?

$$\big[\exists! x : X(x)\big] \vee \big[\exists! y : Y(y)\big] \quad \equiv \quad \exists! z : Z(z)$$

Dans une perspective plus large, peut-on enrichir la hiérarchie polynomiale PH en ajoutant le quantificateur ∃! « il existe un·e unique » ? Et son complémentaire ¬∃! « il n'existe pas un·e unique » (à ne pas confondre avec ∀! « pour tou·te·s sauf un·e », qui correspond à prendre la négation d'une formule propositionnelle liée à un quantificateur ∃!) ? La hiérarchie polynomiale a été introduite vers 1976 [Sto76 ; Wra76], et le quantificateur de comptage ajouté une dizaine d'années plus tard [Wag86 ; Tor91 ; AW93]. On pourrait d'un côté chercher à caractériser la complexité des problèmes de la forme $Q_1 x_1 : \ldots : Q_k x_k : \varphi(x_1, \ldots, x_k)$ où $Q_i \in \{\exists, \forall, \exists!, \neg\exists!\}$ pour $i \in [k]$, et d'un autre côté chercher des problèmes complets pour les modèles de calcul définis inductivement à partir des machines NP, coNP, US, coUS, pour l'opérateur oracle restreint à un certain nombre d'appels.

B. Classes au dessous de P

B.1. Modèle parallèle (PRAM)

Un modèle de calcul parallèle est une collection de *processeurs* (unités de calcul séquentiel) interconnectés de façon à coordonner leurs exécutions et échanger des données [JáJ92]. Nous utiliserons le formalisme des *Parallel Random Access Memory* (**PRAM**). Les instructions sont celles du modèle **RAM** (affectation, bloc d'instructions, conditionnelle, boucles), chaque processeur possède un *identifiant unique*, le mode opératoire est *synchrone* (horloge commune) [a], chaque processeur a accès à une mémoire locale, et les processeurs ont également accès à une mémoire partagée en *lecture concurrente* et *écriture exclusive* (*concurrent read exclusive write*, **CREW**) [b]. Le *temps* d'exécution d'un algorithme parallèle correspond au nombre d'étapes requises par l'algorithme, lorsqu'à chaque étape plusieurs instructions concurrentes peuvent être réalisées par des processeurs différents (le *travail* d'un algorithme parallèle correspond à la somme des temps d'exécution des processeurs).

Afin d'illustrer le modèle **CREW PRAM**, nous ébauchons ci-dessous deux algorithmes parallèles. Le premier calcule le maximum d'un ensemble de n valeurs en temps $\mathcal{O}(\log n)$ sur $\mathcal{O}(n)$ processeurs, et le second les composantes connexes d'un graphe non orienté en temps $\mathcal{O}((\log n)^2)$ sur $\mathcal{O}(n^2)$ processeurs. Ce dernier algorithme a été utilisé pour prédire la dynamique de modèles à l'interface entre piles de sable et réseaux d'automates booléens, dont nous discutons en conclusion. Il est issu du livre de JáJá [JáJ92, Algorithme 5.1 page 209] et exploite une technique nommée *pointer jumping* [JáJ92, Algorithme 2.4 page 52].

Algorithme de calcul du maximum
Complexité : temps $\mathcal{O}(\log n)$ sur $\mathcal{O}(n)$ processeurs [c].
Entrée : un tableau de n valeurs $A : [n] \to \mathbb{N}$.
Sortie : $i \in [n]$ tel que $\forall j \in [n] : A(j) \leq A(i)$.
Début. Il s'agit simplement de construire un arbre binaire équilibré à n feuilles indicées par $[n]$, dont les nœuds internes propagent des feuilles à la racine l'indice de la plus grande valeur dans A. En calculant en parallèle les nœuds de même hauteur

a. Dans le cas d'exécutions concurrentes *asynchrones*, chaque processeur possède sa propre horloge et c'est au programmeur d'implémenter des points de synchronisation si nécessaire.

b. Dans le cas d'*écritures concurrentes* à une même adresse, cela n'est possible dans le modèle **CRCW** que si les processeurs concernés écrivent la même valeur.

c. Dans le modèle **CRCW PRAM**, le calcul du maximum peut être réalisé en temps $\mathcal{O}(1)$ sur $\mathcal{O}(n^2)$ processeurs [JáJ92, Algorithme 2.8 page 72].

dans l'arbre (avec environ $\frac{n}{2}$ processeurs pour la première étape), on obtient l'indice de l'élément maximum à la racine en $\mathcal{O}(\log n)$ étapes. **Fin.**

Algorithme de calcul des composantes connexes
Complexité : temps $\mathcal{O}((\log n)^2)$ sur $\mathcal{O}(n^2)$ processeurs.
Entrée : matrice d'adjacence A d'un graphe non orienté $G = (V, E)$ avec $V = [n]$.
Sortie : $D : V \to V$ avec $D(v)$ le plus petit sommet dans la composante connexe de v.
L'idée consiste à décomposer le graphe G en pseudo forêts orientées (qui correspondent à des sous-ensembles des composantes connexes), les réduire en parallèle à des étoiles enracinées (c'est-à-dire identifier le plus petit sommet), et recommencer sur le graphe des étoiles enracinées (qui vont exponentiellement rapidement inclure les connexions manquantes).
Début.

1. Pour chaque sommet v de G, calculer $C(v) = \min\{u \in V \mid A(u,v) = 1 \text{ et } u \neq v\}$ le plus petit sommet voisin de v, ou $C(v) = v$ si v n'a pas de voisin (le calcul du minimum peut être effectué en temps $\mathcal{O}(\log n)$ grâce à l'algorithme précédent). Le tableau $C : V \to V$ représente une *pseudo forêt orientée* sur V, où chaque sommet a degré sortant 1.
2. Réduire chaque *pseudo arbre orienté* de C à une *étoile enracinée orientée*, en temps $\mathcal{O}(\log n)$ grâce au *pointer jumping* : en parallèle, tant que $C(v) \neq C(C(v))$ on assigne $C(v) = C(C(v))$. La distance d'un nœud à sa racine diminue exponentiellement, et on obtient $C : V \to V$ où tout sommet pointe vers sa racine.
3. Préparer l'itération suivante :
   - Pour tout sommet v non racine (tel que $C(v) \neq v$), nous utiliserons en étape 4 la valeur $C(v)$.
   - Pour tout sommet *racine isolée* (tel que $C(v) = v$ et $\nexists u \neq v : C(u) = v$), définir $D(v) = v$.
   - Construire le nouveau graphe $G' = (V', E')$ sur une copie des sommets *racines non isolées* $V' = \{v \in V \mid C(v) = v \text{ et } \exists u \neq v : C(u) = v\}$, avec les adjacences données par $E' = \{(u,v) \mid \exists u', v' : C(u') = u \text{ et } C(v') = v \text{ et } (u', v') \in E\}$ [d] (les racines représentent leur étoile dans le graphe G, et on se souviendra de la correspondance entre nouveau sommet de G' et sommet de G dans $S : V' \to V$).

   Si $V' \neq \emptyset$ alors recommencer l'étape 1 avec G', sinon continuer à l'étape 4.
4. Propager les informations dans l'ordre inverse des graphes construits à chaque itération, grâce à C qui indique à chaque sommet quelle est sa racine (passer d'un graphe au précédent avec S). Terminer avec le graphe d'entrée G, en écrivant les résultats dans D.

Fin.
Le nombre de sommets du graphe diminue d'un facteur au moins deux à chaque

d. Dans le modèle **CRCW PRAM** la matrice d'adjacence A' de G' est construite avec écriture concurrente des 1. Dans le modèle **CREW PRAM**, $\mathcal{O}(n^2)$ processeurs écrivent en parallèle dans des variables distinctes les adjacences entre étoiles enracinées, puis en parallèle pour chacun des $\mathcal{O}(n^2)$ éléments de A' on calcule en $\mathcal{O}(\log n)$ la disjonction des adjacences données par les variables concernées.

Classe	Description (algorithme, pour $k \in \mathbb{N}$ fixé)
NC	Parallèle, en temps $\mathcal{O}((\log n)^k)$, $\mathsf{NC} = \bigcup_{i\in\mathbb{N}} \mathsf{NC}^i = \bigcup_{i\in\mathbb{N}} \mathsf{PRAM}^i$
PRAM^i avec $i \in \mathbb{N}_+$	Parallèle (**CREW**), en temps $\mathcal{O}((\log n)^i)$ sur $\mathcal{O}(n^k)$ processeurs
NC^i avec $i \in \mathbb{N}_+$	Famille L-uniforme de circuits de taille $\mathcal{O}(n^k)$ et profondeur $\mathcal{O}((\log n)^i)$
AC^0	Famille de circuits de taille polynomiale et profondeur $\mathcal{O}(1)$, avec des portes $\wedge$ et $\vee$ d'arité arbitraire

Table B.1. – Classes de complexité, et leur définition par les modèles de calcul et algorithmes résolvant les problèmes appartenant à chaque classe.

itération (la boucle est donc répétée $\mathcal{O}(\log n)$ fois), car on ne construit qu'un sommet par étoile non isolée (qui contiennent chacune au moins deux sommets). Voir [JáJ92, figure 5.4 page 211] pour une illustration du déroulement de l'algorithme.

B.2. NC, P-complétude, et variantes de CVP

Au dessous de P, c'est le parallélisme qui est au centre des considérations, avec les classes présentées dans la table B.1. Nous renvoyons le lecteur vers le classique de Greenlaw, Hoover et Ruzzo [GHR95] pour de plus amples développements.

La notion de P-complétude pour les réductions $\leq_m^{\mathsf{NC}}$ ou $\leq_m^{\mathsf{L}}$ permet de caractériser des problèmes comme étant « intrinsèquement séquentiels », c'est-à-dire qui n'ont pas d'algorithme parallèle significativement plus efficace que le meilleur algorithme séquentiel (sous l'hypothèse $\mathsf{NC} \neq \mathsf{P}$). À l'opposé, les problèmes dans NC sont dit « efficacement parallélisables » [GHR95]. La classe NC est « robuste », dans le sens où elle peut être définie (voir table B.1) en passant par le modèle de calcul des circuits (NC^i) ou par le modèle des **CREW PRAM** (PRAM^i) ; elle capture donc un aspet fondamental de la complexité algorithmique des problèmes.

Remarquons que l'on a $\mathsf{NC}^i \subseteq \mathsf{PRAM}^i \subseteq \mathsf{NC}^{i+1}$ [JáJ92, page 553]. Pour $\mathsf{PRAM}^i \subseteq \mathsf{NC}^{i+1}$, puisque les circuits n'ont pas de mémoire, on simule les étapes de calcul du programme séquentiellement en déroulant les boucles (les sorties d'une étape sont les entrées de l'étape suivante). Il y a $\mathcal{O}((\log n)^i)$ étapes de calcul, et chacune est simulée avec un circuit de profondeur $\mathcal{O}(\log n)$ car les circuits travaillent en binaire (voir [Per14, figure 5.2 page 139] pour une illustration). Pour $\mathsf{NC}^i \subseteq \mathsf{PRAM}^i$, il faut d'abord, étant donné n, utiliser l'uniformité pour calculer le circuit (voir [GHR95, lemme 2.4.1 page 34], car dans NC^i on a une famille de circuits, alors que dans PRAM^i on a un seul programme). Ensuite, on simule simplement le circuit en parallèle, avec un processeur par porte.

Le problème P-complet canonique consiste à évaluer un circuit.

Circuit Value Problem (**CVP**) *Entrée :* un circuit booléen C, une entrée e pour C, et une porte p de C. *Question :* sur l'entrée e, la porte p est-elle évaluée à 1 dans C ?

Notons quelques restrictions qui intersectent nos intérêts :

- *Monotone CVP* (**MCVP**) sur les circuits monotones (portes $\wedge, \vee$),
- *Planar CVP* (**PCVP**) sur les circuits planaires,
- *Monotone Planar CVP* (**MPCVP**) sur les circuits monotones et planaires.

Théorème B.1 ([Lad75 ; Yan91 ; GHR95])**.** **CVP**, **MCVP** *et* **PCVP** *sont* P*-complets pour les réductions* $\leq_m^{\mathsf{L}}$*, et* **MPCVP** *est dans* NC.

Le théorème B.1 nous dit que l'on peut ajouter la restriction monotone ou la restriction planaire, mais pas les deux car alors le problème tombe dans NC. Par commodité, nous pourrons ajouter les restrictions additionnelles suivantes à **CVP**, **MCVP** et **PCVP** qui restent P-complets :

- degré entrant et sortant au plus 2,
- circuit en couche (en anglais, *layered*) : les sommets du circuit peuvent être partitionnés en couches, avec les entrées dans la première couche et les sorties dans la dernière couche, de façon à ce qu'un sommet de la couche i ait tous ses voisins entrants dans la couche $i-1$; cela correspond intuitivement à dire que l'information s'écoule couche après couche dans le circuit,
- les sommets peuvent être organisés sur une grille (avec des coordonnées $(x, y) \in \mathbb{Z}^2$ pour chaque sommet et des arcs parallèles aux axes $x = 0$ et $y = 0$).

Cela correspond au problème **SAM2CVP** de [GHR95].

Dans **CVP**, on peut également fixer $e = 0^n$ et nommer cette variante **CVP à entrée fixée**, le problème reste P-complet. Pour une discussion sur d'autres variantes de **CVP** qui restent P-complètes, nous redirigeons le lecteur vers [GHR95, section 6.2 page 75].