Une approche basée sur l'ASP pour détecter des attracteurs dans les réseaux booléens circulaires

Tarek Khaled **Belaïd Benhamou**

Aix Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France
{tarek.khaled,belaid.benhamou}@univ-amu.fr

Résumé

En biologie, les réseaux booléens sont traditionnellement utilisés pour représenter et simuler les réseaux de régulation de gènes. Les attracteurs font l'objet d'une attention particulière dans l'analyse de la dynamique d'un réseau Booléen. Ils correspondent à des états stables et à des cycles stables, qui jouent un rôle crucial dans les systèmes biologiques.Dans ce travail, nous étudions une nouvelle représentation de la dynamique des réseaux Booléens qui est basée sur une nouvelle sémantique utilisée dans la programmation par ensemble réponse (Answer Set Programming, ASP). Notre méthode est axé sur l'énumération de tous les attracteurs de réseaux Booléens asynchrones pour les graphes d'interaction circulaires. Nous montrons que la sémantique utilisée permet de concevoir une nouvelle approche pour calculer de manière exhaustive les cycles stables et les états stables de tels réseaux. L'énumération de tous les attracteurs et la distinction entre les deux types d'attracteurs est une étape notable pour mieux comprendre certains aspects critiques en biologie. Nous avons appliqué et évalué l'approche proposée sur des réseaux booléens générés aléatoirement. Les résultats obtenus mettent en évidence les avantages de cette approche et correspondent à certains résultats démontrés en biologie.

Abstract

In biology, Boolean networks are conventionally used to represent and simulate gene regulatory networks. The attractors are the subject of special attention in analyzing the dynamics of a Boolean network. They correspond to stable states and stable cycles, which play a crucial role in biological systems. In this work, we study a new representation of the dynamics of Boolean networks that are based on a new semantics used in answer set programming (ASP). Our work is based on the enumeration of all the attractors of asynchronous Boolean networks having interaction graphs which are circuits. We show that the used semantics allows to design a new approach for computing exhaustively both the stable cycles and the stable states of such networks. The enumeration of all the attractors and the distinction between both types of attractors is a significant step to better understand some critical aspects of biology. We applied and evaluated the proposed approach on randomly generated Boolean networks and the obtained results highlight the benefits of this approach, and match with some demonstrated results in biology.

1 Introduction

Un réseau de régulation des gènes est un ensemble de gènes qui interagissent les uns avec les autres. Chaque gène contient l'information qui détermine sa future fonction. Lorsqu'un gène est actif, un processus appelé transcription a lieu, produisant une copie de l'acide ribonucléique (ARN) de l'information génétique. Cette portion d'ARN peut alors régir la production d'une protéine. Un réseau de régulation des gènes est un système biologique spécifique qui représente la manière dont les protéines/gènes interagissent dans une cellule pour la survie, la reproduction ou la mort de la cellule. Plusieurs représentations peuvent être utilisées pour modéliser les réseaux de régulation des gènes [2]. Des représentations quantitatives peuvent être utilisées. Cependant, ces approches nécessitent des paramètres numériques qui doivent d'abord être mesurés ou calculés, et qui sont généralement difficiles à obtenir. L'autre solution consiste à utiliser des représentations qualitatives. Ce choix ne nécessite pas la connaissance des paramètres indispensables aux représentations quantitatives[25, 7, 17, 14]. Les approches qualitatives donnent généralement moins de précision sur la dynamique des systèmes de régulation que les approches quantitatives. Néanmoins, elles permettent de capturer les propriétés les plus importantes, telles que les attracteurs.

Les réseaux booléens ont été proposés comme modèle mathématique pour les réseaux génétiques [9, 10]. Les réseaux booléens offrent un outil qualitative simple et puissant

pour modéliser les réseaux génétiques [22]. Ils transforment la représentation des interactions génétiques en règles logiques qualitatives. Les réseaux booléens ont une structure constituée d'entités qui correspondent aux gènes ou aux protéines. Chaque gène/protéine prend la valeur *on* ou *off*, ce qui signifie que le gène/protéine est ou n'est pas exprimé. Deux gènes sont connectés si l'expression de l'un d'eux module l'expression de l'autre par activation ou inhibition. D'un point de vue logique, un système biologique peut être considéré comme un ensemble d'éléments en interaction.

Malgré la modélisation simplifiée et qualitative de la réalité biologique, il a été démontré que les réseaux booléens expriment et capturent correctement la dynamique des réseaux de régulation des gènes qui sont principalement caractérisés par leurs attracteurs. Les attracteurs sont les ensembles d'états vers lesquels le système converge. Un attracteur correspond généralement aux caractéristiques/phénotypes observés d'un système biologique [10]. En effet, si un réseau contrôle un phénomène de spécialisation, alors la cellule se spécialise en fonction de l'attracteur vers lequel évolue son réseau booléen sous-jacent. Autrement dit, la cellule acquiert un phénotype particulier ou une fonction physiologique spécifique pour l'attracteur vers lequel converge le réseau booléen. Il est alors essentiel d'identifier les attracteurs des réseaux booléens pour étudier leur dynamique.

Notre objectif dans ce travail est de développer une approche exhaustive pour analyser la dynamique des réseaux booléens et capturer tous les états stables possibles et énumérer tous les cycles stables. Nous nous concentrons ici sur les réseaux de gènes qui sont représentés par des graphes d'interactions circulaire. Nous considérons le mode de mise à jour asynchrone et utilisons le cadre ASP pour représenter et résoudre les problèmes mentionnés ci-dessus. L'ASP [19] est un paradigme déclaratif de résolution de problèmes, issu de la programmation logique et du raisonnement non monotone. Plusieurs solveurs ASP [18, 4, 24] sont disponibles. Ils fournissent une variété de structures et de fonctionnalités pour la modélisation de problèmes [3], aidant l'utilisateur à exprimer les problèmes plus naturellement et à les résoudre efficacement. Dans ce travail, nous utilisons la méthode introduite dans [16, 11, 15] pour traiter les réseaux de gènes. Cette méthode s'appuie sur un processus d'énumération booléen défini pour le paradigme ASP conformément à la sémantique introduite dans [1]. Cette sémantique garantit pour tout programme logique consistant, l'existence d'extensions ou de modèles expliquant le programme considéré. Certaines de ces extensions correspondent à des modèles stables et les autres à des extra-modèles. Les extra-modèles correspondent aux extra-extensions qui ne sont pas capturées par la sémantique des modèles stables [5]. Nous observerons que les extra-modèles jouent un rôle essentiel dans l'approche du codage des attracteurs des cycles stables des réseaux booléens. La représentation des graphes d'interaction comme des programmes logiques interprétés dans la sémantique introduite dans [1] donne certains résultats formels que nous utilisons pour identifier les attracteurs des réseaux. Sur la base de ces résultats théoriques [13], nous avons conçu un algorithme pour énumérer tous les attracteurs. La détection des attracteurs se fait sans passer par la simulation des réseaux booléens, contrairement à l'approche proposée dans [12].

Le reste de l'article est organisé comme suit : Nous commençons par rappeler les notions de bases sur les réseaux booléens et sur la sémantique ASP qui est utilisée dans ce papier dans la section 2. Dans la section 3, nous proposons une nouvelle approche pour la recherche et l'énumération des attracteurs. Nous étudions les relations entre le graphe de transition et la représentation logique de son graphe d'interaction dans la section 4. Nous évaluons dans la Section 5 notre approche sur des réseaux booléens générés aléatoirement. Nous concluons le travail et donnons quelques perspectives dans la section 6.

2 Préliminaires

2.1 Réseaux Booléens

Soit $V = \{v_1, ..., v_n\}$ un ensemble fini d'entités booléennes $v_i \in \{0, 1\}$ (1 pour la valeur vrai et 0 pour la valeur faux) représentant les gènes dans les réseaux de régulation. Une configuration $x = (x_1, \ldots, x_n)$ du système est l'affectation d'une valeur de vérité $x_i \in \{0, 1\}$ à chaque élément de V. L'ensemble de toutes les configurations [8], également appelé *l'espace des configurations*, est noté $X = \{0, 1\}^n$. La dynamique d'un tel système est exprimée par une fonction de transition globale f, et un mode de mise à jour qui définit comment les éléments de V sont mis à jour au cours du temps. La fonction de transition globale f est définie comme $f : X \to X$ telle que $x = (x_1, \ldots, x_n) \mapsto f(x) = (f_1(x), \ldots, f_n(x))$, où chaque fonction f_i [1] $: X \to \{0, 1\}$ est *une fonction de transition locale* qui donne l'évolution de l'état x_i du gène v_i au cours du temps. Les réseaux booléens peuvent être considérés comme des abstractions des réseaux de régulation génique où les variables booléennes x_i représentent l'état des gènes v_i. La valeur vraie pour x_i ($x_i = 1$) signifie que le gène correspondant est actif, la valeur fausse ($x_i = 0$) signifie que le gène est inactif.

2.1.1 Graphe de transitions

La dynamique d'un réseau booléen est décrite par un graphe de transition TG qui est défini par une fonction de transition f et un mode de mise à jour, formellement :

Définition 1. *Soit $X = \{0, 1\}^n$ l'espace de configuration d'un réseau booléen, $f : X \to X$ sa fonction de transition*

1. $f_i(x)$ représente le changement local qui est porté sur l'état x_i du gène v_i

globale associée et $f_i : X \to X$, $i \in \{1, \ldots, n\}$ *sont les fonctions de transition locales formant la fonction* f. *Le graphe de transition représentant la dynamique de* f *est le graphe orienté* $TG(f) = (X, T(f))$ *où l'ensemble des sommets est l'ensemble de toutes les configurations de* X *et l'ensemble des arcs est* $T(f) = \{(x, y) \in X^2; x \neq y, x = (x_1, \ldots, x_i, \ldots, x_n), y = (x_1, \ldots, f_i(x), \ldots, x_n)\}$

Le mode asynchrone est un mode de mise à jour dans lequel un seul composant de la configuration x est mis à jour à chaque instant. C'est-à-dire qu'une seule fonction de transition locale f_i est appliquée sur l'état du gène correspondant x_i à chaque instant. Les différents éléments de x pourraient être mis à jour à des intervalles de temps différents. Par conséquent, les transitions ne sont pas déterministes. Il peut y avoir plusieurs configurations successeurs possibles pour une configuration donnée qui est représenté par un nœud dans le graphe de transition.

Une orbite dans $TG(f)$ est une séquence de configurations $(x^0, x^1, x^2, ...)$ telle que soit $(x^t, x^{t+1}) \in T(f)$ soit $x^{t+1} = x^t$ lorsqu'il n'y a pas de successeurs pour x^t. Un cycle de longueur r est une séquence de configurations $(x^1, \ldots, x^r, x^1)$ avec $r \geq 2$ dont les configurations $x^1, \ldots, x^r$ sont différentes. Nous allons à présent définir la notion d'attracteur dans les systèmes dynamiques. Un état / une configuration $x = (x_1, \ldots, x_n)$ du graphe de transition $TG(f)$ est un état / une configuration stable lorsque $\forall x_i \in x, x_i = f_i(x)$, donc $x = f(x)$. Un état/une configuration stable $x = (x_1, \ldots, x_n)$ forme un attracteur trivial de $TG(f)$. Une séquence d'états / configurations $(x^1, x^2, \ldots, x^r, x^1)$ forme un cycle stable de $TG(f)$ lorsque $\forall t < r, x^{t+1}$ est le successeur unique de x^t et x^1 est le successeur unique de x^r. Un cycle stable dans $TG(f)$ forme un attracteur cyclique. Dans ce qui suit, lorsqu'il n'y a pas de confusion, nous représentons l'ensemble des gènes $V = \{v_1, v_2, \ldots, v_n\}$ par leur seule numérotation $V = \{1, 2, \ldots, n\}$.

Exemple 1. *Considérons* $V = \{1, 2, 3\}$, $X = \{0, 1\}^3$ *et les deux fonctions de transition globale suivantes* f *et* g *définies comme suit :* $f(x_1, x_2, x_3) = (x_3, \neg x_1, x_2)$ *et* $g(x_1, x_2, x_3) = (\neg x_3, \neg x_1, x_2)$.

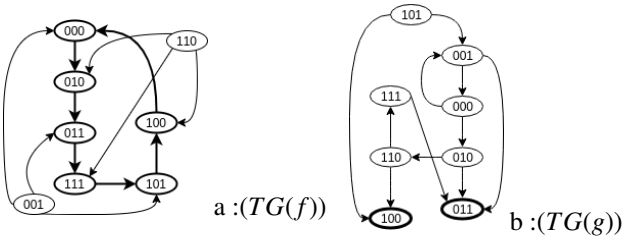


FIGURE 1 – Le graphe de transition d'un circuit booléen positif (b) et d'un circuit booléen négatif (a) de taille 3

Les deux graphes de transition correspondant à la fois à f *et à* g *sont donnés dans la figure 1. Pour chaque arc* (x, y) *des deux graphes de transition, si* $x \neq y$ *alors la configuration* x *diffère de la configuration* y *par une seule composante.* $TG(g)$ *a deux configurations stables* (100) *et* (011)) *indiquées en gras sur la Figure 1 (b). Les deux attracteurs pourraient être écrits comme* $(1, \neg 2, \neg 3)$ *et* $(\neg 1, 2, 3)$ *en prenant en compte les gènes correspondants .* $TG(f)$ *a un cycle stable* ((000), (010), (011), (111), (101), (100)) *de six configurations représentées en gras dans la figure 1 (a). Cet attracteur cyclique pourrait être vu comme* $((\neg 1, \neg 2, \neg 3), (\neg 1, 2, \neg 3), (\neg 1, 2, 3), (1, 2, 3), (1, \neg 2, 3), (1, \neg 2, \neg 3))$ *en considerant les gènes.*

2.1.2 Graphe d'interactions

Les graphes de transition sont un excellent outil pour étudier le comportement dynamique d'une fonction de mise à jour correspondant à un réseau booléen. Cependant, en pratique, les données biologiques proviennent d'expériences qui ne donnent généralement que des corrélations entre les gènes, mais rien sur la dynamique du réseau. Les corrélations entre les gènes dans un réseau de gènes sont traditionnellement représentées par un graphe d'interaction, qui est un graphe dirigé où les arcs sont étiquetés avec un signe - ou +.

Définition 2. *Un graphe d'interaction est un graphe orienté signé* $IG = (V, I)$ *où* $V = \{1, \ldots, n\}$ *est l'ensemble des sommets et* $I \subseteq V \times \{+, -\} \times V$ *est l'ensemble des arcs signés.*

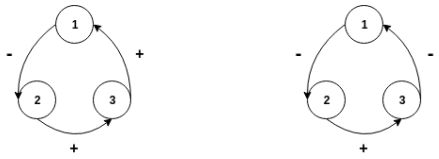
Remarque 1. *Les sommets du graphe d'interaction représentent les différents gènes du réseau de régulation génétique et les arcs expriment les interactions entre eux. Un arc étiqueté par + est dit positif, il dénote une interaction génique positive, tandis qu'un arc étiqueté par - est dit négatif et il exprime une interaction génique négative.*

Définition 3. *Un circuit du graphe d'interactions* $IG = (V, I)$ *de taille* k *est une séquence* $C = (i_1, i_2, \ldots, i_k, i_1)$ *telle que* $(i_j, \{+, -\}, i_{j+1})$ *pour tout* $j \in \{1, \ldots, k-1\}$ *et* $(i_k, \{+, -\}, i_1)$ *sont des arcs du graphe IG. Si tous les sommets de C sont distincts, alors C est dit élémentaire. Si le nombre d'arcs étiquetés par le signe "-" (arcs négatifs) est pair (resp. impair), alors le circuit C est positif (resp. négatif)*

Le graphe d'interaction est une représentation statique des interactions entre les gènes. Chaque nœud v_i du graphe d'interaction est une variable booléenne qui représente l'état du gène i dans le réseau. Plus précisément, si $v_i = 1$ (resp. $v_i = 0$)), alors le gène i est actif (resp. inactif). Un arc positif (resp. négatif) $(v_i, +, v_j)$ (resp. $(v_i, -, v_j)$) compris entre le noeud v_i et le noeud v_j signifie que le gène i est un activateur (resp. ou un inhibiteur) du gène j. Dans la suite, lorsqu'il n'y aura pas de confusion, nous simplifieront la notation en écrivant simplement i pour exprimer le nœud v_i.

Exemple 2. *Considérons le réseau booléen comportant l'ensemble des gènes* $V = \{1, 2, 3\}$, *un espace de configuration*

$X = \{0,1\}^3$ *et deux fonctions de transition globales* f *et* g *définies comme* $f(x_1, x_2, x_3) = (x_3, \neg x_1, x_2)$ *et* $g(x_1, x_2, x_3) = (\neg x_3, \neg x_1, x_2)$. *La figure 2 montre les graphes d'interaction correspondant à la fois à* f *et à* g.



a :($IG(f)$) b :($IG(g)$)

FIGURE 2 – Les deux graphes d'interaction de circuit correspondant aux fonctions de transition globales f et g.

Nous pouvons voir que la fonction f *induit un circuit négatif de taille 3 (Figure 2 (a)) et g induit un circuit positif de taille 3 (Figure 2 (b)). Ces deux graphes d'interaction circulaires correspondent aux graphes de transition présentés dans la figure 1.*

Les graphes d'interaction sont plus compacts que les graphes de transition, donc plus lisibles. Mais, contrairement au graphe de transition, ils ne donnent que des informations statiques sur les interactions. Dans le contexte des réseaux booléens, de nombreux travaux visent à comprendre les relations formelles entre les graphes d'interaction et de transition. Un sujet de recherche majeur porte sur la construction de graphes de transition en utilisant uniquement les fonctions de transition et les graphes d'interaction correspondants. Les auteurs de [20] montrent qu'un circuit positif de taille n admet deux attracteurs dans le mode de mise à jour asynchrone, à savoir deux configurations stables x et $\neg x$ de taille n où $\neg x$ est la configuration booléenne complémentaire de x obtenue en inversant chaque état génétique dans x. D'autre part, un circuit négatif de taille n n'admet qu'un seul attracteur dans le mode de mise à jour asynchrone correspondant à un cycle stable formé par $2n$ configurations représentant sa longueur.

2.2 Answer Set Programming

2.2.1 La nouvelle sémantique pour les programmes logiques généraux

Un programme logique général π est un ensemble de règles de la forme $r : A_0 \leftarrow A_1, A_2, ..., A_m, not\ A_{m+1}, ..., not\ A_n, (0 \leq m < n)$ où $A_{i\in\{0...n\}}$ est un atome et *not* le symbole exprimant la négation par échec. Le corps positif de la règle r est $body^+(r) = \{A_1, A_2, \ldots, A_m\}$, son corps négatif est $body^-(r) = \{A_{m+1}, A_{m+2}, \ldots, A_n\}$ et A_0 est sa tête. Plusieurs sémantiques ont été introduites dans l'ASP pour donner un sens aux programmes logiques. La sémantique des modèles stables [5] est l'une des plus utilisées en ASP. Une nouvelle sémantique pour les programmes logiques généraux est proposée dans [1]. Cette sémantique capture la sémantique des modèles stables et l'étend. Elle est basée sur un langage propositionnel classique L où deux types de variables sont définis. Le sous-ensemble des variables classiques $V = \{A_i : A_i \in L\}$ et le sous-ensemble des extra-variables $nV = \{not\ A_i : not\ A_i \in L\}$. Pour chaque variable $A_i \in V$, il existe une variable correspondante $not\ A_i \in nV$ désignant une sorte de négation par échec faible. Une relation entre les deux types de variables est exprimée par l'ajout au langage L d'un axiome exprimant l'exclusion mutuelle entre elles. Cet axiome d'exclusion mutuelle est exprimé par un ensemble de clauses binaires $ME = \{(\neg A_i \vee \neg not\ A_i) : A_i \in V\}$. Un programme logique général π=$\{r : A_0 \leftarrow A_1, A_2, ..., A_m, not\ A_{m+1}, ..., not\ A_n\}$, $(0 \leq m < n)$ est exprimé dans le langage propositionnel L par un ensemble de clauses de Horn : $HC(\pi) = \{\bigcup_{r\in\pi}(A_0 \vee \neg A_1 \vee, ..., \neg A_m \vee \neg not\ A_{m+1}, ..., \neg not\ A_n)\} \cup ME = \{(\neg A_i \vee \neg not\ A_i) : A_i \in V\}$. Le strong backdoor (STB) [26] du programme logique π est formée par les littéraux de la forme $not\ A_i$ qui apparaissent dans les corps négatifs de ses règles. Formellement, elle est définie par $STB = \{not\ A_i : \exists r \in \pi, A_i \in body^-(r) \subseteq nV\}$. Étant donné un programme π et son STB, une extension de $HC(\pi)$ par rapport au STB, ou simplement une extension de la paire $(HC(\pi), STB)$ est l'ensemble de toutes les clauses consistantes déduites de $HC(\pi)$ en ajoutant un ensemble maximal de littéraux positifs $not\ A_i \in STB$ à $HC(\pi)$. Formellement :

Définition 4. *Soit* $HC(\pi)$ *le codage CNF d'un programme logique* π, *STB son strong backdoor et un sous-ensemble* $S' \subseteq STB$, *l'ensemble* $\mathrm{E} = \mathrm{HC}(\pi) \cup \mathrm{S}'$ *de clauses est alors une extension de* $(\mathrm{HC}(\pi), \mathrm{STB})$ *si les conditions suivantes sont vérifiées :*

1. E *est consistant,*
2. $\forall \mathrm{not\ A_i} \in \mathrm{STB} - \mathrm{S}', \mathrm{E} \cup \{\mathrm{not\ A_i}\}$ *est inconsistant.*

Il est montré dans [1] que chaque $HC(\pi)$ consistant admet au moins une extension par rapport au STB correspondant, formellement :

Proposition 1. *Soit* π *un programme logique et* STB *son strong backdoor. Si* $HC(\pi)$ *est consistant, alors il existe au moins une extension de la pair* $(HC(\pi), STB)$.

Il est démontré dans [1], que l'ensemble des modèles stables d'un programme logique π est en bijection avec l'ensemble des extensions E de $HC(\pi)$ qui satisfont la condition discriminante $(\forall A_i \in V, E \models \neg A_i \Rightarrow E \models A_i)$. Les deux principales propriétés théoriques démontrées sont données dans les deux théorèmes suivants :

Théorème 1. *Si* X *est un modèle stable d'un programme logique* π, *alors il existe une extension* E *de* $(HC(\pi), STB)$ *telle que* $\mathrm{X} = \{\mathrm{A_i} \in \mathrm{V} : \mathrm{E} \models \mathrm{A_i}\}$. *D'autre part,* E *vérifie la condition dite discriminante :* $(\forall A_i \in V, E \models \neg not\ A_i \Rightarrow E \models A_i)$.

Théorème 2. *Si E est une extension de $(HC(\pi), STB)$, qui vérifie la condition $(\forall L_i \in V, E \models \neg notL_i \Rightarrow E \models L_i)$ alors $X = \{L_i : E \models L_i\}$ est un modèle stable de π.*

Exemple 3. *Considérons le programme logique $\pi = \{q \leftarrow not\ r;\quad r \leftarrow not\ q;\quad p \leftarrow not\ p;\quad p \leftarrow not\ r\}$. La représentation en clause de Horn du programme logique π est formé par l'ensemble $HC(\pi) = CR \cup ME$ où $CR = \{q \vee \neg not\ r\,,\ r \vee \neg not\ q\,,\ p \vee \neg not\ p\,,\ p \vee \neg not\ r\}$, $ME = \{\neg a \vee \neg not\ a, \neg r \vee \neg not\ r, \neg p \vee \neg not\ p\}$ et son strong backdoor est donné par $STB = \{not\ r, not\ q, not\ p\}$. On voit que $(HC(\pi), STB)$ admet deux extensions $E_1 = HC(\pi) \cup \{not\ r\}$ et $E_2 = HC(\pi) \cup \{not\ p\}$. En effet, E_1 et E_2 sont maximalement consistants par rapport à l'ensemble STB. Nous pouvons déduire par résolution unitaire que $E_1 \models \{\neg r, q, p, \neg not\ q, \neg not\ p\}$ and $E_2 \models \{\neg not\ r, r, \neg q, \neg not\ p, \neg p\}$. L'extension E_1 satisfait la condition discriminante, mais E_2 ne la satisfait pas. Ainsi, le programme logique a un modèle stable $M_1 = \{p, q\}$ déduit de E_1 par résolution unitaire et l'extra-extension E_2 induit un extra-modèle $M_2 = \{r\}$ où r est vrai et p et q sont faux. Les modèles stables de π sont en bijections avec les extensions de $(HC(\pi), STB)$ satisfaisant la condition discriminante.*

2.2.2 Comment la sémantique est adoptée pour les programmes étendus

Les programmes généraux permettent de modéliser divers problèmes. Cependant, la négation classique est un concept qui est absolument essentiel lorsque des problèmes réels doivent être modélisés de manière déclarative. La sémantique d'un programme logique étendu peut être définie par sa réduction à un programme général [6]. Cette réduction supprime la négation classique, et la sémantique précédemment résumée pour les programmes généraux peut alors être utilisée pour dériver les ensembles réponses du programme logique étendu. Un programme logique étendu est un ensemble de règles de la forme : $r : L_0 \leftarrow L_1, L_2, ..., L_m, not\ L_{m+1}, ..., not\ L_n, (0 \leq m < n)$ où $L_{i\in\{0...n\}}$ sont des littéraux (Atomes A_i ou sa negation $\neg A_i$). Pour réduire un programme logique étendu π en un programme logique général équivalent π', il faut remplacer tous les littéraux négatifs $\neg L$ apparaissant dans π par un nouvel atome L' dans π', puis ajouter les règles de contrainte d'intégrité $(\leftarrow L, L')$ qui interdisent à L et L' d'être vrais dans le même modèle de π'. Cela évite à L et $\neg L$ d'être vrais dans le même modèle du programme logique étendu π. Ainsi, il nous suffit de calculer les modèles stables du programme général résultant π' à partir desquels nous pouvons déduire les ensembles réponses du programme étendu original π.

3 Représentation de graphes d'interaction à l'aide de programmes logiques

Dans ce qui suit, un graphe d'interaction IG sera traduit en un programme logique P_{IG} et représenté sous sa forme clause de Horn $HC(P_{IG})$. Les propriétés liées aux extensions de $HC(P_{IG})$ sont démontrées. Ces dernières seront ensuite utilisées pour étudier la relation entre $HC(P_{IG})$ et le graphe de transition TG. Nous montrerons comment trouver des configurations et des cycles stables du graphe de transition en utilisant simplement le programme logique P_{IG}. Dans le formalisme des réseaux Booléen, on associe à chaque entité ou gène $i \in \{1, \ldots, n\}$ une variable booléenne v_i. Pour alléger la notation, nous utiliserons i au lieu de v_i lorsque cela sera possible. Notre travail ici s'inspire du travail présenté dans [23]. Dans ce travail, les auteurs ont utilisé la logique des hypothèses [21] pour représenter le graphe d'interaction. Ce cadre non-monotone est puissant pour la représentation des connaissances, mais ne contient pas d'outils algorithmiques efficaces. Pour rectifier cette situation, nous avons choisi le framework ASP et la sémantique introduite dans [1]. Cela nous donne un bon compromis entre l'efficacité et le pouvoir d'expressivité.

Définition 5. *Étant donné le graphe d'interaction IG d'un réseau booléen représentant un réseau de régulation de gène, P_{IG} le programme logique représentant IG et un gène i. Nous définissons alors ce qui suit :*

- *i signifie que le gène i est actif dans la cellule*
- *$\neg i$ signifie que le gène i n'est pas actif.*
- *not $\neg i$ (resp. $\neg not\ \neg i$) signifie que la cellule donne (resp. ne donne pas) l'autorisation d'activer i.*
- *not i (resp. $\neg not$ i) signifie que la cellule donne (resp. ne donne pas) le droit de désactiver i.*

Définition 6. *La traduction de IG en un programme logique P_{IG} se fait en traduisant chaque arc de IG en une paire de règles. Plus précisément :*

- *Un arc positif $(i, +, j)$ est traduit en : $j \leftarrow not\ \neg i,\ \neg j \leftarrow not\ i$*
- *Un arc négatif $(i, -, j)$ est traduit en : $j \leftarrow not\ i,\ \neg j \leftarrow not\ \neg i$*

Exemple 4. *Le graphe d'interaction de l'exemple 2 est représenté par le circuit positif qui est exprimé par le programme logique étendus :*

$P(IG(g)) = \{2 \leftarrow not\ 1,\ \neg 2 \leftarrow not\ \neg 1,\ 3 \leftarrow not\ \neg 2,\ \neg 3 \leftarrow not\ 2,\ 1 \leftarrow not\ 3,\ \neg 1 \leftarrow not\ \neg 3\}$.

Le circuit négative est transcrit par le programme logique étendus suivant :

$P(IG(f)) = \{2 \leftarrow not\ 1,\ \neg 2 \leftarrow not\ \neg 1,\ 3 \leftarrow not\ \neg 2,\ \neg 3 \leftarrow not\ 2,\ 1 \leftarrow not\ \neg 3,\ \neg 1 \leftarrow not\ 3\}$.

Le programme logique étendu P_{IG} sera traduit en un programme logique général P'_{IG} qui sera exprimé par un ensemble de clauses de Horn $HC(P'_{IG})$. Une extension de la paire $(HC(P'_{IG}), STB)$ est l'ensemble de toutes les clauses cohérentes dérivées de $HC(P'_{IG})$ lors de l'ajout d'un ensemble maximal de littéraux positifs $not\ A_i \in STB$ dans $HC(P'_{IG})$. Dans ce contexte, l'ensemble STB représente une

collection de permissions pour activer i (Resp. inhiber i). Dans la suite, nous considérerons toujours la représentation clausale $HC(P'_{IG})$ au lieu du programme logique P'_{IG} que nous désignons uniquement par $HC(P_{IG})$ lorsqu'il n'y a pas de confusion. Nous dirons également simplement des extensions de $HC(P_{IG})$ pour signifier des extensions de $(HC(P'_{IG}), STB)$. En ce qui concerne l'utilisation de la nouvelle sémantique [1], le rôle d'une extension semble de collecter un maximum d'autorisations cohérentes. Notez que même si *not* $\neg i$ qui correspond à la cellule donnant la permission de produire i est présente, cette production n'est pas obligatoire. Il peut être produit ou non, selon le contexte (c'est-à-dire l'ensemble de toutes les interactions dans la cellule). De même pour *not* i qui donne l'autorisation de désactiver i. Il est généralement permis d'avoir les deux *not* $\neg i$ et *not* i mais ce n'est pas le cas dans cette représentation. D'un point de vue biologique, nous ne pouvons pas donner la permission d'activer et d'inhiber i en même temps. La proposition 2 ci-dessous confirme cet aspect biologique :

Proposition 2. *Soit $HC(P_{IG})$ un programme logique représentant le graphe d'interaction IG. Alors, Pour chaque $i \in \{1, \ldots, n\}$, $\neg(not\ \neg i \wedge not\ i)$ est vrai.*

Démonstration. Par définition, si IG contient un arc $(i, \{+, -\}, j)$. Avec la traduction de cet arc, nous obtenons deux ensembles de clauses $\{j \vee \neg not\ \neg i,\ \neg j \vee \neg not\ i\}$ ou $\{j \vee \neg not\ i,\ \neg j \vee \neg not\ \neg i\}$. Dans les deux cas, si $not\ \neg i \wedge not\ i$ est vrai. Alors, nous inférons $j \wedge \neg j$. Donc, nous avons une inconsistance. □

Proposition 3. *Soit IG un graphe d'interaction dont l'encodage logique est $HC(P_{IG})$, nous avons :*

1. *$HC(P_{IG})$ est consistant.*
2. *$HC(P_{IG})$ a au moins une seule extension.*

Démonstration.
1. Nous avons noté que $HC(P_{IG})$ est équivalent à un ensemble des clauses de Horn binaires. Chaque clause contient donc au moins un littéral négative et l'affectation de tous les littéraux à faux est un modèle de $HC(P_{IG})$. Alors, $HC(P_{IG})$ est consistant.
2. Comme $HC(P_{IG})$ est consistant, alors $HC(P_{IG})$ a au moins une extension selon la Proposition 1.

□

Les définitions et propositions suivantes sont importantes pour comprendre l'intuition qui se cache derrière la représentation introduite ici. Ils nous permettront de présenter les propriétés et les théorèmes qui établiront des liens entre les graphes d'interactions et de transitions.

Définition 7. *Soit IG un graphe d'interaction ayant l'ensemble d'entités $V = \{v_1, ..., v_n\}$ et E une extension de $HC(P_{IG})$ obtenu en ajoutant à $HC(P_{IG})$ un ensemble maximal consistant de {not i} ou {not $\neg$i}, avec $i \in \{1, \ldots, n\}$.. Nous avons :*

1. *E est complet si pour tous les $i \in V$, not $\neg i \in E$ ou not $i \in E$*
2. *i est libre dans E si $i \notin E$ et $\neg i \notin E$. Autrement, i est fixé.*
3. *Le degré de liberté de E, noté deg(E), est le nombre de sommets libres qui le compose.*
4. *Le miroir de $E = HC(P_{IG}) \cup \{not\ j |\ j \in \{1 \ldots n, \neg 1, \ldots \neg n\}\}$, noté mir(E), est défini comme $mir(E) = HC(P_{IG}) \cup \{not\ \neg j |\ j \in \{1 \ldots n, \neg 1, \ldots \neg n\}\}$.*

Proposition 4. *Soit $HC(P_{IG})$ un programme logique représentant le graphe d'interaction IG et E une extension de $HC(P_{IG})$. Le miroir de E est aussi une extension de $HC(P_{IG})$.*

Démonstration. Par définition, si IG contient un arc $(i, \{+, -\}, j)$ alors sa traduction donne deux ensembles de clauses $\{j \vee \neg not\ \neg i,\ \neg j \vee \neg not\ i\}$ ou $\{j \vee \neg not\ i,\ \neg j \vee \neg not\ \neg i\}$. Une extension est l'ensemble de toutes les clauses cohérentes dérivées de $HC(P_{IG})$ lors de l'ajout d'un ensemble maximal de littéraux positifs *not* i à $HC(P_{IG})$. Si nous inversons chaque littéral *not* i dans l'extension, c'est-à-dire que nous remplaçons *not* i (resp. *not* $\neg i$) par *not* $\neg i$ (resp. *not* i), alors nous avons deux cas :

— Le premier cas correspond à la présence d'un arc positif dans le graphe d'interaction IG. Dans ce cas, nous déduisons j lorsque *not* $\neg i$ est vrai, ou $\neg j$ si *not* i est vrai.

— Le deuxième cas correspond à la présence d'un arc négatif dans le graphe d'interaction IG. Dans ce cas, nous déduisons $\neg j$ lorsque *not* $\neg i$ est vrai et infère j lorsque *not* i est vrai.

Ainsi, il est trivial de voir que l'extension E et son miroir $mir(E)$ sont symétriques. Il en résulte que $mir(E)$ est également une extension. □

Pour certains graphes d'interaction IG particuliers tel que les circuits, nous montrons maintenant que les extensions complètes $HC(P_{IG})$ sont de degré zéro et induisent des ensembles réponses du codage logique $HC(P_{IG})$.

Proposition 5. *Soit IG un graph d'interaction, E est une extension de $HC(P_{IG})$. Si chaque nœud d'un graphe d'interaction IG a au moins un arc entrant, alors toute extension complète de $HC(P_{IG})$ est de degré 0.*

Démonstration. Soit E une extension complète de $HC(P_{IG})$. Pour prouver que E est de degré 0, nous devons prouver que chaque variable j de $HC(P_{IG})$ est liée dans E. En d'autres termes, pour chaque nœud j dans le graphe d'interaction IG, nous avons soit $\neg j \in E$ ou $j \in E$. Par hypothèse, j a un arc entrant positif / négatif $(j, +/-, i)$ dans IG, nous avons donc deux cas :

— Si l'arc est positif, il est exprimé par la paire de clauses $\{j \vee \neg not\ \neg i, \neg j \vee \neg not\ i\}$. Puisque E est complet, nous avons soit *not* $i \in E$ ou *not* $\neg i \in E$. Si *not* $\neg i \in E$, alors j est déduit ($j \in E$). Si *not* $i \in E$, alors $\neg j$ est déduit ($\neg j \in E$).

— Le cas d'un arc négatif est traité de la même manière. Nous aurons les clauses $\{j \vee \neg not\ i, \neg j \vee \neg not\ \neg i\}$. Si *not* $\neg i \in E$, alors nous déduisons $\neg j$ et si *not* $i \in E$, alors nous déduisons j.

Par conséquent, pour toutes les hypothèses, nous déduisons j ou $\neg j$. Ainsi, chaque élément j est lié dans E et le degré de E est 0. □

Proposition 6. *Soit IG un graphe d'interaction, si chaque sommet de IG a au moins un arc entrant, alors toute extension complète de $HC(P_{IG})$ correspond à un ensemble réponse de $HC(P_{IG})$.*

Démonstration. Soit E une extension complète de $HC(P_{IG})$. E correspond à un ensemble réponse si pour un nœud i donné, la condition discriminante est vérifiée pour i et $\neg i$. Les deux conditions discriminantes sont : (1) $\neg not\ i \in E \Rightarrow i \in E$ et (2) $\neg not\ \neg i \in E \Rightarrow \neg i \in E$. Puisque E est complet, alors il est de degré 0 (Proposition 5). Il en résulte que i ou $\neg i$ sont vrais dans E. Nous avons deux cas :

— Si nous avons $i \in E$. Alors, (1) est trivialement vérifié. Selon l'exclusion mutuelle $ME = \{(\neg i \vee \neg not\ i)\}$, on obtient $\neg not\ i \in E$. Dans ce cas, nous avons $\neg i \notin E$ et supposons maintenant que $\neg not\ \neg i \in E$, cela signifie que $not\ \neg i \notin E$. Comme E est complet, nous avons $not\ i \in E$, ce qui contredit le fait que $\neg not\ i \in E$. Ainsi, la condition (2) est vérifiée.

— Si nous avons $\neg i \in E$, alors la condition (2) est vérifiée trivialement. Selon l'exclusion mutuelle ME, on obtient $\neg not\ \neg i \in E$. Dans ce cas, nous avons $i \notin E$ et supposons maintenant que $\neg not\ i \in E$, donc $not\ i \notin E$. Comme E est complet, alors $not\ \neg i \in E$, ce qui contredit le fait que $\neg not\ \neg i \in E$. La condition (1) est donc vérifiée.

Puisque E vérifie la condition discriminante dans les deux cas, alors E induit un ensemble réponse de $HC(P_{IG})$. □

Maintenant, nous allons prouver que tout ensemble réponse de $HC(P_{IG})$ correspond à une extension de degré 0.

Proposition 7. *Soit IG un graphe d'interaction, si chaque sommet de IG a au moins un arc entrant, alors chaque ensemble réponse de $HC(P_{IG})$ correspond à une extension de degré 0.*

Démonstration. Soit E une extension induisant un ensemble réponse de $HC(P_{IG})$. Par définition, E est maximalement consistant par rapport aux littéraux de la forme $not\ i \in E$ ou $not\ \neg i \in E$ et vérifie la condition discriminante (a)$\neg not\ i \in E \Rightarrow i \in E$ et (b) $\neg not\ \neg i \in E \Rightarrow \neg i \in E$ correspondant à la fois à i et $\neg i$.

L'extension E induit alors un ensemble réponse de $HC(P_{IG})$. Nous devons prouver que pour tout $i \in HC(P_{IG})$, nous avons $i \in E$ ou $\neg i \in E$. Il existe trois cas :

1. Le cas où $not\ i \in E$ et $not\ \neg i \notin E$. Il résulte de la proposition 2 que $\neg not\ \neg i \in E$. Ainsi, de la condition discriminante (b), nous obtenons $\neg i \in E$.
2. Le cas où $not\ \neg i \in E$ et $not\ i \notin E$. De la proposition 2 nous obtenons $\neg not\ i \in E$. Ainsi, $i \in E$ puisque la condition discriminante (a) est vraie.
3. Le cas où $not\ i \notin E$ et $not\ \neg i \notin E$. Dans ce cas, nous avons $not\ i \wedge E \models \square$ et $not\ \neg i \wedge E \models \square$. Ainsi, $E \models \neg not\ i$ et $E \models \neg not\ \neg i$. De (a) et (b), nous avons $E \models i$ et $E \models \neg i$. Ainsi, nous obtenons une incohérence qui contredit le fait que E est une extension.

Il en résulte que seul le premier et le deuxième cas sont possibles. Ainsi, nous avons soit $i \in E$ ou $\neg i \in E$.

□

Dans ce qui suit, nous montrons que pour un graphe d'interaction IG représentant un circuit positif de n nœuds, le codage $HC(P_{IG})$ correspondant admet deux ensembles réponses de taille n.

Proposition 8. *Si le graphe d'interaction IG est un circuit positif de taille n, alors sa forme Horn clausale $HC(P_{IG})$ a deux extensions qui induisent deux ensembles réponses de taille n*

Démonstration. La preuve est basée sur les résultats de la Proposition 6 et le fait que dans un circuit positif, chaque gène agit positivement sur lui même à travers le circuit. En effet, si nous donnons au début l'autorisation d'activer le gène i (en supposant $not\ \neg i$) alors nous allons déduire que i est active, inversement si initialement nous donnons l'autorisation de désactiver i (en supposant $not\ i$) alors nous allons déduire que i est inactive ($\neg i$). On peut alors construire deux extensions complètes de degré 0. La première est faite en supposant au début le littéral $not\ \neg i$ et la seconde est son extension miroir qui est obtenue en supposant au début le littéral $not\ i$. Les deux extensions étant complètes et de degré zéro, on déduit grâce à la proposition 6 que chacune induit un ensemble réponse de taille n. □

Exemple 5. *Considérons le graphe d'interaction dans l'exemple 1 qui représente le circuit positif. Nous traduisons ce graphe en un programme logique étendu puis traduit en un programme logique général :*

$P_{IG}(g) = \{2 \leftarrow not\ 1,\ \neg 2 \leftarrow not\ \neg 1,\ 3 \leftarrow not\ \neg 2,\ \neg 3 \leftarrow not\ 2,\ 1 \leftarrow not\ 3,\ \neg 1 \leftarrow not\ \neg 3\}$

$P'_{IG}(g) = \{2 \leftarrow not\ 1,\ 2' \leftarrow not\ 1',\ 3 \leftarrow not\ 2',\ 3' \leftarrow not\ 2,\ 1 \leftarrow not\ 3,\ 1' \leftarrow not\ 3'\}$.

$HC(P'_{IG}(g))$ a deux extension qui correspondent aux modèles stables $E_1 = HC(P'_{IG}(g)) \cup \{not\ 1,\ not2',\ not3'\}$ et $E_2 = HC(P'_{IG}(g)) \cup \{not\ 1',\ not2,\ not3\}$. E_1 et E_2 sont des modèles stables parce-que pour chaque i, $\neg not\ i \in E \Rightarrow i \in E$ et $\neg not\ i' \in E \Rightarrow i' \in E$. Nous remarquons que E_1 est complet parce-que pour chaque $i \in HC(P'_{IG}(g))$, soit not i' (not $\neg i$) appartient à E_1 ou not i appartient à E_1. Nous avons soit $i \in E_1$ ou $i' = \neg i \in E_1$ pour chaque $i \in HC(P'_{IG}(g))$, ce qui signifie que le degré de liberté de E_1 est 0. L'extension E_2 est le miroir de E_1. Le modèle stable induit par E_1 et E_2 sont $M'_1 = \{1', 2, 3\}$ et $M'_2 = \{1, 2', 3'\}$. Les ensembles réponses du programme étendu $P_{IG}(g)$ correspondants sont $M_1 = \{\neg 1, 2, 3\}$ et $M_2 = \{1, \neg 2, \neg 3\}$. On peut voir que les deux ensembles de réponses précédents correspondent aux deux configurations stables du graphe de transition IG (Figure 1-(b)) du circuit positif de l'Exemple 1.

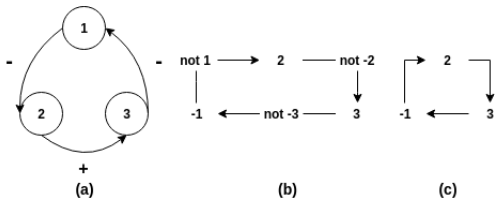


FIGURE 3 – (a) $IG(f)$, (b) Construction de E_1 (c) Le graphe de E_1 (circuit positif)

Une intuition du calcul de E_1 est donné par le processus de construction décrit par la Figure 3. Le graphe d'interaction est représenté par la figure 3-(a).Figure 3-(b) donne les étapes de construction de E_1. En premier E_1 est vide, nous ajoutons à E_1 l'hypothèse not 1. *Comme nous avons* $2 \leftarrow$ *not* 1, *nous allons avoir* 2. *La règle d'exclusion mutuelle* ($\neg 2 \vee \neg$*not* 2) *dit que c'est impossible d'avoir not* 2. *La construction de E_1 se poursuit par l'ajout de l'hypothèse not* $\neg 2$ *à E_1 et nous aurons* 3. *La règle d'exclusion mutuelle* ($\neg 3 \vee \neg$*not* 3) *interdit l'application de not* 3. *Alors, nous ajoutons not* $\neg 3$ *ce qui donne* $\neg 1$. *Si nous nous intéressons uniquement aux littéraux i, on obtient alors le graphe restreint de E_1 représenté par la figure 3(c) représentant le modèle stable correspondant M_1. Ce modèle correspond à l'une des deux configurations stables du graphe de transition de l'exemple 1. De la même manière que nous avons construit E_1, nous construisons l'extension E_2 en commençant par not* $\neg 1$. *D'un point de vue biologique, l'ensemble* $\{\neg 1, 2, 3\}$ *de M_1 représente l'état de chaque gène ;* 2 *et* 3 *sont active et* 1 *est inactive. La même chose pour M_2, l'ensemble* $\{1, \neg 2, \neg 3\}$ *représente l'état des gènes.*

Dans ce qui suit, nous montrerons que chaque graphe d'interaction IG représentant un circuit négatif de n nœuds, a $2n$ extra-extensions de degré 1 induisant $2n$ extra-modèles qui encodent un cycle stable de taille $2n$ dans le graphe de transition correspondant.

Proposition 9. *Si le graphe d'interaction IG est un circuit négative de taille n alors $HC(P_{IG})$ a* $2n$ *extra-extensions de degré 1 qui induisent* $2n$ *extra-modèles de taille* $n-1$.

Démonstration. La preuve est basée sur le fait que dans un circuit négatif chaque gène agit négativement sur lui même à travers le circuit. En effet, si nous donnons au début de l'autorisation pour activer le gène i (en supposant *not* $\neg i$) alors lorsque nous allons fermer le circuit nous allons déduire que i est inactive ($\neg i$). Inversement, si initialement nous autorisons la désactivation de i (en supposant *not* i) alors nous allons déduire que i est active (i) lorsque nous allons fermer le circuit. Nous obtenons une inconsistance dans les deux cas, parce que nous allons déduire i et $\neg i$ au même moment. Cela signifie que nous ne pouvons pas avoir une extension complète dans les deux cas. Donc, nous obtenons une extension et son miroir qui sont toutes deux incomplètes, il n'y aura ni le littéral *not* j ni le littéral *not* $\neg j$, sachant que j est le prédécesseur de i. Ni i ni $\neg i$ ne sont vrais dans les deux extensions obtenues. En revanche, tous les autres éléments différents de i sont liés dans les deux extensions en question. Il s'ensuit que les deux extensions sont donc de degré 1. Il est également évident que les deux extensions ne remplissent pas la condition discriminante. En effet, dans le premier cas, nous aurons $\neg$*not* i sans avoir i et dans le cas du miroir, nous aurons $\neg$*not* $\neg i$ sans avoir $\neg i$. Au final, nous avons donc deux extra-extensions miroir de degré 1 qui induisent deux extra-modèles de taille $n-1$. Chaque fois que nous changeons l'élément de départ i, nous obtiendrons deux extensions supplémentaires miroir de degré 1 qui induiront deux modèles supplémentaires de tailles $n-1$. Au total, il y aura donc $2n$ extra-extensions de degré 1 induisant $2n$ extra-modèles de tailles $n-1$. □

Exemple 6. *Considérons le graphe d'interaction de l'exemple 1 représentant le circuit négative. Nous traduisons ce graphe en un programme logique étendu puis traduit en un programme logique général :* $P_{IG}(f) = \{2 \leftarrow$ *not* $1,\ \neg 2 \leftarrow$ *not* $\neg 1,\ 3 \leftarrow$ *not* $\neg 2,\ \neg 3 \leftarrow$ *not* $2,\ 1 \leftarrow$ *not* $\neg 3,\ \neg 1 \leftarrow$ *not* $3\}$.

$P'_{IG}(f) = \{2 \leftarrow$ *not* $1,\ 2' \leftarrow$ *not* $1',\ 3 \leftarrow$ *not* $2',\ 3' \leftarrow$ *not* $2,\ 1 \leftarrow$ *not* $3',\ 1' \leftarrow$ *not* $3\}$. *Le programme logique $P'_{IG}(f)$ a six extensions qui correspondent aux extra-modèles :*

$E_1 = HC(P'_{IG}(f)) \cup \{$*not* $1,$ *not* $2'\}$, $E_1 \models \{$*not* $1, 2,$ *not* $2', 3\}$

$E_2 = HC(P'_{IG}(f)) \cup \{$*not* $1,$ *not* $3\}$, $E_2 \models \{$*not* $1, 2,$ *not* $3, 1'\}$

$E_3 = HC(P'_{IG}(f)) \cup \{$*not* $1',$ *not* $3'\}$, $E_3 \models \{$*not* $1', 2',$ *not* $3, 1\}$

$E_4 = HC(P'_{IG}(f)) \cup \{$*not* $1',$ *not* $2\}$, $E_4 \models \{$*not* $1', 3',$ *not* $2, 3'\}$

$E_5 = HC(P'_{IG}(f)) \cup \{$*not* $2',$ *not* $3'\}$ $E_5 \models \{$*not* $2', 3,$ *not* $3', 1\}$

$E_6 = HC(P'_{IG}(f)) \cup \{$*not* $2,$ *not* $3\}$, $E_6 \models \{$*not* $2, 3',$ *not* $3, 1'\}$

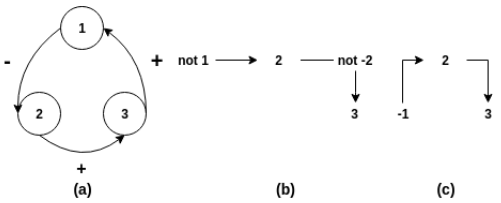


FIGURE 4 – (a) $IG(g)$, (b) E_1 construction (c) Le graphe de E_1 (circuit négatif).

Figure 4(a) montre le circuit négative exprimer par un programme logique. Figure 4(b) montre la construction de l'extension E'_1. Elle est construite en ajoutant à $HC(P'_{IG}(f))$ les littéraux not 1 *et not* $2'$. *Nous pouvons remarquer que dans la Figure 4(b), qu'il est impossible de déduire* $1'$. *En effet, pour obtenir* $1'$, *nous devons utiliser la règle* ($1' \leftarrow$ *not* 3). *Ce qui est impossible parce-que* $\neg$*not* 3 *résulte de l'exclusion mutuelle* ($3' \vee \neg$*not* 3). *D'autre part, nous ne pouvons pas avoir* 1. *Comme not* 1 *est vrai, alors de l'exclusion mutuelle* ($1' \vee \neg$*not* 1) *nous obtenons* $1'$. *Donc, nous ne pouvons pas avoir* 1. *On peut remarquer que l'extension E'_1 n'est pas complète parce-que elle ne contient ni not* 3 *ni not* $3'$. *L'Élément* 1 *est libre dans E'_1 du fait de* $1 \notin E_1$ *et* $1' \notin E'_1$. *Il en résulte que, E'_1 est une extension de degré 1. La figure 4(c) donne la construction de E_1 correspondant à l'extra-modèle M_1. Il est important de voir que la notion de degré de liberté joue ici un rôle clé. Nous remarquons que dans cette figure 4–(c) ne forme pas un circuit car il n'y a pas d'arc entre* 3 *et* $\neg 1$. *Nous remarquons que dans toutes les extensions, nous avons un seul gène manquant.*

4 La relation entre les réseaux Booléens et la representation logique

Dans cette section, nous étudions la relation entre la représentation logique $HC(P_{IG})$ du graphe d'interaction IG et son graphe de transition TG. Pour ce faire, nous verrons que les sommets du graphe de transition TG correspondent à des configurations stables ou à des configurations appartenant à un cycle stable. Ils pourraient représenter en fait des extensions/extra-extensions (ensembles réponses ou extra-modèles) du codage logique $HC(P_{IG})$. Soit un réseau Booléen, IG son graphe d'interaction et TG le graphe de transition correspondant. Nous avons $HC(P_{IG})$ le forme Horn clausale de P_{IG} exprimant IG. Lorsque IG est un cir-

cuit positif, nous allons montrer (Théorème 3) que nous avons un isomorphisme entre les configurations stables de TG et les ensembles réponses (modèles stables) de $HC(P_{IG})$ ayant un degré de liberté 0. De plus lorsque IG est un circuit négatif de taille n , nous allons montrer (Théorème 4) que $HC(P_{IG})$ a $2n$ extra-extensions (extra-modèles) de degré 1 correspondant au cycle stable de TG.

Proposition 10. *Soit un réseau booléen représenté par un graphe d'interaction IG où TG est le graphe de transition associé et $HC(P_{IG})$ la représentation en clause de Horn du programme logique P_{IG} exprimant IG. Si s est un sommet (configuration) de TG représentant une extension/extra-extension E de $HC(P_{IG})$ de degré k, alors s a exactement k successeurs.*

Démonstration. Si i est libre dans l'extension/extra-extension E représentant la configuration s, alors soit $\neg i$ ou i est vrai dans une extension/extra-extension correspondant à une configuration s' accessible depuis s.

Par construction de TG, s' est le seul successeur de s qui vérifie cette propriété. Cette dernière est vérifié pour chaque élément libre i dans E. Donc, si le degré de liberté de E est k, alors il y aura exactement k sommets accessible depuis s. □

Théorème 3. *Soit un réseau Booléen où le graphe d'interaction IG est un circuit positif et $HC(P_{IG})$ la représentation en clause de Horn du programme logique P_{IG}. Nous avons alors :*

1. *Si $X = (x_1, x_2, \ldots, x_n)$ est un ensemble réponse de P_{IG} induit par une extension de $HC(P_{IG})$, alors $(x_1, ..., x_n)$ est une configuration stable du graphe de transition TG*
2. *Si $X = (x_1, ..., x_n)$ est une configuration stable du graphe de transition TG, alors X correspond à un ensemble réponse de P_{IG} induit par une extension de $HC(P_{IG})$.*

Démonstration.

1. Soit E une extension induisant l'ensemble réponse $X = (x_1, x_2, \ldots, x_n)$ et $s = (x_1, x_2, \ldots, x_n)$ le sommet représentant E dans TG. Comme E est une extension complète, alors son degré de liberté est 0. Selon la proposition 10, le seul sommet accessible depuis s est lui-même. Donc, s est une configuration stable de TG.
2. Si $s = (x_1, x_2, \ldots, x_n)$ est une configuration stable du graphe de transition TG, alors aucun arc ne sort de s. Le seul sommet accessible depuis s est lui-même. Il s'ensuit que pour chaque élément x_i ($resp.$ $\neg x_i$) de s, soit x_i est vrai ou $\neg x_i$ est vrai. Ensuite, tous les x_i sont liés dans l'extension E correspondant à la configuration s. Autrement dit, le degré de liberté de E est 0. Il résulte de la proposition 7 que $s = (x_1, x_2, \ldots, x_n)$ forme un ensemble réponse de P_{IG}.

□

Exemple 7. *La partie droite de la figure 5 montre les deux extensions obtenues pour le programme logique correspondant au circuit positif de l'exemple 5. Nous pouvons voir que ces extensions induisent deux ensembles réponses qui codent les deux configurations stables du graphe de transition (partie gauche de la figure 5) dessinées en gras.*

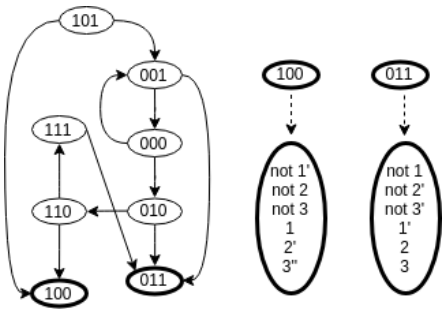


FIGURE 5 – Les configurations stables représentées comme des extensions de $HC(P_{IG})$.

Théorème 4. *Soit un réseau Booléen où le graphe d'interaction IG est un circuit négatif de taille n et P_{IG} le programme logique exprimant IG. L'ensemble $2n$ d'extra-extensions de $HC(P_{IG})$ correspond à un cycle stable du graphe de transition TG associé.*

Démonstration. La proposition 9 garantit l'existence de $2n$ extra-extension (extra-modèles) de degré 1. Nous devons considérer ici le fait que toutes les extra-extension de $2n$ sont de degré 1. Cela implique qu'il y a une seule transition de chaque extra-extension de degré 1 à une autre extra-extension de degré 1, produisant un cycle stable d'extra-extensions de $2n$. Cela correspond à un cycle stable de taille $2n$ dans TG, où chaque extra-extension correspond à une configuration dans le cycle stable de TG.

□

Exemple 8. *La partie droite de la figure 6 montre les extra-extensions obtenues pour le programme logique correspondant au circuit négatif de l'exemple 6. Nous pouvons voir que six extra-extensions de degré 1 induisent six extra-modèles et chacune d'elles identifie une configuration du cycle stable du graphe de transition correspondant (partie gauche de la figure 6, représenté ici en caractères gras).*

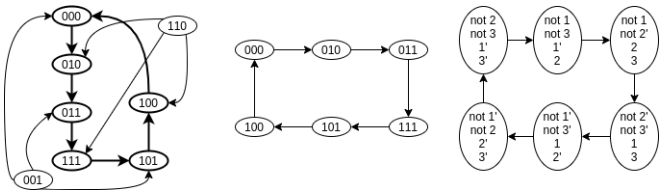


FIGURE 6 – Cycle stable représenté par des extensions de $HC(P_{IG})$.

5 Validation Empirique

Pour démontrer la validité de notre approche sur la découverte des attracteurs de réseaux Booléens, nous l'avons testée sur un vaste ensemble de réseaux générés aléatoirement.

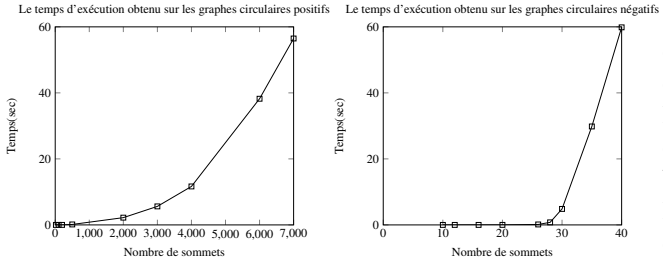


FIGURE 7 – Le temps d'exécution obtenu sur les graphes circulaires générés aléatoirement

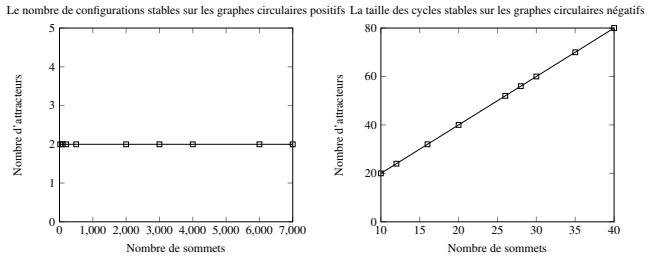


FIGURE 8 – Le nombre d'attracteurs obtenus sur les graphes circulaires générés aléatoirement

Les réseaux circulaires de taille n ont été générés en sélectionnant pour un nœud courant $i \in \{1, 2, \ldots, n\}$ de manière indépendante et uniforme exactement un successeur $j \in \{1, 2, \ldots, i-1, i+1, \ldots, n\}$. L'étiquette s de l'arc (i, s, j) est générée en choisissant au hasard un signe entre positif et négatif ($s \in \{+, -\}$). Le processus est répété pour le nœud j et tous les nœuds successeurs générés jusqu'au dernier nœud, qui doit avoir pour successeur le nœud de départ i rendant le graphe cyclique. Dans ces expérimentations, nous avons appliqué l'algorithme présenté auparavant sur des réseaux Booléens générés de manière aléatoire, jusqu'à 7 000 sommets pour un réseau circulaire positif et jusqu'à 40 pour un réseau circulaire négatif. Le temps d'exécution résultant est présenté dans la figure 7. Comme nous pouvons le constater, le temps d'exécution est très prometteur. Nous calculons les deux attracteurs d'un circuit positif à 7 000 sommets en moins de 60 secondes. Dans le même temps, nous avons calculé pour le circuit négatif des cycles stables pour des réseaux d'une taille de 40 sommets en moins de 60 secondes. Nous pouvons également voir dans la figure 8 que le nombre de configurations stables pour tous les circuits positifs générés est de 2, tandis que pour les circuits négatifs, il n'y a qu'un seul cycle stable de taille $2n$ pour chaque graphe. Ces résultats expérimentaux correspondent bien à la biologie et confirment la validité des propriétés théoriques démontrées dans cet article.

6 Conclusion

Les réseaux booléens représentent une technique de modélisation très répandue pour analyser le comportement dynamique des réseaux de régulation des gènes. En utilisant les réseaux booléens, nous pouvons capturer les attracteurs de réseaux, qui sont souvent utiles pour étudier la fonction biologique d'une cellule. Nous avons discuté dans cet article le cas particulier des circuits qui jouent un rôle essentiel dans les systèmes biologiques. Nous avons prouvé plusieurs propriétés qui établissent une correspondance entre les attracteurs des graphes de transition et les modèles / extra-modèles stables des programmes logiques exprimant les graphes d'interaction circulaire. En particulier, la représentation des cycles stables des circuits négatifs par un ensemble d'extra-modèles montre l'avantage de l'extension offerte par la sémantique utilisée à celle des modèles stables [5]. En plus des résultats théoriques, nous avons proposé une approche qui calcule efficacement tous les ensembles réponses / extra-modèles représentant les configurations stables ou les cycles stables du graphe de transition du réseau booléen considéré. Le fait que la représentation logique d'un circuit positif possède deux modèles miroirs stables et que celle d'un circuit négatif possède un ensemble de $2n$ extra-modèles témoigne bien de la validité de la méthode puisque cela correspond aux résultats connus en biologie [20]. A titre de développement futur, nous souhaitons prendre en compte d'autres modes de mise à jour comme le mode parallèle, puis généraliser l'étude aux blocs séquentiels, qui sont des mises à jour périodiques déterministes. D'autre part, nous nous intéressons à la caractérisation des cycles non stables dans les réseaux booléens.

Références

[1] Belaïd BENHAMOU et Pierre SIEGEL : A new semantics for logic programs capturing and extending the stable model semantics. *Tools with Artificial Intelligence (ICTAI)*, pages 25–32, 2012.

[2] Hidde DE JONG : Modeling and simulation of genetic regulatory systems : a literature review. *Journal of computational biology*, 9(1):67–103, 2002.

[3] Esra ERDEM, Michael GELFOND et Nicola LEONE : Applications of answer set programming. *AI Magazine*, 37:53–68, 2016.

[4] Martin GEBSER, Benjamin KAUFMANN, Andrá NEUMANN et Torsten SCHAUB : Conflict-driven answer set solving. *IJCAI*, 7:386–392, 2007.

[5] Michael GELFOND et Vladimir LIFSCHITZ : The stable model semantics for logic programming. *ICLP/SLP*, 50:1070–1080, 1988.

[6] Michael GELFOND et Vladimir LIFSCHITZ : Classical negation in logic programs and disjunctive databases. *New generation computing*, 9:365–385, 1991.

[7] Katsumi Inoue : Logic programming for boolean networks. *In Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[8] François Jacob et Jacques Monod : Genetic regulatory mechanisms in the synthesis of proteins. *Journal of molecular biology*, 3(3):318–356, 1961.

[9] Stuart A Kauffman : Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–467, 1969.

[10] Stuart A Kauffman : *The origins of order : Self-organization and selection in evolution*. OUP USA, 1993.

[11] Tarek Khaled et Belaid Benhamou : Symmetry breaking in a new stable model search method. *22nd International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR-22), Kalpa Publications in Computing*, 9:58–74, 2018.

[12] Tarek Khaled et Belaid Benhamou : An asp-based approach for attractor enumeration in synchronous and asynchronous boolean networks. *Proceedings 35th International Conference on Logic Programming, ICLP 2019, Las Cruces, NM, USA*, pages 295–301, 2019.

[13] Tarek Khaled et Belaid Benhamou : An asp-based approach for boolean networks representation and attractor detection. *In LPAR*, pages 317–333, 2020.

[14] Tarek Khaled et Belaid Benhamou : Dealing with biology systems in the framework of answer set programming. *Procedia Computer Science*, 176:450–459, 2020.

[15] Tarek Khaled, Belaïd Benhamou et Pierre Siegel : Une nouvelle méthode pour la recherche de modèles stables en programmation logique. *JFPC 2018*, page 63.

[16] Tarek Khaled, Belaïd Benhamou et Pierre Siegel : A new method for computing stable models in logic programming. *Tools with Artificial Intelligence (ICTAI)*, pages 800–807, 2018.

[17] Hannes Klarner, Alexander Bockmayr et Heike Siebert : Computing maximal and minimal trap spaces of boolean networks. *Natural Computing*, 14(4):535–544, 2015.

[18] Fangzhen Lin et Yuting Zhao : Assat : Computing answer sets of a logic program by sat solvers. *Artificial Intelligence*, pages 115–137, 2004.

[19] Victor W. Marek et Miros L. Truszczynski : Stable models and an alternative logic programming paradigm. *The Logic Programming Paradigm*, pages 375–398, 1999.

[20] Elisabeth Remy, Brigitte Mossé, Claudine Chaouiya et Denis Thieffry : A description of dynamical graphs associated to elementary regulatory circuits. *Bioinformatics*, 19(suppl_2):ii172–ii178, 2003.

[21] Camilla Schwind et Pierre Siegel : A modal logic for hypothesis theory. *Fundamenta Informaticae*, 21:89–101, 1994.

[22] Ilya Shmulevich, Edward R Dougherty et Wei Zhang : From boolean to probabilistic boolean networks as models of genetic regulatory networks. *Proceedings of the IEEE*, 90(11):1778–1792, 2002.

[23] Pierre Siegel, Andrei Doncescu, Vincent Risch et Sylvain Sené : Towards a boolean dynamical system representation in a monmonotonic modal logic. 2018.

[24] Patrik Simons, Ilkka Niemelä et Timo Soininen : Extending and implementing the stable model semantic. *Artificial Intelligence*, 138:181–234, 2002.

[25] Nam Tran et Chitta Baral : Hypothesizing about signaling networks. *Journal of Applied Logic*, 7:253–274, 2009.

[26] Ryan Williams, Carla P Gomes et Bart Selman : Backdoors to typical case complexity. *International joint conference on artificial intelligence*, 18:1173–1178, 2003.