Recherche de bassins d'attraction dans des réseaux biologiques

Alexis Tricot

Avril-Juillet 2018

Table des matières

1 Introduction

L'étude de réseaux biologiques comportant un grand nombre de gènes est un des challenges importants de la bioinformatique moderne. La dynamique de tels réseaux s'avère souvent très complexe, quelle que soit la modélisation choisie ; cependant, la connaissance de la dynamique d'un système biologique est cruciale à la compréhension des interactions entre gènes et des états particuliers dudit système. En particulier, l'étude des attracteurs apporte des informations importantes : ce sont des ensembles d'états dont on ne peut plus sortir une fois atteints, qui peuvent par exemple modéliser une différentiation ou un comportement pathologique. Une meilleure compréhension des attracteurs d'un réseau et des conditions qui les forment peut offrir la possibilité de concevoir des solutions pour les éviter ou au contraire les rejoindre.
Le but de notre travail est d'identifier les attracteurs de réseaux d'automates modélisant des réseaux biologiques. Nous adopterons cette modélisation qui vient des réseaux de Thomas [8].
Un certain nombre de travaux sur les attracteurs de réseaux booléens asynchrones ont déjà été menés [1, 6] et il existe des études très récentes sur le sujet [7]. On présente ici notre méthode de recherche, qui a la particularité de s'intéresser aux bassins d'attraction et de mêler étude statique et dynamique. Notre méthode ne prétend pas être exhaustive mais elle fournit une quantité appréciable d'information sur la dynamique générale du réseau. De plus, notre algorithme se veut très rapide afin de permettre l'étude de réseaux très larges. Cette méthode a par ailleurs été implémentée dans le langage C.

2 Chronologie générale du stage

Initialement, l'objectif de mon stage était d'imaginer et d'implémenter un algorithme de recherche d'attracteurs en utilisant un solveur Satisfiability Modulo Theory (SMT). Grossièrement, un solveur SMT utilise un solveur SAT et des "théories" qui permettent de modéliser un problème, problème qui est ensuite résolu par le solveur SAT. L'utilisation de solveurs SAT ou d'outils de programmation logique sur des problèmes de ce type est fréquente. La puissance des solveurs SAT actuels permet d'obtenir des algorithmes rapides. L'étude [6] utilise le langage ASP pour repérer des bifurcations (point d'entrée dans un attracteur). L'objectif de mon travail était d'essayer d'appliquer la flexibilité d'un solveur SMT au problème de la recherche d'attracteurs dans un réseau biologique. J'ai commencé par étudier le problème en lisant un certain nombre d'articles sur le sujet. Je me suis ensuite interressé aux différents solveurs SMT et j'en ai utilisé deux : Yices et SMCHR [1]. Je me suis longuement interrogé sur la forme que pouvaient prendre les attracteurs et les bassins d'attraction. J'ai progressivement décidé de relâcher certaines contraintes (attracteurs/bassins, automates binaires, exhaustivité). Au fur et à mesure de l'avancée de mon tra-

1. Respectivement *http ://yices.csl.sri.com/* et *https ://www.comp.nus.edu.sg/ gregory/smchr/*

vail, l'utilisation d'un solveur est devenue de moins en moins adéquate pour la solution vers laquelle je me dirigeais. Finalement, la solution que je présente ici est purement algorithmique et n'utilise pas de solveurs, SMT ou autres. Il est toutefois possible que l'algorithme puisse être amélioré en y intégrant des méthodes de programmation logique.
Chronologiquement, le premier mois de mon stage m'a servi à m'approprier le sujet ainsi que les différents outils informatiques que je devais utiliser. Le deuxième mois a été consacré à réfléchir au problème, à la structure des attracteurs et bassins d'attraction et aux différentes méthodes envisageables pour les rechercher. J'ai finalement employé les deux derniers mois de mon stage à développer l'algorithme que je présente par la suite.

3 Le modèle

Afin de modéliser un réseau biologique, nous utilisons des réseaux d'automates. Chaque automate représente un gène et sa protéine et l'état de l'automate représente plus généralement l'état d'activité (ou de présence) de la protéine. Mon travail utilise des automates à deux états (0 et 1) qui modélisent donc l'activité ou la non-activité. Notre équipe a travaillé précédemment avec des automates multi-valués (plus de deux états par automate) ; cela-dit, ces automates provenant de réseaux de Thomas ont la particularité qu'il n'y a pas de "saut d'état", c'est-à-dire que les transitions se font entre états adjacents (pas de transition entre a_0 et a_2 par exemple, il y a obligatoirement passage par a_1). Cette particularité permet de transformer un automate à 3 ou 4 états en deux automates booléens. Cette transformation est présentée en **Annexe 1**. Les réseaux que j'ai rencontrés vérifiant ces conditions, j'ai choisi de limiter ma méthode aux réseaux d'automates booléens.

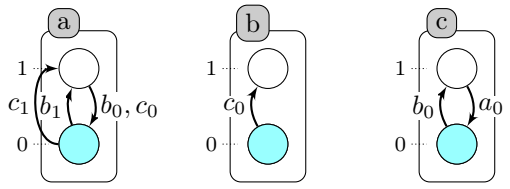


FIGURE 1 – Automates booléens

Sur la figure 1, on a 3 automates booléens, chacun dans l'état initial respectif a_0, b_0 et c_0. On a représenté par des flèches les transitions possibles d'un état à l'autre ; les labels sur les flèches sont les conditions requises pour la transition en question. Par exemple, il a deux transitions possibles pour passer de a_0 à a_1 : quand on a b_1 ou quand on a c_1. En revanche, pour passer de a_1 à a_0, il n'y a qu'une transition qui requiert b_0 et c_0 simultanément.
On appelle état local un état d'un automate particulier (par exemple a_0 est un état local) et état global un état où tous les automates sont dans un état local particulier. Par exemple, 000 ou $a_0b_0c_0$ est un état global du réseau d'auto-

mate de la figure 1. On différenciera de même les transitions locales (entre deux états locaux d'un même automate) des transitions globales (entre deux états globaux).
On trouvera une description plus formelle des réseaux d'automates dans [6] (papier de notre équipe). Cette formalisation peut apporter un meilleure compréhension du modèle mais elle n'est pas necéssaire pour comprendre la suite de ce rapport.
Pour compléter la dynamique du modèle, il faut choisir une sémantique, c'est-à-dire décider combien de transitions on peut emprunter simultanément. La sémantique réalisant toutes les transitions réalisables à chaque étape est dite synchrone ; celle où une seule transition est réalisée à la fois est dite asynchrone. Dans le cas d'automates booléens, la sémantique synchrone forme un graphe de transitions (voir figure 2) déterministe (chaque état global n'a qu'un successeur). Plusieurs études [3] ont déjà été menées sur le sujet des attracteurs de réseaux booléens synchrones. Le déterminisme du système implique que les attracteurs sont les cycles ou les points fixes. L'article [4] fournit un certain nombre d'arguments en faveur de la modélisation des réseaux de régulation biologique par la sémantique asynchrone plutôt que par la sémantique synchrone. Nous adopterons donc ici la sémantique asynchrone. Notons que le graphe de transitions est alors beaucoup plus complexe (il compte bien plus de transitions). Si le réseau contient n automates, chaque état global a jusqu'à n successeurs : le modèle est fortement indéterministe. Cet indéterminisme rend l'étude dynamique du système très gourmande en temps de calcul pour des réseaux comptant beaucoup d'automates et de transitions locales.

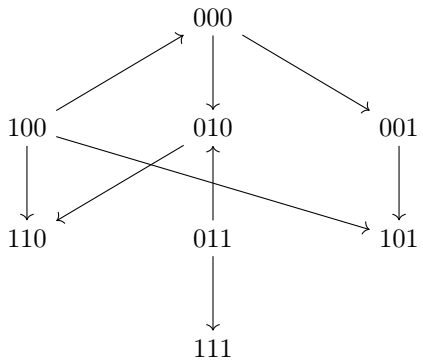


FIGURE 2 – Graphe de transition du réseau d'automates de la figure 1

La figure 2 présente le graphe de transition entre les différents états globaux possibles du système. On observe la sémantique asynchrone : un seul automate change d'état à la fois. Les successeurs d'un état global sont des états globaux semblables en tout automate sauf un. Ce graphe forme un réseau booléen particulier.

4 Attracteurs et bassins d'attraction

Pour parler d'attracteurs nous avons d'abord besoin d'une notion d'accessibilité. Sans plus de formalisation, on note qu'un état global *s'* est accessible depuis un autre état global *s* s'il existe une série de transitions dans le graphe de transitions permettant de passer de *s* à *s'*.
On définit un attracteur de la manière suivante :

Définition 1. *Un ensemble A d'états globaux est un attracteur du système si et seulement si A est :*

- ***terminal*** *: les états accessibles depuis A sont dans A.*
- ***fortement connecté*** *: tout état de A est accessible depuis tout autre état de A.*

Si A est seulement terminal, A est un ***bassin d'attraction****.*

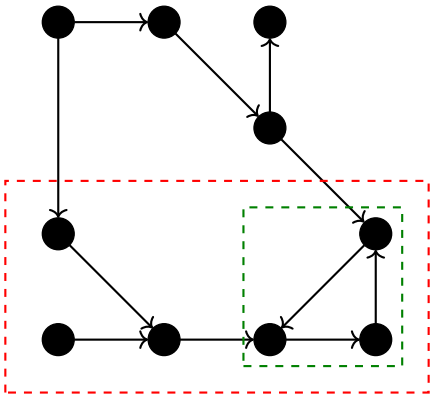


FIGURE 3 – Un attracteur et un bassin d'attraction.

La figure 3 présente un exemple d'un attracteur et un bassin d'attraction dans un graphe. Les points noirs représentent des états globaux du graphe et les flèches les transitions entre eux.

A partir de la définition 1 on peut émettre quelques remarques à propos des attracteurs et des bassins d'attraction :

- Un **attracteur** est un **bassin d'attraction**.
- Un **bassin d'attraction** contient au moins un **attracteur**.
- Un **bassin d'attraction** qui ne contient pas de **bassin** autre que lui-même est un **attracteur**.

Cette dernière remarque offre une raison supplémentaire de rechercher des bassins plutôt que des attracteurs directement. En effet, la condition "être fortement connecté" peut être très coûteuse à vérifier, alors qu'on peut trouver des sous-ensembles terminaux plus simplement. Certaines conditions dans les transitions créent des bassins où certains automates sont constants. Ce sont des bassins de cette forme que l'on va chercher. L'orientation que j'ai choisie est

donc de chercher des bassins les plus petits (au sens de l'inclusion) possible. Les bassins les plus restreints, c'est-à-dire ceux au sein desquels on n'a pas trouvé d'autres bassins, sont alors susceptibles d'être des attracteurs ou au moins d'en contenir.

Idée importante Les bassins d'attraction, même sous la forme particulière que nous allons étudier, peuvent être en trop grand nombre pour être tous calculés ou même stockés en mémoire. Heureusement, on n'a pas besoin de tous les connaître. Si on connaît deux bassins, et que leur intersection est non-vide, alors cette intersection est un bassin également. De plus, ce bassin-intersection ne contient aucun des deux premiers bassins. Ainsi, après avoir obtenu un certain nombre de bassins, on va calculer les intersections minimales que l'on peut réaliser avec ces bassins. Ces intersections minimales seront des bassins d'attraction au sein desquels notre méthode n'aura pas trouvé de sous-bassin, et donc susceptibles d'être des attracteurs.

5 La recherche de bassins

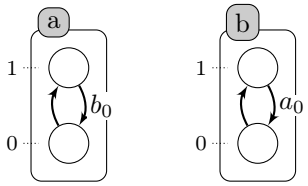


FIGURE 4 – Deux automates créant un bassin

Les transitions des deux automates de la figure 4 ci-contre créent un bassin. En effet, l'état (global) a_1b_1 est terminal : pour sortir de a_1 il nous faut b_0, mais pour aller en b_0 il nous faut a_0. En fait, tout réseau d'automate contenant ces automates a et b aura pour bassin d'attraction l'ensemble des états vérifiant a_1b_1. On notera un tel sous-ensemble d'états globaux $\{a_1, b_1, *\}$. On appelle a_1 et b_1 les *conditions* d'un tel bassin. Cette notation se généralise naturellement pour un plus grand nombre de conditions.

On va rechercher des bassins de cette forme, pour plusieurs raisons :

- **recherche :** on peut rechercher des bassins de cette forme en regardant les transitions. On évite ainsi de s'intéresser au graphe de transitions, ce qui serait coûteux et probablement nécessaire pour trouver des bassins mettant en jeu des interaction plus complexes.
- **vérification :** on peut facilement vérifier si un ensemble de conditions $\{a_1, b_1, *\}$ crée un bassin. Il suffit de comparer les conditions des transitions des automates impliqués.
- **intersection :** on peut facilement obtenir l'intersection de deux bassins, et il est immédiat de savoir si elle est vide ou non. Par exemple, l'intersection de $\{a_1, b_1, *\}$ et $\{c_0, d_0, *\}$ est $\{a_1, b_1, c_0, d_0, *\}$. L'intersection de deux bassins est vide si et seulement si ils contiennent des conditions contradictoires, par exemple a_0 et a_1.
- **sous-bassins :** les inclusions de bassins de cette forme les uns dans les autres sont également claires. Par exemple, si $\{a_1, b_1, c_0, *\}$ et $\{a_1, b_1, *\}$

sont des bassins, le premier est inclut dans le deuxième.

Il existe bien entendu des bassins d'attraction différents de ceux-ci, qui ne seront pas repérés par cette recherche. Cependant, cette approche fournit déjà une certaine quantité d'information sur la dynamique du système. En effet, les différentes intersections des bassins fournissent des zones générales de la dynamique du système. Cela est illustré par la figure 5, à partir de deux bassins ayant chacun un sous-bassin (diagonales en bleu et en rouge).

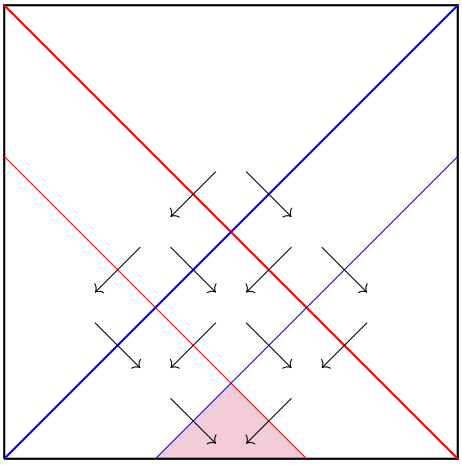


FIGURE 5 – Représentation abstraite de l'intersection de bassins d'attraction
Le carré noir représente l'ensemble du graphe de transition. Les diagonales et les flèches représentent un bassin : la "frontière" ne peut être passée que dans un sens. La zone colorée en bas représente un bassin minimal.

Nous allons désormais présenter l'algorithme de recherche de bassins. L'objectif de cet algorithme n'est pas de tester toutes les conditions pour trouver tous les bassins, car comme on l'a évoqué précédemment, le nombre de bassins, même en se limitant à ceux de la forme qui nous intéresse, peut être exponentiel par rapport au nombre d'automates. L'idée est de trouver un maximum de bassins **indépendants** ; c'est-à-dire que si un bassin C peut s'exprimer comme l'intersection de deux bassins A et B, on ne veut trouver (et stocker) que A et B (C étant alors *implicitement* connu).
Le fonctionnement général de l'algorithme est le suivant :

1. on trouve un premier bassin *e.g.* $\{a_1,b_0\}$
2. on cherche dans ce bassin un autre bassin *e.g.* $\{a_1,b_0,d_1,e_0\}$
3. on relance la recherche dans le dernier bassin trouvé
4. on procède ainsi jusqu'à obtenir un bassin **C** au sein duquel on ne trouve plus de bassin
5. on conserve ce dernier bassin et ce bassin seulement.
6. on recommence depuis l'étape **1** un certain nombre de fois.

7. on regarde finalement les *intersections minimales* de tous les bassins trouvés.

Quelques remarques sur les différentes étapes :
— la technique de recherche d'un bassin est expliquée plus bas dans le paragraphe **Recherche d'un bassin**.
— lors des étapes **3, 4, 5** où l'on cherche des sous-bassins, on oriente la recherche de manière que les sous-bassins dépendent du bassin dans lequel on les cherche.
— le dernier sous-bassin trouvé, est **minimal** pour notre recherche seulement : le fait qu'on ne trouve pas de bassin à l'intérieur ne veut pas dire qu'il n'y en a pas.
— on peut élargir la recherche en cherchant plus d'un sous-bassin à chaque étape. Cela revient à sacrifier du temps de calcul au profit de davantage d'exhaustivité.

Recherche d'un bassin Afin de trouver un bassin (depuis l'étape **1** aussi bien qu'un sous-bassin), on définit une fonction de dénombrement des transitions. L'idée est de regarder en priorité les transitions qui sont le moins souvent réalisées.

Définition 1. *On définit la fonction de dénombrement de transitions F, pour un état local a_0 et un ensemble de conditions $\boldsymbol{C}$:*

$F(a_0, \boldsymbol{C})$ *est égale au nombre d'états globaux vérifiant a_0 et $\boldsymbol{C}$ et pouvant effectuer la transition de a_0 vers a_1.*

Une transition pouvant être réalisée de plusieurs façons différentes, F peut être exprimée comme le cardinal d'une union. On peut ainsi calculer F en utilisant la formule du crible. Le calcul de cette fonction F est expliqué plus en détail en **Annexe 2**. Le calcul peut être compliqué et un peu coûteux à effectuer si la transition étudiée peut être réalisée de nombreuses façon différentes. La fonction F étant appelée régulièrement dans le processus de l'algorithme, on veut éviter ce coût. Heureusement, on n'a pas besoin de la valeur exacte de F mais seulement de pouvoir comparer ces valeurs entre elles, aussi on utilisera une approximation en limitant le calcul aux termes importants.
Une fois munis de cette fonction, on cherche des bassins en ajoutant progressivement des automates dans un état local.
— On commence par ajouter a_0 qui minimise $F(a_0, \{\})$.
— On ajoute ensuite b_0 qui minimise $F(a_0, \{b_0\}) + F(b_0, \{a_0\})$, le deuxième terme anticipant le "blocage de b_0".
— On procède de même en ajoutant c_0 qui minimise $F(a_0, \{b_0, c_0\}) + F(b_0, \{a_0, c_0\}) + F(c_0, \{a_0, b_0\})$.
— *etc.*
La procédure s'arrête quand on trouve une somme égale à zéro : tous les termes sont égaux à zéro, on a trouvé un ensemble dont aucune transition ne sort, c'est-à-dire un bassin. La procédure peut également s'arrêter sans que la somme vaille zéro ; on recommence alors avec une autre condition initiale.

6 Implémentation

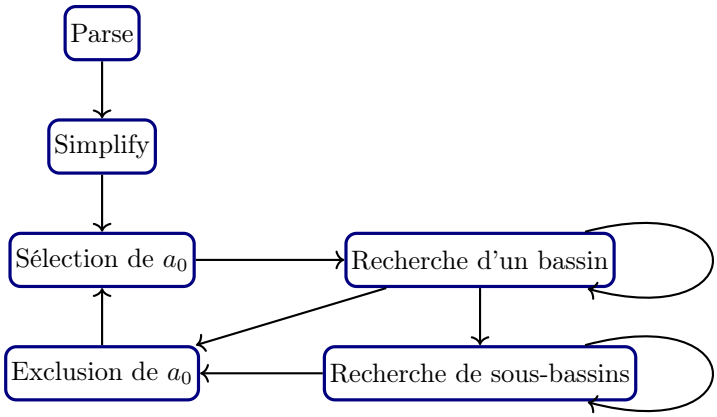


FIGURE 6 – Schéma de l'implémentation de l'algorithme

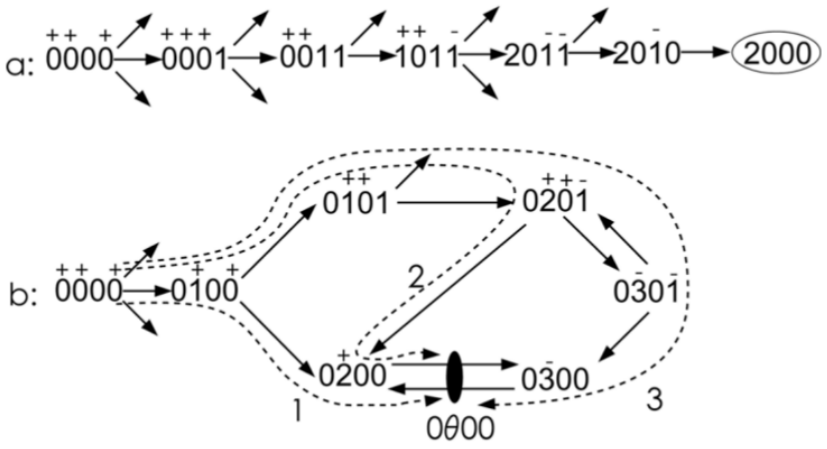
La figure 6 présente le fonctionnement général de l'algorithme. On récupère tout d'abord le modèle sous la forme d'un réseau d'automates et d'un état initial lors de l'étape *Parse*. L'étape *Simplify* est une étape de simplification du modèle : on retire un certain nombre de transitions et d'automates en fonction de l'état initial. On obtient un réseau plus petit que le réseau de départ mais qui lui est équivalent. L'étape *Sélection* correspond à la sélection de l'état local initial de la recherche de bassins. C'est à partir de cet état a_0 qu'on va commencer à ajouter d'autres états locaux jusqu'à obtenir un bassin d'attraction (ou une condition bloquante). Il est sélectionné à l'aide de la fonction de dénombrement des transitions comme évoqué précédemment. On passe ensuite à l'étape de *Recherche d'un bassin* où on va chercher à obtenir un bassin d'attraction en ajoutant progressivement des états locaux. Si on ne trouve pas de bassin, on reprend l'étape de sélection avec un autre état local initial différent de a_0 (on passe donc par l'étape d'*exclusion*). Si on trouve un bassin d'attraction, on recherche de la même façon un sous-bassin à l'intérieur (étape *Recherche de sous-bassins*), et cela récursivement jusqu'à rencontrer un sous-bassin au sein duquel on ne trouve plus de sous-bassin. On passe alors à l'étape d'*exclusion de* a_0 qui consiste à préparer le redémarrage à l'étape de *sélection* depuis un autre état initial.

7 Résultats

7.1 Résultats sur un petit modèle : modèle du phage lambda

Le modèle du phage lambda est un réseau biologique constitué de 4 gènes : *CI, Cro, CII* et *N*. Chaque gène est représenté par un automate, chaque au-

tomate ayant respectivement 3, 4, 2 et 2 états. On peut construire un réseau de 6 automates booléens équivalent à ce réseau, où les automates *CI* et *Cro* sont réprésentés par deux automates chacun. Il est connu [2, 5] que ce réseau possède deux attracteurs, dits lytique et lysogénique, qui correspondent chacun à mode de fonctionnement du phage. Ces attracteurs sont représentés sur la figure 7. En considérant les états dans l'ordre *{CI, CII, Cro, N}*, ces attracteurs correspondent aux états *2000* et *0[2-3]00* (resp. lysogénique et lytique).



Source: Taken from Thieffry (1993, p.138)

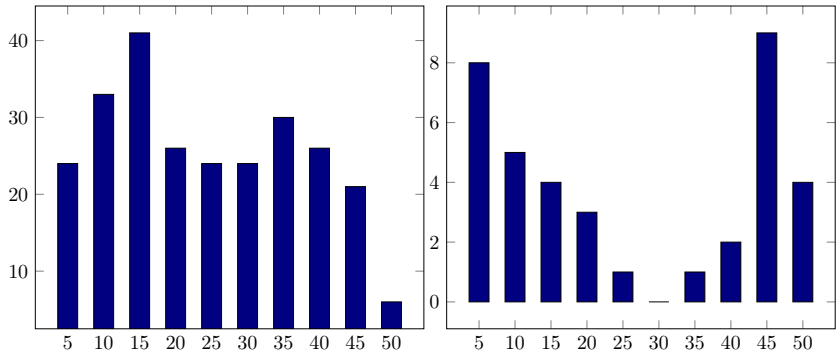
FIGURE 7 – Attracteurs du réseau du phage lambda

On utilise notre méthode pour détecter les bassins du réseau booléen équivalent comptant 6 automates. On traduit ensuite ces résultats dans le cadre du réseau original. Les bassins minimaux trouvés sont alors les suivants : *20*** et *0[2-3]00*.

Remarque : notre algorithme trouve des bassins où les automates sont constants, or on trouve ici un automate (*Cro*) variant entre les états 2 et 3. Cela est dû au fait qu'on a représenté l'automate à 4 états par deux automates à 2 états, et la traduction d'un de ces deux automates étant constant donne ce résultat. On remarquera que c'est ici un avantage.

On observe qu'on trouve exactement le cycle lytique comme étant un bassin minimal. Pour ce qui est du cycle lysogénique, on en trouve un sur-bassin où les deux derniers automates sont encore libres. Cela illustre le fait que notre méthode n'est pas exhaustive : on ne trouve pas le dernier sous-bassin. Evidemment, notre méthode est plus adaptée aux grands réseaux dans lesquels une recherche exhaustive est exclue. Cependant, on observe que même sur un très petit réseau comme celui-ci, on obtient tout de même un des attracteurs et un sur-bassin du deuxième.

7.2 Résultats sur des modèles plus larges



(a) Avec sur-bassins (b) Sans sur-bassin

FIGURE 8 – Nombre de bassins d'attraction trouvés en fonction de leur taille (en nombre d'automates)

Le modèle étudié est un réseau modélisant la différentiation des lymphocytes T auxiliaires (*T helper*, *CD4+* ou *Th*). Ces lymphocytes jouent un rôle dans la régulation de la réponse immunitaire. Sous certains conditions d'activation, ces cellules se différencient en sous-types Th spécifiques. On étudie ainsi les attracteurs de ce réseau dans le but d'identifier ces processus de différentiation.
Le réseau comporte 103 automates booléens et un total de 380 transitions locales. C'est donc un réseau à la fois très complexe et très large (de l'ordre de 10^{30} états globaux). Les études [6] et [9] proposent une étude dynamique du système (entier ou simplifié). L'approche dynamique est vite limitée pour un tel réseau en raison du temps important nécessaire aux calculs d'accessibilité. Notre méthode mi-statique mi-dynamique est conçue pour fournir rapidement de l'information sur un système aussi large, elle y est donc bien adaptée.

Résultats Les résultats sont présentés sur la figure 8. Chaque diagramme présente le nombre de bassins trouvés en fonction de leur taille. Par exemple, le premier bâton indique le nombre de bassins trouvés ayant pour conditions entre 0 et 5 automates. La figure de gauche présente les résultats en incluant les sur-bassins trouvés lors de la recherche, celle de droite sans les sur-bassins. Ces résultats sont obtenus en recommençant la recherche 100 fois à partir de conditions initiales différentes. Le temps de calcul est de l'ordre de 1.5 seconde.

8 Discussion

Répétons-le : la méthode développée ici ne prétend pas être exhaustive. Notre objectif est d'obtenir des ensembles susceptibles d'être des attracteurs, ou du moins de contenir des attracteurs. On se restreint à un certain type de bassin du

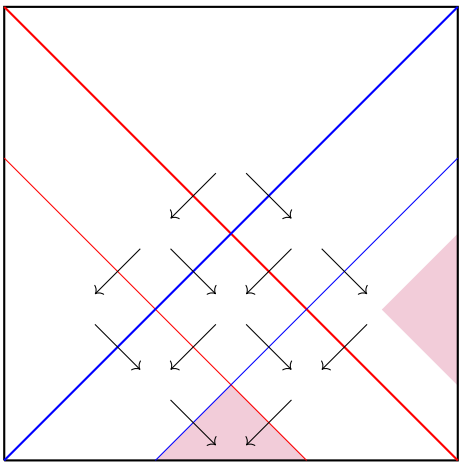


FIGURE 9 – Illustration de la non-exhaustivité de la recherche de bassins

fait que nos réseaux booléens ne sont pas quelconques mais proviennent d'automates. Ainsi, notre recherche rate sans doute un certain nombre de bassins, qu'ils soient de cette forme ou non. La figure 9 illustre la non-exhaustivité de notre processus : on peut rater certains bassins entre deux sous-bassins, et ainsi rater un attracteur.
Une autre limite que l'on peut évoquer est l'accessibilité des bassins. En effet, il peut être important d'ajouter à la définition d'un bassin d'attraction (et d'un attracteur) qu'il soit accessible depuis l'état initial du système. On garantit ici la non-accessibilité de l'état initial depuis tout élément d'un bassin (définition d'un bassin d'attraction), mais on choisit de ne pas considérer l'accessibilité du bassin. Celle-ci peut être vérifiée par un calcul d'accessibilité, mais cette possibilité n'est pas intégrée dans l'outil que nous avons développé.

Complexité Notre méthode de recherche est particulièrement rapide : la recherche d'un bassin et de ses sous-bassins a une complexité polynomiale par rapport au nombre d'automates (en fait même un peu mieux). La rapidité du processus dépend finalement du nombre de bassins que l'on désire trouver. La flexibilité de l'algorithme permet d'effectuer davantage de recherches en fonction de la quantité d'information que l'on souhaite obtenir sur le système.
Savoir combien de bassins on espère trouver présuppose cependant une bonne connaissance préalable du système. Il est a priori difficile d'estimer à l'avance quelle sera l'efficacité de notre recherche. Cette efficacité dépend en effet beaucoup de la structure particulière du système étudié. Notre approche a pour motivation d'être adaptée au cas particulier des réseaux d'automates modélisant des réseaux biologiques.

9 Conclusion

Nous avons développé ici une méthode de recherche de bassins d'attraction dans des réseaux d'automates. Si elle ne permet pas de trouver tous les bassins d'attraction du système, elle fournit cependant une quantité appréciable d'information sur sa dynamique générale et la position potentielle de ses attracteurs. Bien que notre méthode vise à être efficace sur des modèles très larges où une recherche exhaustive est exclue, on a également observé des résultats sur le modèle du phage lambda. Ce résultat nous encourage à être optimiste quant à l'exhaustivité de nos résultats sur des modèles plus larges. Il serait cependant intéressant de pouvoir établir un ordre de grandeur de la proportion de bassins qui échappent à la recherche. Cette proportion est probablement très variable en fonction de la structure du réseau et de ses bassins.
Bien que notre algorithme utilise les caractéristiques particulières des réseaux biologiques, l'essentiel du travail est informatique. La prochaine étape est donc de réfléchir à des applications biologiques directes pour notre méthode.

Références

[1] Saadatpour A., Albert I., and Albert R. Attractor analysis of asynchronous boolean models of signal transduction networks. *Journal of Theoretical Biology*, 2010.

[2] Thieffry D. and Thomas R. Dynamical behaviour of biological regulatory networks-ii. immunity control in bacteriophage lambda. *Bull Math Biol.*, 1995.

[3] Dubrova E. and Teslenko M. A sat-based algorithm for computing attractors in synchronous boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2011.

[4] Harvey I. and Bossomaier T. Time out of joint : Attractors in asynchronous random boolean networks. 1999.

[5] Ahmad J., Roux O., Bernot G., Comet J-P., and Richard A. Analysing formal models of genetic regulatory networks with delays. *Int. J. Bioinformatics Research and Applications, Vol. 4, No. 3*, 2008.

[6] Fippo Fitime L., Roux O., Guziolowski C., and Paulevé L. Identification of bifurcation transitions in biological regulatory networks using answer-set programming. *Algorithms Mol Biol.*, 2017.

[7] Soumya Paul, Cui Su, Jun Pang, and Andrzej Mizera. A decomposition-based approach towards the control of boolean networks. *https ://doi.org/none*, 2018.

[8] Thomas R. Boolean formalization of genetic control circuits. *J. theor. Biol.*, 1973.

[9] Abou-Jaoudé W., Monteiro P.T., Naldi A., Grandclaudon M., Soumelis V., Chaouiya C., and Thieffry D. Model checking to assess t-helper cell plasticity. *Front Bioeng Biotechnol*, 2015.

A Annexe 1 : passage du multivalué au booléen

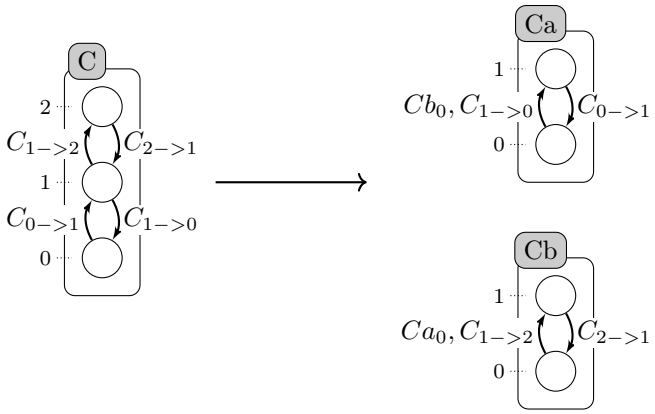


FIGURE 10 – Automate à 3 états transformé en deux automates booléens équivalents

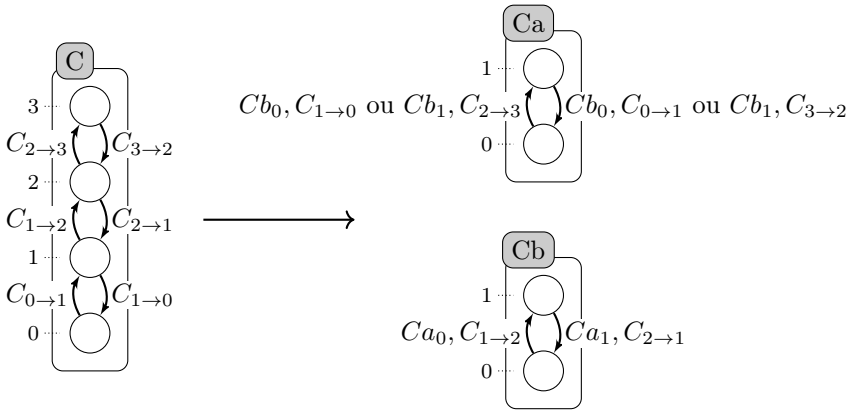


FIGURE 11 – Automate à 4 états transformé en deux automates booléens équivalents

TABLE 1 – Table de correspondance des états locaux

C	0	1	2	3
$Ca\ Cb$	10	00	01	11

B Annexe 2 : fonction de dénombrement de transitions

On a dans notre réseau un automate a dont la transition de l'état a_0 vers a_1 est régie par des conditions de la forme :

$$\begin{array}{lll} a_0 \to a_1 & & \\ & \text{when } b_0 \text{ and } c_0 & (c_1) \\ & \text{when } b_0 \text{ and } d_0 & (c_2) \\ & \text{when } b_1 \text{ and } e_0 & (c_3) \\ & \vdots & \\ & & (c_n) \end{array}$$

On veut obtenir le nombre d'états globaux vérifiant a_0 et dont sort une transition $a_0 \to a_1$. Cela revient à compter les états globaux qui vérifient au moins une des conditions $(c_i)_{1 \leq i \leq n}$ ci-dessus. On note C_i l'ensemble des états globaux vérifiant a_0 et c_i pour $1 \leq i \leq n$. Le nombre que nous cherchons à calculer est alors $card(\bigcup_{i=1}^{n} C_i)$. En appliquant la formule du crible, cela revient à calculer les intersections de toutes les combinaisons possibles entre les C_i. On souhaite éviter ce calcul très coûteux et on se limite donc aux termes d'ordre 4 ou moins (on néglige les intersections de plus de 4 C_i). Si le nombre de conditions n est inférieur ou égal à 4 (ce qui est le plus fréquent dans les réseaux étudiés ici), cette approximation est exacte. Pour $n > 4$, l'approximation est suffisante pour obtenir un ordre de grandeur suffisamment précis pour l'utilisation du nombre obtenu. En effet, on utilise ce nombre pour comparer les transitions entre elles, aussi peut-on se contenter d'un ordre de grandeur.
Le calcul des intersections se fait ensuite simplement. Par exemple, pour l'automate a ci-dessus, l'intersection de C_1 et C_2 a pour cardinal le nombre d'états globaux vérifiant a_0,b_0,c_0 et d_0. A contrario, l'intersection de C_1 ou C_2 avec C_3 est vide et a pour cardinal 0 car aucun état global ne vérifie à la fois b_0 et b_1.