



HAL
open science

Techniques de model-checking pour l'inférence de paramètres et l'analyse de réseaux biologiques

Emmanuelle Gallet

► **To cite this version:**

Emmanuelle Gallet. Techniques de model-checking pour l'inférence de paramètres et l'analyse de réseaux biologiques. Autre. Université Paris Saclay (COmUE), 2016. Français. NNT : 2016SACL035 . tel-01484341

HAL Id: tel-01484341

<https://theses.hal.science/tel-01484341v1>

Submitted on 7 Mar 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NNT : 2016SACLC035

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PARIS-SACLAY
PRÉPARÉE À CENTRALESUPÉLEC

Ecole doctorale n°573
Interfaces
Spécialité de doctorat : Informatique

par

EMMANUELLE GALLET

Techniques de model-checking pour l'inférence de paramètres
et l'analyse de réseaux biologiques

Thèse présentée et soutenue à Châtenay-Malabry, le 08 décembre 2016.

Composition du Jury :

M.	PAUL-HENRY COURNÈDE	Professeur CentraleSupélec	(Président du jury)
M.	GILLES BERNOT	Professeur Université Nice Sophia Antipolis	(Rapporteur)
Mme	MARIE BEURTON-AIMAR	Maître de conférences Université Bordeaux-I	(Rapporteur)
Mme	PASCALE LE GALL	Professeur CentraleSupélec	(Directrice de thèse)
M.	PAOLO BALLARINI	Maître de conférences CentraleSupélec	(Co-encadrant)
M.	MATTHIEU MANCENY	Enseignant-chercheur ISEP	(Co-encadrant)

Cette thèse a été préparée au

MICS

LogiMAS

CentraleSupélec

Grande Voie des Vignes

92295 Châtenay-Malabry Cedex

Site <http://www.mas.ecp.fr>

TECHNIQUES DE MODEL-CHECKING POUR L'INFÉRENCE DE PARAMÈTRES ET L'ANALYSE DE RÉSEAUX BIOLOGIQUES

Résumé

Dans ce mémoire, nous présentons l'utilisation de techniques de model-checking pour l'inférence de paramètres de *réseaux de régulation génétique* (GRN) et l'analyse formelle d'une voie de signalisation.

Le cœur du mémoire est décrit dans la première partie, dans laquelle nous proposons une approche pour inférer les paramètres biologiques régissant les dynamiques de modèles discrets de GRN. Les GRN sont encodés sous la forme d'un méta-modèle, appelé *GRN paramétré*, de telle façon qu'une instance de paramètres définit un modèle discret du GRN initial. Sous réserve que les propriétés biologiques d'intérêt s'expriment sous la forme de formules LTL, les techniques de model-checking LTL sont combinées à celles d'exécution symbolique et de résolution de contraintes afin de sélectionner les modèles satisfaisant ces propriétés. L'enjeu est de contourner l'explosion combinatoire en terme de taille et de nombre de modèles discrets. Nous avons implémenté notre méthode en Java, dans un outil appelé SP \cup TNI κ .

La seconde partie décrit une collaboration avec des neuropédiatres, qui ont pour objectif de comprendre l'apparition du phénotype protecteur ou toxique des microglies (un type de macrophage du cerveau) chez les prématurés. Cette partie exploite un autre versant du model-checking, celui du model-checking statistique, afin d'étudier un type de réseau biologique particulier : la voie de signalisation Wnt/ β -caténine, qui permet la transmission d'un signal de l'extérieur à l'intérieur des cellules via une cascade de réactions biochimiques. Nous présentons ici l'apport du model-checker stochastique COSMOS, utilisant la *logique stochastique à automate hybride* (HASL), un formalisme très expressif nous permettant une analyse formelle sophistiquée des dynamiques de la voie Wnt/ β -caténine, modélisée sous la forme d'un processus stochastique à événements discrets.

Mots clés : analyse formelle de réseaux biologiques, réseau de régulation génétique, modèle discret de Thomas, inférence de paramètres, model-checking LTL, voie Wnt/bêta-caténine, modélisation stochastique, model-checking HASL.

MODEL CHECKING TECHNIQUES FOR PARAMETER INFERENCE AND ANALYSIS OF BIOLOGICAL NETWORKS

Abstract

In this thesis, we present the use of model checking techniques for inference of parameters of *Gene Regulatory Networks* (GRNs) and formal analysis of a signalling pathway.

In the first and main part, we provide an approach to infer biological parameters governing the dynamics of discrete models of GRNs. GRNs are encoded in the form of a meta-model, called *Parametric GRN*, such that a parameter instance defines a discrete model of the original GRN. Provided that targeted biological properties are expressed in the form of LTL formulas, LTL model-checking techniques are combined with symbolic execution and constraint solving techniques to select discrete models satisfying these properties. The challenge is to prevent combinatorial explosion in terms of size and number of discrete models. Our method is implemented in Java, in a tool called SP \cup TNI κ .

The second part describes a work performed in collaboration with child neurologists, who aim to understand the occurrence of toxic or protective phenotype of microglia (a type of macrophage in the brain) in the case of preemies. We use an other type of model-checking, the statistical model-checking, to study a particular type of biological network: the Wnt/ β -catenin pathway that transmits an external signal into the cells via a cascade of biochemical reactions. Here we present the benefit of the stochastic model checker COSMOS, using the *Hybrid Automata Stochastic Logic* (HASL), that is an very expressive formalism allowing a sophisticated formal analysis of the dynamics of the Wnt/ β -catenin pathway, modelled as a discrete event stochastic process.

Keywords: formal analysis of biological networks, genetic regulatory network, Thomas discrete modelling, parameters inference, LTL model checking, Wnt/ β -catenin pathway, stochastic modelling, HASL model checking.

MICS

LogiMAS – CentraleSupélec – Grande Voie des Vignes – 92295 Châtenay-Malabry Cedex

Remerciements

Je tiens à remercier les personnes qui m'ont accompagnée au cours de ma thèse, et plus largement tout au long de mon parcours scolaire et lors du début de ma vie professionnelle.

Je voudrais tout d'abord remercier ma directrice ainsi que mes deux co-encadrants de thèse : Pascale Le Gall, Paolo Ballarini et Matthieu Manceny, pour avoir guidé mes travaux, m'avoir offert la possibilité de les présenter en conférence et m'avoir encouragée jusqu'à la fin.

Je remercie également mes rapporteurs, Marie Beurton-Amar et Gilles Bernot, pour avoir accepté de rapporter ma thèse malgré leur charge de travail et les délais courts, ainsi que Paul-Henry Cournède pour avoir accepté de faire partie de mon jury. Merci pour leurs questions et les discussions qui ont suivies lors de ma soutenance, qui m'ont permis de jeter un regard neuf sur mes travaux.

Au cours de ma thèse, j'ai eu le plaisir de travailler avec la microbiologiste Janine Guespin, le docteur Pierre Gressens et son équipe ainsi qu'avec le chercheur en informatique Loïc Paulevé ; je les remercie de m'avoir consacré du temps et semé les graines de, je l'espère, futures collaborations avec ceux qui poursuivront mon travail.

En marge de mes activités de recherche, j'avais également une charge d'enseignement à l'École Centrale. Merci une nouvelle fois à Pascale et Paolo, mais aussi à Marc Aiguier, Céline Hudelot et Wassila Ouerdane pour m'avoir confié ces enseignements et laissé une certaine liberté en me faisant participer à l'élaboration des cours, des TP et des TD. Merci aux élèves à qui j'ai enseigné et grâce à qui cette expérience fut enrichissante humainement et professionnellement, en particulier à ceux que j'ai eu le plaisir de recroiser après la fin de leurs cours avec moi.

Je remercie également les membres du personnel de mon laboratoire et de l'école doctorale qui m'ont accompagnée lors des nombreuses démarches administratives qui ont jalonné ma thèse et m'ont permis de me concentrer sur mon travail de recherche et d'enseignement : Emmanuelle Coplo, Sylvie Dervin, Annie Glomeron et Suzanne Thuron.

Je remercie également mes anciens condisciples du laboratoire MAS (MICS) avec qui j'ai eu le plaisir de partager un bureau, une aile, un couloir, des repas RU, des pique-niques, des cinémas et du théâtre (mais pas de baby-foot) : en particulier, Adrien, Anthony, Benoit, Dung, Emmanuel, Grégoire, Hichem, Imen, Jean-Christophe, José, Nassim, Nesrine, Nguyen et Sonia.

Mon parcours scolaire aurait sans doute été différent sans le soutien et les encouragements de divers professeurs que j'ai eu la chance de croiser tant au lycée que lors de mes études supérieures.

Je voudrais ainsi remercier en particulier certaines professeurs de l'isep : Denis Escaffré, qui m'a permis de prendre confiance dans ce que je pouvais réaliser, Olivier Hermant, qui fut mon professeur mais également mon tuteur d'apprentissage, et encore une fois, Matthieu, grâce à qui cette thèse a pu se faire, sur la base d'un travail préliminaire commencé sous sa direction en dernière année d'étude d'ingénieur.

Je n'oublie pas non plus mes collègues d'OWI (anciens et actuels) qui m'ont soutenue à la fin de ma thèse, en particulier Daniel, Mustafa, Niels, Patrycja, Sihem, Wei-Ming et Youssef.

Enfin, je remercie également mes amis qui ne m'ont pas tenu rigueur de mon hibernation doctorale, en particulier Carine, Charlotte, Cyril, Elsa, Florent, Hakima, Louise, Mélanie, Michaël, Nelson et Téphanie.

Je remercie également ma famille d'avoir eu la patience d'attendre la fin de mon *interminable* thèse et qui ont eux aussi subi ma période d'hibernation mais qui se sont néanmoins toujours assurés que je ne manquais ni de chocolat ni de noisette et m'ont abreuvée en cidre à la moindre occasion.

Sommaire

Résumé	v
Remerciements	vii
Sommaire	ix
Introduction	1
I Inférence de paramètres de modèles de Thomas par vérification de propriétés LTL	7
1 Préliminaires	11
2 Présentation des réseaux de régulation génétique	23
3 Méthode d'inférence de paramètres de Thomas basée sur le model-checking LTL et la résolution de contraintes	47
4 Implémentation	71
5 Etudes de cas	85
II Analyse de la voie de signalisation Wnt/β-caténine à l'aide du model-checking statistique HASL	105
6 Préliminaires	109
7 Analyse de la voie de signalisation Wnt/ β -caténine à l'aide du model-checking statistique HASL	137
Conclusion	155
Bibliographie	161
Annexes	173
A Mode d'emploi de SPuTNIk	173

B Etudes de cas traitées avec SPuTNIk	183
Table des matières	187

Introduction

Contexte

Les systèmes biologiques peuvent être vus comme une imbrication de multiples réseaux (notamment des réseaux de régulation génétique, métaboliques ou de voie de transduction) gouvernant le fonctionnement du vivant selon différentes strates d'observation (macroscopique vs microscopique par exemple). Ces réseaux mettent en jeu des entités de nature distincte (cellules, protéines, hormones, nutriments...) qui interagissent selon des modes différents (par contact, à distance proche ou éloignée, intracellulaire, extracellulaire...) et à des vitesses différentes.

Comme indiqué dans [SSP06], la biologie est une science guidée par l'expérimentation : les mécanismes en jeu sont trop complexes pour être compris *in extenso*, mais des expériences *in vivo* (au sein du vivant, i.e. sur un organisme vivant) ou *in vitro* (dans le verre, en éprouvette, i.e. en dehors d'un organisme vivant) permettent de récolter des données quantitatives ou des observations qualitatives sur un sous-système particulier.

Les techniques d'expérimentation s'améliorant, la disponibilité de données biologiques croît de plus en plus rapidement. Certains organismes ont été particulièrement étudiés et servent ainsi d'*organismes modèles* ; on peut citer par exemple le cas du virus bactériophage λ [Pta04; Opp+05].

La réalisation d'expériences *in vivo* ou *in vitro* nécessite néanmoins un investissement (matériel, humain, en cobayes, financier, temporel) important qui conduit à vouloir maîtriser au mieux la conception de ces expérimentations afin de réduire au maximum leur nombre, et à anticiper et calibrer leur portée.

Depuis déjà plusieurs années, l'informatique apporte une aide considérable en biologie ; par exemple en modélisant, simulant, analysant ou prévoyant le comportement (qualitatif) ou la quantité en terme de population ou de concentration (quantitatif) des espèces composant un système donné. Les tests *in silico* (au sein de la silice, i.e. informatiquement) ont pour avantage de réduire les investissements précédemment cités.

Les tests *in silico* peuvent par exemple éliminer à priori des expériences ou au contraire en suggérer de nouvelles. En effet, en lien avec les hypothèses de modélisation sous-jacentes aux tests *in silico*, ces tests peuvent étayer des argumentations correspondantes au schéma directeur

suivant : les expériences *in vivo* ou *in vitro* envisagées ne pourraient respectivement pas donner lieu à des observations exploitables ou au contraire, à des observations susceptibles de corroborer des pistes non usuelles (qui n'auraient pas été conceptualisées sans l'aide de l'informatique).

En informatique, les méthodes formelles regroupent l'ensemble des démarches couplant la définition d'un formalisme mathématique (impliquant en général des ingrédients de logique ou/et de mathématiques discrètes comme les graphes) et des techniques outillées d'exploitation des objets décrits par ces formalismes. Elles étaient initialement dédiées à la conception et à la vérification rigoureuses des systèmes logiciels (analyse statique, preuve, test, model-checking). En quelques mots, l'utilisation des méthodes formelles se ramène à concevoir un modèle dans le cadre du formalisme mathématique afin d'appliquer la technique associée au formalisme (qui peut parfois nécessiter l'expertise humaine). Le résultat obtenu n'est pertinent que dans le périmètre des hypothèses sous-jacentes à la démarche de modélisation. Par exemple, une preuve à la Hoare ne tient lieu d'argument de correction du logiciel que sous réserve que le compilateur et les systèmes d'exploitation soient considérés eux aussi comme exempts d'erreurs. Par le double bénéfice issu du savoir-faire de « modélisation sous hypothèses » et d'« analyse outillée de modèles », l'emploi des méthodes formelles pour la biologie des systèmes a fait ses preuves. Le succès d'écoles d'été comme *Formal Methods for Computational Systems Biology*, de conférences comme *Computational Methods for Systems Biology*, ou encore le nombre d'articles de recherche [DL04 ; CH09 ; Bon+09] attestent de l'intérêt de la communauté scientifique pour la coopération entre méthodes formelles et biologie des systèmes.

Au sein des méthodes formelles, le *model-checking* regroupe un ensemble de méthodes de vérification automatique de systèmes modélisés sous la forme d'un graphe étiqueté fini (représentant de façon plus ou moins abstraite l'ensemble des états du système et des changements d'états possibles) au regard de la satisfaction d'une spécification ou *propriété*. Les propriétés en question portent sur les chemins du graphe d'états et, en ce sens, expriment des propriétés que l'on peut qualifier de *temporelles*. Les algorithmes de model-checking visent à identifier de façon exhaustive les états du système qui vérifient une propriété temporelle donnée. Les techniques de model-checking ont émergé à partir des années 80 dans le contexte de vérification de programmes concurrents, indépendamment par CLARKE et EMERSON [CE81] d'une part, et par QUEILLE et SIFAKIS [QS82] d'autre part. Les préoccupations récurrentes de la communauté du model-checking sont essentiellement doubles : elles concernent l'expressivité des logiques temporelles en jeu et la performance des algorithmes, notamment en lien avec la taille des graphes d'états, qui peut devenir conséquente lorsqu'il s'agit d'analyser des systèmes issus d'études de cas réelles. Dans le cadre de l'analyse des systèmes réactifs, en général décrits sous la forme de sous-systèmes communicants entre eux, les techniques de model-checking sont désormais considérées comme une méthode mature, largement acceptée et pratiquée.

Depuis les années 80, la famille du model-checking s'est agrandie et diversifiée. Il existe aujourd'hui un certain nombre de types de model-checking, chacun lié à une logique tempo-

relle spécifique. À elles toutes, elles permettent de vérifier un grand nombre de propriétés des systèmes biologiques, portant sur des modèles continus ou discrets, probabilistes ou non probabilistes. Citons notamment les logiques temporelles suivantes :

- LTL (*Linear Temporal logic*) [Pnu77] : les propriétés sont exprimées sur un chemin à l'aide d'opérateurs temporels modaux ;
- CTL (*Computation Tree Logic*) [EC80] : il s'agit d'une logique de temps arborescente, i.e. on considère plusieurs chemins futurs sous la forme d'arbres, les opérateurs logiques sont tous précédés d'un quantificateur de chemin ;
- CTL* [EH86] : extension de LTL et CTL sous la forme d'une logique arborescente ;
- PCTL (*Probabilistic Computation Tree Logic*) [HJ94] : extension de CTL pour quantifier des probabilités de propriétés (temps discret) ;
- CSL (*Continuous Stochastic Logic*) [Bai+03b] : extension de PCTL pour les CTMC (*Continuous Time Markov Chain*) ;
- HASL (*Hybrid Automata Stochastic Logic*) [Bal+10] : logique linéaire statistique dédiée à la vérification de DESP (*Discrete Event Stochastic Process*).

Selon la nature des propriétés exprimées, la modélisation du système sous forme de graphes d'états intègre plus ou moins d'informations (notion de temps ou de probabilités par exemple).

Les systèmes biologiques, vus comme des systèmes interagissant avec leur environnement, possèdent beaucoup d'analogies avec les systèmes réactifs analysés au travers des techniques de model-checking. Sous réserve que le système biologique de l'étude puisse être modélisé à l'aide du formalisme sous-jacent à la technique de model-checking ciblé et que le langage de logique temporelle permette d'exprimer les propriétés biologiques d'intérêt, l'expert biologique peut appliquer les algorithmes de model-checking afin de confronter le modèle aux propriétés temporelles. Selon le contexte, les résultats peuvent corroborer une connaissance ou hypothèse biologique, et ainsi conforter la construction du modèle, ou au contraire, remettre en question soit le modèle, soit la propriété, et donner lieu à des révisions. À l'issue d'une séquence d'allers-retours, la modélisation est suffisamment affinée pour aider le biologiste à formuler de nouvelles hypothèses ou prédictions, appelées à être validées ultérieurement. Selon un scénario idéal, le résultat des expériences *in silico* peut ainsi suggérer de nouvelles pistes d'investigation en biologie. Cette démarche « modélisation-analyse par model-checking/révision de modèle » a été mise en avant dans de nombreux travaux [Bat+05 ; Fil+06 ; CFS06].

Les deux freins principaux à l'emploi du model-checking pour la biologie sont :

1. la difficulté de construire un modèle pertinent, en raison du manque de données biologiques à propos du système à traiter, ou bien en raison d'une trop grande simplification du modèle qui pourrait aller jusqu'à obtenir un modèle trop abstrait pour être exploité ;
2. la maîtrise de la complexité (en temps, en espace mémoire) de traitement pour appliquer les algorithmes de model-checking, eu égard à la taille du modèle considéré ou au nombre de variantes de modèles (par exemple paramétrés par des paramètres biologiques de valeur inconnue) à considérer ou encore à la complexité intrinsèque de la propriété.

Organisation du manuscrit

Dans ce manuscrit, nous explorons deux voies distinctes de l'emploi des techniques de model-checking pour l'aide à la compréhension de systèmes biologiques. Elles sont respectivement abordées dans la première et dans la seconde partie, suivant le plan annoncé ci-dessous.

Partie principale. La première partie de ce manuscrit concerne l'inférence des paramètres biologiques pour les *réseaux de régulation génétique* (GRN) dans le cadre de la modélisation discrète de R. Thomas [Td90 ; TTK95].

En quelques mots, la modélisation de Thomas s'appuie sur une discrétisation des concentrations des protéines en un nombre de niveaux qualitatifs (chacun correspondant à un seuil de régulation de protéines du GRN), de telle sorte que l'ensemble des états du GRN puisse donner lieu à un modèle susceptible d'être analysé au travers de propriétés logiques et temporelles (en particulier exprimées en LTL).

La variation des niveaux d'expression des protéines au cours du temps est régie par un ensemble de paramètres, appelés *paramètres du modèle de Thomas*, qui sont expérimentalement difficiles à obtenir et dont les possibilités d'instanciation sont grandes.

Dans cette partie, nous présentons une approche de vérification de propriété LTL pour les modèles d'un GRN qui étend les approches classiques de model-checking LTL avec des techniques d'exécution symbolique et de résolution de contraintes, afin de caractériser l'ensemble des modèles du GRN correspondant à une formule LTL ciblée.

La partie I est organisée selon le plan suivant :

- Chapitre 1. Ce chapitre préliminaire rassemble les notations mathématiques ainsi que certaines définitions de base (en grande majorité classiques) employées dans la partie I pour spécifier le cadre formel de modélisation des GRN. Compte tenu du formalisme mathématique employé dans nos définitions, nous avons choisi de rédiger un chapitre préliminaire essentiellement technique. Il est conseillé de s'y référer en cours de lecture.
- Chapitre 2. Ce chapitre introduit le contexte biologique étudié dans cette partie, i.e. les réseaux de régulation génétiques (GRN). Il aborde également l'état de l'art de la modélisation discrète des GRN ainsi que l'état de l'art des méthodes outillées pour inférer les paramètres de Thomas.
- Chapitre 3. Ce chapitre présente formellement notre méthode d'inférence des paramètres, basée sur le model-checking LTL combiné à des techniques d'exécution symbolique et de résolution de contraintes.
- Chapitre 4. Ce chapitre contient la description de l'implémentation de notre méthode, en particulier le détail de certaines algorithmes d'optimisation du parcours lié à la procédure de model-checking. Cette implémentation a conduit à la création d'un outil nommé SP_{VTNI} κ , dont le mode d'emploi est présenté en annexe A.

Chapitre 5. Tout au long des chapitres 2 et 3, un « exemple jouet » est utilisé à titre d’illustration. Les résultats obtenus avec SP \cup TN κ pour des études de cas plus complexes sont regroupés dans le chapitre 5 (ainsi que dans l’annexe B).

Travail collaboratif. La seconde partie de ce manuscrit porte sur l’étude des dynamiques de la voie de signalisation Wnt/ β -caténine. Il s’agit d’un travail préliminaire, issu d’une collaboration avec l’unité mixte de recherche 1141 (*Neuroprotection du cerveau en développement*) dirigée par le Dr. Pierre Gressens.

Les voies de signalisation sont des cascades de réactions biochimiques, qui permettent de transmettre des messages de l’extérieur à l’intérieur d’une cellule, permettant l’adaptation de la cellule en fonction de stimuli spécifiques. Les particularités des voies de signalisation (faible nombre d’espèces en jeu, sensibilité aux variations dues au bruit) peut amener à privilégier une modélisation stochastique.

Nous présentons dans cette partie l’utilisation du model-checker probabiliste COSMOS pour analyser les dynamiques de plusieurs modèles de la voie Wnt/ β -caténine.

Le contenu de la partie II est organisé de la manière suivante :

Chapitre 6. Ce chapitre rappelle tout d’abord les notions fondamentales nécessaires pour la compréhension du cadre de modélisation et de vérification formelle utilisé dans cette partie. Il contient ainsi des éléments de théorie des probabilités, puis introduit la représentation de réactions biochimiques sous la forme d’équations chimiques avant d’aborder la modélisation stochastique de réseaux biologiques.

Ces éléments en main, le lecteur trouvera ensuite un condensé de la description formelle du model-checking statistique sur lequel repose COSMOS, l’outil que nous utilisons dans le chapitre suivant. Ceci nous amènera à présenter le formalisme des *réseaux de Petri stochastiques généralisés* (GSPN), le *langage de spécification aux automates hybrides* (HASL), les *automates hybrides linéaires* (LHA) ainsi que la procédure d’estimation basée sur la synchronisation entre un LHA et un modèle GSPN.

Chapitre 7. Ce chapitre contient les résultats que nous avons obtenus avec COSMOS lors de notre étude de la voie de signalisation Wnt/ β -caténine. Il décrit tout d’abord la version du modèle de la voie Wnt/ β -caténine que nous avons choisie d’étudier, puis présente le modèle GSPN correspondant. Ce GSPN est associé à des spécifications au format HASL qui visent à capturer des aspects particuliers de sa dynamique (détection de pics, analyse des oscillations). Enfin, la fin du chapitre commente les résultats des expériences effectuées à l’aide de l’outil COSMOS.

Les deux parties de ce manuscrit peuvent se lire indépendamment et intègrent chacune une présentation détaillée du contexte, à la fois en terme de techniques de model-checking utilisées et de systèmes biologiques étudiés.

Contributions

La partie I contient les contributions principales de cette thèse, i.e. la mise au point et l'implémentation d'une méthode formelle d'inférence des paramètres de modèles de Thomas contrôlant les dynamiques de réseaux de régulation génétique.

Elle repose sur l'utilisation de techniques inspirées du model-checking LTL, d'exécution (semi)-symbolique et de résolution de contraintes afin de réduire l'explosion combinatoire du nombre de dynamiques à envisager. Cette méthode a été implémentée dans un outil appelé $SP\upsilon TNI\kappa$, facilement utilisable pour tester des hypothèses *in silico*.

La liste de nos publications relatives à la partie I est donnée ci-dessous (classée par type puis par audience) :

- Journal national :
[Gal+15] (version étendue de [Gal+14a], sélectionné pour un numéro spécial de TSI);
- Acte de conférence internationale avec comité de lecture :
[Gal+14b] (conférence ICFEM 2014);
- Acte de conférence nationale avec comité de lecture :
[Gal+14a] (conférence AFADL 2014);
- Poster en conférence internationale :
[Gal+13] (conférence CMSB 2013)

Par ailleurs, nous avons débuté en milieu de thèse une collaboration avec des neuropédiatres de l'unité mixte de recherche 1141 (*Neuroprotection du cerveau en développement*) dirigée par le Dr. Pierre Gressens. Dans ce cadre, nous avons mis au point des modèles de la voie de signalisation Wnt/ β -caténine sous la forme de réseaux de Petri stochastiques, qui nous ont permis d'étudier le comportement de ce réseau biologique à l'aide du model-checker COSMOS.

Cette étude a amené à la publication d'un article « préliminaire » dans une conférence internationale [Bal+14] (conférence ISoLA 2014).

Ce manuscrit a été composé en \LaTeX , en utilisant la classe `yathesis`¹. Sauf mention contraire, les illustrations présentes dans ce manuscrit ont été créées par l'auteur de ce manuscrit, en utilisant le package `PGF/TikZ`.

1. classe \LaTeX écrite par Denis Bitouzé, dédiée à l'écriture de manuscrits de thèse (ou d'HDR) en suivant les règles françaises.

Première partie

**Inférence de paramètres de modèles
de Thomas par vérification de
propriétés LTL**

Plan de la partie I

1	Préliminaires	11
1.1	Autour des ensembles	11
1.2	Contraintes sur les entiers	13
1.3	Éléments de la théorie des langages	18
1.4	Systèmes de transition	18
1.5	Logique Temporelle Linéaire	20
2	Présentation des réseaux de régulation génétique	23
2.1	Contexte : les réseaux de régulation génétique	24
2.2	État de l'art de la modélisation de GRN	26
2.3	Le modèle multivalué de R. Thomas	32
2.4	Sélection de modèles à partir d'observations temporelles	41
3	Méthode d'inférence de paramètres de Thomas basée sur le model-checking LTL et la résolution de contraintes	47
3.1	GRN Paramétré	49
3.2	Synchronisation d'un PGRN avec un automate de Büchi	58
3.3	Exécution symbolique du Produit	60
3.4	Inférence des paramètres	67
4	Implémentation	71
4.1	SPuTNIK	71
4.2	Algorithme de simplification du produit	73
4.3	Algorithme d'exécution des arbres	75
4.4	Pistes d'optimisation	79
5	Etudes de cas	85
5.1	Inductibilité de la cytotoxicité de <i>Pseudomonas aeruginosa</i>	86
5.2	Cycle de vie du phage λ	98

Préliminaires

Dans la partie principale de cette thèse, nous nous intéressons à l'utilisation de techniques de model-checking pour inférer les paramètres de réseaux de régulation génétique.

Ce chapitre contient des rappels de notations et de définitions classiques employées dans la partie I suivant le plan donné ci-dessous. Étant donnée sa nature technique, le lecteur est invité à parcourir rapidement ces préliminaires dans un premier temps, afin de s'imprégner du formalisme mathématique employé, puis à s'y référer au besoin au cours de sa lecture.

1.1	Autour des ensembles	11
1.2	Contraintes sur les entiers	13
1.3	Éléments de la théorie des langages	18
1.4	Systèmes de transition	18
1.5	Logique Temporelle Linéaire	20

1.1 Autour des ensembles

Nous utiliserons dans ce manuscrit les notations ensemblistes usuelles.

Pour A et B deux ensembles :

- A^B désigne l'ensemble des applications de l'ensemble B vers l'ensemble A .
Une fonction f de B vers A sera notée $f : B \rightarrow A$;
- 2^A désigne l'ensemble de tous les sous-ensembles de A ;
- $id_A : A \rightarrow A$ (ou simplement id quand il n'y a pas d'ambiguïté sur l'ensemble A concerné) est la fonction identité de A ;
- $|A|$ est le cardinal de l'ensemble A ;
- \emptyset représente l'ensemble vide ;

- $A \cup B$ et $A \setminus B$ sont les unions et différences ensemblistes de A et B , i.e. respectivement $\{x \mid x \in A \vee x \in B\}$ et $\{x \mid x \in A \wedge x \notin B\}$;
- pour un sous-ensemble $C \subseteq A$ et pour une application $f : A \rightarrow B$, $f|_C$ est l'application $C \rightarrow B$ vérifiant $\forall c \in C, f|_C(c) = f(c)$;
- pour A et C deux ensembles disjoints, pour $f : A \rightarrow B$ et $g : C \rightarrow B$, $f + g$ est l'application de $A \cup C$ dans B définie par :

$$\begin{cases} \forall a \in A, (f + g)(a) = f(a) \\ \forall c \in C, (f + g)(c) = g(c) \end{cases}$$

Un ensemble E est dit ordonné s'il est muni d'une relation d'ordre total \leq , i.e. une relation d'ordre vérifiant :

$$\forall x, y \in E, x \neq y \Rightarrow (x \leq y \vee y \leq x)$$

\mathbb{N} et \mathbb{Z} dénotent respectivement les ensembles des entiers naturels ($\{0, 1, 2, \dots\}$) et des entiers relatifs ($\{\dots, -2, -1, 0, 1, 2, \dots\}$). Ces deux ensembles sont munis des opérations et relations usuelles, à savoir l'addition $+$, la soustraction $-$, la multiplication \times et les relations binaires usuelles $=, \neq, <, >, \leq$ et \geq .

Un intervalle de \mathbb{N} (respectivement de \mathbb{Z}) est noté $[x, y]$ avec x et y éléments de \mathbb{N} (respectivement \mathbb{Z}) vérifiant $x \leq y$. L'intervalle $[x, y]$ est alors défini comme l'ensemble de tous les éléments z de \mathbb{N} (respectivement \mathbb{Z}) vérifiant $x \leq z$ et $z \leq y$.

Un index I est un ensemble fini ordonné dont les éléments sont appelés des indices. Par convention, notons i_1, \dots, i_n les éléments de I énumérés selon l'ordre sous-jacent à I (avec $n = |I|$).

Soit une famille d'ensemble $(A_i)_{i \in I}$ indexée par I . $\prod_{i \in I} A_i$ est le produit cartésien $A_{i_1} \times \dots \times A_{i_n}$ des ensembles A_{i_1}, \dots, A_{i_n} construit selon l'ordre sous-jacent à I . Un élément de $A_{i_1} \times \dots \times A_{i_n}$ est appelé un n -uplet et est noté de façon générique $(x_{i_1}, \dots, x_{i_n})$ avec x_{i_j} élément de A_{i_j} pour j dans $[1, n]$. Pour i indice de I , la i -ème composante de x est l'élément x_{i_j} pour peu que l'indice i ait le j -ème rang dans l'énumération des indices de I selon la relation d'ordre.

Dans le cas particulier où les ensembles A_{i_j} sont des intervalles $[a_{i_j}, b_{i_j}]$ de \mathbb{N} ou de \mathbb{Z} avec $a_{i_j} < b_{i_j}$, on considère les applications $i_j \uparrow : A_{i_1} \times \dots \times A_{i_n}$ et $i_j \downarrow : A_{i_1} \times \dots \times A_{i_n}$, avec $i_j \in I$ définies comme suit, avec $x = (x_{i_1}, \dots, x_{i_n}) \in A_{i_1} \times \dots \times A_{i_n}$:

$$i_j \uparrow (x) = \begin{cases} (x_{i_1}, \dots, x_{i_j} + 1, \dots, x_{i_n}) & \text{si } x_{i_j} < b_{i_j} \\ x & \text{si } x_{i_j} = b_{i_j} \end{cases}$$

$$i_j \downarrow (x) = \begin{cases} (x_{i_1}, \dots, x_{i_j} - 1, \dots, x_{i_n}) & \text{si } x_{i_j} > a_{i_j} \\ x & \text{si } x_{i_j} = a_{i_j} \end{cases}$$

Ainsi, les fonctions $i_j \uparrow$ et $i_j \downarrow$ sont des applications qui associent à un n -uplet d'entiers, un n -uplet dont la composante indexée par i_j est respectivement augmentée et diminuée d'une

unité, sauf si la composante en question correspond à la borne de l'intervalle définissant le domaine de définition de la composante (A_{i_j}).

Dans la suite, pour des soucis de lisibilité, les ensembles A_{i_j} seront souvent laissés implicites et nous privilégierons les notations postfixées $x[i_j \uparrow]$ et $x[i_j \downarrow]$ respectivement aux notations $i_j \uparrow(x)$ et $i_j \downarrow(x)$.

Exemple 1. Choisissons l'index $I = \{\alpha, \beta\}$ avec $\alpha < \beta$ et les intervalles $[0, 2]$ pour A_α et $[0, 1]$ pour A_β . Considérons par exemple l'élément $x_0 = (1, 1)$ de $A_\alpha \times A_\beta$. On obtient alors :

$$\begin{aligned} x_0[\alpha \uparrow] &= (2, 1) \\ x_0[\alpha \downarrow] &= (0, 1) \\ x_0[\beta \uparrow] &= (1, 1) \\ x_0[\beta \downarrow] &= (1, 0) \end{aligned}$$

Notation 1.1.1. Soit A un ensemble. Un multi-ensemble M sur A est une application $M : A \rightarrow \mathbb{N}$. Pour a élément de A , $M(a)$ est appelé la multiplicité de a et représente le nombre d'occurrences de l'élément a dans le multi-ensemble M .

Pour l'ensemble $A = \{\alpha, \beta, \gamma\}$, le multi-ensemble M défini sur A par $M(\alpha) = 1$, $M(\beta) = 2$ et $M(\gamma) = 0$ est aussi noté de la façon suivante : $\{\alpha, \beta, \beta\}$.

1.2 Contraintes sur les entiers

Dans cette section, nous introduisons la famille de formules portant sur les entiers que nous serons amenés à utiliser. Ces formules sont susceptibles d'être prises en charge par les solveurs de contraintes portant sur les domaines finis.

Nous serons amenés à simplifier ces formules dans un souci d'efficacité de nos algorithmes, c'est pourquoi nous avons choisi un ensemble restreint de formules, simplement appelées *contraintes* par la suite.

Définition 1.2.1. Soit V un ensemble de variables.

L'ensemble $Contraintes(V)$ des contraintes définies sur V est l'ensemble défini inductivement par :

- la constante \top , représentant la valeur de vérité « vrai », est dans $Contraintes(V)$;
- la constante \perp , représentant la valeur de vérité « faux », est dans $Contraintes(V)$;
- les atomes de la forme ¹

$$x \times v \bowtie c$$

avec x et c deux éléments de \mathbb{Z} , v une variable de V et \bowtie un symbole dans l'ensemble $\{=, \neq, <, >, \leq, \geq\}$, sont dans $\text{Contraintes}(V)$;

- $\neg\varphi$ est dans $\text{Contraintes}(V)$ pour φ dans $\text{Contraintes}(V)$,
- $\varphi_1 \odot \varphi_2$ est dans $\text{Contraintes}(V)$ pour φ_1 et φ_2 dans $\text{Contraintes}(V)$ et pour \odot élément de l'ensemble $\{\wedge, \vee, \Rightarrow\}$.

Exemple 2. Soit φ_0 la formule $(3v_1 \leq 7) \wedge (v_2 \neq 3)$. φ_0 est une formule de $\text{Contraintes}(\{v_1, v_2\})$.

Notation 1.2.1. Pour φ une contrainte, on note $\text{Var}(\varphi) \subseteq V$ l'ensemble des variables apparaissant dans φ .

Exemple 3. Ainsi, $\text{Var}(\varphi_0)$ est l'ensemble $\{v_1, v_2\}$.

Définition 1.2.2. Une interprétation des variables de V est une application $\nu : V \rightarrow \mathbb{Z}$.

Définition 1.2.3. Soit $\nu : V \rightarrow \mathbb{Z}$ une interprétation et soit φ une contrainte de $\text{Contraintes}(V)$.

ν satisfait φ , noté $\nu \models \varphi$, si et seulement si :

- si φ est \top , ν satisfait \top ;
- si φ est \perp , ν ne satisfait pas \top ;
- si φ est de la forme $x \bowtie v \bowtie c$ avec x et c dans \mathbb{Z} , v variable de V et \bowtie dans $\{=, \neq, <, >, \leq, \geq\}$, alors ν satisfait φ si et seulement si $x \bowtie \nu(v) \bowtie c$, vue comme une expression dans \mathbb{Z} , est interprétée comme vraie avec les interprétations usuelles de \times et \bowtie dans \mathbb{Z} ;
- si φ est de la forme $\neg\psi$, alors ν satisfait φ si et seulement si ν ne satisfait pas ψ ;
- si φ est de la forme $\varphi_1 \wedge \varphi_2$, ν satisfait φ si et seulement si ν satisfait φ_1 et ν satisfait φ_2 ;
- si φ est de la forme $\varphi_1 \vee \varphi_2$, ν satisfait φ si et seulement si ν satisfait φ_1 ou ν satisfait φ_2 ;
- si φ est de la forme $\varphi_1 \Rightarrow \varphi_2$, ν satisfait φ si et seulement si, si ν satisfait φ_1 , alors ν satisfait φ_2 .

1. En pratique, le symbole \times sera omis si cela ne nuit pas à la lecture de la formule.

Définition 1.2.4. Une interprétation ν satisfait un ensemble Φ de contraintes de $\text{Contraintes}(V)$ si et seulement si pour toute contrainte φ de Φ , $\nu \models \varphi$.
On note alors $\nu \models \Phi$.

Définition 1.2.5. Une contrainte φ de $\text{Contraintes}(V)$ est dite

- *satisfaisable* s'il existe une interprétation $\nu : V \rightarrow \mathbb{Z}$ vérifiant $\nu \models \varphi$;
- *valide* si pour toute interprétation $\nu : V \rightarrow \mathbb{Z}$, on a $\nu \models \varphi$.

Définition 1.2.6. Pour deux formules φ_1 et φ_2 de $\text{Contraintes}(V)$,

- φ_1 est dite incluse dans φ_2 si et seulement si la formule $\varphi_1 \Rightarrow \varphi_2$ est valide, i.e. pour toute interprétation $\nu : V \rightarrow \mathbb{Z}$, si $\nu \models \varphi_1$ alors $\nu \models \varphi_2$.
- φ_1 et φ_2 sont dites équivalentes à un ensemble de contraintes $\Phi \subseteq \text{Contraintes}(V)$ près, et on note $\varphi_1 \equiv_{\Phi} \varphi_2$, si et seulement si pour toute interprétation $\nu : V \rightarrow \mathbb{Z}$ vérifiant $\nu \models \Phi$, $\nu \models \varphi_1$ ssi $\nu \models \varphi_2$.

En particulier, deux formules φ_1 et φ_2 sont dites équivalentes, notées $\varphi_1 \equiv \varphi_2$, si elles sont équivalentes à l'ensemble vide \emptyset de contraintes près.

Définition 1.2.7. Soit $U \subseteq V$ un sous-ensemble des variables, φ une contrainte de $\text{Contraintes}(V)$ et $\nu : U \rightarrow \mathbb{Z}$ une interprétation définie sur l'ensemble des variables U .

$\llbracket \varphi \rrbracket_{\nu}$ est la contrainte définie inductivement par :

- si φ est \top ou \perp , $\llbracket \varphi \rrbracket_{\nu}$ est φ ;
- si φ est de la forme $x \times v \bowtie c$ avec x et c dans \mathbb{Z} , et \bowtie dans $\{=, \neq, <, >, \leq, \geq\}$,
 - si v est une variable de U alors $\llbracket \varphi \rrbracket_{\nu}$ est \top (respectivement \perp) si l'expression (sans variable) $x \times \nu(v) \bowtie c$ est interprétée comme vraie (respectivement fausse) avec les interprétations usuelles de \times et \bowtie dans \mathbb{Z} ,
 - si v est une variable de $V \setminus U$, alors $\llbracket \varphi \rrbracket_{\nu}$ est φ ;
- si φ est de la forme $\neg \psi$
 - si $\llbracket \psi \rrbracket_{\nu}$ est \top , alors $\llbracket \varphi \rrbracket_{\nu}$ est \perp ,
 - si $\llbracket \psi \rrbracket_{\nu}$ est \perp , alors $\llbracket \varphi \rrbracket_{\nu}$ est \top ,
 - sinon, $\llbracket \varphi \rrbracket_{\nu}$ est $\neg \llbracket \psi \rrbracket_{\nu}$;

- si φ est de la forme $\varphi_1 \wedge \varphi_2$,
 - si $\llbracket \varphi_1 \rrbracket_v$ ou $\llbracket \varphi_2 \rrbracket_v$ est \perp , alors $\llbracket \varphi \rrbracket_v$ est \perp ,
 - si $\llbracket \varphi_1 \rrbracket_v$ (respectivement $\llbracket \varphi_2 \rrbracket_v$) est \top , alors $\llbracket \varphi \rrbracket_v$ est $\llbracket \varphi_2 \rrbracket_v$ (respectivement $\llbracket \varphi_1 \rrbracket_v$),
 - sinon, $\llbracket \varphi \rrbracket_v$ est $\llbracket \varphi_1 \rrbracket_v \wedge \llbracket \varphi_2 \rrbracket_v$;
- si φ est de la forme $\varphi_1 \vee \varphi_2$,
 - si $\llbracket \varphi_1 \rrbracket_v$ ou $\llbracket \varphi_2 \rrbracket_v$ est \top , alors $\llbracket \varphi \rrbracket_v$ est \top ,
 - si $\llbracket \varphi_1 \rrbracket_v$ (respectivement $\llbracket \varphi_2 \rrbracket_v$) est \perp , alors $\llbracket \varphi \rrbracket_v$ est $\llbracket \varphi_2 \rrbracket_v$ (respectivement $\llbracket \varphi_1 \rrbracket_v$),
 - sinon, $\llbracket \varphi \rrbracket_v$ est $\llbracket \varphi_1 \rrbracket_v \vee \llbracket \varphi_2 \rrbracket_v$;
- si φ est de la forme $\varphi_1 \Rightarrow \varphi_2$,
 - si $\llbracket \varphi_1 \rrbracket_v$ est \perp , alors $\llbracket \varphi \rrbracket_v$ est \top ,
 - si $\llbracket \varphi_1 \rrbracket_v$ est \top , alors $\llbracket \varphi \rrbracket_v$ est $\llbracket \varphi_2 \rrbracket_v$,
 - sinon, $\llbracket \varphi \rrbracket_v$ est $\llbracket \varphi_1 \rrbracket_v \Rightarrow \llbracket \varphi_2 \rrbracket_v$.

Exemple 4. Reprenons la contrainte φ_0 (définie comme étant la formule $(3v_1 \leq 7) \wedge (v_2 \neq 3)$) :

- pour $\nu_1 : \{v_2\} \rightarrow \mathbb{Z}$ avec $\nu_1(v_2) = 2$, on obtient $\llbracket \varphi_0 \rrbracket_{\nu_1} = (3v_1 \leq 7)$;
- pour $\nu_2 : \{v_2\} \rightarrow \mathbb{Z}$ avec $\nu_2(v_2) = 3$, on obtient $\llbracket \varphi_0 \rrbracket_{\nu_2} = \perp$.

La définition 1.2.7 appelle quelques commentaires :

1. Remarquons que par construction, si $\nu : V \rightarrow \mathbb{Z}$ est une interprétation des variables de V , alors tous les atomes apparaissant dans φ donnent lieu à une interprétation en vrai ou faux. Par conséquent, $\llbracket \varphi \rrbracket_v$ est nécessairement de la forme \top ou \perp et vaut \top (respectivement \perp) si et seulement si $\nu \models \varphi$ (respectivement $\nu \not\models \varphi$).

Si $U \subsetneq V$, alors $\llbracket \varphi \rrbracket_v$ pour $\nu : U \rightarrow \mathbb{Z}$ est une contrainte de $\text{Contraintes}(V \setminus U)$, vérifiant pour toute interprétation $\rho : V \rightarrow \mathbb{Z}$ telle que $\rho|_U = \nu$, alors $\rho \models \varphi$ si et seulement si $\rho|_{V \setminus U} \models \llbracket \varphi \rrbracket_v$. Ainsi, ν pour $U \subsetneq V$ peut être vue comme une interprétation partielle des variables de φ .

2. Intuitivement, pour une interprétation $\nu : U \rightarrow \mathbb{Z}$,

$$\llbracket _ \rrbracket_v : \text{Contraintes}(V) \rightarrow \text{Contraintes}(V \setminus U)$$

est une application associant à toute contrainte φ une contrainte ϕ équivalente à φ à l'ensemble Cst_ν de contraintes $\{u = \nu(u) \mid u \in U\}$ près, i.e :

$$\forall \varphi \in \text{Contraintes}(V), \varphi \equiv_{Cst_\nu} \llbracket \varphi \rrbracket_v$$

Par construction, l'application $\llbracket _ \rrbracket_v$ donnée ci-dessus respecte la structure de la formule donnée en argument.

L'implémentation de cette application fournira une contrainte équivalente à la formule donnée en argument à Cst_v près, éventuellement sous un autre format que celui de la définition (par exemple sous un format privilégié, de type forme normale conjonctive).

3. Pour deux sous-ensembles disjoints U_1 et U_2 de V , considérons deux interprétations $v_1 : U_1 \rightarrow \mathbb{Z}$ et $v_2 : U_2 \rightarrow \mathbb{Z}$. Les deux contraintes $\llbracket \llbracket \varphi \rrbracket_{v_1} \rrbracket_{v_2}$ et $\llbracket \llbracket \varphi \rrbracket_{v_2} \rrbracket_{v_1}$ sont équivalentes. Nous utiliserons les notations $\llbracket \varphi \rrbracket_{v_1, v_2}$ ou $\llbracket \varphi \rrbracket_{v_2, v_1}$ pour noter une contrainte équivalente à φ à $Cst_{v_1} \cup Cst_{v_2}$ près.

Définition 1.2.8. Soit I un index dont les indices rangés selon la relation d'ordre associée à I sont i_1, \dots, i_n . Soit $V = \{v_{i_1}, \dots, v_{i_n}\}$ un ensemble de variables indexées par I et soit $x = (x_{i_1}, \dots, x_{i_n})$ un n -uplet de \mathbb{Z}^n indexé par I .

Nous notons $v_x : V \rightarrow \mathbb{Z}$ l'interprétation associant à chaque variable $v_{i_j} \in V$ l'élément x_{i_j} du n -uplet x , i.e. $\forall j, 1 \leq j \leq n, v_x(v_{i_j}) = x_{i_j}$.

Restriction à \mathbb{N} En pratique, dans la suite, les domaines des variables pourront être restreints à l'ensemble des entiers positifs \mathbb{N} , par exemple si les variables satisfont à des contraintes de domaines finis sur \mathbb{N} de type $v \leq b \wedge v \geq a$ avec a et b entiers naturels vérifiant $a \leq b$ (aussi simplement dénotées $v \in [a, b]$). En présence de telles contraintes de domaines, il est possible de considérer les seules interprétations de la forme $v : V \rightarrow \mathbb{N}$.

Problème de satisfaction de contraintes Notre contexte d'utilisation des contraintes dans les prochains chapitres sera essentiellement celui du *problème de satisfaction de contraintes* dans un domaine fini à valeurs dans \mathbb{N} ; les contraintes considérées porteront sur des variables dont le domaine de valeurs est lui-même borné grâce à des contraintes, ce qui réduit d'emblée le domaine des valeurs possibles.

Un problème de satisfaction de contraintes classique est celui connu sous le nom du problème des « huit dames », qui consiste à trouver comment placer huit dames sur un plateau de huit cases sur huit, de manière à ce que chaque dame soit seule sur sa rangée, sa colonne et sa diagonale.

Suivant le problème à traiter, plusieurs résultats peuvent être recherchés : obtention d'au moins une solution (i.e. une interprétation satisfaisant l'ensemble des contraintes), recherche de toutes les solutions ou bien vérification qu'il n'existe pas de solution.

L'idée générale de la résolution est de parcourir l'espace de recherche en construisant des arbres de recherche et en cherchant à optimiser leur parcours. Les *solveurs de contraintes* reposent sur des procédures qui leurs sont propres (méthodes de filtrage par propagation de contraintes pour éliminer des solutions, heuristiques de parcours de l'arbre de recherche pour atteindre rapidement les domaines intéressants...), qui leur permettent d'être plus ou moins efficaces

suyant le problème à traiter.

Dans notre cas, nous avons ainsi eu recours aux solveurs Choco [CHO10] et Z3 [DB08]. Nous reviendrons sur nos choix techniques dans le chapitre 4, section 4.1.2.

1.3 Éléments de la théorie des langages

Dans la théorie des langages, on appelle *alphabet* un ensemble dont les éléments sont des *lettres* (ou *symboles*). A partir de ces lettres, des *mots* (aussi appelés *séquences*) *finis* ou *infinis* peuvent être composés.

Mots finis Soit A un alphabet. Un mot w de longueur $|w| = n$ sur A est une suite de n lettres, qu'on définit comme une application de $[1, n]$ vers A . Pour chaque entier i dans $[1, n]$, on note a_i la lettre associée par un mot $w : [1, n] \rightarrow A$; ce mot est lui-même noté $a_1.a_2.\dots.a_n$.

On note A^* l'ensemble des mots finis sur A . A^* contient en particulier le mot vide, noté ε , de longueur 0. A^* est muni de l'opérateur de concaténation « . », pour lequel ε est élément neutre. Soit $w = a_1.\dots.a_n$ et $w' = a'_1.\dots.a'_p$ (avec $n, p \in \mathbb{N}$) deux mots de A^* , alors $w.w'$ est le mot $a_1.\dots.a_n.a'_1.\dots.a'_p$ (de longueur $n + p$ par construction).

Pour $w = a_1.\dots.a_n$ dans A^* de longueur non nulle et pour i dans \mathbb{N} avec $1 \leq i \leq n$, $w[i]$ désigne la i -ème lettre (i.e. a_i) de w . D'autre part, pour $i \in [1, n]$, on note w_i : le i -ème suffixe de w . Par exemple, pour $w = a_1.\dots.a_n$, $w_1 = w$ et $w_2 = a_2.\dots.a_n$.

Enfin, pour w mot sur A et p entier de \mathbb{N} , w^p dénote le mot obtenu en concaténant p fois le mot w . Ainsi w^0 est ε , w^2 est le mot $w.w$.

Mots infinis De la même façon, on définit les mots infinis sur A , dont l'ensemble est noté A^ω . Un mot infini sur A est une application de \mathbb{N}^* vers A . Pour $w : \mathbb{N}^* \rightarrow A$, le i -ème symbole de w est $w(i)$ avec $i \in \mathbb{N}^*$, et on note abusivement w sous la forme $a_1.\dots.a_i.\dots$. Par souci d'homogénéité avec les notations des mots finis, $w(i)$ est aussi noté $w[i]$. Le suffixe quant à lui sera noté de la même manière que pour un mot fini : w_i : (avec $i \geq 1$) est le mot infini défini par $\forall j \in \mathbb{N}^*, w_i(j) = w(i + j - 1)$. En particulier, on a $w_1 = w$.

Soit w un mot fini sur A de longueur k . w^ω désigne le mot infini obtenu en concaténant w à lui-même infiniment souvent. Autrement dit, w^ω s'écrit $w.w.w.w.\dots$ avec $\forall j \in [1, k], \forall n \in \mathbb{N}, w^\omega[j + k \times n] = w[j]$.

1.4 Systèmes de transition

Dans le cadre de cette partie, les systèmes que nous souhaitons modéliser peuvent être spécifiés avec des *systèmes de transition*. Les systèmes de transitions permettent de modéliser le comportement d'un système à l'aide :

- d'un ensemble d'états représentant abstraitement des étapes ou configurations particulières du système
- d'un ensemble de transitions reliant deux états et indiquant les changements d'état possibles : une transition relie ainsi un état dit *courant* à un état dit *futur* ou *successeur*.

Dans ce manuscrit, nous ne considérons que des systèmes de transition finis, i.e. pour lesquels les ensembles d'états et de transitions sont finis. De plus, par défaut, nos systèmes de transition seront non-déterministes, c'est-à-dire sans restriction sur le nombre de successeurs d'un même état.

Selon le contexte, les états et les transitions pourront être munis d'informations, aussi appelées de façon générique *étiquettes*, en général capturées par une application dédiée ayant pour ensemble de départ respectif les états ou les transitions. On parle alors de système de transition *étiqueté* (abrégié en LTS, pour *Labelled Transition System*).

Définition 1.4.1 (Système de transition étiqueté). Un système de transition étiqueté est

un sextuplet $\mathcal{LTS} = (S, S_0, L_e, L_t, l_e, \rightarrow)$ avec :

- S un ensemble d'états fini ;
- $S_0 \subseteq S$ l'ensemble des états initiaux ;
- L_e un ensemble dont les éléments sont ici appelés *étiquettes d'états* ;
- L_t un ensemble dont les éléments sont ici appelés *étiquettes de transitions* ;
- $l_e : S \rightarrow L_e$ application associant à chaque état son étiquette ;
- $\rightarrow \subseteq S \times L_t \times S$ une relation ternaire sur S , appelée *relation de transition*.

Soit un système de transition étiqueté $\mathcal{LTS} = (S, S_0, L_e, L_t, l_e, \rightarrow)$.

Notation 1.4.1. Une transition $tr = (s, l, s')$ de \rightarrow sera notée sous la forme $s \xrightarrow{l} s'$. De plus, nous utiliserons les notations suivantes :

- $source(tr)$,
- $label(tr)$,
- $target(tr) = s'$

pour accéder respectivement à l'état source (s), l'étiquette (l), et l'état cible (s') de la transition tr .

Un *chemin fini* σ de \mathcal{LTS} est une séquence finie d'états consécutifs $\sigma = s_1.s_2.\dots.s_n \in S^*$ telle que : $\forall i \in [1, n-1], \sigma[i] \xrightarrow{l_i} \sigma[i+1]$ est une transition de \mathcal{LTS} .

De la même façon, un *chemin infini* σ de \mathcal{LTS} est une séquence infinie d'états consécutifs $\sigma = s_1.s_2.\dots \in S^\omega$ telle que : $\forall i \in \mathbb{N}^+, \sigma[i] \xrightarrow{l_i} \sigma[i+1]$ est une transition de \mathcal{LTS} .

Si $\sigma[1] \in S_0$, le chemin (fini ou infini) est dit *initialisé*; on dira également dans ce cas que σ est une exécution de LTS à partir de $\sigma[1]$.

De plus, un état s appartient au chemin σ (noté $s \in \sigma$) s'il existe $i \in \mathbb{N}^+$ tel que $\sigma[i] = s$.

Par la suite, on s'intéressera à une famille particulière de systèmes de transition qui vérifie la propriété suivante :

Propriété 1.4.1. *Soit V un ensemble de variables.*

Tous les états du système interprètent les formules de $Contraintes(V)$ en vrai ou faux. En pratique, cela signifie que la fonction l_e permet d'associer à chaque état une valeur dans \mathbb{Z} ou \mathbb{N} pour toutes les variables de V , i.e. que l'étiquette de chaque état e contient une interprétation $v_e : V \rightarrow \mathbb{Z}$.

Un tel LTS sera dit défini sur V dans la suite.

Abus de notation (Système de transition). Lorsque des composantes d'un système de transition seront soit triviales (i.e. sans intérêt) soit évidentes d'après le contexte, elles pourront être omises lors de la donnée des systèmes de transition.

Ainsi, un système présenté sous la forme d'un couple (S, \rightarrow) désignera un système de transition dont les états et les transitions ne sont pas étiquetés, et dont tous les états sont des états initiaux. Nous aurons aussi l'occasion de manipuler des systèmes de transition où seuls les transitions (respectivement états) sont étiquetés, et pour lesquels le 6-uplet définissant le système de transition étiqueté sera réduit de une (respectivement deux) composante.

1.5 Logique Temporelle Linéaire

La *Logique Temporelle Linéaire* (LTL) [Pnu81] est une logique utilisée pour la vérification automatisée des systèmes réactifs; elle permet d'exprimer des propriétés d'un système en fonction du temps (le temps étant vu comme une succession d'étapes discrètes). Une formule LTL décrit les propriétés temporelles d'une succession infinie d'états du système; autrement dit, elle caractérise les propriétés des chemins infinis correctement initialisés.

LTL est souvent présentée comme étant une surcouche d'un fragment de logique propositionnelle. Les atomes propositionnels en jeu décrivent l'état courant du système (dans l'exemple classique de modélisation d'un ascenseur, ils peuvent correspondre en langage naturel à : portes ouvertes ou fermées, bouton d'appel enclenché...) tandis que la surcouche est constituée des *opérateurs temporels* suivants [BK08] :

- **X** : « au prochain état » ;
- **G** : « dans tous les états » ;
- **F** : « dans un état futur indéterminé » ;
- **U** : « jusqu'à un état donné » .

Dans notre cas, ce qui tiendra lieu de fragment propositionnel correspond à l'ensemble des formules $Contraintes(V)$ précédemment introduites. Ainsi, l'état courant de nos systèmes sera caractérisé par des variables entières auxquelles seront associées des valeurs, autrement dit par une interprétation $v : V \rightarrow \mathbb{Z}$.

Propriété 1.5.1. Soit $\mathcal{S} = (S, S_0, L_e, L_t, l_e, \rightarrow)$ un système de transition défini sur V (i.e. pour lequel une interprétation $v_e : Contraintes(V) \rightarrow \mathbb{Z}$ est associée à chaque état $e \in S$ via l'application l_e).

Un chemin σ du système de transition \mathcal{S} satisfait une formule φ , notée $\sigma \models \varphi$, si et seulement si :

- pour φ dans $Contraintes(V)$, $v_{\sigma[1]} \models \varphi$
- pour $\varphi = \neg\varphi_1$ ou $\varphi = \varphi_1 \Delta \varphi_2$ avec $\Delta \in \{\wedge, \vee, \Rightarrow\}$, la sémantique usuelle des connecteurs booléens s'applique (e.g. pour $\varphi = \neg\varphi_1$, $\sigma \models \varphi$ ssi $\sigma \not\models \varphi_1$)
- pour $\varphi = \mathbf{X}\varphi_1$, $\sigma_{:2} \models \varphi_1$;
- pour $\varphi = \mathbf{G}\varphi_1$, $\forall i \in \mathbb{N}^*$, $\sigma_{:i} \models \varphi_1$
- pour $\varphi = \mathbf{F}\varphi_1$, $\exists i \in \mathbb{N}^*$, $\sigma_{:i} \models \varphi_1$
- $\varphi = \varphi_1 \mathbf{U} \varphi_2$, $\exists j \in \mathbb{N}^*$, $\sigma_{:j} \models \varphi_2$ et $\forall 1 \leq i < j$, $\sigma_{:i} \models \varphi_1$.

Automate de Büchi Un automate de Büchi [Buc60] est un automate fini défini comme suit :

Définition 1.5.1 (Automate de Büchi sur V). Un automate de Büchi sur un ensemble de variables V est un quadruplet $B = (Q_B, q_B^0, A_B, \delta_B)$ où :

- $(Q_B, \{q_B^0\}, \emptyset, Contraintes(V), l_\emptyset, \delta_B)$ est un système de transition étiqueté avec l_\emptyset la fonction partout indéfinie ;
- $A_B \subseteq Q_B$ est l'ensemble des états acceptants.

Un chemin initialisé σ est dit reconnu par B si et seulement si $\text{inf}(\sigma) \cap A_B \neq \emptyset$ où $\text{inf}(\sigma)$ est l'ensemble des états de Q_B se retrouvant infiniment souvent dans σ .

Produit d'un système de transition et d'un automate de Büchi Le produit d'un système de transition et d'un automate de Büchi amène à considérer le système de transition dont la définition est donnée ci-dessous :

Définition 1.5.2 (Produit $\mathcal{S} \otimes B$). Soit V un ensemble de variables. Pour $\mathcal{S} = (S, S_0, L_e, L_t, l_e, \rightarrow)$ un système de transition défini sur V et pour $B = (Q_B, q_B^0, A_B, \delta_B)$ un automate de Büchi défini sur V , nous notons $\mathcal{S} \otimes B = (S \times Q_B, S_0 \times \{q_B^0\}, \delta)$ le système de transition dont la relation de transition δ est définie ainsi : $((e, q_B), (e', q'_B)) \in \delta$ si et seulement s'il existe une transition $(q_B, \varphi, q'_B) \in \delta_B$ et une transition $e \rightarrow e'$ telle que $v_e \models \varphi$.

Notation 1.5.1 (Composante d'un chemin). Soit \mathfrak{S} un chemin associé à un produit.

Pour i dans $\{1, 2\}$, nous notons \mathfrak{S}_i la projection de \mathfrak{S} sur sa i^{e} composante.

Propriété 1.5.2. Soit φ une formule LTL sur un ensemble de variables V et $\mathcal{S} = (E, \rightarrow)$ un système de transition défini sur V .

Il existe un automate de Büchi $B = (Q_B, q_B^0, A_B, \delta_B)$ sur V tel que : pour tout chemin σ de \mathcal{S} , $\sigma \models \varphi$ si et seulement il existe un chemin \mathfrak{S} de $\mathcal{S} \otimes B$, tel que $\mathfrak{S}_{|1} = \sigma$ et $\mathfrak{S}_{|2}$ est reconnu par B . Par extension, σ est dit reconnu par B .

Un tel automate de Büchi est dit associé à φ et est noté de façon générique $B(\varphi)$.

Remarquons que pour une formule LTL φ donnée, il n'y a pas un unique automate de Büchi associé. Dans notre cas, nous avons choisi d'utiliser l'outil `ltl2BA` [GO01] (via la passerelle `LTL2BA4J` [Bod11]) pour générer les automates de Büchi à partir d'une formule LTL. Nous expliquerons notre choix dans le chapitre 4, section 4.1.2.

Présentation des réseaux de régulation génétique

Dans ce chapitre, nous présentons le contexte biologique de la première partie de ce manuscrit, dans laquelle nous nous intéressons à l'étude des réseaux de régulation génétique (abrégiés en GRN, pour *Genetic Regulatory Network*). Nous avertissons que nous proposons ici une version simplifiée des mécanismes liés à la régulation génétique, afin de familiariser le lecteur avec le contexte biologique étudié. S'il souhaite une explication plus complète, le lecteur est invité à consulter des ouvrages tels que [PVP08 ; Wat+13].

Nous détaillons ensuite l'état de l'art de la modélisation discrète des GRN, tout en définissant les différents éléments constituant le modèle dit de René Thomas sur lequel nous nous basons. La fin de ce chapitre met en évidence la problématique à laquelle nous nous confrontons.

2.1	Contexte : les réseaux de régulation génétique	24
2.2	État de l'art de la modélisation de GRN	26
2.2.1	Modélisation par système d'ODE	26
2.2.2	Modélisation discrète booléenne	27
2.2.3	Modélisation discrète multivaluée	28
2.3	Le modèle multivalué de R. Thomas	32
2.3.1	Graphe d'interactions	33
2.3.2	Espace des états	35
2.3.3	Dynamique	35
2.3.4	Paramètres du modèle de Thomas	37
2.3.5	Contraintes sur les paramètres	40
2.4	Sélection de modèles à partir d'observations temporelles	41

2.1 Contexte : les réseaux de régulation génétique

Les GRN permettent de modéliser l'effet de protéines régulatrices sur *l'expression génétique*. L'expression génétique découle de la *théorie fondamentale de la biologie moléculaire* [Cri58] : il s'agit du processus biologique par lequel une *protéine* (un assemblage d'acides aminés) est synthétisée à partir de l'information codée dans un *gène* (une sous-séquence d'un brin d'ADN¹).

De manière simplifiée, la théorie fondamentale repose sur deux mécanismes, appelés *transcription* et *traduction*. Ces deux étapes sont schématisées dans la figure 2.1.

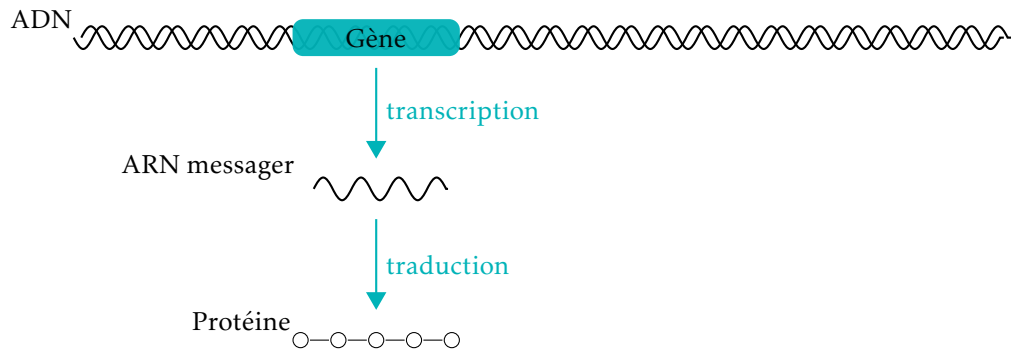


FIGURE 2.1 – Schéma de synthèse d'une protéine à partir de l'information génétique.

Dans un premier temps, le gène est transcrit en *ARNm*² par une enzyme appelée *ARN polymérase*, qui se fixe initialement sur une sous-séquence d'ADN proche du gène, appelée *promoteur*. La liaison de l'ARN polymérase à l'ADN enclenche l'ouverture de la double hélice de l'ADN, permettant la copie progressive du gène ciblé en ARNm ; l'ARN polymérase poursuit la transcription tant qu'il n'a pas atteint le *site de terminaison*, annonçant la fin de la séquence du gène copié. La double hélice de l'ADN est alors refermée et l'ARNm, ainsi que l'ARN polymérase, sont libérés.

Dans un second temps, l'ARNm est traduit en protéine par un *ribosome* (un complexe constitué de protéines et d'ARN) de la manière suivante : en se déplaçant le long de l'ARNm, le ribosome interprète séquentiellement chaque triplet de nucléotides de l'ARNm et recrute les acides aminés correspondant à chacun ; ceux-ci sont assemblés au fur et à mesure pour former finalement une protéine.

Dans la suite, pour des raisons inhérentes à la modélisation, nous simplifierons (plus encore) les mécanismes de l'expression génétique en considérant que les protéines sont les produits directs de l'activité d'un gène et qu'un gène *code* pour une protéine particulière, i.e. qu'un seul type de protéine est synthétisable à partir d'un gène.

1. L'*Acide DésoxyriboNucléique* est une macromolécule formée de deux brins complémentaires enroulés en forme de double hélice, composés de nucléotides. Une *séquence* d'ADN correspond à une succession ordonnée de nucléotides d'un brin.

2. L'*Acide RiboNucléique messenger* est une molécule formée de nucléotides correspondant au complémentaire d'une séquence d'ADN copiée. Contrairement à l'ADN, elle est dégradée au cours du temps.

Conformément à sa composition et à sa structure, une protéine peut avoir des fonctions très variées : elles peuvent notamment servir d'enzyme ou de régulateur, transmettre des signaux, rigidifier des tissus ou encore permettre la mobilité d'une cellule. Les protéines, tout comme l'ARN, sont soumises à une dégradation (ainsi, même dans le cas où elles sont produites en continu, le nombre total de protéines présentes dans la cellule est plafonné).

L'information génétique, contenue dans l'ensemble des gènes appelé *génome*, est commune à toutes les cellules d'un organisme (excepté en cas de mutation). Cependant, elle ne s'exprime pas de la même façon dans toutes les cellules car certains gènes sont spécifiques à un type de cellules et n'ont pas de raison d'être exprimés en dehors de celles-ci.

Ainsi, chez l'homme par exemple, on peut trouver des cellules aussi variées que les neurones, les cellules hépatiques, les globules blancs, etc. (et qui sont elles-mêmes divisées en sous-types), qui possèdent toutes une copie du même génome mais ne synthétisent pas les mêmes protéines (entre autres produits).

La spécialisation des cellules à partir d'un génome commun s'appelle la *différenciation*, elle permet d'associer une structure et une fonction particulière à chacune grâce à l'expression génétique.

L'expression génétique permet également aux cellules de s'adapter en fonction du contexte (une modification du milieu telle qu'un changement de température ou une carence en nutriments notamment, ou d'autres facteurs épigénétiques³) en modifiant par exemple la quantité de protéines produites.

L'expression génétique est influencée par la présence d'un certain type de protéines, appelées *protéines régulatrices*, qui favorisent ou au contraire empêchent l'une des étapes de l'expression génétique. De cette manière, ces protéines régulent la synthèse des protéines associées au gène cible. Notons qu'une protéine peut en particulier réguler le gène à partir duquel elle est synthétisée.

Certaines protéines se positionnent par exemple sur l'opérateur (qui est une sous-séquence de l'ADN proche du gène cible et du promoteur), ce qui empêche l'ARN polymérase de se fixer sur le promoteur et bloque ainsi la transcription du gène. Les protéines régulatrices peuvent également avoir le rôle inverse en empêchant un répresseur de se lier à l'opérateur ou en favorisant la liaison de l'ARN polymérase au promoteur.

D'autres mécanismes de régulation dans lesquels les protéines régulatrices agissent en tant qu'activatrices ou inhibitrices existent, que ce soit au niveau de la transcription ou de la traduction, mais nous ne les détaillerons pas plus ici (voir par exemple [Alb+02]).

Un ensemble de gènes peuvent dépendre d'un promoteur commun et être transcrits en un seul ARNm (leur régulation est donc liée), on dit alors qu'ils appartiennent à un *opéron* [Jac+05].

Exemple 5. *Tout au long de ce chapitre, nous illustrerons nos propos à l'aide d'un exemple « jouet » : un GRN composé de deux gènes, notés α et β , et de trois interactions. Nous verrons par la suite*

3. L'épigénétique concerne les modifications d'un phénotype (i.e. l'ensemble des caractères apparents d'un organisme) sans mutation génétique, qui peuvent dans certains cas se transmettre à la descendance.

(section 5.1 p.86) que ce GRN correspond à un cas d'étude biologique réel simplifié : l'étude de la cytotoxicité de la bactérie *Pseudomonas aeruginosa* [Gal+15].

Dans ce GRN, le gène α synthétise des protéines (appelées « protéines α ») activatrices de β et de lui-même, tandis que les protéines synthétisées à partir de β (appelées « protéines β ») inhibent l'expression du gène α . Il y a donc en jeu deux activations (α sur α d'une part et α sur β d'autre part) et une inhibition (β sur α).

Ce GRN est représenté schématiquement en figure 2.2. Les traits pleins reliant un gène à la protéine associée correspondent au processus de synthèse (transcription et traduction) ; les traits discontinus entre les protéines et un gène représentent les régulations (d'activation ou d'inhibition suivant le signe étiquetant l'interaction, respectivement « + » et « - »).

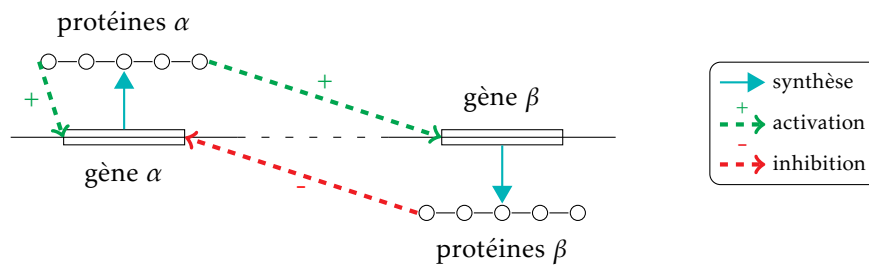


FIGURE 2.2 – GRN composé de deux gènes α et β

2.2 État de l'art de la modélisation de GRN

Depuis les années 60, de nombreux formalismes mathématiques ou informatiques ont été proposés pour modéliser les réseaux de régulation génétique : sous la forme d'équations différentielles ordinaires, partielles, de réseaux bayésiens ou sous d'autres formes stochastiques, de réseaux de Petri, de réseaux discrets booléens ou multivalués ou encore d'automates hybrides. Dans un premier temps, nous esquisserons la modélisation sous la forme d'équations différentielles ordinaires, avant de nous intéresser à la modélisation discrète, et en particulier au modèle dit de René Thomas. Pour un aperçu plus complet de l'état de l'art sur la modélisation des GRN, le lecteur pourra par exemple consulter [Jon02 ; SBB00].

2.2.1 Modélisation par système d'ODE

Classiquement, la modélisation mathématique des GRN se base sur des systèmes d'équations différentielles ordinaires (ODE) décrivant la variation des concentrations des différentes espèces en présence (voir par exemple [Goo65 ; C+99]).

Par nature, ces modèles constituent une approche quantitative des réactions sous-jacentes aux phénomènes de régulation entre gènes et protéines régulatrices. Dans ce cadre, la régulation

d'un gène est modélisée par une *équation de vitesse de réaction*, exprimant la vitesse de synthèse de la protéine associée au gène en fonction de la concentration de ses régulateurs.

La non-linéarité des ODE⁴ ne permet généralement pas une résolution analytique de ces systèmes. Une alternative possible est de recourir à des techniques de résolution numérique (cf. [MS95 ; SA85] par exemple pour l'étude du phage λ). Cependant, les simulations nécessitent de connaître précisément les valeurs des constantes qui apparaissent dans les ODE, correspondant aux paramètres cinétiques du système (notamment les constantes de production et de dégradation des équations des vitesses de réaction). Or ces paramètres sont difficilement évaluables à cause du manque de données expérimentales *in vivo* ou *in vitro*.

2.2.2 Modélisation discrète booléenne

En 1969, KAUFFMAN propose dans [Kau69] une modélisation qualitative s'appuyant sur une approche discrète (vis-à-vis du temps et des états). Dans ce formalisme, les GRN sont représentés par un modèle booléen (dit aussi *modèle binaire*) dans lequel l'expression des gènes est activée ou inactivée par le biais d'un switch binaire on/off enclenché par la présence ou l'absence des protéines régulatrices correspondantes (elles-mêmes étant représentées par des gènes activés ou inactivés). Chaque gène est ainsi associé à une variable qui peut prendre la valeur 0 (pour la valeur booléenne faux, le gène est considéré inactivé et les protéines correspondantes sont absentes) ou 1 (pour vrai, gène activé et protéines présentes).

Une fonction booléenne permet de connaître de manière déterministe le prochain état atteint en fonction des gènes activés dans l'état courant. Cette fonction booléenne est à déterminer : soit au hasard, soit par tâtonnement en testant toutes les possibilités une à une, en utilisant par exemples des connaissances sur le comportement du GRN.

La transition entre les états est *synchrone*, i.e. plusieurs gènes peuvent évoluer simultanément. L'évolution au cours du temps des valeurs des variables associées aux gènes est appelée *dynamique* du GRN et caractérise son comportement.

Exemple 6. Pour le GRN introduit précédemment (figure 2.2), on peut imaginer le modèle dont les entrées et sorties de la fonction booléenne sont décrites dans le tableau 2.1.

Dans ce modèle, l'état correspondant à $\alpha = 0$ et $\beta = 0$ (i.e. les gènes α et β sont inactifs) sera suivi de l'état dans lequel $\alpha = 1$ et $\beta = 0$ (i.e. α est actif tandis que β reste inactif).

À l'instant T		À l'instant T+1	
α	β	α	β
0	0	1	0
0	1	0	0
1	0	1	1
1	1	1	1

TABLEAU 2.1 – Exemple de modèle booléen modélisant le GRN en figure 2.2.

4. cf. section 2.2.3

La modélisation booléenne des GRN a été étudiée par d'autres auteurs, citons le travail antérieur de SUGITA dans [Sug61 ; Sug63 ; SF63] et celui postérieur de THOMAS dans [Tho73 ; Tho79] par exemple.

2.2.3 Modélisation discrète multivaluée

En 1990, THOMAS et al. proposent une modélisation discrète multivaluée des GRN dans [Td90] (voir aussi [Tho91 ; TCT93 ; TTK95 ; TT95]), connue sous le nom de *modèle de Thomas* (ou *méthode logique généralisée*). THOMAS y introduit la notion de seuils dans les modèles discrets, i.e. une protéine ne peut réguler un gène cible que si elle est présente en quantité suffisante, cette quantité étant spécifique d'une régulation donnée.

Par rapport au modèle booléen, la modélisation multivaluée présente l'avantage de prendre en compte de manière plus fine le phénomène de régulation. Par exemple, dans le cas du GRN précédent, les protéines α peuvent être en quantité suffisante pour réguler le gène β mais insuffisante pour activer le gène α : cette nuance n'est possible que dans un modèle multivalué.

La notion de seuil provient de la discrétisation des concentrations du modèle continu des ODE⁵. Elle est liée à la forme sigmoïdale des courbes \mathcal{V} représentant la vitesse de synthèse d'une protéine en fonction de la concentration d'un de ses régulateurs.

En effet, expérimentalement, les biologistes déterminent qu'une protéine p est activatrice d'un gène g si en augmentant progressivement la concentration de p , ils observent une valeur à partir de laquelle la synthèse des protéines de g augmente. On considère généralement que la vitesse de synthèse des protéines de g sature à partir d'un certain seuil de concentration de p , i.e. que la vitesse de synthèse est constante au-dessus d'un seuil spécifique à la régulation de g par p [JM61 ; YY71 ; Tho98].

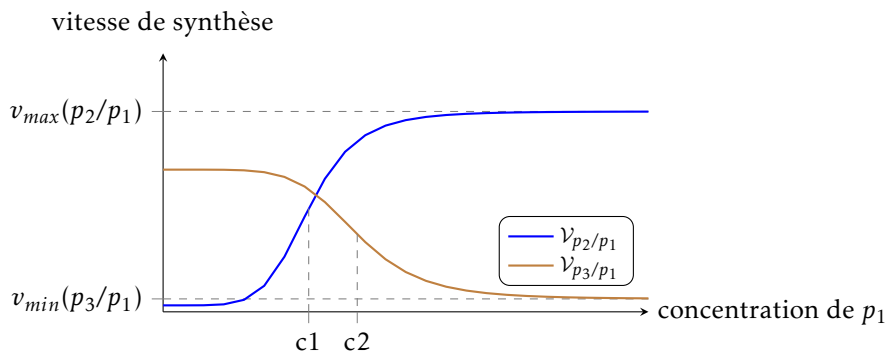
À l'inverse, un inhibiteur de g est expérimentalement caractérisé par une diminution du taux de synthèse de g à partir d'une valeur de concentration donnée de cet inhibiteur. La vitesse maximale est dans ce cas atteinte pour une valeur de l'inhibiteur inférieure à ce seuil. Un phénomène de saturation se produit également dans le cas d'une inhibition : quand la concentration de p est supérieure à la valeur de seuil, la vitesse de synthèse ne diminue plus.

Ces observations biologiques expliquent le profil des courbes \mathcal{V} en sigmoïde : elles sont croissantes ou décroissantes suivant le type de la régulation associée (activation ou inhibition), la régulation est faible tant qu'une concentration minimale spécifique n'est pas atteinte, et pour une concentration bien supérieure à ce seuil, la régulation atteint un palier maximal (cas d'activation) ou minimal (cas d'inhibition).

Pour illustrer notre propos, considérons trois protéines p_1 , p_2 et p_3 telles que p_1 est un activateur du gène qui synthétise p_2 et un inhibiteur du gène qui synthétise p_3 . Notons respectivement \mathcal{V}_{p_2/p_1} et \mathcal{V}_{p_3/p_1} les courbes caractérisant la vitesse de synthèse de p_2 et p_3 en fonction de la concentration de p_1 . Une illustration possible du profil de ces deux courbes est esquissée en

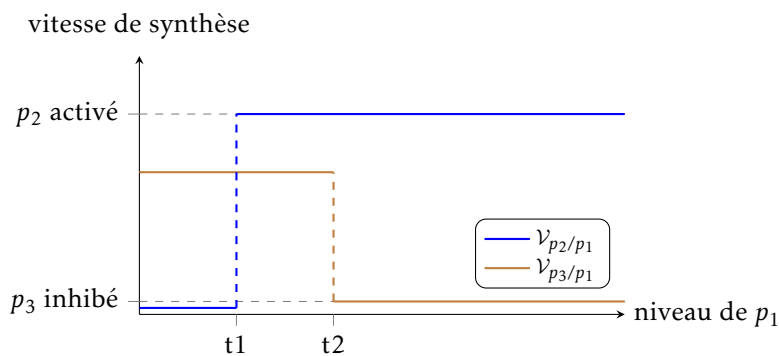
5. cf. section 2.2.1

figure 2.3.

FIGURE 2.3 – Profil type de la synthèse des espèces régulées p_2 et p_3 en fonction de la concentration du régulateur p_1 .

\mathcal{V}_{p_2/p_1} et \mathcal{V}_{p_3/p_1} sont des courbes respectivement croissante et décroissante. Elles atteignent un palier respectivement maximal (de vitesse $v_{max}(p_2/p_1)$) et minimal (de vitesse $v_{min}(p_3/p_1)$) à partir d'un seuil de concentration de p_1 propre à chaque régulation (respectivement c_1 et c_2). Précisons ici que le fait que $v_{max}(p_2/p_1)$ soit supposée supérieure à $v_{min}(p_3/p_1)$ d'après la représentation des courbes, n'a pas d'incidence sur notre propos.

Ces courbes continues \mathcal{V} peuvent être approchées par des fonctions en escalier, c'est ce qu'on appelle *l'interprétation logique*. L'interprétation logique des courbes de la figure 2.3 est représentée en figure 2.4.

FIGURE 2.4 – Représentation logique des courbes \mathcal{V}_{p_2/p_1} et \mathcal{V}_{p_3/p_1} .

Pour obtenir cette interprétation logique, on considère que l'abscisse du point d'inflexion de \mathcal{V} correspond à la concentration minimale à partir de laquelle un régulateur p active ou inhibe un gène g . Cette concentration est appelée *seuil de régulation* de p sur g . Dans la figure 2.3, ces seuils sont notés c_1 (associé à \mathcal{V}_{p_2/p_1} , et donc à la régulation de la protéine p_2 par le gène associé à p_1) et c_2 (associé à \mathcal{V}_{p_3/p_1}).

Pour une courbe donnée, on néglige l'effet régulateur potentiel de p_1 en-dessous du seuil de régulation, tandis qu'au-dessus, on considère que le plateau de saturation est atteint. Les deux paliers de la fonction en escalier obtenue à partir de \mathcal{V} correspondent ainsi à deux états logiques différents : un état dans lequel il n'y a pas d'effet de régulation, et un état où la régulation est effective.

Dans l'interprétation logique, la concentration de p est de plus discrétisée : on parle alors de *niveaux d'expression du gène* associés à p . Dans la figure 2.4, l'abscisse t correspondant au saut entre les paliers est un entier naturel découlant de la discrétisation de la concentration de p : c'est une valeur relative respectant l'ordonnement des seuils de concentrations de l'ensemble des régulations de p . Ainsi, dans notre exemple, on note t_1 et t_2 les seuils discrétisés obtenus à partir de c_1 et c_2 ; comme $c_1 < c_2$, alors $t_1 < t_2$.

Les seuils de régulation considérés dans le modèle de Thomas correspondent aux seuils t discrétisés de l'interprétation logique des courbes de régulation. Du point de vue des gènes, ils correspondent à un niveau d'expression minimal qu'un gène doit atteindre pour que les protéines qu'il synthétise régulent un ou des gènes cibles du GRN.

Le nombre de seuils de régulation différents pour une protéine, et donc la valeur maximale du niveau d'expression du gène correspondant, est au plus égal au nombre de gènes qu'elle régule (en général il est égal au nombre de gènes régulés, i.e. chaque régulation correspond à un seuil différent).

Dans le cadre de notre exemple jouet, le niveau maximum de α est au plus 2 car il régule deux gènes, α et β (il peut cependant être fixé à 1 si on considère que les seuils de régulation de α et β sont identiques, ce qui n'est généralement pas le cas, à l'exception des gènes appartenant à un même opéron) ; β par contre ne régule qu'un seul gène, α , et donc son niveau maximum est 1. Le plus petit palier, 0, correspond à l'absence de la protéine associée au gène, ou à une concentration trop faible pour réguler un gène du GRN.

Cette approche est qualitative, chaque niveau d'expression est lié à un effet de régulation différent. Par exemple, on peut imaginer que le gène α puisse atteindre trois niveaux d'expression différents 0, 1 et 2, caractérisés de la manière suivante :

- 0 signifie que α est inactif et qu'il ne régule aucun gène,
- 1 correspond au niveau à partir duquel α peut réguler un gène (le gène β par exemple),
- 2 correspond au niveau à partir duquel α peut réguler un autre gène (α).

Le fonctionnement d'un GRN est caractérisé par sa *dynamique*, autrement dit par l'évolution au cours du temps des concentrations des protéines, ou des niveaux d'expression des gènes. Ces dynamiques sont assujetties à la connaissance de paramètres biologiques indicatifs des effets de compétition entre protéines activatrices et inhibitrices d'un même gène.

Par exemple, quand α et β ont tous les deux un niveau d'expression supérieur à leur seuil respectif de régulation sur α , on ignore a priori si l'effet d'activation de α sur α est plus important que l'effet d'inhibition de β sur α et donc si le niveau d'expression de α va avoir tendance à augmenter, à diminuer ou à rester stable.

Chaque dynamique peut être caractérisée par un ensemble de paramètres, dits *paramètres du modèle de Thomas* (ou *paramètres logiques*) et définis comme les *points focaux* (constants) représentant le niveau vers lequel tend un gène en fonction des *ressources* disponibles (i.e. les régulateurs qui sont au-dessus de leur seuil de régulation, que nous appellerons *régulateurs effectifs* dans la suite).

Dans le cadre discret, l'évolution au cours du temps d'un GRN est caractérisée par une succession d'états, chacun étant déterminé par l'ensemble de valeurs des niveaux d'expression des gènes du GRN au temps t discret correspondant. Il existe deux approches antagonistes pour modéliser l'écoulement du temps : l'approche synchrone et l'approche asynchrone.

Dans le cas synchrone, l'évolution est déterministe et plusieurs niveaux d'expression peuvent évoluer en même temps de manière à atteindre immédiatement l'état ciblé. Ce cadre est notamment utilisé dans la modélisation booléenne de KAUFFMAN [Kau69] (cf. section 2.2.2).

Dans le cas asynchrone, un seul niveau d'expression peut évoluer à la fois, en considérant que chaque niveau d'expression tend à évoluer vers la valeur qu'il aurait dans l'état ciblé par l'approche synchrone. Une transition synchrone définit de fait plusieurs transitions asynchrones : l'état source de ces transitions asynchrones est le même que celui de la transition synchrone, mais par contre chaque transition asynchrone correspond à une augmentation ou une diminution d'un seul niveau d'expression d'une seule unité.

Alors que lors d'une transition synchrone le point focal est nécessairement atteint, cela n'est plus le cas lorsque la transition est rendue asynchrone. En effet, les paramètres caractérisant la prochaine évolution dépendent des niveaux d'expressions des gènes dans l'état courant : le point focal évolue avec chaque transition asynchrone suivie, car l'état évolue lui aussi. Par conséquent, l'approche asynchrone apporte un niveau de complexité supplémentaire pour la détermination des dynamiques par rapport à l'approche synchrone.

L'approche asynchrone présente deux avantages par rapport à l'approche synchrone. D'une part, elle permet de considérer qu'il peut exister plusieurs choix d'états atteignables à partir de l'état courant, ce qui relâche une contrainte forte. D'autre part, elle permet également de s'affranchir de la possibilité de processus de régulation simultanés, qui sont peu probables en pratique. En effet, cela suppose la simultanéité du changement de niveau d'expression de plusieurs gènes à priori distants, qui reposent elle-même sur la simultanéité des mécanismes sous-jacents, i.e. la régulation de plusieurs gènes d'une part et la variation significative des concentrations des protéines correspondantes d'autre part. C'est pour ces raisons que l'approche asynchrone est privilégiée par THOMAS et al. [Td90].

Enfin, troisième possibilité, on peut également se placer dans un cadre à la fois synchrone et asynchrone comme GONZALEZ et al. ([Gon+06 ; Fau+06]) qui déclinent le modèle de Thomas en autorisant des transitions synchrones en plus des transitions asynchrones.

Dans le cas asynchrone, il peut exister plusieurs possibilités d'évolution à partir de l'état courant. Signalons que plusieurs choix peuvent être pris en conséquence :

- considérer que toutes les transitions se valent et donc exploiter tous les cas possibles (cas

de base du modèle de Thomas [Td90]);

- ajouter la notion de délai ou de priorité aux transitions (dans [Ahm+09] par exemple, AHMAD et al. proposent un modèle hybride en ajoutant des contraintes temporelles au modèle de Thomas pour simuler l'évolution continue des niveaux d'expression);
- se placer dans le cadre probabiliste, en attachant une probabilité de franchissement aux transitions.

Ces deux derniers choix permettent de brider le non-déterminisme dû à l'asynchronicité.

Les paramètres de Thomas capturent de manière plus abstraite les paramètres des ODE. Il est admis (ou pour certaines propriétés, démontré) que les dynamiques discrètes capturent l'essentiel des caractéristiques qualitatives des dynamiques continues. THOMAS a par exemple conjecturé dans [Tho81] deux propriétés importantes des GRN, portant sur le lien entre les circuits de rétroaction positifs (i.e. qui contiennent un nombre pair d'inhibitions) d'une part ou négatifs (i.e. qui contiennent un nombre impair d'inhibitions) d'autre part, présent dans un graphe d'interactions (qui représente les régulations entre les gènes du GRN, nous le définirons formellement dans la suite, cf. définition 2.3.1) et l'existence respective de plusieurs états stables ou d'oscillations entretenues. Ces conjonctures ont été démontrées au fil des ans dans plusieurs cadres différents : tout d'abord, continu (dans [PMO95 ; Sno98 ; Sou03 ; Sou06] par exemple), puis discret booléen [RRT08], et enfin discret généralisé [Ric06 ; RC07 ; Ric09 ; Ric10b].

Qu'il s'agisse de modélisation se basant sur des ODE ou de modélisation discrète, la question centrale est celle de la détermination des paramètres du modèle, qui déterminent précisément le comportement des dynamiques issues du GRN. Le modèle de Thomas a donné lieu à de nombreux travaux de modélisation, appliqués par exemple à la production de mucus de *Pseudomonas aeruginosa* [Ber+04], au cycle de vie du phage λ [TT95], à la morphogénèse d'*Arabidopsis thaliana* [MTA99], ou encore à la régulation de la réponse immunitaire [Mur+96]). Il a conduit en particulier à la mise au point de méthodes outillées d'aide à la détermination des paramètres (présentées en détail dans la section 2.4), cadre dans lequel nous nous plaçons également.

2.3 Le modèle multivalué de R. Thomas

Dans cette section, nous détaillons le cadre bien établi de l'approche multivaluée de R. Thomas présentée en section 2.2. En particulier, nous introduisons ou redéfinissons des notations propres à cette thèse (proches de celles de [Ber+04] ou [Bar+12]), afin de faciliter la mise en œuvre de notre approche dans la suite.

Remarque. Par souci de simplification, nous assimilons gène et protéine en considérant qu'un gène est associé à un seul type de protéines produites.

2.3.1 Graphe d'interactions

Un GRN est classiquement représenté par un *graphe d'interactions* (abrégé en IG pour *Interaction Graph*), qui est un graphe orienté et étiqueté dans lequel les sommets sont associés aux gènes du GRN et les arcs aux interactions (de régulation) entre protéines et gènes (définition 2.3.1).

Définition 2.3.1 (Graphe d'interactions). Un *graphe d'interactions* est un graphe orienté étiqueté $\Gamma = (G, I)$ où :

- G est un ensemble fini de sommets, appelés *gènes*
- $I \subseteq G \times \{+, -\} \times \mathbb{N}^* \times G$ est l'ensemble des arcs, appelés *interactions*, il satisfait les propriétés suivantes :

p1. [Unicité des couples d'interactions]

$$\forall (g_1, s, t, g_2), (g'_1, s', t', g'_2) \in I, (g_1 = g'_1 \wedge g_2 = g'_2) \Rightarrow (s = s' \wedge t = t')$$

p2. [Couverture du domaine des seuils]

$$\forall (g_1, s, t, g_2) \in I, t > 1 \Rightarrow \exists (g_1, s', t', g'_2) \in I, t' = t - 1$$

Soit $\Gamma = (G, I)$ un graphe d'interactions. Une interaction $(g, s, t, g') \in I$ entre les gènes g et g' dans G est étiquetée par un couple (s, t) , avec $s \in \{+, -\}$ représentant l'effet de g sur g' (« + » correspondant à une activation, « - » à une inhibition) et t indiquant le seuil de régulation, i.e. le niveau minimal que la source g a besoin d'atteindre pour influencer l'expression génétique de la cible g' .

L'ensemble des gènes G est muni d'une relation d'ordre : g_1, g_2, \dots, g_n (avec $|G| = n$). En pratique, les gènes ne seront pas forcément notés g_i avec $i \in \mathbb{N}$, mais plutôt avec une chaîne de caractères correspondant à leur nom ; dans ce cas, la relation d'ordre sera donnée par l'ordre lexicographique.

Notation 2.3.1 (Seuils associés à un gène). Pour un gène $g \in G$, nous notons m_g le seuil maximal étiquetant une interaction issue de g : $m_g = \max\{t \mid \exists (g, s, t, g') \in I\}$.

Dans le cas particulier où un gène n'est la source d'aucune interaction, on pose arbitrairement $m_g = 1$ pour pouvoir observer l'influence des régulations qui s'exercent sur lui.

L'ensemble des seuils associés à g est noté $\mathbb{X}_g = \{0, \dots, m_g\} \subset \mathbb{N}$.

La propriété **p1** de la définition 2.3.1 spécifie qu'il ne peut pas exister plus d'une interaction entre deux gènes de G .

La propriété **p2** de la définition 2.3.1 signifie que pour n'importe quel gène g de G , avec un domaine de valeurs associé $\mathbb{X}_g = \{0, \dots, m_g\} \subset \mathbb{N}$, chaque niveau intermédiaire doit étiqueter au moins une interaction en sortie de g . Cette propriété résulte de la discrétisation des niveaux de

concentration des protéines provenant du formalisme de R.Thomas (les seuils sont des paliers relatifs les uns par rapport aux autres, cf. sous-section 2.2.3).

Notation 2.3.2 (Signe et seuil d'une interaction). D'après la propriété **p1** de la définition 2.3.1, s'il existe une interaction de g sur g' telle que $(g, s, t, g') \in I$, alors il y a unicité du signe et du seuil de l'interaction. Nous pouvons donc noter $s(g, g') = s$ le signe de l'interaction et $t(g, g') = t$ le seuil de régulation.

L'interaction $(g, s, t, g') \in I$ est notée alternativement $g \xrightarrow{[s,t]} g'$.

Les notations suivantes permettent d'exprimer les régulateurs d'un gène donné, notamment en fonction du signe des régulations.

Notation 2.3.3 (Ensemble des régulateurs). Pour un gène g dans G , nous notons $G^-(g)$ le sous-ensemble de G correspondant aux régulateurs de g , défini par :

$$G^-(g) = \{g' \mid \exists (g', s, t, g) \in I\}$$

$G^-(g)$ est lui-même partitionné en deux sous-ensembles $G_+^-(g)$ et $G_-^-(g)$, qui correspondent respectivement aux ensembles des *activateurs* et des *inhibiteurs* de g :

$$G_+^-(g) = \{g' \mid g' \in G^-(g), s(g', g) = +\}$$

$$G_-^-(g) = \{g' \mid g' \in G^-(g), s(g', g) = -\}$$

Exemple 7. La figure 2.5 est un exemple de graphe d'interactions pouvant correspondre au GRN schématisé en figure 2.2. Ce n'est qu'une possibilité parmi d'autres, car les seuils des régulations dont α est la source pourraient être fixés à des valeurs différentes.

Dans ce graphe d'interactions, α active sa propre expression génétique à partir d'un seuil plus élevé que celui qu'il a besoin d'atteindre pour réguler β : $t(\alpha, \alpha) > t(\alpha, \beta)$. β quant à lui n'agit que sur α , en tant qu'inhibiteur.

Avec les notations introduites précédemment : $G^-(\alpha) = \{\alpha, \beta\}$, $G_+^-(\alpha) = \{\alpha\}$ et $G_-^-(\alpha) = \{\beta\}$, $G^-(\beta) = G_+^-(\beta) = \{\alpha\}$ et $G_-^-(\beta) = \{\}$.

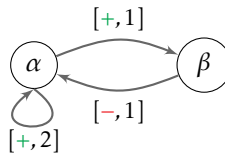


FIGURE 2.5 – Graphe d'interactions correspondant au GRN de l'exemple 5

2.3.2 Espace des états

Les niveaux d'expression des gènes d'un GRN évoluent au cours du temps en fonction des régulations entre les gènes. La discrétisation (en niveaux d'expression) des concentrations des protéines, par rapport au modèle continu, nous amène à considérer un ensemble d'états dans lesquels le niveau d'expression de chaque gène est fixé à une valeur particulière. L'ensemble de ces états représente l'ensemble des possibilités de combinaisons de niveaux d'expression, conformément au domaine de valeur des gènes (dépendant de leur seuil de régulation maximal).

Définition 2.3.2 (Espace des états et niveau d'expression). L'espace des états associé à un graphe d'interactions Γ , noté \mathbb{X} , est défini comme le produit cartésien des niveaux d'expression des gènes, ordonnés selon la relation d'ordre de G ; c'est un sous-ensemble de $\mathbb{N}^{|G|}$.

$$\mathbb{X} = \prod_{g \in G} \mathbb{X}_g$$

Pour un état $x \in \mathbb{X}$, on note $x_g \in \mathbb{X}_g$ le niveau d'expression du gène g dans x .

Exemple 8. L'espace des états associé au graphe d'interactions de la figure 2.5 est :

$$\mathbb{X} = \{(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)\}$$

En particulier, le couple $(2, 1)$ correspond à l'état où $x_\alpha = 2$ et $x_\beta = 1$.

2.3.3 Dynamique

Une évolution des niveaux d'expression des gènes correspond à un changement d'état. Dans le cadre du modèle de R. Thomas, les évolutions sont désynchronisées et pas-à-pas : entre deux états consécutifs, il ne peut y avoir qu'un seul gène dont le niveau d'expression est modifié et la variation correspondante est d'une seule unité. Ainsi l'état $(2, 1)$ ne peut pas être atteint directement à partir de l'état $(0, 0)$, plusieurs changements d'état intermédiaires sont nécessaires ; par exemple, $(0, 0)$ évolue en $(1, 0)$, puis en $(2, 0)$ et enfin en $(2, 1)$.

En reprenant les notations introduites dans le chapitre Préliminaires (section 1.1, p.12), nous notons $x[g \uparrow]$ (respectivement $x[g \downarrow]$) l'état obtenu à partir de x en augmentant (diminuant) x_g d'une unité.

L'évolution des niveaux d'expression au cours du temps peut être représentée sous la forme d'un système de transition particulier, appelé *dynamique*, dont les contraintes sont données en définition 2.3.3.

Définition 2.3.3 (Dynamique associée à un graphe d'interactions). Un système de transition $\mathcal{S} = (E, \rightarrow)$ est une *dynamique* d'un graphe d'interactions s'il vérifie les propriétés suivantes :

p1. [Couverture de l'espace des états]

$$E = \mathbb{X};$$

p2. [Évolution asynchrone]

$$\forall x, x' \in \mathbb{X}, \text{ si } x \rightarrow x' \text{ alors } x' = x \text{ ou } \exists g \in G, x' = x[g \uparrow] \vee x' = x[g \downarrow];$$

p3. [Stabilité]

$$\forall x \in \mathbb{X}, \text{ si } x \rightarrow x \text{ alors il n'existe pas d'état } x' \in \mathbb{X} \text{ distinct de } x, \text{ tel que } x \rightarrow x'.$$

Une dynamique associée à un graphe d'interactions dont l'espace des états est \mathbb{X} , est notée de façon générique $\mathcal{D} = (\mathbb{X}, \rightarrow)$.

La propriété **p1** de la définition 2.3.3 définit l'ensemble des états de la dynamique comme étant l'espace des états de l'IG associé.

La propriété **p2** de la définition 2.3.3 spécifie que seules les transitions d'états asynchrones et pas-à-pas sont autorisées, i.e. qu'un seul gène au plus évolue lors d'un changement d'état et d'une seule unité de seuil.

La propriété **p3** de la définition 2.3.3 spécifie que la présence d'une boucle dans un état interdit toute autre transition sortante.

Exemple 9. La figure 2.6 représente une dynamique associée à l'IG en figure 2.5. Dans cette dynamique, on note l'existence d'une boucle dans l'état $(2, 1)$, i.e. cet état est stable. Il existe également un cycle entre les états $(0, 0)$, $(1, 0)$, $(1, 1)$ et $(0, 1)$.

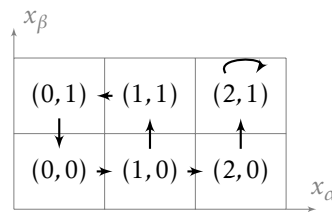


FIGURE 2.6 – Une dynamique associée à l'IG en figure 2.5

Dans la suite, pour étudier les dynamiques, nous aurons besoin de raisonner sur les niveaux d'expression des gènes de G , en les manipulant sous forme symbolique (i.e. à l'aide de variables). Pour ce faire, nous introduisons dans la définition 2.3.4 la notion de *variable d'état* et nous nous

munissons d'une fonction d'interprétation nous permettant d'associer chaque variable d'état de Γ au niveau d'expression correspondant dans un état $x \in \mathbb{X}$ donné.

Définition 2.3.4 (Variables d'état et interprétation). $\mathbb{G} = \{\chi_g \mid g \in G\}$ représente l'ensemble des variables, dites *variables d'état*, associées aux gènes de G .

Par ailleurs, nous définissons la famille d'interprétations $(\nu_x)_{x \in \mathbb{X}}$ telle que $\nu_x : \mathbb{G} \rightarrow \mathbb{N}$ et $\nu_x(\chi_g) = x_g$ (cf. chapitre Préliminaires, section 1.2, p.13).

2.3.4 Paramètres du modèle de Thomas

Dans le modèle de R. Thomas, on considère que le niveau d'expression d'un gène g de G dans un état x de \mathbb{X} tend à évoluer pour se rapprocher de la valeur d'un *point focal* (ou *attracteur*) particulier, qui dépend des régulateurs qui agissent sur g dans l'état x .

Ce *point focal* est représenté par un *paramètre* (dit *paramètre du modèle de Thomas*) dont la valeur (constante) est en général difficile d'accès pour les biologistes. Un paramètre est associé à un gène g mais également à un sous-ensemble de régulateurs de g . Ce sous-ensemble correspond aux *régulateurs effectifs* de g dans un état x donné, i.e. les régulateurs qui ont un niveau d'expression supérieur à leur seuil de régulation de g dans x (définition 2.3.5).

Définition 2.3.5 (Régulateurs effectifs d'un gène). Soit un état x pris dans \mathbb{X} et un gène g de G . Un régulateur $g' \in G^-(g)$ de g est dit *effectif* dans l'état x ssi $x_{g'} \geq t(g', g)$; autrement dit, si le niveau d'expression de g' dans l'état x est supérieur au seuil $t(g', g)$ de l'interaction $(g', s, t, g) \in I$.

Nous notons $\omega_g^* : \mathbb{X} \rightarrow 2^G$ la fonction qui associe à chaque état $x \in \mathbb{X}$, l'ensemble des régulateurs effectifs de g dans l'état x , telle que :

$$\omega_g^*(x) = \{g' \mid \exists (g', s, t, g) \in I, x_{g'} \geq t(g', g)\}$$

Exemple 10. Dans l'état $(2, 0)$, les régulateurs effectifs de α et de β sont respectivement $\omega_\alpha^*((2, 0)) = \{\alpha\}$ et $\omega_\beta^*((2, 0)) = \{\alpha\}$.

Dans l'état $(0, 0)$, on a : $\omega_\alpha^*((0, 0)) = \{\}$ et $\omega_\beta^*((0, 0)) = \{\}$.

Un paramètre de g (définition 2.3.6) caractérise l'influence que peuvent exercer ses régulateurs effectifs sur son évolution. Il n'est pas associé à un seul et unique état car un ensemble de régulateurs effectifs de g peut correspondre à plusieurs états différents. Par exemple, $\{\alpha\}$

correspond à l'ensemble des régulateurs effectifs de β dans tous les états où $x_\alpha \geq 1$, soit : $(1, 0)$, $(2, 0)$, $(1, 1)$ et $(2, 1)$.

Définition 2.3.6 (Paramètres). Soit $g \in G$ un gène. L'ensemble des *paramètres* de g est noté \mathbb{K}_g avec :

$$\mathbb{K}_g = \{K_{g,\omega} \mid \omega \subseteq 2^{G^-(g)}\}$$

où $K_{g,\omega}$ est le paramètre associé au gène g et à l'ensemble ω de régulateurs de g .

L'ensemble des paramètres biologiques de Γ est noté $\mathbb{K} = \cup_{g \in G} \mathbb{K}_g$.

Exemple 11. Les sous-ensemble des régulateurs de α et β sont respectivement

$$2^{G^-(\alpha)} = \{\{\}, \{\alpha\}, \{\beta\}, \{\alpha, \beta\}\} \quad \text{et} \quad 2^{G^-(\beta)} = \{\{\}, \{\alpha\}\}.$$

L'ensemble des paramètres liés au graphe d'interactions est donc :

$$\mathbb{K} = \{K_{\alpha,\{\}}, K_{\alpha,\{\alpha\}}, K_{\alpha,\{\beta\}}, K_{\alpha,\{\alpha,\beta\}}, K_{\beta,\{\}}, K_{\beta,\{\alpha\}}\}.$$

Un paramètre de g synthétise le résultat des actions combinées des régulateurs effectifs de g , qui influencent conjointement g . Sa valeur peut être difficile à déterminer, en particulier dans le cas où l'ensemble des régulateurs effectifs contient à la fois des activateurs et des inhibiteurs de g .

Dans l'état $(2, 1)$ par exemple, $\omega_\alpha^*((2, 1)) = \{\alpha, \beta\}$ avec α et β respectivement activateur et inhibiteur de α . Sans connaissance supplémentaire, nous ignorons les forces d'influence respectives de α et β sur α , qui pourraient amoindrir l'effet d'activation au profit de l'inhibition (ou l'inverse), ou s'il peut y avoir un effet de coordination entre les deux régulateurs provoquant une régulation de α uniquement activatrice ou uniquement inhibitrice.

Grâce à une fonction d'interprétation \mathcal{K} , appelée *instance de paramètres*, nous associons les paramètres de \mathbb{K} à un niveau d'expression représentant la valeur du paramètre correspondant (définition 2.3.7). Par construction, le domaine d'instance des paramètres de \mathbb{K}_g est identique au domaine de valeur du gène associé g .

Définition 2.3.7 (Instance de paramètres). Une instance de paramètres \mathbb{K} de Γ est définie comme une fonction $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$, qui associe chaque paramètre $K_{g,\omega}$ dans \mathbb{K} à une valeur dans \mathbb{X}_g .

Exemple 12. Soit l'instance de paramètres $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$ définie par :

- $\mathcal{K}(K_{\alpha,\{\}})=2$,
- $\mathcal{K}(K_{\alpha,\{\alpha\}})=2$,
- $\mathcal{K}(K_{\alpha,\{\beta\}})=0$,
- $\mathcal{K}(K_{\alpha,\{\alpha,\beta\}})=2$,
- $\mathcal{K}(K_{\beta,\{\}})=0$,
- $\mathcal{K}(K_{\beta,\{\alpha\}})=1$.

Dans l'état $(2,0)$, α tend vers $\mathcal{K}(K_{\alpha,\{\alpha\}})=2$ et β tend vers $\mathcal{K}(K_{\beta,\{\alpha\}})=1$. L'ensemble des états suivants est donc réduit à un seul état : $(2,1)$.

Dans l'état $(0,0)$, α tend vers $\mathcal{K}(K_{\alpha,\{\}})=2$ et β tend vers $\mathcal{K}(K_{\beta,\{\}})=0$. En vertu de la progression pas-à-pas du modèle de Thomas, l'ensemble des états suivants est donc : $\{(1,0)\}$.

Enfin, dans l'état $(1,0)$, α tend vers $\mathcal{K}(K_{\alpha,\{\}})=2$ et β tend vers $\mathcal{K}(K_{\beta,\{\alpha\}})=1$. Du fait de l'asynchronisme du modèle de Thomas, l'ensemble des états suivants est donc : $\{(1,1), (2,0)\}$.

Considérons une instance de paramètres $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$. La différence entre la valeur $\mathcal{K}(K_{g,\omega})$ (avec $K_{g,\omega} \in \mathbb{K}$) et celle de x_g dans un état $x \in \mathbb{X}$ tel que ω est l'ensemble des régulateurs effectifs de g dans x , spécifie l'évolution possible de x_g : augmentation, diminution ou stabilité. Nous pouvons donc déterminer les états atteignables à partir de l'état x , en considérant les valeurs des paramètres de tous les gènes, correspondant à leurs régulateurs respectifs dans x . De cette manière, nous obtenons une dynamique de Γ associée à l'instance de paramètres \mathcal{K} ; les conditions d'existence des transitions d'états d'une dynamique concrète résultant d'une instance de paramètres \mathcal{K} sont spécifiées en définition 2.3.8.

Définition 2.3.8 (Dynamique induite par une instance de paramètres). Pour une instance de paramètres $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$, $\mathcal{D}_{\mathcal{K}} = (\mathbb{X}, \rightarrow_{\mathcal{K}})$ est la dynamique de Γ induite par \mathcal{K} minimalement définie telle que :

$$\forall x \in \mathbb{X} \forall g \in G,$$

- $x \rightarrow_{\mathcal{K}} x[g \uparrow]$ si $x_g < \mathcal{K}(K_{g,\omega_g^*(x)})$ [augmentation du gène g];
- $x \rightarrow_{\mathcal{K}} x[g \downarrow]$ si $x_g > \mathcal{K}(K_{g,\omega_g^*(x)})$ [diminution du gène g];
- $x \rightarrow_{\mathcal{K}} x$ si $\forall g \in G, x_g = \mathcal{K}(K_{g,\omega_g^*(x)})$ [état stable].

Une dynamique induite par une instance de paramètres \mathcal{K} et associée à un graphe d'interactions dont l'espace des états est \mathbb{X} , est notée de façon générique $\mathcal{D}_{\mathcal{K}} = (\mathbb{X}, \rightarrow_{\mathcal{K}})$.

Ainsi, une transition en sortie d'un état $x \in \mathbb{X}$ correspondant à une augmentation (respectivement diminution) unitaire suivant l'axe d'un gène $g \in G$ est présente seulement si x_g est inférieur (respectivement supérieur) à la valeur du paramètre associé à g et à ses régulateurs effectifs dans x (condition d'augmentation/diminution).

D'autre part, une transition ayant pour origine et pour cible un même état x (autrement dit une boucle) existe à condition que le niveau d'expression de chaque gène $g \in G$ dans x soit égal à

la valeur du paramètre associé à g et à ses régulateurs effectifs dans x (*condition d'état stable*).

Exemple 13. La dynamique représentée dans la figure 2.6 est la dynamique induite par l'instance de paramètres indiquée dans l'exemple 12.

2.3.5 Contraintes sur les paramètres

Remarquons qu'aucune contrainte spécifique n'est incluse dans la définition d'une *instance de paramètres* (définition 2.3.7). Par conséquent, n'importe quelle fonction $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$ peut être associée aux paramètres \mathbb{K} d'un graphe d'interactions : en effet, la dynamique concrète issue d'une instance de paramètres \mathcal{K} non contrainte pourrait être en inadéquation avec des évidences biologiques basiques. Afin d'éviter cela, nous introduisons la notion d'*instance de paramètres bien formée*.

Définition 2.3.9 (Instance de paramètres bien formée). Une instance de paramètres \mathcal{K} de Γ est dite *bien formée* si elle satisfait les contraintes (exprimées sur $\text{Contraintes}(\mathbb{K})$)⁶ suivantes :

Contrainte de définition C_{def} :

$$\forall g \in G, \bigwedge_{g' \in G_+(g), \omega \subseteq G^-(g) \setminus \{g'\}} K_{g,\omega} \leq K_{g,(\omega \cup \{g'\})} \wedge \bigwedge_{g' \in G_-(g), \omega \subseteq G^-(g) \setminus \{g'\}} K_{g,\omega} \geq K_{g,(\omega \cup \{g'\})}$$

Contrainte d'observation C_{obs} :

$$\forall g \in G, \bigwedge_{g' \in G_+(g)} \left(\bigvee_{\omega \subseteq G^-(g) \setminus \{g'\}} K_{g,\omega} < K_{g,(\omega \cup \{g'\})} \right) \wedge \bigwedge_{g' \in G_-(g)} \left(\bigvee_{\omega \subseteq G^-(g) \setminus \{g'\}} K_{g,\omega} > K_{g,(\omega \cup \{g'\})} \right)$$

Contrainte Min/Max $C_{min/max}$:

$$\forall g \in G, K_{g,G^-(g)} = 0 \wedge K_{g,G_+(g)} = m_g$$

La *contrainte de définition*, ou contrainte de Snoussi [ST93], spécifie que si un activateur (respectivement inhibiteur) $g' \in G$ d'un gène $g \in G$ devient effectif (*i.e.* son niveau d'expression franchit le seuil de régulation de l'interaction de g' sur g), alors le niveau d'expression de g ne peut pas diminuer (respectivement augmenter).

La *contrainte d'observation* indique comment est identifié l'effet d'un régulateur sur sa cible. Si $g' \in G$ est un activateur (respectivement inhibiteur) de $g \in G$, alors il existe au moins un état dans lequel l'augmentation du niveau d'expression de g' au-dessus de son seuil de régulation tend à une augmentation (respectivement diminution) du niveau d'expression de g .

Enfin, la *contrainte min/max* spécifie que la valeur du paramètre correspondant à l'ensemble de régulateurs effectifs composé de l'ensemble des activateurs (respectivement inhibiteurs) d'un gène est maximale (respectivement minimale, c'est-à-dire égale à 0).

Exemple 14. Une instance de paramètres \mathcal{K} du graphe d'interactions de la figure 2.5 est bien formée si elle satisfait les contraintes suivantes :

$$\begin{aligned} C_{def} &\stackrel{\text{def}}{=} K_{\alpha,\{\}} \leq K_{\alpha,\{\alpha\}} \wedge K_{\alpha,\{\beta\}} \leq K_{\alpha,\{\alpha,\beta\}} \wedge K_{\alpha,\{\}} \geq K_{\alpha,\{\beta\}} \wedge K_{\alpha,\{\alpha\}} \geq K_{\alpha,\{\alpha,\beta\}} \wedge K_{\beta,\{\}} \leq K_{\beta,\{\alpha\}} \\ C_{obs} &\stackrel{\text{def}}{=} (K_{\alpha,\{\}} < K_{\alpha,\{\alpha\}} \vee K_{\alpha,\{\beta\}} < K_{\alpha,\{\alpha,\beta\}}) \wedge (K_{\alpha,\{\}} > K_{\alpha,\{\beta\}} \vee K_{\alpha,\{\alpha\}} < K_{\alpha,\{\alpha,\beta\}}) \wedge K_{\beta,\{\}} < K_{\beta,\{\alpha\}} \\ C_{min/max} &\stackrel{\text{def}}{=} K_{\alpha,\{\alpha\}} = 2 \wedge K_{\alpha,\{\beta\}} = 0 \wedge K_{\beta,\{\alpha\}} = 1 \wedge K_{\beta,\{\}} = 0 \end{aligned}$$

Une formulation équivalente de $C_{def} \wedge C_{obs} \wedge C_{min/max}$ est :

$$K_{\alpha,\{\alpha\}} = 2 \wedge K_{\alpha,\{\beta\}} = 0 \wedge K_{\beta,\{\alpha\}} = 1 \wedge K_{\beta,\{\}} = 0 \wedge (K_{\alpha,\{\}} < 2 \vee 0 < K_{\alpha,\{\alpha,\beta\}}) \wedge (K_{\alpha,\{\}} > 0 \vee 2 > K_{\alpha,\{\alpha,\beta\}})$$

Même si ces contraintes correspondent à des intuitions biologiques communément admises, elles ne sont pas toujours toutes prises en compte par les biologistes. Dans la suite, elles seront par défaut utilisées et regroupées sous le terme $CPBF = C_{def} \wedge C_{obs} \wedge C_{min/max}$. Elles peuvent cependant être relâchées sur demande, contrainte par contrainte, gène par gène ou encore interaction par interaction.

Ainsi, si on souhaite que la contrainte de définition ne soit pas prise en compte pour un gène $g \in G$, on notera $C_{def \setminus \{g\}}$ la nouvelle contrainte de définition obtenue satisfaite par tous les $g' \in G \setminus \{g\}$. On étendra cette notation pour exclure un ensemble de gènes et pour les autres contraintes de $CPBF$.

De la même façon, on peut souhaiter réduire l'effet de régulation d'un gène g'' sur un gène g , ce qui revient à relâcher la contrainte de définition (et/ou d'observation) pour l'interaction $(g'', s, t, g) \in I$. On obtient ainsi pour la contrainte de définition, notée $C_{def \setminus \{(g'', s, t, g)\}}$:

$$C_{def \setminus \{g\}} \wedge \bigwedge_{g' \in G_+^-(g) \setminus \{g''\}, \omega \subseteq G^-(g) \setminus \{g''\}} K_{g,\omega} \leq K_{g,(\omega \cup \{g''\})} \wedge \bigwedge_{g' \in G_+^-(g) \setminus \{g''\}, \omega \subseteq G^-(g) \setminus \{g''\}} K_{g,\omega} \geq K_{g,(\omega \cup \{g''\})}$$

En généralisant sur le même principe, on peut exclure l'effet d'un ensemble de régulateurs et appliquer le même principe à la contrainte d'observation.

2.4 Sélection de modèles à partir d'observations temporelles

Pour un graphe d'interactions Γ donné, il existe a priori de nombreuses instances de paramètres $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$ qui induisent des dynamiques $\mathcal{D}_{\mathcal{K}}$ différentes. En pratique, les valeurs des paramètres biologiques ne sont généralement pas identifiables directement. Cependant, les dynamiques de Γ peuvent être connues à l'aide d'observations partielles ou des hypothèses des biologistes.

Les biologistes peuvent par exemple observer des états stables ou des oscillations lors d'expériences *in vivo* ou *in vitro* ou simplement postuler leur existence. De même, ils peuvent être amenés à observer des séries temporelles, i.e. une succession d'états (non nécessairement consé-

cutifs). Ces trois types d'observation sont illustrés en figure 2.7 : la figure 2.7a contient un état stable (l'état $(2, 1)$), la figure 2.7b contient un cycle de 6 états $((0, 0) \rightarrow (1, 0) \rightarrow (2, 0) \rightarrow (2, 1) \rightarrow (1, 1) \rightarrow (0, 1) \rightarrow (0, 0))$ et la figure 2.7c décrit une série temporelle composée de 3 états consécutifs $((0, 0), (2, 0)$ et $(2, 1)$), éventuellement séparés par d'autres états.

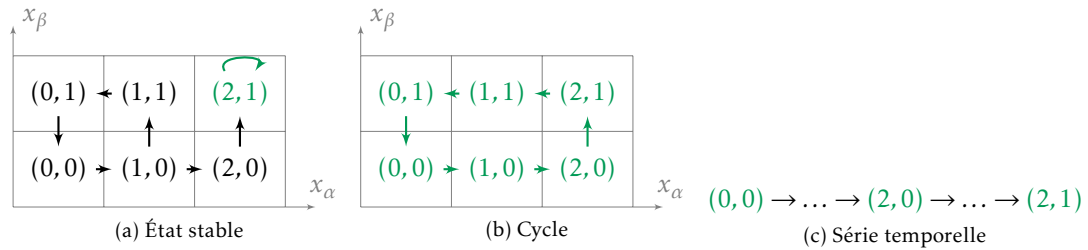


FIGURE 2.7 – Exemples de dynamiques associées à l'IG en figure 2.5 dont une partie du comportement (associée à la couleur verte) peut typiquement être observée *in vivo* ou *in vitro*.

Comme précédemment évoqué dans l'introduction de ce manuscrit, une partie des observations biologiques (notamment celles citées ci-dessus) peuvent être traduites sous forme de formules logiques temporelles, par exemple en LTL (*Linear Temporal Logic*) [Pnu77] ou en CTL (*Computation Tree Logic*) [EC80]. Ainsi, le modèle de René Thomas se prête assez naturellement à l'utilisation du *model-checking* [CGP99; BK08].

BERNOT, COMET, RICHARD et GUESPIN sont les premiers à avoir eu recours aux techniques de *model-checking* pour l'inférence des paramètres de Thomas. Dans [Ber+04], BERNOT et al. utilisent le *model-checking* CTL selon la procédure simplifiée suivante (représentée en figure 2.8) : étant donnée une propriété biologique exprimée sous la forme d'une formule CTL φ (qui correspond à la *spécification* que le système doit satisfaire), chaque instance de paramètres est considérée l'une après l'autre, afin de réaliser un test de satisfaction de φ pour chaque dynamique associée à l'instance. Pour chaque dynamique qui vérifie φ , l'instance de paramètres qui lui est associée représente une solution ; dans le cas contraire, la dynamique est un contre-exemple.

À l'issue de ce travail, il est donc possible de savoir si un ensemble de propriétés CTL peut être vérifié par au moins une dynamique ; plus encore, l'ensemble des instances de paramètres compatibles avec ces connaissances peut être énuméré.

La méthode de BERNOT et al. est implémentée dans un outil appelé SMBioNet [Ric10a], qui peut ainsi être utilisé pour inférer les paramètres de Thomas conformément à des connaissances biologiques. La vérification des formules CTL est réalisée en utilisant le *model-checker* NuSMV [Cim+02].

Dans [Ber+04] et [Fil+06], SMBioNet est utilisé pour inférer les paramètres d'un GRN lié à *Pseudomonas aeruginosa*. Nous reviendrons sur cet exemple dans le chapitre 4, section 5.1, p.86.

Cette approche doit cependant faire face à un écueil majeur, i.e. « la difficulté du passage à

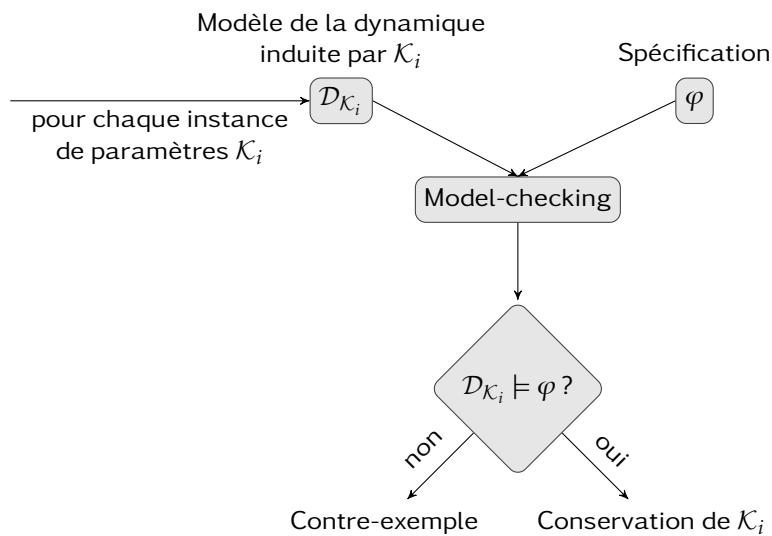


FIGURE 2.8 – Aperçu de la procédure de model-checking classique appliquée à la discrimination de modèles de dynamique

l'échelle », car le nombre de dynamiques associées à un GRN s'exprime en double exponentielle de la structure du réseau. Ainsi, le nombre $|\mathcal{K}|$ d'instances de paramètres d'un graphe d'interactions dépend du nombre de gènes et du nombre d'interactions. Pour un gène $g \in G$ donné, le nombre de paramètres $|\mathbb{K}_g|$ s'exprime en fonction de son nombre de régulateurs, il est égal à $2^{|\mathcal{G}^-(g)|}$. Chacun de ces paramètres peut être associé à $m_g + 1$ valeurs différentes par une instance de paramètres $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$. On obtient ainsi :

$$|\mathcal{K}| = \prod_{g \in G} (m_g + 1)^{2^{|\mathcal{G}^-(g)|}}$$

Dans le cas de notre exemple jouet, nous obtenons un total de 324 possibilités d'instance de paramètres. Un autre exemple, qui sera traité dans le chapitre 5 (section 5.2, p.98), est un réseau plus grand composé de 4 gènes et 10 interactions ; il contient 24 paramètres et correspond à un total de presque 7 milliards d'instances de paramètres différentes possibles.

En pratique, le model-checking CTL classique n'est donc pas envisageable pour espérer passer à l'échelle afin de traiter des GRN comportant plus de quelques gènes en un temps raisonnable.

À la suite du travail de BERNOT, COMET, RICHARD et GUESPIN [Ber+04], plusieurs pistes de méthodes d'inférence des paramètres des modèles de Thomas ont été suivies pour pouvoir traiter des GRN plus grands. Examinons dans un premier temps celles qui se basent sur SMBioNet.

KHALIS, COMET, RICHARD et BERNOT proposent dans [BCK08 ; Kha+09] d'enrichir le modèle de Thomas en intégrant explicitement de nouvelles données biologiques issues de la connaissance de la formation de complexes protéiques (ce qui peut correspondre à des situations de coopéra-

tion ou de concurrence entre protéines régulatrices).

Ces connaissances sont en fait des restrictions sur le domaine de valeurs des paramètres, cela permet ainsi de réduire l'espace de solutions initial. Ils se basent néanmoins toujours sur leur outil SMBioNet, et utilise donc le model-checking CTL classique.

À titre d'exemple, ils effectuent une comparaison dans [BCK08] entre le temps de calcul nécessaire au traitement du modèle de Thomas et celui du modèle enrichi pour une étude de cas portant sur la régulation de la résorption de la queue du têtard par des hormones thyroïdiennes (le réseau est composé de 8 gènes et 14 interactions). Le nombre de possibilités d'instances de paramètres de Thomas dans le cadre de Thomas originel est de l'ordre de 2.7×10^{11} ; dans le modèle enrichi, ce nombre est ramené à moins de 2.6×10^8 possibilités. En utilisant SMBioNet, ils passent ainsi de 54h (modèle original) à 20 minutes (modèle enrichi, prise en compte de la formation de complexes ainsi que la contrainte de définition, cf. section 2.3.5 p.40) de temps de traitement pour obtenir les solutions.

Plus récemment, SAEED et AHMAD ont proposé dans [SA14] une implémentation parallèle de SMBioNet [Ric10a]. Leur outil partitionne l'espace des paramètres en plusieurs sous-ensembles de combinaisons de paramètres, puis attribue chaque sous-ensemble à un processeur différent. Chaque processeur traite indépendamment les jeux de paramètres qui lui ont été attribués en utilisant SMBioNet.

En testant sur un réseau composé de 5 gènes et 7 interactions, ils affirment ainsi avoir amélioré la vitesse de traitement d'inférence de ses paramètres d'un facteur 7 par rapport à SMBioNet (le temps de traitement de ce réseau est de 48,12s avec un seul processus, contre 7,12s avec 8 processus exécutés en parallèle).

D'autres travaux, tels que [Fro+07 ; Cor+09], se placent dans le cadre de la programmation par contraintes pour inférer les paramètres de Thomas. Dans [Fro+07], FROMENTIN, COMET, LE GALL et ROUX encodent les conditions de transition (augmentation, diminution, stabilité des niveaux d'expression) sous la forme de contraintes sur les paramètres et ils prennent en compte des connaissances biologiques sur les dynamiques du GRN sous la forme de formules CTL, qu'ils expriment également sous la forme de contraintes à satisfaire.

Leur méthode est implémentée dans l'outil SeMoCo-GRN. Dans [Fro+07], ils estiment le gain en temps de traitement par rapport à SMBioNet en testant le cas d'étude présenté dans [TT95] (dont l'espace de solutions associé est de l'ordre de 3 millions) : SeMoCo-GRN trouve les 32 solutions en 800ms, tandis que SMBioNet a besoin de 8m30s.

Dans [Cor+09 ; CFT10], CORBLIN, TRIPODI, FANCHON, ROPERS et TRILLING utilisent des techniques de programmation logique avec contraintes combiné avec un solveur SAT pour identifier les paramètres. Les dynamiques et les observations et hypothèses biologiques sont décrites par le biais de contraintes sur les paramètres, et les comportements à vérifier sont exprimées uniquement sur des chemins finis (du type : *on suppose que le comportement sera vérifié en 6 étapes maximum*). Ils proposent une approche pas-à-pas de prise en compte des connaissances biologiques (ajout, suppression, adaptation) afin de faire face aux éventuelles incertitudes sur

les observations ou les hypothèses biologiques ou pour tester de nouvelles pistes. Leur méthode est implémentée dans l'outil GNBox.

Dans [Cor+12], des données spatio-temporelles sont prises en compte afin de sélectionner les graphes d'interaction possibles. Cette méthode est implémentée dans l'outil SysBiOX.

Enfin, certains ont choisi d'utiliser le model-checking LTL pour inférer les paramètres de Thomas, allié à d'autres pistes pour tenter de franchir le passage à l'échelle.

Dans [Ito+13], ITO, ICHINOSE, SHIMAKAWA, IZUMI, HAGIHARA et YONEZAKI font ainsi appel au model-checking LTL. En quelques mots, la procédure classique du model-checking LTL consiste tout d'abord à exprimer sous la forme d'une formule LTL φ la spécification donnée en entrée (ici les connaissances biologiques). Ensuite, un automate de Büchi \mathcal{B} reconnaissant la *négation* de la formule LTL est construit (voir chapitre Préliminaires, section 1.5, p.21). La dernière étape consiste à détecter les chemins menant à un cycle acceptant dans le produit du graphe de transition d'état \mathcal{G} représentant le système (ici la dynamique associée à une instance de paramètres donnée) et de \mathcal{B} (voir chapitre Préliminaires, section 1.5, p.21). Si un tel chemin est trouvé, alors il existe un chemin dans \mathcal{G} qui satisfait $\neg\varphi$ et, par conséquence, la propriété attendue n'est pas satisfaite sur tous les chemins de \mathcal{G} : \mathcal{G} est donc un contre-exemple. À l'inverse, si aucun de ses chemins ne vérifie $\neg\varphi$ alors tous ses chemins vérifient φ : \mathcal{G} est donc un modèle de φ .

Appliqué à la vérification des dynamiques d'un GRN, on comprend que la procédure classique du model-checking LTL souffre du même problème d'explosion combinatoire que le model-checking CTL. Pour pallier ce problème, Ito et al. proposent de découper l'automate de Büchi issu de la formule LTL en exprimant les connaissances biologiques en plusieurs automates de Büchi de façon à réaliser une analyse modulaire du produit [Ito+13].

Dans [Kla+12a ; Bar+12], KLARNER et al. ; BARNAT et al. cherchent à rassembler les calculs entre plusieurs combinaisons de paramètres, encodées par des vecteurs binaires, en se basant sur l'heuristique que des combinaisons proches partagent en grande partie la même dynamique. Ils se basent sur le *model-checking coloré*, qui associe une couleur à chaque dynamique, et le model-checking LTL étendu aux opérations booléennes sur les vecteurs.

Dans leur première version, seules les formules LTL représentant des séries temporelles étaient prises en compte, ils ont par la suite étendu aux automates de Büchi généraux mais tout en gardant une spécialisation pour les séries temporelles, leurs calculs étant plus rapides dans ce cas. Très récemment, ils ont adapté leur méthode au model-checking CTL [Bri+15].

Dans les prochains chapitres, nous proposons une autre méthode d'inférence des paramètres des modèles de Thomas qui tout comme dans [Ito+13] et [Kla+12a ; Bar+12], s'inspire du model-checking LTL.

À la différence de [Kla+12a ; Bar+12] qui implicitement énumère l'ensemble des dynamiques par le biais de l'utilisation d'un vecteur binaire, nous préférons manipuler les dynamiques à l'aide de variables comme pour les approches fondées sur la programmation logique avec contraintes. Cependant, contrairement à [Cor+09 ; CFT10], nous ne nous restreignons pas à des chemins

dont la taille est prédéterminée. En sus, comme [BCK08 ; Kha+09], nous laissons la possibilité d'apporter des connaissances (statiques) supplémentaires sur les paramètres biologiques.

Pour faire face à l'explosion combinatoire, nous utiliserons des techniques d'exécution symbolique combinées à la résolution de contraintes en mettant en œuvre des mécanismes de coupures dédiés pour minimiser autant que possible le parcours de la structure paramétrée par les paramètres biologiques.

Notre approche est détaillée dans le chapitre 3. Une version préliminaire et simplifiée a été publiée dans [Mat+07] : il s'agissait d'une adaptation de l'outil AGATHA [Big+03] développé au CEA (LIST), dédié à l'analyse symbolique de modèles de systèmes réactifs. Ce travail tirait parti du savoir-faire en termes d'exécution symbolique et de parcours de structure arborescente, mais n'était pas spécialisé pour l'étude des GRN, et en particulier n'intégrait pas de mécanismes de coupures.

Méthode d'inférence de paramètres de Thomas basée sur le model-checking LTL et la résolution de contraintes

Dans ce chapitre, nous présentons notre méthode permettant d'inférer les paramètres du modèle de Thomas. Nous utilisons les notations introduites dans le chapitre précédent, en particulier : $\Gamma = (G, I)$ représente un graphe d'interactions (définition 2.3.1), l'espace des états associés à Γ est noté \mathbb{X} (définition 2.3.2), et les variables d'état et de paramètre de Γ sont respectivement notées \mathbb{G} (définition 2.3.4) et \mathbb{K} (définition 2.3.6).

L'approche que nous proposons vise à réduire l'effet de l'explosion combinatoire qui se produit en appliquant la procédure classique du model-checking LTL, comme nous l'avons expliqué à la fin du chapitre précédent.

Notre approche s'inspire néanmoins du model-checking LTL classique, mais, au lieu de vérifier chaque dynamique induite par une instance de paramètre $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$ de Γ , nous définissons un méta-modèle qui permet de représenter l'ensemble des dynamiques possibles en une seule structure. Ce méta-modèle, que nous appelons PGRN (pour *Parametric GRN*, soit *GRN paramétré*) est paramétré par les états \mathbb{G} et les paramètres \mathbb{K} de Γ .

Afin de conserver l'indépendance des dynamiques par rapport aux paramètres instanciés pendant la procédure adaptée du model-checking LTL, les paramètres sont manipulés uniquement symboliquement, c'est-à-dire sous forme de variables non instanciées (variables dans \mathbb{K}). Pour cela, nous utilisons un solveur de contrainte (cf. chapitre 1, section 1.2 p.13). L'exécution du produit de l'automate de Büchi (correspondant à la formule LTL à satisfaire) et du PGRN associé à Γ

est réalisée en associant chaque état du chemin parcouru à une contrainte dans $Contraintes(\mathbb{K})$, correspondante aux conditions que doivent satisfaire les instances de paramètres pour atteindre ces états. En déterminant alors les *cycles acceptants* du produit, nous obtenons une contrainte dans $Contraintes(\mathbb{K})$ qui est satisfaite (uniquement) par les instances de paramètres correspondant aux dynamiques de Γ qui satisfont la formule LTL.

La figure 3.1 résume notre approche.

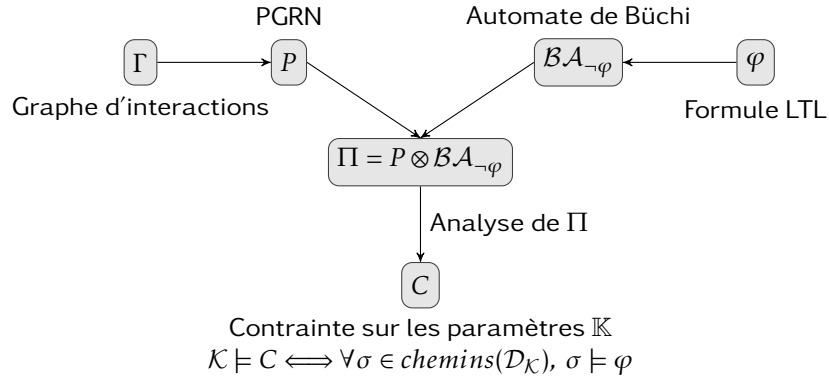


FIGURE 3.1 – Vue schématique de notre adaptation du model-checking LTL pour l'inférence de paramètres

Dans la suite, nous commencerons par définir le PGRN et le produit d'un PGRN et d'un automate de Büchi, puis nous détaillerons notre procédure d'exécution symbolique du produit.

3.1 GRN Paramétré	49
3.1.1 Caractérisation des états par contraintes	49
3.1.2 GRN Paramétré	51
3.1.3 Dynamiques associées à un PGRN	54
3.2 Synchronisation d'un PGRN avec un automate de Büchi	58
3.3 Exécution symbolique du Produit	60
3.3.1 Vue d'ensemble	60
3.3.2 Construction des Arbres d'Exécution	62
3.3.3 Recherche des cycles acceptants	66
3.4 Inférence des paramètres	67

Tout au long de ce chapitre, nous continuerons d'utiliser « l'exemple jouet » introduit dans le chapitre précédent (chapitre 2 p.25).

3.1 GRN Paramétré

3.1.1 Caractérisation des états par contraintes

Avant de définir le GRN Paramétré associé à un graphe d'interactions, nous avons besoin de caractériser formellement les états $x \in \mathbb{X}$ dans lesquels le niveau d'expression x_g d'un gène $g \in G$ est susceptible d'augmenter, de diminuer ou de rester stable.

Ces états sont définis à l'aide de contraintes qu'ils doivent respecter. Les contraintes s'expriment à la fois en fonction des variables d'état \mathbb{G} et des paramètres \mathbb{K} du graphe d'interactions car l'évolution du niveau d'expression d'un gène est guidée par le paramètre correspondant aux *régulateurs effectifs* de g dans un état $x \in \mathbb{X}$ (définition 2.3.5). Ces contraintes seront ensuite utilisées pour caractériser les *gardes* des transitions du PGRN.

Régulateurs effectifs

La *contrainte des régulateurs effectifs* permet d'identifier les états de \mathbb{X} dans lesquels des gènes ω régulateurs de g (et uniquement eux) sont *effectifs*. Elle s'exprime en fonction des variables d'état de \mathbb{G} , i.e. $P_{g,\omega}(\chi)$ appartient à $\text{Contraintes}(\mathbb{G})$.

Définition 3.1.1 (Contrainte des régulateurs effectifs). Soit $g \in G$ un gène et $\omega \in G^-(g)$ un ensemble de gènes régulateurs de g . Les états où ω sont les régulateurs effectifs de g sont les états vérifiant la contrainte suivante :

$$P_{g,\omega}(\chi) \stackrel{\text{def}}{=} \left(\bigwedge_{g' \in \omega} \chi_{g'} \geq t(g', g) \right) \wedge \left(\bigwedge_{g' \in G^-(g) \setminus \omega} \chi_{g'} < t(g', g) \right) \quad (3.1)$$

Exemple 15. Par exemple, pour $g = \alpha$ et $\omega = \{\beta\}$ on obtient la contrainte suivante :

$$P_{\alpha, \{\beta\}}(\chi) \stackrel{\text{def}}{=} (\chi_\beta \geq t(\beta, \alpha)) \wedge (\chi_\alpha < t(\alpha, \alpha))$$

soit :

$$P_{\alpha, \{\beta\}}(\chi) \stackrel{\text{def}}{=} (\chi_\beta \geq 1) \wedge (\chi_\alpha < 2)$$

On en déduit que les états où β est le seul régulateur effectif de α sont les états $(0, 1)$ et $(1, 1)$.

Caractérisation des évolutions des états

En utilisant la *contrainte des régulateurs effectifs*, nous pouvons établir les contraintes liées aux possibilités d'évolution des niveaux d'expression des gènes en fonction des paramètres associés aux régulateurs effectifs dans chaque état.

Ainsi, les *contraintes d'augmentation, de diminution et de stabilité* (définition 3.1.2) permettent d'identifier les états de $x \in \mathbb{X}$ dans lesquels le niveau d'expression du gène g peut respectivement augmenter, diminuer ou rester constant, en fonction des paramètres \mathbb{K} . Elles s'expriment toutes les trois en fonction des variables d'état de \mathbb{G} et des variables de paramètres \mathbb{K} , i.e. elles sont définies sur $Contraintes(\mathbb{G} \cup \mathbb{K})$.

Définition 3.1.2 (Contraintes d'évolution des états). Soit $g \in G$ un gène. Nous définissons les contraintes suivantes :

Contrainte d'augmentation :

$$Increase(g)(\chi) \stackrel{\text{def}}{=} \bigvee_{\omega \in G^-(g)} (P_{g,\omega}(\chi) \wedge \chi_g < K_{g,\omega}) \quad (3.2)$$

Contrainte de diminution :

$$Decrease(g)(\chi) \stackrel{\text{def}}{=} \bigvee_{\omega \in G^-(g)} (P_{g,\omega}(\chi) \wedge \chi_g > K_{g,\omega}) \quad (3.3)$$

Contrainte de stabilité :

$$Stable(g)(\chi) \stackrel{\text{def}}{=} \bigvee_{\omega \in G^-(g)} (P_{g,\omega}(\chi) \wedge \chi_g = K_{g,\omega}) \quad (3.4)$$

Dans le détail, $Increase(g)(\chi)$ caractérise les états sources $x \in \mathbb{X}$ dont un des états cibles possibles correspond à une augmentation du niveau d'expression du gène g , en fonction des paramètres \mathbb{K} . De la même manière, $Decrease(g)(\chi)$ et $Stable(g)(\chi)$ caractérisent les états $x \in \mathbb{X}$ permettant une transition vers un état dans lequel, respectivement, x_g diminue et x_g reste au même niveau.

Dans le contexte du PGRN, les *contraintes d'augmentation, de diminution et de stabilité* seront interprétées en fonction des niveaux d'expression, correspondant à un état $x \in \mathbb{X}$, donnés par une fonction d'interprétation $v_x : \mathbb{G} \rightarrow \mathbb{N}$. Il s'agit d'une interprétation partielle des variables de $Increase(g)(\chi)$, $Decrease(g)(\chi)$ et $Stable(g)(\chi)$ car les paramètres \mathbb{K} ne sont pas instanciés (par exemple, $\llbracket Increase(g)(\chi) \rrbracket_{v_x}$ appartient à $Contraintes(\mathbb{K})$).

Comme nous le verrons par la suite, un *solveur de contraintes* peut déterminer si les contraintes dans $Contraintes(\mathbb{K})$ sont satisfaisables ou non sans pour autant avoir besoin d'instancier les paramètres, nous en aurons besoin lors de l'exécution symbolique (section 3.3).

Exemple 16. Pour α , la contrainte d'augmentation (non simplifiée) est la suivante :

$$\begin{aligned} \text{Increase}(\alpha)(\chi) &\stackrel{\text{def}}{=} \left(\chi_\alpha < 2 \wedge \chi_\beta < 1 \wedge \chi_\alpha < K_{\alpha, \{\}} \right) \\ &\vee \left(\chi_\alpha \geq 2 \wedge \chi_\beta < 1 \wedge \chi_\alpha < K_{\alpha, \{\alpha\}} \right) \\ &\vee \left(\chi_\alpha < 2 \wedge \chi_\beta \geq 1 \wedge \chi_\alpha < K_{\alpha, \{\beta\}} \right) \\ &\vee \left(\chi_\alpha \geq 2 \wedge \chi_\beta \geq 1 \wedge \chi_\alpha < K_{\alpha, \{\alpha, \beta\}} \right) \end{aligned}$$

Cette contrainte permet de déterminer l'ensemble des états dans lesquels α peut augmenter d'une unité. Chaque disjonction de $\text{Increase}(\alpha)(\chi)$ correspond à un ensemble différent de régulateurs effectifs ; il y a quatre possibilités pour α : l'ensemble vide, α seul, β seul et enfin, α et β ensemble.

3.1.2 GRN Paramétré

Doté des contraintes d'augmentation, de diminution et de stabilité, nous pouvons à présent définir le GRN Paramétré (PGRN) associé à Γ qui permet de représenter en une seule structure l'ensemble des dynamiques de Γ (i.e. l'ensemble des évolutions possibles des niveaux d'expression des gènes) en fonction des variables d'états \mathbb{G} et des paramètres \mathbb{K} .

Un PGRN est un système de transition étiqueté qui ne contient que deux états appelés T , comme *Transitoire*, et S , comme *Stable*. L'état *transitoire* correspond aux états de \mathbb{G} où le niveau d'expression d'au moins un gène est susceptible d'être modifié (augmentation ou diminution de son niveau d'expression). L'état *stable* quant à lui est associé aux états dans lesquels plus aucun gène n'évolue.

Les états T et S sont reliés par quatre types de transition, correspondant aux contraintes énumérées précédemment (définition 3.1.2).

Définition 3.1.3 (PGRN). Étant donné Γ un graphe d'interactions, un PGRN associé à Γ est un système de transition étiqueté $P = (Q_P, Q_{P_0}, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_P)$ avec :

- $Q_P = \{T, S\}$ l'ensemble des états
- $Q_{P_0} = \{T\}$ l'état initial
- $\delta_P \subseteq Q_P \times \text{Contraintes}(\mathbb{G} \cup \mathbb{K}) \times \text{Affectations}(\mathbb{X}) \times Q_P$ l'ensemble des transitions avec :
 $\text{Affectations}(\mathbb{X}) = \{a_P \mid a_P : \mathbb{X} \rightarrow \mathbb{X}\}$ l'ensemble des applications entre les états de \mathbb{X} .

Une transition de δ_P sera notée $(q_P, g_P, a_P, q'_P) \in \delta_P$ ou bien $q_P \xrightarrow{g_P [a_P]} q'_P$, avec g_P représentant la garde et a_P l'affectation de la transition. L'ensemble des transitions de P est le suivant :

Transitions d'augmentation :

$$\forall g \in G, T \xrightarrow{\text{Increase}(g)(\chi) [g^\dagger]} T \quad (3.5)$$

Transitions de diminution :

$$\forall g \in G, T \xrightarrow{\text{Decrease}(g)(x) [g\downarrow]} T \quad (3.6)$$

Transition menant à la stabilité :

$$T \xrightarrow{\bigwedge_{g \in G} \text{Stable}(g)(x) [id]} S \quad (3.7)$$

Transition de stabilité :

$$S \xrightarrow{\top [id]} S \quad (3.8)$$

Les transitions dites *d'augmentation* et de *diminution* sont propres à l'état T , elles correspondent aux évolutions entre deux états de \mathbb{G} liées à l'augmentation ou à la diminution du niveau d'expression d'un gène g d'une unité, provoquée par la différence de valeur entre le niveau d'expression de g et le paramètre correspondant aux régulateurs effectifs de g dans l'état donné.

La *transition menant à la stabilité* représente le passage de l'état transitoire à l'état stable. Sa garde spécifie que les niveaux d'expression de l'ensemble des gènes du graphe d'interactions doivent être respectivement égaux au paramètre associé à leurs régulateurs effectifs dans l'état considéré.

Enfin, la transition dite de *stabilité*, dont la source et la cible sont l'état stable, n'a ni garde restrictive ni affectation effective : elle n'influe pas sur l'état \mathbb{G} , garantissant le caractère de stabilité de S .

Exemple 17. Le PGRN construit à partir du graphe d'interactions de la figure 2.5 est donné en figure 3.2. Les gardes des transitions ont été simplifiées en prenant en compte la CPBF (cf. définition 2.3.9), suivant la procédure suivante :

- Prise en compte du domaine de valeur des états ;
ainsi, la contrainte $(x_\alpha < 2 \wedge x_\beta < 1 \wedge x_\alpha < K_\alpha(\{\}))$ devient : $(x_\alpha < 2 \wedge x_\beta = 0 \wedge x_\alpha < K_\alpha(\{\}))$.
- Instanciation des paramètres dont la valeur peut être fixée par la CPBF ;
en particulier, $K_\alpha(\{\beta\})$ est fixé grâce à la contrainte Min/Max : $K_\alpha(\{\beta\}) = 0$.
- Suppression des propositions formant les disjonctions qui sont alors évaluées à \perp ;
par exemple, la contrainte $(x_\alpha < 2 \wedge x_\beta \geq 1 \wedge x_\alpha < K_\alpha(\{\beta\})) \models \perp$ car $K_\alpha(\{\beta\}) = 0$.

Notons que quel que soit Γ , le PGRN sous la forme donnée par la définition 3.1.3 a toujours par construction deux états et une *transition menant à la stabilité* ainsi qu'une *transition de stabilité*. Par ailleurs, le nombre de *transitions d'augmentation* et de *transitions de diminution* est égal à deux fois le nombre de gènes de Γ .

De plus, les gardes des transitions d'augmentation et de diminution sont par construction sous la forme de disjonctions de plusieurs conjonctions. Or, lors de l'utilisation d'un solveur de

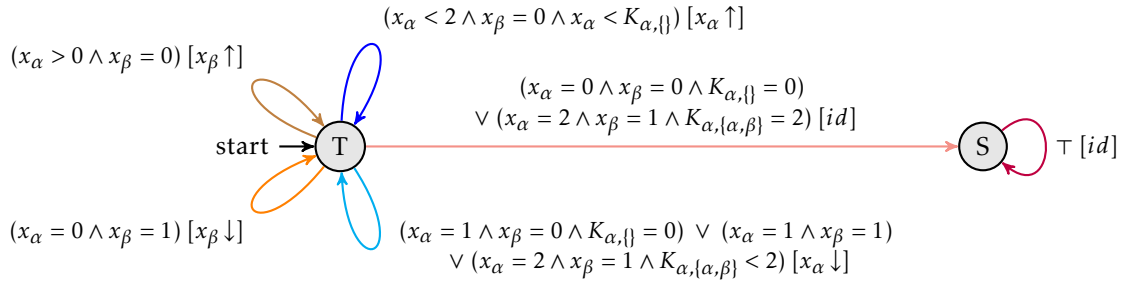


FIGURE 3.2 – PGRN associé à l’IG en figure 2.5, simplifié en tenant compte de la CPBE.

contraintes (dont nous verrons l’utilité plus en détails par la suite), les contraintes contenant des disjonctions de conjonction sont plus difficiles (consommant plus de mémoire et de temps) à résoudre que celles ne contenant que des conjonctions. En pratique, nous pourrions ainsi préférer remplacer une transition $q_P \xrightarrow{g_P [a_P]} q'_P$ dont la garde est une disjonction de n propositions (composées uniquement de conjonctions), soit :

$$g_P = \bigvee_{i=1}^n (P_{g, \omega_i}(\chi) \wedge \chi_g \bowtie K_{g, \omega_i})$$

avec : $\omega_1 \dots \omega_n \subset G^-(g)$ et $\bowtie \in \{<, >, =\}$

par n transitions de la forme :

$$q_P \xrightarrow{g_{P_i} [a_P]} q'_P$$

avec : $\forall i \in [1, n] \ g_{P_i} = (P_{g, \omega_i}(\chi) \wedge \chi_g \bowtie K_{g, \omega_i})$.

Le PGRN obtenu à partir du PGRN en figure 3.2 en séparant les transitions comportant des disjonctions en plusieurs transitions est représenté en figure 3.3.

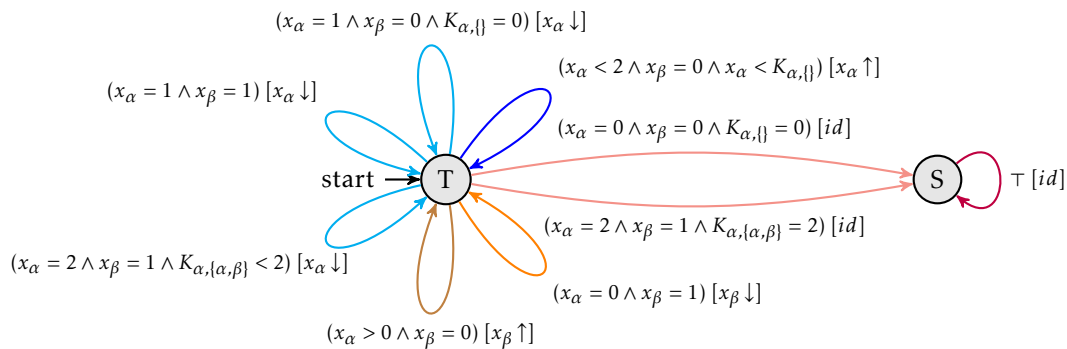


FIGURE 3.3 – PGRN associé à l’IG en figure 2.5, dont les transitions composées de gardes avec des disjonctions ont été séparées en plusieurs transitions.

Notons enfin que pour un gène g , le nombre n de propositions composant la disjonction de la garde de la transition d'augmentation (ou de diminution) est égale à $2^{G^-(g)}$, i.e. le nombre de combinaisons de régulateurs effectifs de g . Parmi ces n propositions, certaines ne sont cependant pas compatibles avec la CPBF (cf définition 2.3.9), i.e. la conjonction de ces contraintes avec les contraintes composant la CPBF n'est pas satisfaisable. En pratique, nous ne construirons pas les transitions contenant des gardes non satisfaisables, comme pour notre exemple présenté en figure 3.3.

3.1.3 Dynamiques associées à un PGRN

Le PGRN permet de représenter l'ensemble des possibilités d'évolution des niveaux d'expression des gènes en fonction des états dans \mathbb{X} et des paramètres dans \mathbb{K} . L'exemple suivant permet d'illustrer cette capacité.

Considérons un chemin σ dans le graphe d'interactions de notre exemple jouet en figure 2.5 constitué de trois états consécutifs : $\sigma = x_1.x_2.x_3$, avec par exemple $x_1 = (2, 0)$, $x_2 = (2, 1)$ et $x_3 = (2, 1)$. La variation entre x_1 et x_2 correspond à une augmentation de x_β d'une unité, et celle entre x_2 et x_3 est nulle (les deux états sont identiques). Appliqué au PGRN associé au graphe d'interactions, le chemin dans le PGRN est :

$$\sigma_P = (T, (2, 0)) \xrightarrow{(x_\alpha > 0 \wedge x_\beta = 0)[x_\beta \uparrow]} (T, (2, 1)) \xrightarrow{(x_\alpha = 2 \wedge x_\beta = 1 \wedge K_{\alpha, \{\alpha, \beta\}} = 2)[id]} (S, (2, 1))$$

Supposons à présent que l'état suivant de σ est $x_4 = (1, 1)$, soit une diminution de x_α d'une unité. Dans le PGRN, cette situation est incompatible avec les transitions qui ont été franchies dans le chemin σ_P menant à l'état stable. Cependant, l'état x_2 est identique à l'état x_3 , nous pouvons donc considérer que le système est passé directement de l'état x_2 à l'état x_4 (sans dommage car nous n'avons pas dans notre modèle de notion de temps passé dans un état). Ceci revient à franchir la transition de diminution de α dans le PGRN à partir de l'état transitoire. On obtient alors le chemin suivant :

$$\sigma'_P = (T, (2, 0)) \xrightarrow{(x_\alpha > 0 \wedge x_\beta = 0)[x_\beta \uparrow]} (T, (2, 1)) \xrightarrow{(x_\alpha = 2 \wedge x_\beta = 1 \wedge K_{\alpha, \{\alpha, \beta\}} < 2)[x_\alpha \downarrow]} (T, (1, 1))$$

Si on fait le parallèle avec une dynamique associée à une instance de paramètres qui permettrait d'atteindre les états x_1 , x_2 puis x_4 , l'état x_3 ne serait pas non plus parcouru entre x_2 et x_4 car aucune instance de paramètre ne peut permettre la stagnation puis l'augmentation (ou la diminution). Ceci est cohérent avec le fait qu'une boucle (ici $x_2 \rightarrow x_3$) est interdite dans une dynamique si une transition sortant vers un autre état est possible ($x_2 \rightarrow x_4$) (cf propriété **p3** de la définition 2.3.3).

Quelles que soient les variations des niveaux d'expression, nous avons ainsi l'intuition que nous pouvons appliquer la même méthode pour retrouver un chemin dans le PGRN (notons cependant que la simplification avec la CPBF n'est pas compatible avec l'ensemble des dynamiques).

Nous allons voir dans la suite de ce manuscrit comment établir une correspondance formelle entre une dynamique induite par une instance de paramètres et une dynamique dite *annotée*, associée à un PGRN et à une instance de paramètres.

Dynamique annotée

Une *dynamique annotée* correspond au système de transition obtenu à partir d'un PGRN pour décrire les évolutions des états (correspondant à un état du PGRN et au niveau d'expression des gènes) en considérant une instance de paramètres donnée.

Définition 3.1.4 (Dynamiques annotées induites par une instance de paramètres). Soit $P = (Q_P, Q_{P_0}, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_P)$ un PGRN associé à un graphe d'interactions $\Gamma = (G, I)$ et soit $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$ une instance de paramètres. La dynamique annotée associée à P et \mathcal{K} est un système de transition $D_{\mathcal{K}} = (Q_D, Q_{D_0}, \delta_D)$ avec :

- $Q_D \subset Q_P \times \mathbb{X}$ l'ensemble des états ;
- $Q_{D_0} \subset Q_D$ l'ensemble des états initiaux ;
- $\delta_D \subset Q_D \times Q_D$ l'ensemble des transitions.

Les états et les transitions de $D_{\mathcal{K}}$ sont mutuellement définis à partir des propriétés suivantes :

- $Q_{D_0} = Q_{P_0} \times \mathbb{X}$;
- si $(q_P, x) \in Q_D$ et $(q_P, g_P, a_P, q'_P) \in \delta_P$ tel que $\llbracket g_P \rrbracket_{v,x,\mathcal{K}} = \top$, alors $(q'_P, a_P(x)) \in Q_D$ et $((q_P, x), (q'_P, a_P(x))) \in \delta_D$.

Exemple 18. La figure 3.4 représente la dynamique annotée associée au PGRN en figure 3.2 et à l'instance de paramètres donnée dans l'exemple 12.

L'ensemble des états de la dynamique annotée est : $(T, (0, 0)), (T, (1, 0)), (T, (2, 0)), (T, (0, 1)), (T, (1, 1)), (T, (2, 1))$ et enfin $(S, (2, 1))$, qui est le seul état stable (caractérisé par la composante S) atteint .

Lien entre dynamique annotée et dynamique d'un GRN

Une dynamique annotée décrit notamment l'évolution des niveaux d'expression des gènes du graphe d'interactions considéré en fonction d'une instance de paramètres. La propriété suivante permet d'établir une correspondance entre une dynamique d'un graphe d'interactions Γ induite par une instance de paramètres \mathcal{K} et une dynamique annotée associée au PGRN de Γ et à \mathcal{K} .

Propriété 3.1.1. Soit $P = (Q_P, Q_{P_0}, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_P)$ le PGRN associé à un graphe d'interactions Γ et $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$ une instance de paramètres de Γ . On note $D_{\mathcal{K}} = (Q_D, \delta_D)$ la dynamique annotée associée à P et \mathcal{K} et $\mathcal{D}_{\mathcal{K}} = (\mathbb{X}, \rightarrow_{\mathcal{K}})$ la dynamique de Γ induite par \mathcal{K} .

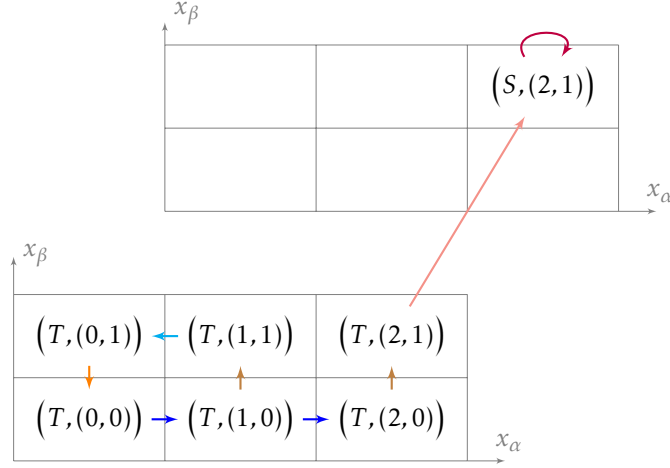


FIGURE 3.4 – La dynamique annotée associée au PGRN en figure 3.2 et à l'instance de paramètres donnée dans l'exemple 12.

On note $\text{proj}(D_{\mathcal{K}}) = (\mathbb{X}, \rightsquigarrow_{\mathcal{K}})$ le système de transition dont la relation de transition est définie ainsi :

$$\forall x, x' \in \mathbb{X}, \quad x \rightsquigarrow_{\mathcal{K}} x' \Leftrightarrow \exists ((q_p, x), (q'_p, x')) \in \delta_D$$

Alors le système de transition $\mathcal{D}_{\mathcal{K}}$ et $\text{proj}(D_{\mathcal{K}})$ sont isomorphes : $\text{proj}(D_{\mathcal{K}}) = \mathcal{D}_{\mathcal{K}}$.

Démonstration. Montrons que $\text{proj}(D_{\mathcal{K}}) = \mathcal{D}_{\mathcal{K}}$ se ramène à démontrer que $\rightarrow_{\mathcal{K}} = \rightsquigarrow_{\mathcal{K}}$, qui à son tour revient à démontrer les inclusions réciproques d'ensembles correspondantes $\rightarrow_{\mathcal{K}} \subseteq \rightsquigarrow_{\mathcal{K}}$ et $\rightsquigarrow_{\mathcal{K}} \subseteq \rightarrow_{\mathcal{K}}$:

Commençons par démontrer $\rightarrow_{\mathcal{K}} \subseteq \rightsquigarrow_{\mathcal{K}}$. Soit $x, x' \in \mathbb{X}$ deux états génériques et supposons que $x \rightarrow_{\mathcal{K}} x'$. Par définition de $\mathcal{D}_{\mathcal{K}}$ (définition 2.3.8), nous avons à distinguer trois cas possibles pour $x \rightarrow_{\mathcal{K}} x'$: $x' = x[g \uparrow]$, $x' = x[g \downarrow]$ et $x' = x$.

Supposons tout d'abord $x' = x[g \uparrow]$. Si $x \rightarrow_{\mathcal{K}} x[g \uparrow]$ alors, par définition de la relation $\rightarrow_{\mathcal{K}}$ (définition 2.3.8), on a :

$$x_g < \mathcal{K}(K_{g, \omega_g^*(x)})$$

avec $\omega_g^*(x)$ l'ensemble des régulateurs effectifs de g dans l'état x .

Par ailleurs, par définition des régulateurs effectifs (définition 2.3.5),

$$\forall g' \in \omega_g^*(x), \quad x_{g'} \geq t(g', g).$$

En appliquant la *contrainte des régulateurs effectifs* (définition 3.1.1), on obtient alors :

$$\llbracket P_{g, \omega_g^*(x)} \rrbracket_{\mathcal{K}} = \top.$$

D'après la *contrainte d'augmentation* (définition 3.1.2), on en déduit :

$$\llbracket \text{Increase}(g)(\chi) \rrbracket_{x, \mathcal{K}} = \top.$$

Ceci implique (définition 3.1.3) l'existence de la transition $T \xrightarrow{\text{Increase}(g)(\chi) [g\uparrow]} T \in \delta_P$ dans le PGRN P , et par conséquent (définition 3.1.4) l'existence de la transition $((T, x), (T, [x\uparrow])) \in \delta_D$ dans la dynamique annotée $D_{\mathcal{K}}$.

Donc, par projection, on obtient : $x \rightsquigarrow_{\mathcal{K}} x'$.

Pour les deux autres cas ($x' = x[g\downarrow]$ et $x' = x$), la preuve est similaire à celle qui vient d'être exécutée.

On en déduit $\rightarrow_{\mathcal{K}} \subseteq \rightsquigarrow_{\mathcal{K}}$.

Démontrons à présent $\rightsquigarrow_{\mathcal{K}} \subseteq \rightarrow_{\mathcal{K}}$. Soit $x, x' \in \mathbb{X}$ deux états génériques et supposons $x \rightsquigarrow_{\mathcal{K}} x'$. D'après la définition de la relation $\rightsquigarrow_{\mathcal{K}}$ (propriété 3.1.1), il existe une transition

$$((q_P, x), (q'_P, x')) \in \delta_D$$

dans la dynamique annotée $D_{\mathcal{K}} = (Q_D, Q_{D_0}, \delta_D)$. Nous avons alors trois cas à considérer : $x' = x[g\uparrow]$, $x' = x[g\downarrow]$ et $x = x'$.

Supposons le cas $x' = x[g\uparrow]$. Si $x' = x[g\uparrow]$ alors il existe une transition

$$((q_P, x), (q'_P, x[g\uparrow])) \in \delta_D.$$

D'après la définition 3.1.4, on en déduit qu'il existe une transition

$$(q_P, g_P, \uparrow, q'_P) \in \delta_P$$

dans le PGRN $P = (Q_P, \delta_P)$ avec $\llbracket g_P \rrbracket_{v_x, \mathcal{K}} = \top$.

D'après la définition 3.1.3, cette transition ne peut correspondre qu'à une transition du type $T \xrightarrow{\text{Increase}(g)(\chi) [g\uparrow]} T \in \delta_P$ donc :

$$\llbracket \text{Increase}(g)(\chi) \rrbracket_{v_x, \mathcal{K}} = \top.$$

$\llbracket \text{Increase}(g)(\chi) \rrbracket_{v_x, \mathcal{K}} = \top$ implique qu'il y a au moins une disjonction de la *contrainte d'augmentation* (définition 3.1.2) qui est évaluée à vrai, i.e. qu'il existe $\omega' \subset G^-(g)$ tel que :

$$\llbracket P_{g, \omega'}(\chi) \wedge \chi_g < K_{g, \omega'} \rrbracket_{v_x, \mathcal{K}} = \top$$

soit :

$$\llbracket P_{g, \omega'}(\chi) \rrbracket_{v_x} = \top$$

et :

$$\llbracket \mathcal{X}_g < K_{g,\omega'} \rrbracket_{v_x, \mathcal{K}} = \top.$$

Or, par application de *la contrainte des régulateurs effectifs* (définition 3.1.1), nous obtenons que si $\llbracket P_{g,\omega'}(\mathcal{X}) \rrbracket_{v_x} = \top$ alors l'ensemble des régulateurs ω' doit correspondre à l'ensemble des régulateurs effectifs du gène g dans l'état x (i.e. $\omega' = \omega_g^*(x)$).

Donc, par substitution, nous en déduisons :

$$\llbracket \mathcal{X}_g < K_{g,\omega_g^*(x)} \rrbracket_{v_x, \mathcal{K}} = \top$$

soit :

$$x_g < \mathcal{K}(K_{g,\omega_g^*(x)})$$

d'où : $x \rightarrow_{\mathcal{K}} x[g \uparrow]$ d'après la définition 2.3.8.

Pour les deux autres cas ($x' = x[g \downarrow]$ et $x' = x$), la preuve est similaire à celle qui vient d'être exécutée.

On en déduit $\rightsquigarrow_{\mathcal{K}} \subseteq \rightarrow_{\mathcal{K}}$.

On a ainsi démontré que $proj(D_{\mathcal{K}}) = \mathcal{D}_{\mathcal{K}}$.

La propriété 3.1.1 spécifie que pour une instance de paramètres $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{X}$, la dynamique annotée $D_{\mathcal{K}}$ correspond à la dynamique $\mathcal{D}_{\mathcal{K}}$ de Γ induite par \mathcal{K} (définition 2.3.8).

Ainsi, à partir des transitions du type $((q_p, x), (q'_p, a_p(x)))$ dans δ_D , si on supprime la première composante des états (qui correspond à l'état du PGRN, ici q_p et q'_p), on obtient alors des transitions du type $x \rightarrow a_p(x)$, qui correspondent aux transitions d'une dynamique induite par la même instance de paramètres.

La première composante (l'état T ou S du PGRN associé) est en fait une des techniques nous permettant d'optimiser la traversée des états du PGRN : en annotant par S les états du produit stables, nous pouvons plus facilement identifier les états stables, et ainsi leur appliquer un traitement spécifique. Nous reviendrons dans le chapitre suivant sur cette spécificité (chapitre 4, section 4.3 p.75).

Exemple 19. *La dynamique représentée en figure 3.4 est la dynamique annotée associée au PGRN de la figure 3.2 et à l'instance de paramètres donnée dans l'exemple 12. Elle correspond également à la dynamique induite par cette même instance de paramètres et représentée en figure 2.6.*

3.2 Synchronisation d'un PGRN avec un automate de Büchi

Comme indiqué en introduction (section 2.4, p.41), nous cherchons à déterminer les instances de paramètres $\mathcal{K} : \mathbb{K} \rightarrow \mathbb{N}$ d'un graphe d'interactions Γ , telles que les dynamiques correspondantes $\mathcal{D}_{\mathcal{K}}$ ont un comportement particulier, i.e. vérifient une certaine propriété exprimée sous la

forme d'une formule LTL φ . Sur le modèle du model-checking LTL classique, nous construisons un automate de Büchi $B_{\neg\varphi}$ à partir de la négation de la formule LTL (voir chapitre Préliminaire section 1.5, p.21). Nous pouvons alors construire le produit entre le PGRN associé à Γ et cet automate $B_{\neg\varphi}$ (définition 3.2.1 donnée ci-dessous).

Définition 3.2.1 (Produit du PGRN et de l'Automate de Büchi). Soit $\Gamma = (G, I)$ un graphe d'interactions et $P = (Q_P, Q_{P_0}, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_P)$ le PGRN associé à Γ . Soit $B_{\neg\varphi} = (Q_B, q_B^0, A_B, \delta_B)$ un automate de Büchi associé à une formule LTL $\neg\varphi$ définie sur \mathbb{G} .

Le produit synchronisé $\Pi_{\neg\varphi}$ entre P et $B_{\neg\varphi}$, appelé *Produit* dans la suite, est défini par $\Pi_{\neg\varphi} = P \otimes B_{\neg\varphi} = (Q_\Pi, q_\Pi^0, A_\Pi, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_\Pi)$ avec :

- $Q_\Pi = Q_P \times Q_B$ l'ensemble des états
- $q_\Pi^0 = (T, q_B^0)$ l'état initial
- $A_\Pi = Q_P \times A_B$ l'ensemble des états acceptants
- $\delta_\Pi \subseteq Q_\Pi \times \text{Contraintes}(\mathbb{G} \cup \mathbb{K}) \times \text{Assignements}(\mathbb{X}) \times Q_\Pi$ l'ensemble des transitions
 $((q_P, q_B), g_P \wedge g_B, a_P, (q'_P, q'_B))$ telles que : $(q_P, g_P, a_P, q'_P) \in \delta_P$ et $(q_B, g_B, q'_B) \in \delta_B$.

Une transition du Produit $((q_P, q_B), g_P \wedge g_B, a_P, (q'_P, q'_B))$ dans δ_Π pourra alternativement être notée de la façon suivante : $(q_P, q_B) \xrightarrow{g_P \wedge g_B[a_P]} (q'_P, q'_B)$.

Notons qu'une garde $g_P \wedge g_B$ est une contrainte définie à la fois sur les variables d'état de \mathbb{G} et sur les paramètres de \mathbb{K} , i.e. $g_P \wedge g_B$ est définie sur $\text{Contraintes}(\mathbb{G} \cup \mathbb{K})$. Lors de l'exécution du Produit, la satisfaisabilité de $g_P \wedge g_B$ sera évaluée par un solveur de contrainte (détaillé dans la suite).

Exemple 20. Reprenons le cadre de notre exemple jouet.

Supposons qu'il soit connu que l'état $(2, 1)$ est un état stable, i.e. une fois l'état $(2, 1)$ atteint, le système reste toujours dans cet état. Cette propriété peut s'exprimer à l'aide de la formule LTL suivante :

$$\varphi \stackrel{\text{def}}{=} \mathbf{G}((x_\alpha = 2 \wedge x_\beta = 1) \Rightarrow \mathbf{X}(x_\alpha = 2 \wedge x_\beta = 1))$$

En langage courant, cette formule peut être traduite par « si le système atteint l'état $(2, 1)$, alors l'état suivant sera nécessairement $(2, 1)$ ».

La négation de cette formule φ s'exprime ainsi :

$$\neg\varphi \stackrel{\text{def}}{=} \mathbf{F}((x_\alpha = 2 \wedge x_\beta = 1) \Rightarrow \mathbf{X}(x_\alpha < 2 \vee x_\beta = 0))$$

Un automate de Büchi correspondant à cette formule $\neg\varphi$ est donné en figure 3.5. Notons qu'il ne s'agit que d'un automate possible parmi d'autres car plusieurs automates de Büchi peuvent correspondre à la même formule LTL. L'automate présenté ici a été généré par l'outil LTL2BA4J [Bod11].

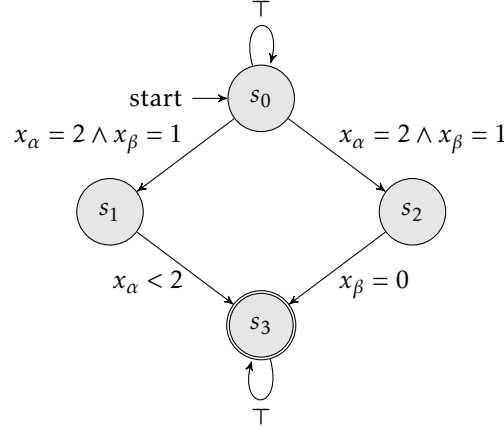


FIGURE 3.5 – $B(\neg\varphi)$ avec $\neg\varphi \stackrel{\text{def}}{=} \mathbf{F}((x_\alpha = 2 \wedge x_\beta = 1) \Rightarrow \mathbf{X}(x_\alpha < 2 \vee x_\beta = 0))$.

Le produit entre le PGRN de notre exemple jouet (figure 3.2) et cet automate de Büchi (figure 3.5) est représenté figure 3.6. Les transitions dont la garde $g_P \wedge g_B$ n'est pas satisfaisable, qui correspondent au cas où il n'existe pas d'interprétation $v_x : G \rightarrow \mathbb{N}$ telle que $\llbracket g_P \wedge g_B \rrbracket_{v_x}$ est satisfaisable, ne sont pas représentées (il n'est pas nécessaire d'interpréter les paramètres car les gardes des transitions de $B_{\neg\varphi}$ ne sont exprimées que sur \mathbb{G}).

Afin d'alléger notre propos dans la suite, nous utiliserons la notation $\Pi_{\neg\varphi}$ pour le produit d'un PGRN et d'un automate de Büchi sans mentionner explicitement les éléments $\Pi_{\neg\varphi}$ qui s'y réfèrent, dont la liste est la suivante :

- le graphe d'interactions Γ ,
- l'espace des états \mathbb{X} ,
- l'ensemble des variables d'état \mathbb{G} ,
- l'ensemble des paramètres \mathbb{K} ,
- le PGRN $P = (Q_P, Q_{P_0}, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_P)$
- et l'automate de Büchi $B_{\neg\varphi} = (Q_B, q_B^0, A_B, \delta_B)$.

3.3 Exécution symbolique du Produit

3.3.1 Vue d'ensemble

Le produit Π entre un PGRN P et un automate de Büchi $B_{\neg\varphi}$ correspond à la synchronisation de P avec $B_{\neg\varphi}$. Autrement dit, pour toute transition $t_\Pi = ((q_P, q_B), g_P \wedge g_B, a_P, (q'_P, q'_B))$ dans δ_Π , il existe $t_P = (q_P, g_P, a_P, q'_P)$ dans δ_P et $t_{B_{\neg\varphi}} = (q_B, g_B, a_B, q'_B)$ dans δ_B . La garde de cette transition t_Π est obtenue en faisant la conjonction des gardes de t_P et de $t_{B_{\neg\varphi}}$, soit $g_P \wedge g_B$.

Le Produit peut donc être vu à la fois comme un PGRN et comme un automate de Büchi. Notamment, pour une instance de paramètres donnée, le PGRN définit une dynamique annotée.

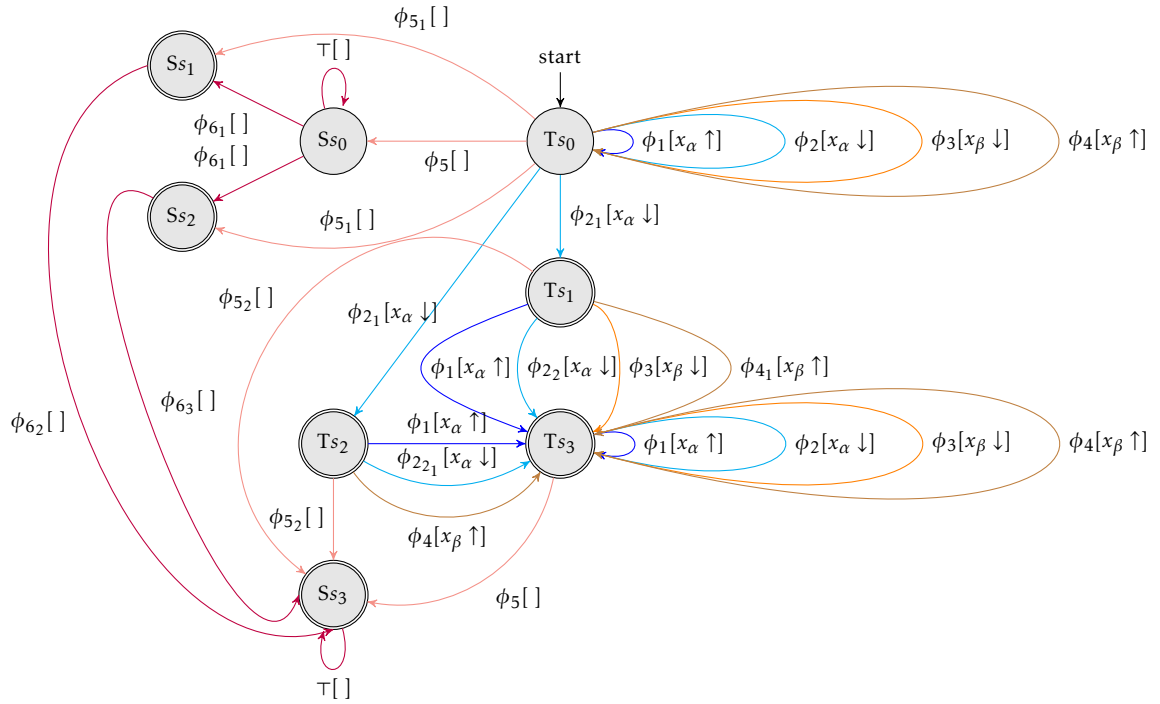


FIGURE 3.6 – Produit associée au PGRN en figure 3.2 et à l’automate de Buchi en figure 3.5 avec :

$$\phi_1 \stackrel{\text{def}}{=} x_\alpha < 2 \wedge x_\beta = 0 \wedge x_\alpha < K_\alpha(\{\})$$

$$\phi_2 \stackrel{\text{def}}{=} (x_\alpha = 1 \wedge x_\beta = 0 \wedge K_\alpha(\{\}) = 0) \vee (x_\alpha = 1 \wedge x_\beta = 1) \vee (x_\alpha = 2 \wedge x_\beta = 1 \wedge K_\alpha(\{\alpha, \beta\}) < 2)$$

$$\phi_{2_1} \stackrel{\text{def}}{=} x_\alpha = 2 \wedge x_\beta = 1 \wedge K_\alpha(\{\alpha, \beta\}) < 2$$

$$\phi_{2_2} \stackrel{\text{def}}{=} (x_\alpha = 1 \wedge x_\beta = 0 \wedge K_\alpha(\{\}) = 0) \vee (x_\alpha = 1 \wedge x_\beta = 1)$$

$$\phi_{2_{2_1}} \stackrel{\text{def}}{=} x_\alpha = 1 \wedge x_\beta = 0 \wedge K_\alpha(\{\}) = 0$$

$$\phi_3 \stackrel{\text{def}}{=} x_\alpha = 0 \wedge x_\beta = 1$$

$$\phi_4 \stackrel{\text{def}}{=} x_\alpha > 0 \wedge x_\beta = 0$$

$$\phi_{4_1} \stackrel{\text{def}}{=} x_\alpha = 1 \wedge x_\beta = 0$$

$$\phi_5 \stackrel{\text{def}}{=} (x_\alpha = 0 \wedge x_\beta = 0 \wedge K_\alpha(\{\}) = 0) \vee (x_\alpha = 2 \wedge x_\beta = 1 \wedge K_\alpha(\{\alpha, \beta\}) = 2)$$

$$\phi_{5_1} \stackrel{\text{def}}{=} x_\alpha = 2 \wedge x_\beta = 1 \wedge K_\alpha(\{\alpha, \beta\}) = 2$$

$$\phi_{5_2} \stackrel{\text{def}}{=} x_\alpha = 0 \wedge x_\beta = 0 \wedge K_\alpha(\{\}) = 0$$

$$\phi_{6_1} \stackrel{\text{def}}{=} x_\alpha = 2 \wedge x_\beta = 1$$

$$\phi_{6_2} \stackrel{\text{def}}{=} x_\alpha < 2$$

$$\phi_{6_3} \stackrel{\text{def}}{=} x_\beta = 0$$

Or cette dynamique est telle qu'elle définit un chemin dans l'automate de Büchi (il suffit pour cela de considérer les transitions franchies dans le PGRN, et de ne conserver que les parties des gardes issues de l'automate de Büchi), donc si ce chemin passe infiniment souvent dans un état acceptant de l'automate de Büchi, alors il vérifie la propriété $\neg\varphi$.

Bien entendu, il ne s'agit pas de générer toutes les dynamiques annotées (car on retrouverait alors le problème de l'explosion combinatoire du nombre de dynamiques) mais de définir l'ensemble des chemins qui existent dans le produit, sans instancier les paramètres.

Pour cela, nous nous appuyons sur des techniques d'exécution symbolique afin de ne pas évaluer les paramètres $K_{i,\omega}$. Ces techniques ont été initialement développées pour le test de programmes [Kin75]. Il s'agit de manipuler symboliquement les variables au lieu de leur associer des valeurs concrètes. Chaque exécution symbolique est munie d'une *condition de chemin* exprimant les conditions nécessaires que doivent respecter les variables symboliques pour exécuter le chemin en question.

Les techniques d'exécution symbolique ont été étendues aux systèmes de transition symbolique [Gas+06] en dépliant ces systèmes pour obtenir des arbres symboliques. Les chemins du système de transition (du Produit) étant infinis, les arbres symboliques sont eux-mêmes potentiellement infinis. Des critères de sélection ou repliements sont donc utilisés pour arrêter la construction des chemins, et rendre les arbres finis.

Dans la suite, nous nous attacherons donc :

1. à définir les arbres d'exécution symbolique composés de nœuds et de transitions ;
2. à identifier les situations de retour sur un nœud déjà rencontré qui autorisent à stopper l'exécution symbolique des chemins, et à identifier les cycles acceptants à partir de ces situations de retour ;
3. à déterminer les contraintes que les paramètres doivent vérifier pour qu'il existe un chemin menant à ces cycles acceptant.

3.3.2 Construction des Arbres d'Exécution

Le Produit inclut deux familles de variables : celles représentant les niveaux d'expression des gènes du GRN (les variables χ_i) et celles représentant les paramètres biologiques (les variables $K_{i,\omega}$). Ces deux ensembles de variables sont de nature différente.

Ainsi, les niveaux d'expression χ_i sont *évalués* et *dynamiques*, leurs valeurs sont toutes toujours connues et elles sont susceptibles d'être modifiées après franchissement d'une transition.

Les paramètres $K_i(\omega)$ quant à eux sont *symboliques* et *statiques*, leurs valeurs sont inconnues mais elles peuvent (ou non) être contraintes au fur et à mesure du parcours du Produit, par les gardes des transitions successivement franchies. Nous avons donc choisi de réaliser une exécution semi-symbolique : les variables d'états sont évaluées, alors que les paramètres de Thomas restent symboliques.

Une exécution du produit $\Pi_{-\varphi}$ à partir d'un état initial x dans \mathbb{X} est obtenue en franchissant successivement différentes transitions du produit. À chaque transition franchie, l'état courant x est mis à jour en fonction de l'affectation de la transition et une transition n'est franchie que si l'état courant x permet que la garde de la transition soit satisfaisable. De plus, le franchissement d'une transition n'est possible que si les paramètres vérifient les contraintes de la garde. Au fur et à mesure que les transitions sont franchies, les contraintes des gardes sur les paramètres s'additionnent pour définir une contrainte plus forte appelée *condition de chemin*.

Un arbre d'exécution à partir d'un état initial x représente l'ensemble des exécutions du produit à partir de x . Il est composé de nœuds dits *symboliques* qui mémorisent l'état du Produit, de l'état x courant et de la condition de chemin ayant permis d'atteindre ce nœud. Les arcs de l'arbre indiquent les transitions qui sont franchissables à partir du nœud.

Noeuds de l'arbre d'exécution symbolique

Définition 3.3.1 (Nœuds symboliques). Soit $\Pi_{-\varphi} = (Q_{\Pi}, q_{\Pi}^0, A_{\Pi}, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_{\Pi})$ un Produit.

L'ensemble des nœuds symboliques du Produit est noté $SS(\Pi_{-\varphi})$ (« SS » pour *Symbolic State*) et défini par :

$$SS(\Pi_{-\varphi}) \subseteq Q_{\Pi} \times \mathbb{X} \times \text{Contrainte}(\mathbb{K}) \times (Q_{\Pi} \times \mathbb{X})^* \times \mathbb{N}$$

Le sous-ensemble $SS^0(\Pi_{-\varphi})$ représente l'ensemble des nœuds initiaux de $\Pi_{-\varphi}$:

$$SS^0(\Pi_{-\varphi}) = \{(q_{\Pi}^0, x, \tau, \epsilon, 0) \in SS(\Pi_{-\varphi})\}.$$

Un élément de $SS(\Pi_{-\varphi})$ est noté $ss = ((q_P, q_{B_{-\varphi}}), x, pc, \sigma, occ)$. Il correspond à un nœud créé lors du dépliement du Produit $\Pi_{-\varphi}$; ses composantes contiennent les informations relatives à la trajectoire empruntée dans le Produit $\Pi_{-\varphi}$ pour l'atteindre. Elles correspondent à :

- $q_{\Pi} = (q_P, q_{B_{-\varphi}})$ désigne l'état de $\Pi_{-\varphi}$;
- $x \in \mathbb{X}$ correspond à l'état courant à la création du nœud ;
- pc (pour *path condition* soit *condition de chemin*) est la contrainte définie sur $\text{Contrainte}(\mathbb{K})$ qui doit être satisfaite pour atteindre l'état ss à partir de l'état initial de la trajectoire ;
- $\sigma \in (Q_{\Pi} \times \mathbb{X})^*$ est le chemin exprimé sur $(Q_{\Pi} \times \mathbb{X})$ qui conduit à ss à partir du nœud initial de la trajectoire ($\sigma = \epsilon$ si ss est le nœud initial de l'exécution) ;
- $occ \in \mathbb{Z}$ (pour *occurrence*) est l'indice i du premier état σ_i identique atteint dans le chemin σ ; il est égal à -1 si un tel nœud n'existe pas. La composante occ sera utilisée pour la détection de cycle dans les trajectoires symboliques.

Étant donné un nœud symbolique ss de la forme $ss(q_\Pi)$, $ss(q_P)$, $ss(q_{B_{-\varphi}})$, $ss(x)$, $ss(pc)$, $ss(\sigma)$ et $ss(occ)$ désigneront respectivement les composantes $(q_P, q_{B_{-\varphi}})$, q_P , $q_{B_{-\varphi}}$, x , pc , σ et occ

Nous noterons $ss^0(x)$ le nœud initial dans $SS^0(\Pi_{-\varphi})$ correspondant à l'état x dans \mathbb{X} . Notons que pour un Produit $\Pi_{-\varphi}$ donné, il existe un nœud initial différent pour chaque état x dans \mathbb{X} , i.e. $|SS^0(\Pi_{-\varphi})| = |\mathbb{X}|$; comme nous le verrons ensuite, chaque nœud initial $ss^0(x)$ correspond à un *arbre d'exécution* spécifique.

Arcs de l'arbre d'exécution symbolique

Pour obtenir une exécution de $\Pi_{-\varphi}$, nous devons franchir les transitions de $\Pi_{-\varphi}$ nous permettant de construire un ensemble de nœuds successifs. Pour cela, pour un nœud ss donné, nous devons déterminer quelles transitions sont empruntables (franchissables); la définition 3.3.2 caractérise les conditions de franchissement.

Définition 3.3.2 (Transition franchissable). Soit $\Pi_{-\varphi} = (Q_\Pi, q_\Pi^0, A_\Pi, Contraintes(\mathbb{G} \cup \mathbb{K}), \delta_\Pi)$ un Produit et $ss = ((q_P, q_B), x, pc, \sigma, occ) \in SS(\Pi_{-\varphi})$ un nœud de l'exécution symbolique du Produit. Une transition $t_\Pi = ((q_P, q_B), g_P \wedge g_B, a_P, (q'_P, q'_B)) \in \delta_\Pi$ est franchissable depuis ss si et seulement si la contrainte $(\llbracket g_P \wedge g_B \rrbracket_x \wedge pc)$ est satisfaisable.

Toute transition franchissable depuis un nœud $ss \in SS(\Pi_{-\varphi})$ est empruntée, ce qui conduit à la création potentielle de plusieurs branches d'exécution. Le franchissement d'une transition $t_\Pi \in \delta_\Pi$ conduit à la création d'un nouveau nœud dans $SS(\Pi_{-\varphi})$, dont les caractéristiques dépendent de ss et de t_Π (définition 3.3.3).

Définition 3.3.3 (Franchissement d'une transition). Soit $\Pi_{-\varphi} = (Q_\Pi, q_\Pi^0, A_\Pi, Contraintes(\mathbb{G} \cup \mathbb{K}), \delta_\Pi)$ un Produit, soit $ss = ((q_P, q_B), x, pc, \sigma, occ) \in SS(\Pi_{-\varphi})$ un nœud de l'exécution symbolique du Produit et soit $t_\Pi = ((q_P, q_B), g_P \wedge g_B, a_P, (q'_P, q'_B)) \in \delta_\Pi$ une transition franchissable depuis ss .

Le franchissement de t_Π depuis ss conduit au nœud $ss' = (q'_\Pi, x', pc', path', occ')$, défini par :

- $q'_\Pi = (q'_P, q'_B)$;
- $x' = a_P(x)$;
- $pc' = \llbracket g_P \wedge g_B \rrbracket_x \wedge pc$;
- $\sigma' = \sigma.(q'_\Pi, x')$;
- $occ' = \begin{cases} occ & \text{si } ((q'_P, q'_{B_{-\varphi}}), x') \notin \sigma \\ \min(\{i \mid \sigma[i] = ((q'_P, q'_{B_{-\varphi}}), x')\}) & \text{sinon.} \end{cases}$

Dans la suite, nous adoptons la notation $ss \xrightarrow{t_\Pi} ss'$ pour représenter le franchissement de t_Π depuis ss .

Les éléments du nouveau nœud ss' , en accord avec la définition 3.3.3, sont les suivants :

- l'état du Produit composant ss' est l'état cible (q'_p, q'_b) de la transition franchie t_Π ;
- l'état x' atteint dans l'espace des états \mathbb{X} est obtenu à partir de x (la composante de ss donnant l'état) par application de l'affectation $a_p(x)$ de t_Π ;
- la nouvelle condition de chemin pc' est obtenue en effectuant la conjonction de la condition de chemin pc du nœud ss et de la garde de la transition franchie $\llbracket g_p \wedge g_b \rrbracket_x$;
- le chemin σ' qui mène à ss' est obtenu en ajoutant le nouveau couple atteint (état du PGRN, état dans \mathbb{X}) au chemin qui mène à ss , i.e. $\sigma' = \sigma.(q'_\Pi, x')$;
- la valeur de l'indice occ' est mise à jour par rapport à occ si le couple (état du PGRN, état dans \mathbb{X}) de ss' a été précédemment rencontré dans σ .

Notons en particulier que la condition de chemin obtenue pc' est bien définie sur $Contrainte(\mathbb{K})$ car pc et $\llbracket g_p \wedge g_b \rrbracket_x$ sont elles-mêmes définies sur $Contrainte(\mathbb{K})$.

De plus, remarquons que, par construction, si la composante occ d'un nœud ss est différente de 0 (valeur correspondante à celle des nœuds initiaux), alors la composante occ de tous les nœuds créés à la suite de ss est elle-aussi différente de 0. La valeur 0 permet ainsi de partitionner les nœuds d'une exécution selon qu'un cycle a été ou non découvert dans le chemin σ de ss .

L'exécution est dite *partiellement symbolique* car, à chaque étape du dépliement, la garde $g_p \wedge g_b$ de chaque transition franchie reste partiellement instanciée, i.e. les variables d'états $Var(g_p \wedge g_b) \cap \mathbb{G}$ sont instanciées en fonction de l'état courant $x \in \mathbb{X}$, tandis que les variables de paramètres $Var(g_p \wedge g_b) \cap \mathbb{K}$ ne sont pas instanciées.

Nœuds atteignables

La notion de franchissement d'une transition à partir d'un nœud conduit naturellement à la notion d'atteignabilité d'un nœud. Intuitivement, un nœud symbolique ss' est atteignable à partir de ss s'il existe une suite de transitions $t_{\Pi,1} \dots t_{\Pi,j}$ dont l'exécution mène de ss à ss' .

Définition 3.3.4 (Nœud atteignable). Soit $ss, ss' \in SS(\Pi_{-\varphi})$ deux nœuds du Produit $\Pi_{-\varphi}$.

ss' est dit *atteignable* à partir de ss s'il existe une séquence de transitions $t_{\Pi,i}$ dans δ_Π et une séquence de nœuds ss_i dans $SS(\Pi_{-\varphi})$ avec $1 \leq i \leq j$, et telles que $ss = ss_1$ et

$$ss_1 \xrightarrow{t_{\Pi,1}} ss_2 \xrightarrow{t_{\Pi,2}} \dots \xrightarrow{t_{\Pi,j-1}} ss_m \xrightarrow{t_{\Pi,j}} ss'.$$

Nous notons $ss \xrightarrow{*}_{\Pi} ss'$ la séquence de transitions dans δ_Π permettant d'atteindre ss' à partir de ss .

De plus, nous notons $Reach$ la fonction de $SS(\Pi_{-\varphi})$ vers $2^{SS(\Pi_{-\varphi})}$ associant chaque nœud ss de $SS(\Pi_{-\varphi})$ au sous-ensemble de $SS(\Pi_{-\varphi})$ correspondant aux nœuds atteignables à partir de ss , i.e. $\forall ss \in SS(\Pi_{-\varphi}), Reach(ss) = \{ss' \in SS(\Pi_{-\varphi}) \mid ss \xrightarrow{*}_{\Pi} ss'\}$.

Si ss' est atteignable à partir de ss , ss' est appelé *descendant* de ss , et ss est un *ancêtre* de ss' .

Arbre d'exécution symbolique

En se basant sur la notion d'atteignabilité des nœuds, nous introduisons maintenant la notion *d'arbre d'exécution symbolique*, un système de transition particulier dérivé à partir d'un état dit *initial* dans \mathbb{X} et noté x_0 .

L'*arbre d'exécution symbolique* de $\Pi_{-\varphi}$ depuis x_0 est $SET(\Pi_{-\varphi}, x_0)$ et est obtenu en déterminant tous les nœuds pouvant être atteints (par le biais de la relation de transition) si nous prenons le nœud $ss^0(x_0) = (q_{\Pi}^0, x_0, \top, \varepsilon, 0)$ comme nœud initial de l'exécution.

Définition 3.3.5 (Arbre d'exécution symbolique depuis un état x_0). Soit $\Pi_{-\varphi} = (Q_{\Pi}, q_{\Pi}^0, A_{\Pi}, \delta_{\Pi})$ un Produit.

L'*Arbre d'exécution symbolique* de $\Pi_{-\varphi}$ depuis x_0 est noté $SET(\Pi_{-\varphi}, x_0)$ et correspond au système de transitions $(Reach((q_{\Pi}^0, x_0, \top, \varepsilon, 0)), \rightarrow)$ où $\rightarrow \subset Reach((q_{\Pi}^0, x_0, \top, \varepsilon, 0)) \times Reach((q_{\Pi}^0, x_0, \top, \varepsilon, 0))$ est défini par $\rightarrow_{\Pi_{-\varphi}, x_0} = \{(ss, ss') \mid ss, ss' \in SS(\Pi_{-\varphi}, x_0) \wedge \exists t_{\Pi} \in \delta_{\Pi}, ss \xrightarrow{t_{\Pi}} ss'\}$

Notons que pour un Produit $\Pi_{-\varphi}$ donné, il est possible de définir autant de nœuds initiaux que d'états x dans \mathbb{X} , (i.e. $|SS^0(\Pi_{-\varphi})| = |\mathbb{X}|$), et donc autant d'arbres d'exécution symbolique spécifiques.

3.3.3 Recherche des cycles acceptants

Les arbres d'exécution symbolique définis précédemment sont des structures de données infinies. En effet, les propriétés biologiques sont exprimées sur des séquences infinies, donc les exécutions du Produit et donc les branches des SET sont elles aussi infinies.

Cependant, si on ne considère pas les conditions de chemin, le nombre de nœuds possibles dans un SET est fini. En effet, ce nombre est majoré par le produit du nombre d'états ($|\mathbb{X}|$) et du nombre de sommets du Produit, qui est lui-même majoré par le produit du nombre de sommets de l'automate de Büchi ($|Q_B|$) et du nombre de sommets ($|Q_P| = 2$) du PGRN, soit par $2|\mathbb{X}||Q_B|$. Pour notre exemple jouet, cela revient à un maximum de 48 « patrons » (sans pc) de nœuds.

Ainsi, dans un arbre d'exécution symbolique, il existe nécessairement dans chaque branche un nœud $ss' = (q_{\Pi}, x, pc', \sigma', occ')$ ayant pour ancêtre un nœud $ss = (q_{\Pi}, x, pc, \sigma, occ)$ (même sommet dans Q_{Π} et même état dans \mathbb{X}).

Pour toute branche infinie de l'arbre symbolique, par construction, il existe une infinité de tels couples (ss, ss') partageant le même sommet dans Q_Π et le même état dans \mathbb{X} . En considérant parmi tous ces couples (ss, ss') , le couple pour lequel $|\sigma(ss')|$ est minimal, on appelle ce nœud ss' de *nœud de retour* et le nœud ss correspond de *nœud d'origine*.

Les transitions franchissables à partir du nœud de retour ss' sont incluses dans les transitions franchissables depuis le nœud d'origine ss .

En effet, ss' est atteignable à partir de ss et donc la condition de chemin pc' correspond à la condition de chemin pc à laquelle ont été ajoutées les gardes franchies pour aller de ss à ss' , i.e. pc' est de la forme $pc' = pc \wedge \llbracket g_{\Pi_i} \rrbracket_x \wedge \llbracket g_{\Pi_j} \rrbracket_x \wedge \dots$, et donc pc' implique pc .

Par définition, les transitions franchissables à partir de ss' sont les transitions $t_\Pi = ((q_P, q_B), g_P \wedge g_B, a_P, (q'_P, q'_B)) \in \delta_\Pi$ telles que $(\llbracket g_P \wedge g_B \rrbracket_x \wedge pc')$ est satisfaisable, ce qui définit bien un sous-ensemble des t_Π telles que $(\llbracket g_P \wedge g_B \rrbracket_x \wedge pc)$ est satisfaisable, qui sont les transitions franchissables à partir de ss .

Autrement dit, l'ensemble des instanciations de paramètres vérifiant pc' est inclus dans l'ensemble des instanciations de paramètres vérifiant pc .

Ainsi, en arrêtant la construction des branches des arbres d'exécution symbolique lorsque le nœud de retour est atteint, on ne perd pas d'information (puisque'il y a inclusion de l'arbre issu du nœud de retour de celui de l'arbre issu du nœud origine).

Les différentes conditions de chemins de nœuds de retour caractérisent les cycles que l'on peut rencontrer dans les différentes exécutions infinies. En effet, le chemin depuis le nœud origine jusqu'au nœud de retour peut s'exécuter infiniment souvent à condition que la condition de chemin du nœud de retour soit vérifiée. S'il existe un nœud acceptant entre le nœud origine et le nœud de retour, ce nœud de retour est dit acceptant.

Notation 3.3.1. Soit $\Pi_{-\varphi} = (Q_\Pi, q_\Pi^0, A_\Pi, \delta_\Pi)$ un Produit, et x_0 un état.

On note $AR_{\Pi_{-\varphi}}(x_0)$ l'ensemble de *nœuds de retour acceptants* de l'arbre d'exécution symbolique de $\Pi_{-\varphi}$ depuis x_0 .

3.4 Inférence des paramètres

L'exécution symbolique du Produit $\Pi_{-\varphi} = P \otimes B_{-\varphi}$ nous conduit à construire un ensemble $\{\mathcal{S}(x) \mid x = (x_i)_{i \in V}\}$ d'arbres d'exécution symbolique où $\mathcal{S}(x)$ est un arbre de l'ensemble des chemins possibles à partir de l'état x .

Ainsi, si un jeu de paramètres $K_i(\omega)$ vérifie (au moins) une des conditions de chemin acceptantes de $\mathcal{S}(x)$, cela signifie qu'il existe un chemin dans Π commençant en x et passant infiniment souvent par le cycle associé à cette condition de chemin acceptante. Autrement dit, le jeu de paramètres $K_i(\omega)$ définit une dynamique où au moins un des chemins commençant en x vérifie

la propriété $\neg\varphi$.

Réciproquement, si le jeu de paramètres $K_i(\omega)$ ne vérifie aucune des conditions de chemins acceptantes, alors il n'existe pas de chemin partant de x vérifiant $\neg\varphi$, donc tous les chemins partant de x vérifient φ .

Si on cible toutes les dynamiques qui vérifient φ (i.e. les dynamiques telles que tous les chemins, quel que soit l'état de départ, vérifient φ), il suffit de rechercher les jeux de paramètres vérifiant la *conjonction des négations des conditions de chemin acceptantes de tous les SET associés à $\Pi_{\neg\varphi}$* .

On peut également s'intéresser aux dynamiques telles qu'au moins un chemin vérifie φ , celles-ci correspondent aux jeux de paramètres vérifiant la *disjonction des conditions de chemin acceptantes de tous les SET associés au Produit $\Pi_\varphi = P \otimes B_\varphi$* .

Propriété 3.4.1. Soit un PGRN $P = (Q_P, S_{P_0}, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_P)$, et soit $B_{\neg\varphi} = (Q_B, q_B^0, A_B, \delta_B)$ un automate de Büchi.

Soit $\Pi_{\neg\varphi} = (Q_\Pi, q_\Pi^0, A_\Pi, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_\Pi)$ le produit associé et soit $(\text{SET}(\Pi_{\neg\varphi}, x))_{x \in \mathbb{X}}$ l'ensemble des arbres d'exécution symbolique construits à partir de $\Pi_{\neg\varphi}$.

Notons $AR_{\Pi_{\neg\varphi}} = \cup_{x \in \mathbb{X}} AR_{\Pi_{\neg\varphi}}(x)$ l'ensemble de tous les nœuds de retours (de tous les arbres construits sur $\Pi_{\neg\varphi}$).

Soit \mathcal{K} une instance de paramètres.

Tous les chemins de la dynamique $\mathcal{D}_\mathcal{K}$ satisfont φ si et seulement si :

$$\mathcal{K} \models \bigwedge_{ss \in AR_{\Pi_{\neg\varphi}}} \neg pc(ss)$$

Il existe un chemin de la dynamique $\mathcal{D}_\mathcal{K}$ qui satisfait $\neg\varphi$ si et seulement si :

$$\mathcal{K} \models \bigvee_{ss \in AR_{\Pi_{\neg\varphi}}} pc(ss)$$

Pour distinguer ces deux types de propriétés, nous préfixerons dans la suite les formules LTL par A et E respectivement.

Notation 3.4.1. Soit φ une formule LTL.

Soit \mathcal{K} une instance de paramètres.

$\mathcal{D}_\mathcal{K} \models A\varphi$ si et seulement si pour tout chemin σ de $\mathcal{D}_\mathcal{K}$, $\sigma \models \varphi$

$\mathcal{D}_\mathcal{K} \models E\varphi$ si et seulement si il existe un chemin σ de $\mathcal{D}_\mathcal{K}$, $\sigma \models \varphi$.

Par cohérence avec l'usage de LTL, une formule φ sans préfixe A ou E désignera une formule de type A (autrement dit, désignera la formule $A\varphi$).

Remarquons que le traitement des formules de type A ($A\varphi$) est analogue à l'approche LTL classique, au sens où il demande de considérer dans un premier temps un automate de Büchi

associé à la négation de la formule ($B_{\neg\varphi}$), alors que le traitement des formules de type E ($E\varphi$) demande de considérer l'automate de Büchi directement associé à la formule (B_φ).

Dans ce chapitre, nous avons réussi à exprimer la recherche des paramètres des réseaux de régulation génétique sous la forme d'un problème de résolution de contraintes. Grâce à la nature finie des systèmes de transition manipulés, nous avons établi que les mécanismes d'exécution symbolique pouvaient être coupés sans perdre d'information. Dans le chapitre suivant, nous allons détailler d'autres éléments algorithmiques visant à couper la construction au plus tôt des arbres d'exécution symbolique.

Chapitre 4

Implémentation

Ce chapitre est consacré à l’implémentation de notre méthode présentée dans le chapitre 3 sous la forme d’un logiciel Java appelé `SPuTNIk`. Après une brève présentation du logiciel, nous nous intéresserons aux algorithmes mis en place afin d’optimiser les traitements, notamment la simplification du produit entre le PGRN et l’automate de Büchi ainsi que le parcours des arbres d’exécution symbolique, mettant en particulier en œuvre des mécanismes de *coupure* des branches.

4.1	SPuTNIk	71
4.1.1	Entrées et sorties de <code>SPuTNIk</code>	72
4.1.2	Informations techniques	73
4.2	Algorithme de simplification du produit	73
4.3	Algorithme d’exécution des arbres	75
4.4	Pistes d’optimisation	79
4.4.1	Parallélisation du parcours	79
4.4.2	Autres pistes	83

4.1 SPuTNIk

Conformément à la méthode décrite au chapitre précédent, nous avons développé un prototype appelé `SPuTNIk` — pour *Symbolic Parameters of Thomas’ Networks Inference* — afin de déterminer les jeux de paramètres biologiques d’un GRN compatibles avec des connaissances biologiques.

`SPuTNIk` a été conçu pour être facile à prendre en main, même sans connaissance en informatique, et assez flexible. Dans ce sens, une interface graphique a été réalisée afin de mettre à la disposition de l’utilisateur diverses facilités d’entrée des données. Le cas échéant, les données

sont fournies dans un fichier d'entrée créé facilement à la main en suivant la syntaxe imposée, ou éventuellement exporté à partir de l'interface graphique. SPuTNIk peut ainsi manipuler des fichiers importés mais aussi extraire un certain nombre d'informations collectées pendant l'exécution à destination de l'utilisateur, retranscrites pendant l'exécution ou copiées dans un fichier.

Nous ne détaillerons pas dans ce chapitre comment utiliser SPuTNIk, son manuel d'utilisation illustré est consultable en annexe A, à partir de la p.173.

4.1.1 Entrées et sorties de SPuTNIk

SPuTNIk attend plusieurs informations en entrée de la part de l'utilisateur : le graphe d'interactions, un ensemble (facultatif) de formules LTL, et un ensemble (facultatif) de contraintes sur les paramètres.

Le graphe d'interaction $\Gamma = (G, I)$ est décrit en donnant le nom des gènes composant le GRN (correspondant à l'ensemble des étiquettes associées aux gènes g dans G) et ses interactions (chaque interaction (g, s, t, g') dans I est définie par quatre données : g la source, g' la cible, s le signe et t le seuil de l'interaction).

Il est possible de préciser une ou plusieurs formules LTL $(\varphi_i)_{i \in \mathbb{N}}$ à tester sur le système. Ces formules peuvent être de type A ou E (cf. notation 3.4.1).

La précision des formules LTL n'est cependant que facultative. Si aucune formule LTL n'est donnée, SPuTNIk générera tous les instanciations de paramètres compatibles avec le GRN et les contraintes précisées en option (notamment la CPBF, cf. section 2.3.5, définition 2.3.9 p.40).

À l'inverse, si plusieurs formules de type A et/ou E sont données, SPuTNIk effectuera dans un premier temps le parcours des arbres symboliques obtenus à partir de la conjonction des formules de type A , puis effectuera dans un second temps les parcours des arbres symboliques obtenus à partir de chaque formule de type E . En d'autres termes, l'ensemble des formules de type A sont prises en compte lors d'un seul parcours, tandis que chaque formule de type E est liée à un parcours qui lui est propre. Une fois tous les parcours réalisés, l'ensemble des solutions correspond à l'intersection des solutions obtenues pour chaque parcours. Une option permet de préciser si les conditions acceptantes prélevées lors du parcours correspondant aux formule de type A doivent être utilisées en tant que conditions initiales lors du parcours des formules de type E .

Enfin, il est également possible de spécifier des contraintes sur les paramètres, sous la forme d'une liste de contraintes définies sur $\text{Contrainte}(\mathbb{K})$. Ces contraintes permettent de réduire l'espace de solutions. Elles peuvent par exemple permettre de tester rapidement si une instance donnée vérifie une propriété sur le système.

À noter que d'autres paramètres facultatifs existent, en particulier pour modifier l'affichage (affichage console en cours d'exécution, type de données exportées à la fin, ...) et le comportement de l'outil (nombre de nœuds parcourus, temps max, ...).

À partir de l'ensemble de ces informations, $SP_{\forall TNI_K}$ calcule la liste des instanciations de paramètres satisfaisant la(les) formule(s) LTL $(\varphi_i)_{i \in \mathbb{N}}$ données en entrée.

4.1.2 Informations techniques

$SP_{\forall TNI_K}$ est écrit en Java 7 et fait appel à plusieurs bibliothèques afin de gérer la conversion d'une formule LTL en automate de Büchi, ainsi que pour la résolution des contraintes.

La conversion d'une formule LTL en automate de Büchi est réalisée par la bibliothèque LTL2BA4J [Bod11], qui est en fait une passerelle vers la bibliothèque `ltl2ba` [GO01]. La bibliothèque `ltl2ba` s'appuie sur un algorithme efficace de traduction LTL vers Büchi, que ce soit en terme de taille de l'automate de Büchi, de temps de calcul ou d'occupation mémoire.

La partie résolution de contraintes est réalisée en faisant appel au solveur de contraintes Z3 [DB08] qui sert à vérifier la satisfaisabilité des conditions de chemin.

D'un point de vue technique, nous mettons à profit la possibilité du solveur Z3 de gérer les contraintes à vérifier sous la forme d'une *pile* ; dans ce mode, le solveur établit la satisfaisabilité de la conjonction de l'ensemble des contraintes présents dans sa pile. Concrètement, pendant la traversée en profondeur d'une branche, nous ajoutons dans la pile du solveur la garde de chaque transition en cours de test de franchissement : si la transition peut être franchie, la contrainte est conservée dans la pile, sinon elle est dépilée. Lorsque la fin d'une branche est atteinte, les contraintes ajoutées successivement dans la pile sont dépilées conjointement à la poursuite du parcours en profondeur d'une autre branche de l'arbre d'exécution symbolique. Ces opérations successives d'empilements et de dépilements permettent de faciliter la gestion des contraintes à satisfaire et améliorent les temps de résolution.

Précisons que la CPBF, si elle est requise, est initialement ajoutée à la base de la pile et n'est jamais supprimée.

Nous utilisons également Z3 en phase finale afin de déterminer les jeux de paramètres solutions.

Dans la première version de $SP_{\forall TNI_K}$, nous utilisons le solveur de contrainte Choco [CHO10]. Cependant, après avoir réalisé des tests avec d'autres solveurs de contrainte (en nous appuyant sur les résultats de différents solveurs SMT dans la compétition annuelle SMT-COMP), nous nous sommes finalement réorientés vers Z3, qui semblait avoir de meilleures performances dans notre cas¹.

4.2 Algorithme de simplification du produit

La recherche des paramètres est réalisée en recherchant des cycles acceptants par l'exécution symbolique du Produit issu de la synchronisation d'un PGRN et d'un automate de Büchi. Il

1. Précisons cependant que des améliorations majeures ont été apportées à Choco depuis nos premiers tests (Choco 4 [PFL16]), mais le code actuel de $SP_{\forall TNI_K}$ a été adapté aux particularités de Z3 et ne nous permet pas de refaire simplement ce test de performance.

est possible, en effectuant une analyse statique du Produit, de détecter des transitions non franchissables, ou menant à des cycles finaux non acceptants. Ces transitions peuvent alors être retirées du Produit sans modifier le résultat final de l'exécution symbolique mais en réduisant le temps de parcours du Produit, car cela évite d'emprunter des branches inutiles ou de tester plusieurs transitions qui, statiquement, ne peuvent pas être franchies.

Illustrons cette simplification sur le Produit de notre exemple fil rouge présenté dans la figure 3.6 (page 61). Plus précisément, la figure 4.1 redonne la forme générale de ce produit, et nous allons nous intéresser aux transitions en traits pleins.

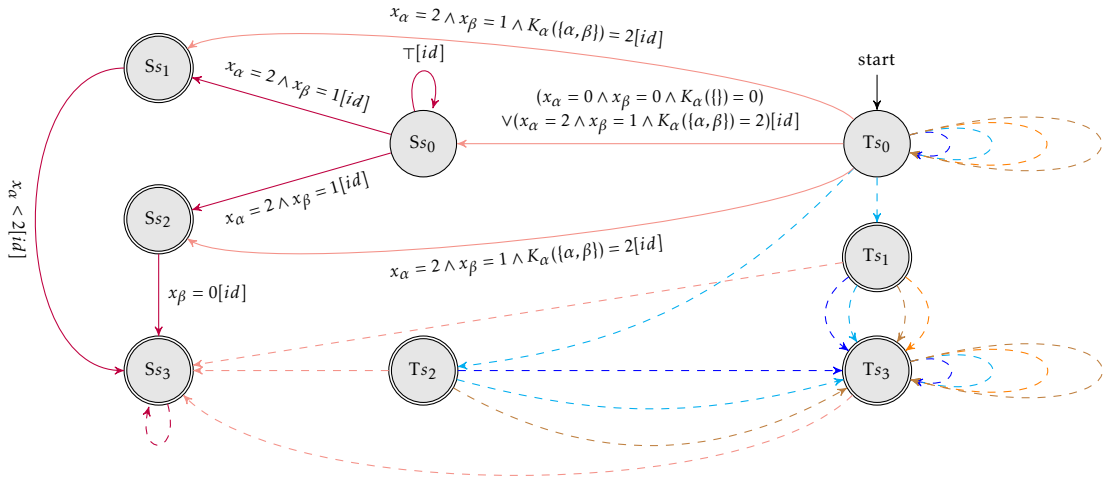


FIGURE 4.1 – Aperçu de la passe de simplification du Produit. Nous nous intéressons ici uniquement aux transitions représentées par un trait pleins.

Les étapes de simplification sont les suivantes :

1. *Suppression des transitions en sortie non satisfaisables.*

Il s'agit des transition sortantes d'un sommet q_Π dont la garde ne peut être satisfaite au vu des gardes et des affectations des transitions en entrée de ce sommet.

Par exemple, la garde de la transition $Ss_1 \rightarrow Ss_3$ est $x_\alpha < 2$. Or, les transitions en entrée de Ss_1 sont soit $x_\alpha = 2 \wedge x_\beta = 1[id]$ (pour la transition venant de Ss_0) soit $x_\alpha = 2 \wedge x_\beta = 1 \wedge K_\alpha(\{\alpha, \beta\}) = 2[id]$ (pour la transition venant de Ts_0). De fait, la transition $Ss_1 \rightarrow Ss_3$ ne pourra jamais être franchie, car les transition arrivant en Ss_1 impliquent $x_\alpha = 2$, ce qui est incompatible avec la garde de sortie $x_\alpha < 2$. La transition $Ss_1 \rightarrow Ss_3$ peut donc être supprimée du Produit. La transition en sortie de Ss_2 peut être supprimée pour la même raison.

Pour un sommet q_Π donné, les transitions en sortie à supprimer correspondent à l'en-

semble suivant :

$$\left\{ (q_{\Pi}, g_{\Pi}, a_{\Pi}, q'_{\Pi}) \in \delta_{\Pi} \mid \forall (q''_{\Pi}, g''_{\Pi}, a''_{\Pi}, q_{\Pi}) \in \delta_{\Pi}, \forall x \in \mathbb{X} \text{ tels que } \llbracket g''_{\Pi} \rrbracket_x \text{ est satisfaisable} \right. \\ \left. \llbracket g_{\Pi} \rrbracket_{a''_{\Pi}(x)} \text{ n'est pas satisfaisable} \right\}$$

2. *Suppression des états sans transition en entrée ou sortie.*

Dans notre exemple, suite à la suppression de la transition $Ss_1 \rightarrow Ss_3$, l'état Ss_3 n'a plus de transition en sortie, on peut donc le supprimer du graphe, ce qui entraîne la suppression des transitions $Ts_0 \rightarrow Ss_1$ et $Ss_0 \rightarrow Ss_1$. Pour des raisons similaires, Ss_2 et $Ts_0 \rightarrow Ss_2$, $Ss_0 \rightarrow Ss_2$ peuvent être supprimés.

3. *Suppression des boucles non acceptantes.*

Dans notre exemple, la suppression de Ss_1 , de Ss_2 et des transitions associées réduit l'état Ss_0 à n'avoir plus qu'une seule transition en sortie : $Ss_0 \rightarrow Ss_0$. Il s'agit d'une boucle sur un état qui n'est pas acceptant, il ne pourra donc jamais donner lieu à un cycle acceptant, et il peut être supprimé du graphe (de même que les transitions en entrée de cet état).

Ces trois étapes sont répétées tant que cela est possible. Le graphe obtenu après simplification est un sous-graphe du Produit, et c'est à partir de lui que l'exécution est réalisée. La figure 4.2 présente le graphe obtenu après simplification du Produit de la figure 3.6.

4.3 Algorithme d'exécution des arbres

Une fois que le Produit a été simplifié, nous pouvons alors en réaliser l'exécution symbolique telle que présentée dans le chapitre précédent. Nous avons néanmoins optimisé cette exécution par un processus de coupure. Il s'agit d'identifier les nœuds des arbres d'exécution dont l'analyse est inutile car les chemins vers lesquels ces nœuds pourraient mener ont déjà été analysés (soit dans une autre branche du même arbre ou d'un autre arbre précédemment construit). On peut alors « couper » la branche correspondant à ce nœud sans perte de solution.

Le processus de coupure s'appuie sur deux critères :

— *Critère d'inclusion des nœuds.*

Considérons deux nœuds $ss = (q, x, pc, \sigma, occ)$ et $ss' = (q, x, pc', \sigma', occ')$ sans lien d'ascendance mais tels que $pc \Rightarrow pc'$. Autrement dit, les instanciations de paramètres qui vérifient pc vérifient aussi pc' . Alors, tout comme pour le repliement présenté dans le chapitre précédent, l'ensemble des transitions franchissables à partir de ss est un sous-ensemble des transitions franchissables à partir de ss' . Si ss' (et ses descendants) ont déjà été analysés alors il n'est pas nécessaire d'analyser ss . Dans pareil cas, on dira que ss est un nœud de coupure *inclus* dans ss' .

— *Critère des conditions de chemin acceptantes.*

Il est également possible d'utiliser les conditions de chemins acceptantes déjà rencontrées pour couper certains nœuds. Notons $\{apc_1, \dots, apc_l\}$ l'ensemble des conditions de chemins

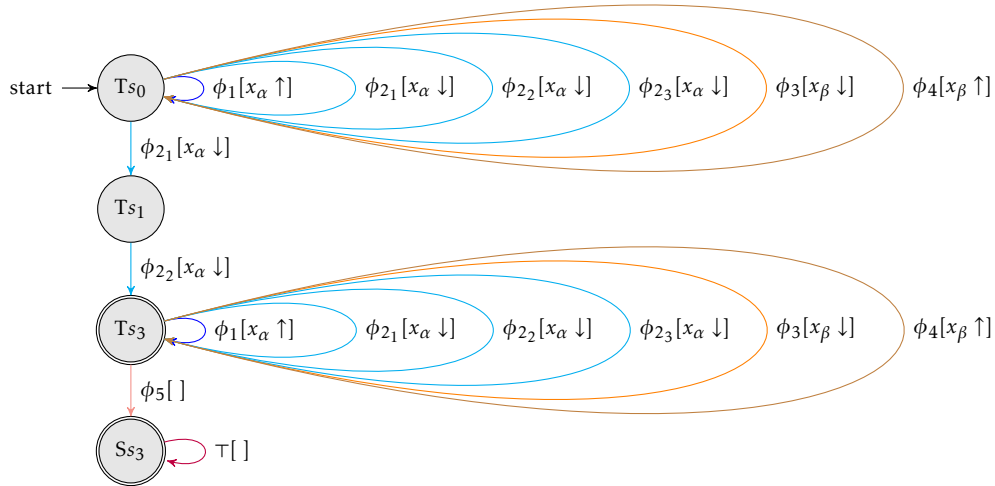


FIGURE 4.2 – Produit associée au PGRN en figure 3.2 et à l’automate de Büchi en figure 3.5 après simplification avec :

$$\begin{aligned} \phi_1 &\equiv x_\alpha < 2 \wedge x_\beta = 0 \wedge x_\alpha < K_\alpha(\{\}) \\ \phi_{2_1} &\equiv x_\alpha = 2 \wedge x_\beta = 1 \wedge K_\alpha(\{\alpha, \beta\}) < 2 \\ \phi_{2_2} &\equiv x_\alpha = 1 \wedge x_\beta = 1 \\ \phi_{2_3} &\equiv x_\alpha = 1 \wedge x_\beta = 0 \wedge K_\alpha(\{\}) = 0 \\ \phi_3 &\equiv x_\alpha = 0 \wedge x_\beta = 1 \\ \phi_4 &\equiv x_\alpha > 0 \wedge x_\beta = 0 \\ \phi_5 &\equiv x_\alpha = 0 \wedge x_\beta = 0 \wedge K_\alpha(\{\}) = 0. \end{aligned}$$

acceptantes connues et supposons que le nœud à créer $ss = (q, x, pc, \sigma, occ)$ soit tel que pc implique l’une des conditions de chemins acceptantes connues apc_i . Alors, toutes les conditions de chemin des nœuds construits à partir de ss sont de la forme $pc \wedge \dots$ (puisque’ils descendent de ss). Ainsi, les conditions de chemins des nœuds descendants de ss impliquent pc et donc impliquent apc_i .

L’algorithme 1 décrit la mise en œuvre de la recherche des conditions de chemin acceptantes. Cet algorithme s’appuie sur un *parcours en profondeur*. Pour les besoins de présentation de l’algorithme, trois variables globales sont utilisées :

- le Produit Π ,

- la liste *acceptingPC* des conditions de chemin acceptantes
- et la liste *nodesList* des nœuds du SET en cours de construction.

Algorithme 1 : Vue d'ensemble de *DFS_transitoire(node)*

Données : *node*, global Π , global *acceptingPC*, global *nodesList*

```

1 pour chaque transition de  $\Pi$  en sortie du sommet de node faire
2   si transition est SATISFAISABLE alors
3     CALCULER succ, le nœud successeur atteint par cette transition;
4     si la pc de succ n'IMPLIQUE pas l'une des pc dans acceptingPC alors
5       si succ est un NŒUD DE RETOUR vers un ancêtre anc alors
6         si il existe un NŒUD ACCEPTANT entre anc et succ alors
7           Ajouter la pc de succ à acceptingPC;
8       sinon si succ n'est pas INCLUS dans un nœud de nodesList alors
9         si succ n'est pas STABLE alors
10          DFS_transitoire(succ);
11        sinon
12          DFS_stable(succ);
13        Ajouter succ à nodesList;

```

Le premier appel à la fonction *DFS* est réalisé à partir d'un nœud racine. Les lignes 1 à 3 de l'algorithme permettent le calcul des successeurs d'un nœud du SET. Rappelons que nous ne parcourons que les nœuds dont la condition de chemin satisfait la CPBF. Trois tests sont alors réalisés successivement, correspondant aux mécanismes de repliement et de coupure décrits précédemment :

- La ligne 4 teste si la condition de chemin du successeur implique l'une des conditions de chemin acceptantes de la liste *acceptingPC* (coupure suivant le critère des conditions de chemin acceptantes.). En cas de réponse positive, l'étude de ce successeur n'est pas poursuivie et l'itération suivante de la boucle *pour chaque* teste une autre transition partant du sommet du nœud courant.
- En cas de réponse négative à ce test, la ligne 5 vérifie l'existence d'un cycle, en recherchant dans les ancêtres du successeur testé un nœud ayant le même état et lié au même sommet de Π . Si un tel ancêtre est trouvé et qu'un nœud acceptant est présent dans ce cycle, alors la condition de chemin du successeur est acceptante et est ajoutée à la liste *acceptingPC* (lignes 6 à 7).
- Si aucun des deux tests précédents n'est positif, la ligne 8 vérifie si le successeur est inclus dans un nœud déjà créé (présent dans *nodesList*), ce qui correspond au premier type de coupure présenté précédemment. Si un tel nœud existe déjà, l'algorithme passe au successeur suivant. Sinon, le chemin est poursuivi à partir du successeur.

Dans le cas où aucun des critères de coupure, ni le repliement ne peut s'appliquer, la construction de l'arbre se poursuit à partir du successeur. On distingue ici deux cas, suivant la

valeur de la première composante du sommet de *succ*. En effet, ce sommet provient du PGRN, et peut donc prendre seulement deux valeurs. Si cette composante est T , on poursuit le parcours avec la même fonction (ligne 10); sinon (la composante est alors égale à S), on dit que l'état est *stable* et on poursuit le parcours avec une version simplifiée de la fonction courante, appelée *DFS_stable()* (ligne 12). Le successeur est ensuite ajouté à *nodesList* (ligne 13).

La version simplifiée de l'algorithme (algorithme 2) prend en compte que, par construction, tous les successeurs d'un nœud stable sont eux-mêmes stables (car la seule transition sortant de l'état S du PGRN est une boucle). De plus, par construction également, les gardes des transitions reliant deux états stables sont toujours de la forme $g_{\Pi} = g_P \wedge g_B$ avec $g_P = \top$ et l'affectation de la transition est $a_{\Pi} = id$. Le premier ancêtre stable est appelé *racine stable*, et ses spécificités impliquent donc :

- il est inutile de faire le test des conditions de chemin (ligne 4 de l'algorithme 1) car ce test a déjà été réalisé avec la racine stable (et la condition de chemin n'est plus modifiée),
- il n'est pas nécessaire de tester la satisfaisabilité de la conjonction de la condition de chemin de l'ancêtre et de la garde de la transition empruntée, ni de substituer des états à réaliser dans la *pc* (ligne 2),
- il n'y a pas de mise à jour de l'état (ligne 3).

Si un nœud est acceptant dans un parcours stable, alors toutes les explorations des branches effectuées à partir de la racine stable peuvent être stoppées. En effet, la condition de chemin de la racine stable est identique à la condition de chemin de tous ses descendants.

Algorithme 2 : Vue d'ensemble de *DFS_stable(node)*

Données : *node*, global Π , global *acceptingPC*, global *nodesList*

```

1 pour chaque transition de  $\Pi$  en sortie du sommet de node faire
2   si transition est SATISFAISABLE alors
3     CALCULER succ, le nœud successeur atteint par cette transition;
4     si succ est un NŒUD DE RETOUR vers un ancêtre anc alors
5       si il existe un NŒUD ACCEPTANT entre anc et succ alors
6         Ajouter la pc de succ à acceptingPC;
7         Retourner true;
8     sinon si succ n'est pas INCLUS dans un nœud de nodesList alors
9       estAcceptant = DFS_stable(succ);
10      si estAcceptant == true alors
11        Retourner true;
12      Ajouter succ à nodesList;
13 Retourner false;
```

Sous l'hypothèse d'une formule de la forme $A\varphi$, nous procédons à la conjonction des contraintes sur les paramètres et des négations de chacune des conditions de chemin acceptantes stockées dans la liste *acceptingPC* construite à l'issue de l'exploration de tous les arbres.

Les instanciations de paramètres qui vérifient cette conjonction de contraintes définissent les dynamiques qui satisfont la propriété biologique $A\varphi$.

4.4 Pistes d'optimisation

Même si les premiers résultats de $SP\cup TNI_k$ sont intéressants, il est probable qu'il existe encore une marge possible d'optimisations pour traiter des modèles plus conséquents dans un temps raisonnable. Dans cette section, nous décrivons les pistes d'optimisation sur lesquelles nous avons commencé à travailler, ou que nous estimons prometteuses.

4.4.1 Parallélisation du parcours

Idées

Lors d'une utilisation de $SP\cup TNI_k$, la partie du programme prenant le plus de temps est le parcours du Produit (i.e. l'exécution symbolique); l'algorithme correspondant est donné dans le chapitre 4 section 4.3 (algorithme 1 p.77). Cet algorithme est présenté sous une forme séquentielle. Une optimisation possible pour accélérer ce traitement est de le répartir entre plusieurs processus s'exécutant en parallèle. Pour cela, il faut réfléchir à la meilleur façon de découper les tâches à traiter. Plusieurs répartitions sont envisageables, dont les trois pistes principales suivantes :

— *Répartir selon des paramètres fixés.*

Il s'agit de fixer certains paramètres à une valeur distincte dans chaque processus, ce qui revient à multiplier le nombre d'arbres indépendants à construire (en fonction du nombre de paramètres fixés et du domaine de valeur de ces paramètres).

— *Répartir selon le domaine des états.*

Il s'agit de répartir le traitement entre plusieurs arbres d'exécution n'agissant que sur un domaine d'états (une sous-partie de \mathbb{X}) défini à l'avance (ce qui implique donc une communication entre les arbres au niveau des limites du domaine).

— *Répartir selon des sous-graphes du Produit.*

Il s'agit de diviser le Produit en plusieurs sous-graphes, chacun étant parcouru par un processus différent.

La première méthode, i.e. fixer les paramètres, se rapproche en fait de la méthode utilisée par KLARNER et al. (décrite dans [Kla+12a]), qui consiste à partager les processus selon des groupes dont les paramètres (fixés) sont de valeur proche. Dans notre cas, cette méthode reviendrait en fait à lancer en parallèle plusieurs traitements en modifiant les contraintes sur les paramètres données en entrée de manière à fixer une partie des paramètres à la valeur souhaitée. Comme le temps de calcul du Produit est supposé négligeable par rapport à son exécution, nous pouvons évaluer directement le gain de temps espéré à partir de l'implémentation séquentielle : l'exécution séquentielle la plus longue donne une idée du temps d'exécution du processus le

plus long, qui définit le temps d'exécution total des processus en parallèle. Une question liée à cette méthode est de déterminer le meilleur choix minimum de paramètres à contraindre en fonction du type de la formule. Néanmoins, avec une telle méthode, nous nous éloignons des choix initiaux que nous avons fait, i.e. conserver les paramètres sous forme symbolique pendant le parcours. Nous avons donc choisi de nous concentrer sur d'autres pistes de parallélisation.

La seconde méthode, i.e. répartir les calculs selon le domaine des états, nous a paru très coûteuse en terme de communication entre les arbres (en particulier s'il existe un cycle entre des états n'appartenant pas au même domaine). Elle reste cependant une idée intéressante qui pourrait être envisagée dans un futur développement de SP ν TNI κ . Comme dans le cas précédent, le type de formule à traiter pourrait être également étudié pour proposer une méthode de cloisonnement propre à chacun.

Enfin, c'est sur la troisième méthode, i.e. répartir selon des sous-graphes du Produit, que nous avons concentré nos efforts. Cette méthode revient à diviser le travail sans avoir à fixer les paramètres ni besoin de communiquer entre les processus. Dans la suite, nous revenons plus en détail sur ses avantages et ses inconvénients.

Parallélisation suivant les SCC du Produit

La version parallèle de SP ν TNI κ repose sur la décomposition du Produit en *composantes fortement connexes* (abrégé en SCC pour *Strongly Connected Components*). La décomposition en SCC nous permet de répartir les calculs de parcours des arbres entre plusieurs processeurs travaillant en autonomie.

Décomposition en composantes fortement connexes. La définition qui suit donne la définition d'une composante fortement connexe prélevée du Produit.

Définition 4.4.1 (SCC du Produit). Soit un Produit $\Pi_{-\varphi} = (Q_{\Pi}, q_{\Pi}^0, A_{\Pi}, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_{\Pi})$. Une SCC \mathcal{C} de Π est un sous-graphe de Π tel que : $\mathcal{C} = (Q_{\mathcal{C}}, q_{\mathcal{C}}^0, A_{\mathcal{C}}, \text{Contraintes}(\mathbb{G} \cup \mathbb{K}), \delta_{\mathcal{C}})$ avec :

- $Q_{\mathcal{C}} \subset Q_{\Pi}$, $Q_{\mathcal{C}} \neq \emptyset$;
- $A_{\mathcal{C}} \subset A_{\Pi}$;
- $\delta_{\mathcal{C}} \subset \delta_{\Pi}$

construit à partir de Π en utilisant les propriétés suivantes :

$$\forall q_1, q_2 \in Q_{\mathcal{C}}, \exists \sigma = q_1 \cdots q_2 \in \text{paths}(\mathcal{C}) \wedge \sigma \in \text{paths}(\Pi)$$

$$\forall q_1, q_2 \in Q_{\mathcal{C}} \text{ tels que } t = q_1 \xrightarrow{[g,a]} q_2 \in \delta_{\Pi} \text{ alors } t \in \delta_{\mathcal{C}}$$

$$\forall q_1 \in Q_{\mathcal{C}}, q_2 \in Q_{\Pi} \wedge q_2 \notin Q_{\mathcal{C}}, \nexists \sigma = q_1 \cdots q_2 \in \text{paths}(\Pi)$$

Notation 4.4.1 (SCC initiale, non acceptante, stable). Une SCC est dite *initiale* si elle contient le sommet initial du Produit (si $q_{\Pi}^0 \in Q_C$).

Une SCC vérifiant $A_C = \emptyset$ est dite *non acceptante*. Autrement dit, Q_C ne contient pas de sommet correspondant à un sommet acceptant du Produit.

Une SCC *stable* est une SCC possédant des sommets stables, autrement dit des sommets du produit dont la première composante correspond à l'état stable du PGRN.

Comme nous le verrons par la suite, nous avons défini des traitements spécifiques aux SCC stables et/ou non acceptantes.

La première étape de la parallélisation consiste donc à découper le Produit en un ensemble de composantes fortement connexes. Pour cela, nous utilisons l'algorithme de Tarjan [Tar72] grâce auquel nous obtenons la liste des SCC.

Par construction, chaque sommet du Produit appartient à une et une seule SCC. Chaque composante va ensuite être exécutée indépendamment.

Exécution des composantes fortement connexes Chaque SCC du Produit est parcourue indépendamment, en parallèle, en suivant le principe d'exécution du Produit présenté dans le chapitre 3, section 3.3. Chaque processus a ainsi en charge l'exécution d'une composante afin de déterminer les conditions de cycles acceptants (locales) dans la SCC qui lui est attribuée.

Les racines des différents arbres d'exécution construits à partir d'une SCC correspondent aux *entrées* possibles dans la composante. Exception faite de la SCC initiale, les entrées dans une SCC sont établies en fonction des gardes et des affectations des transitions dont la cible est un sommet appartenant à la composante traitée.

Les racines de la SCC initiale sont par contre construites à partir de l'espace des états et de la pc τ , de la même façon que les racines du parcours séquentiel (section 4.3).

Par construction, un cycle ne peut pas être partagée entre plusieurs SCC ; la détermination des cycles se déroule donc comme dans le parcours séquentiel. Cependant, par rapport au parcours séquentiel, nous ne conservons pas seulement la pc acceptante (qualifiée de *pc localement acceptante* ici) correspondante mais également à partir de quelle entrée dans la SCC le cycle correspondant peut être atteint. En effet, toutes les entrées dans la SCC que nous avons provisionnées ne sont pas nécessairement atteignables depuis les racines du Produit, et par conséquent toutes les cycles acceptants (locales) détectés ne sont pas nécessairement atteignables.

Afin de pouvoir déterminer les conditions d'atteignabilité effective des SCC, nous collectons également les conditions de sortie des SCC, en fonction des entrées.

Signalons que nous conservons les techniques de coupure présentées précédemment (section 4.3), avec certaines ajustements cependant.

Ainsi, la coupure due à une pc acceptante est dans ce cas locale : chaque processus ne considère que les pc localement acceptantes de sa SCC. De plus, seule les pc localement acceptantes

découvertes dans le même arbre d'exécution ne sont prises en compte car chaque pc est collectée en fonction de l'entrée dans la SCC. Nous avons donc moins de cas de ce type de coupure dans ce mode de parcours. La suite du chemin ne nous intéresse plus pour la détection de cycles dans ce cas (car la pc est déjà mise de côté), mais nous devons cependant déterminer les conditions de sortie ; un parcours spécifique (avec moins de calculs à réaliser) est utilisé, sur le principe du parcours du Produit à partir d'un état stable (cf. algorithme 2 p.78).

Par rapport à la coupure entre des nœuds n'appartenant pas à la même branche, nous utilisons la même méthode de repérage car ce type de coupure est nécessairement interne à une SCC (car un sommet du Produit appartient à une SCC unique). Nous avons par contre une analyse supplémentaire à faire, qui consiste à déterminer si le nœud de coupure possède des descendants acceptants avec une pc identique ou moins spécifique que la pc du nœud de retour, qui signifierait que des cycles acceptants pourraient être atteints dans la suite du parcours. Si cela est le cas, nous ajoutons les pc localement acceptantes correspondantes (conjonction de la pc du nœud de retour avec les gardes franchies entre le nœud de coupure et le nœud acceptant) que nous lions à l'entrée de la SCC permettant d'atteindre le nœud de retour. Dans tous les cas, nous pouvons stopper le chemin. Nous récoltons cependant les conditions de sortie à partir des conditions de sortie du nœud de coupure qui sont compatibles avec le nœud de retour (reposant sur le satisfaisabilité de la conjonction des pc du nœud de retour avec chacune des conditions de sortie).

Enfin, les SCC non acceptantes sont rapides à analyser : on recherche uniquement les conditions de sortie. Dans le cas d'une SCC stable, le parcours à partir d'un état stable est traité différemment, avec moins de calculs à réaliser (même type de parcours que dans le cas de détection des sorties à partir d'une coupure).

Consolidation Quand toutes les composantes ont été traitées (en parallèle), on sélectionne les pc localement acceptantes qui peuvent être atteintes depuis les racines du Produit. Pour cela, on examine les pc en suivant un ordre topologique de leurs composantes : si l'entrée liée à la pc correspond à au moins une sortie atteignable d'une composante précédemment examinée (i.e. il existe une transition permettant d'atteindre le cycle acceptant à partir d'un sommet d'une autre composante, lui-même atteignable depuis une des racines) alors la pc est acceptante. Ainsi on retrouve l'ensemble des pc acceptantes correspondant aux conditions d'atteignabilité des cycles du Produit passant infiniment souvent par un sommet acceptant. Comme pour le parcours séquentiel, la conjonction des négations des pc acceptantes permet de déterminer l'ensemble des jeux de paramètres satisfaisant la formule LTL sur tous les chemins.

Résultats Le traitement indépendant des SCC augmente le nombre de calculs à réaliser et de nœuds à parcourir (car les entrées des arbres d'exécution liés aux SCC sont établies en fonctions des transitions en entrée de la composante, alors que les gardes de ces transitions ne seraient pas nécessairement franchies pendant le parcours du Produit, ou pourraient être plus

contraintes, à cause des transitions précédemment franchies). Cependant, nous pouvons réaliser ces traitements en parallèle car ils sont indépendants (aucune donnée n'a besoin d'être échangée entre les processus de traitement des SCC), nous faisant ainsi espérer un gain de temps total ; seul le post-traitement des pc localement acceptantes mises de côté nécessite que le parcours de toutes les composantes soient achevées.

Après un test sur notre exemple jouet, nous nous sommes cependant aperçus que ce traitement n'était pas aussi prometteur que nous l'espérions ; en effet, le traitement local d'une SCC prend plus de temps que le temps d'exécution total du Produit. Ce résultat pourrait cependant être tempéré selon la taille des modèles ou la forme des graphes à considérer.

4.4.2 Autres pistes

Nous avons également réfléchi à d'autres pistes nous permettant de réduire le temps de parcours des arbres d'exécution ou d'optimiser la mémoire. Une possibilité serait de ne conserver en mémoire qu'une partie seulement des nœuds parcourus (à une fréquence aléatoire par exemple). Ceci permet, d'une part, d'alléger l'espace occupé, mais également de procéder à moins de calculs de coupure (car l'ensemble des nœuds à comparer est plus petit), et ainsi de gagner du temps. À l'inverse, des branches inutiles peuvent être construites, correspondant à des chemins qui auraient été coupés en amont à cause de la détection d'une coupure d'un nœud non mémorisé. Il pourrait être intéressant d'étudier s'il existe une fréquence d'optimisation optimale qui permettrait de contre-balancer les avantages et les inconvénients de cette méthode. D'autres « petites » optimisations de ce genre pourraient peut-être améliorer les performances de SPvTNI_k.

D'autre part, nous avons testé deux solveurs de contrainte au cours du développement de SPvTNI_k : Choco[CHO10] tout d'abord, puis Z3[DB08]. Les performances de Z3 sont meilleures pendant l'exécution du Produit (en partie dues à l'utilisation du système de pile de contraintes, décrites dans la section 4.1.2 du chapitre 4), mais à contrario, nous avons noté que l'énumération des solutions (à l'étape finale) est lente. Cette énumération repose sur la résolution de la conjonction des négations des pc acceptantes ; Z3 ne donne qu'une seule solution à la fois, pour obtenir une autre solution il est ainsi nécessaire de résoudre la conjonction du système initial et de la négation de la contrainte définissant la solution déjà déterminée. Et ainsi de suite pour obtenir les autres solutions.

Dans le but de réduire le temps d'énumération (qui était par exemple moindre avec Choco, mais d'autres solveurs pourraient également être envisagés), il nous paraît donc intéressant de changer de solveur à cette étape mais cela demande une réécriture des pc acceptantes pour qu'elles soient compatibles avec le deuxième solveur choisi.

Chapitre 5

Etudes de cas

Dans ce chapitre, nous présentons deux études de cas classiques (et pour chacune, deux variations) réalisées avec notre outil SP_UTNI_K.

5.1 Inductibilité de la cytotoxicité de <i>Pseudomonas aeruginosa</i>	86
5.1.1 Version minimale	88
5.1.2 Version étendue	94
5.2 Cycle de vie du phage λ	98
5.2.1 Version 1	100
5.2.2 Version 2 : séries temporelle	102

D'autres études de cas (cette fois-ci intégralement tirées de la littérature) sont également présentées en annexe B. Chaque étude de cas peut être lue indépendamment.

La première étude de cas, *Inductibilité de la cytotoxicité de Pseudomonas aeruginosa*, est détaillée en pas-à-pas afin d'illustrer notre approche. Elle est constituée d'un modèle simple tiré de la littérature ([Fil+06]) que nous avons ensuite étendu afin de montrer la facilité d'adaptation de notre outil pour tester différentes configurations du modèle original. Cette extension a été réalisée avec l'aide d'une biologiste, Dr. Janine Guespin-Michel (l'une des auteurs de [Fil+06]). Cette étude de cas a été présentée dans notre article [Gal+15].

La deuxième étude de cas porte sur un modèle très étudié en biologie : le *Cycle de reproduction du phage λ* . Elle se divise également en deux temps. La première partie porte sur la vérification de propriétés adaptées directement à partir des connaissances biologiques du système. Dans la deuxième partie nous vérifions les propriétés testées par KLARNER et al. dans [Kla+12a], constituées d'une sous-partie de formules LTL bien spécifiques, appelées *séries temporelles*. Cela sera l'occasion de comparer l'expressivité et les performances de notre outil SP_UTNI_K avec leur outil Parsybone. Cette étude de cas a été évoquée dans notre article [Gal+14b].

5.1 Inductibilité de la cytotoxicité de *Pseudomonas aeruginosa*

Notre première étude de cas porte sur l'inductibilité de la cytotoxicité de la bactérie *Pseudomonas aeruginosa*, i.e. la régulation de la capacité de *P. aeruginosa* à altérer certaines cellules cibles via la sécrétion de toxines.

Les êtres humains entrent fréquemment en contact avec *P. aeruginosa*, en particulier en milieu hospitalier où elle est en cause dans de nombreux cas d'infections nosocomiales. C'est un pathogène opportuniste : elle affecte rarement une personne en bonne santé mais elle est en revanche une cause de mortalité importante chez les patients immunodéprimés, notamment chez ceux qui sont atteints de mucoviscidose [KKK09].

L'un des mécanismes pathogènes de *P. aeruginosa* réside dans un de ses systèmes de production et de sécrétion de toxines, appelé *système de sécrétion de type III* (ou T3SS pour *Type III Secretion System*). Le T3SS est un assemblage de protéines composé en particulier d'une structure en forme d'aiguille (appelée *appareil du T3SS*), qui permet de transférer directement les toxines du cytosol de la bactérie au cytosol de la cellule hôte. Ces toxines altèrent leur fonctionnement, voire dans certains cas, provoquent leur mort.

La production de toxines ainsi que l'activation de l'appareil du T3SS (ouverture/fermeture du canal de sécrétion) sont régulés par un ensemble de protéines, parfois sous la forme d'un complexe.

Le GRN contrôlant l'inductibilité de la cytotoxicité de *P. aeruginosa* est présenté en figure 5.1. Il est composé des gènes¹ suivants :

- l'opéron² lié aux gènes ExsC, ExsE et ExsA (dont le promoteur est pC) avec les rôles (simplifiés) suivants :
 - ExsA régule l'expression des (43) gènes qui forment le T3SS dans *P. aeruginosa* [Bal+12]. Il active en particulier l'opéron lié à pC (il régule ainsi ExsC, ExsE et ExsA), ExsD et également le gène psc, qui contrôle la formation de l'appareil de sécrétion, ainsi que des gènes codant des exoenzymes (ExoS, ExoT, ExoU et ExoY).
 - ExsE est sécrétée à condition que le canal de sécrétion soit ouvert (ce qui dépend de la réception d'un signal d'activation, par exemple le contact de l'aiguille avec une cellule cible). Si à l'inverse le canal de sécrétion est fermé, ExsE forme un complexe avec ExsC.
 - ExsC est un anti-anti-activateur de ExsA. En effet, si ExsE n'est pas libre de former un complexe avec lui, il en forme un avec ExsD, empêchant ainsi ce dernier de former un complexe avec ExsA.

L'affinité entre ExsC et ExsE (respectivement entre ExsD et ExsC) est plus grande qu'entre ExsC et ExsD (ExsD et ExsA).

1. On rappelle que par simplicité les protéines synthétisées à partir d'un gène sont nommées de la même manière que le gène en question.

2. On rappelle qu'un opéron est un ensemble de gènes situés à proximité les uns des autres, liés à un même promoteur et donc à priori exprimés de façon conjointe [Jac+05].

- ExsD est un anti-activateur de ExsA, il se lie à ExsA pour former un complexe et l'empêche alors d'activer la transcription du T3SS.
- le groupe Tox correspondant aux exoenzymes (toxiques) sécrétées par T3SS : ExoS, ExoT, ExoU et ExoY. Une fois injectée dans la cellule hôte, chacune a un mode d'action et un facteur de virulence qui lui est propre.
- le groupe T3SS des gènes codant l'appareil du T3SS, en particulier psc.

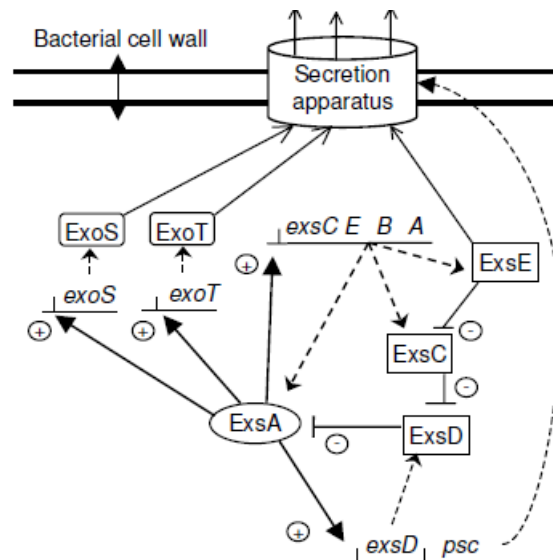


FIGURE 5.1 – GRN contrôlant l'inductibilité de la cytotoxicité de *P. aeruginosa* tiré de [Fil+06]

Des expériences biologiques ont montré l'existence de deux types de souches particulières de la bactérie : l'un *inductible* et l'autre *non inductible* (dans [Fil+06], les deux souches correspondantes analysées sont respectivement CHA et PAO1).

Dans le cas de souches dites *inductibles*, l'activation du T3SS conduit à un état *induit* dans lequel les toxines sont produites et injectées dans la cellule cible. Dans ce cas, le phénotype (i.e. le caractère observable) de la bactérie est *cytotoxique*. L'activation du T3SS est déclenchée par le contact entre la bactérie et la cellule cible ou bien par l'appauvrissement du milieu de culture en ions calcium (Ca^{2+}) [Fra97 ; Val+99].

Dans le cas de l'autre type de souches, dites *non inductibles*, le T3SS n'est jamais activé, en dépit des signaux d'activation. Le phénotype de la bactérie est donc *non cytotoxique* dans ce cas.

Les biologistes s'intéressent à la raison de cette différence, en espérant pouvoir favoriser le phénotype non cytotoxique de *P. aeruginosa*.

Comme aucune mutation entre les deux types de souches n'a été trouvée, une des hypothèses permettant d'expliquer ce phénomène repose sur la modification des phénotypes cytotoxique et non cytotoxique (qui sont conditionnés par l'expression des gènes) en fonction de l'environnement, via un mécanisme appelé *switch épigénétique*. Les switches épigénétiques caractérisent

le fait que même en dehors de mutations génétiques, des cellules peuvent acquérir un autre phénotype, qui correspond à un comportement différent.

Du point de vue de la modélisation, ces deux phénotypes *stables* correspondent à plusieurs bassins d'attractions. Dans [RC07], RICHARD et al. ont démontré (dans le cas discret) que cette situation de multi-stationnarité n'était possible que si le graphe d'interaction $\Gamma = (G, I)$ modélisant les régulations du système comportait au moins un *circuit de rétroaction positif* (dans lequel chaque élément du circuit a une influence, directe ou indirecte, positive sur lui-même), i.e. un cycle $i_1 \rightarrow i_2 \rightarrow \dots \rightarrow i_k = i_1$ (avec i_1, \dots, i_k dans I) comportant un nombre pair d'inhibitions, soit $\exists n \in \mathbb{N}, \left| \{(g, s, t, g') \mid (g, s, t, g') \in \{i_1, \dots, i_k\} \text{ avec } s = -\} \right| = 2n$.

Nous allons présenter plusieurs versions du GRN en cause dans l'inductibilité de la cytotoxicité de *Pseudomonas aeruginosa*. La première version est tirée de la littérature : nous nous appuyons sur le modèle proposé dans [Fil+06]. Il s'agit d'un modèle du système simplifié et réduit au minimum. Nous présenterons ensuite plusieurs versions dites *étendues* à partir de ce GRN : pour mettre au point ces modèles, nous nous sommes appuyés sur l'expertise de Mme Janine Guespin, docteur en biologie et co-auteur de [Fil+06].

5.1.1 Version minimale

Graphe d'interactions

Le GRN pilotant l'inductibilité de la cytotoxicité de *P. aeruginosa* peut être réduit à trois gènes, comme présenté dans l'article [Fil+06]. Nous appellerons cette version « GRN minimal » dans la suite.

Le GRN minimal est composé des entités suivantes : le gène ExsA, le gène ExsD et le groupe ToxT3SS, qui regroupe les gènes formant Tox et T3SS présentés lors du préambule de cette étude de cas. Cette simplification repose sur les hypothèses suivantes :

- Le système initial est exempt de toxines et l'étude commence à partir du moment où le signal est activé.
- Tout au long de l'étude, on considère que le milieu est pauvre en calcium (signal d'activation du T3SS).

Par conséquent, il n'y a que deux états possibles du système :

- *non inductible*
- *induit*

(i.e. l'état *inductible non induit* est impossible).

Du point de vue des régulations, comme nous l'avons indiqué précédemment, ExsA est un activateur de ToxT3SS, de ExsD et de lui-même. Son expression est par contre inhibée par ExsD.

Ces informations ne suffisent toutefois pas pour pouvoir établir un graphe d'interactions unique. En effet, comme ExsA active trois gènes (ou groupe de gènes), il y a treize combinaisons possibles pour les seuils de régulations des interactions dont ExsA est la source. Parmi ces treize

possibilités, FILOPON et al. considèrent les deux suivantes :

$$t(\text{ExsA} \rightarrow \text{ExsA}) > t(\text{ExsA} \rightarrow \text{ExsD})$$

$$\text{ou } t(\text{ExsA} \rightarrow \text{ExsA}) < t(\text{ExsA} \rightarrow \text{ExsD}),$$

avec dans les deux cas : $t(\text{ExsA} \rightarrow \text{ToxT3SS}) = 2.$

Comme les autres composants du GRN minimal régulent au plus un gène, ces deux possibilités correspondent à deux graphes d'interactions différents. Tout comme [Fil+06], nous considérons dans la suite ces deux versions du graphe d'interactions.

Dans le premier graphe d'interactions $\Gamma_{5.2}$ (présenté en figure 5.2), ExsA active ExsD dès qu'il atteint son premier niveau d'expression ; il active également sa propre expression ainsi que l'expression de ToxT3SS à partir de son niveau 2.

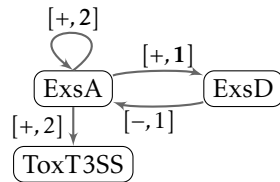


FIGURE 5.2 – Version 1 : $\Gamma_{5.2}$

Dans la deuxième graphe d'interactions $\Gamma_{5.3}$ (présenté en figure 5.3), ExsA active sa propre expression dès qu'il atteint son premier niveau d'expression ; il active également ExsD ainsi que l'expression de ToxT3SS à partir de son niveau 2.

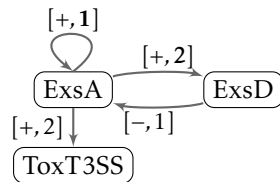


FIGURE 5.3 – Version 2 : $\Gamma_{5.3}$

Pour ces deux cas, nous obtenons alors les phénotypes suivants en fonction des niveaux d'expression :

- non inductible : impossibilité de produire des toxines (soit $x_{\text{ToxT3SS}} = 0$) ;
- induit : possibilité de produire des toxines ($x_{\text{ExsA}} = 2$) ;
- cytotoxique : production de toxines ($x_{\text{ToxT3SS}} = 1$).

Notons que le graphe d'interactions $\Gamma_{5.2}$ correspond, à peu de choses près, à l'exemple jouet que nous avons utilisé pour illustrer notre approche dans les chapitres précédents. Plus précisément, $\Gamma_{5.2}$ contient un élément supplémentaire : ToxT3SS. Mais comme celui-ci est uniquement en *sortie* du GRN (i.e. il n'influe sur aucun gène du GRN), il peut éventuellement ne pas être

considéré en adaptant les formules logiques sur ExsA, à la manière de [Fil+06] (nous reviendrons sur cette adaptation dans la suite).

Paramètres et PGRN

Quel que soit le graphe d'interaction ($\Gamma_{5.2}$ ou $\Gamma_{5.3}$), huit paramètres sont à considérer :

- $K_{\text{ExsA},\{\}};$
- $K_{\text{ExsA},\{\text{ExsA}\}};$
- $K_{\text{ExsA},\{\text{ExsD}\}};$
- $K_{\text{ExsA},\{\text{ExsA},\text{ExsD}\}};$
- $K_{\text{ExsD},\{\}};$
- $K_{\text{ExsD},\{\text{ExsA}\}};$
- $K_{\text{ToxT3SS},\{\}};$
- $K_{\text{ToxT3SS},\{\text{ExsA}\}}.$

Le nombre d'instanciations possibles des paramètres dans \mathbb{K} est égal à 1296.

La figure 5.4 présente la forme générale du PGRN associé aux graphes d'interactions $\Gamma_{5.2}$ et $\Gamma_{5.3}$. Il y a six (deux fois le nombre de gènes) transitions de T à T , une transition de T à S , et une transition de S à S . Les gardes des transitions dépendent de la version du graphe d'interactions. Par exemple, les conditions d'augmentation de ExsD dépendent du seuil de $\text{ExsA} \rightarrow \text{ExsD}$. Ainsi, on obtient pour $\Gamma_{5.2}$:

$$\text{Increase}(\text{ExsD}) \stackrel{\text{def}}{=} (\chi_{\text{ExsA}} < 1 \wedge \chi_{\text{ExsD}} < K_{\text{ExsD},\{\}}) \vee (\chi_{\text{ExsA}} \geq 1 \wedge \chi_{\text{ExsD}} < K_{\text{ExsD},\{\text{ExsA}\}})$$

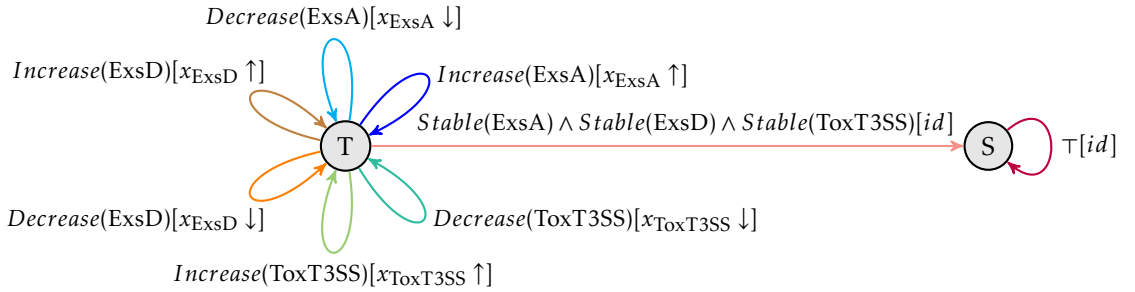


FIGURE 5.4 – Forme générale du PGRN associé aux graphes d'interaction $\Gamma_{5.2}$ et $\Gamma_{5.3}$

En appliquant les contraintes CPBF sur les paramètres de $\Gamma_{5.2}$, nous obtenons (après simplification) les conditions suivantes pour une instance sur les paramètres \mathcal{K} :

- $K_{\text{ExsA},\{\text{ExsA}\}} = 2;$
- $K_{\text{ExsA},\{\text{ExsD}\}} = 0;$
- $K_{\text{ExsD},\{\}} = 0;$
- $K_{\text{ExsD},\{\text{ExsA}\}} = 1;$

- $K_{\text{ToxT3SS},\{\}} = 0$;
- $K_{\text{ToxT3SS},\{\text{ExsA}\}} = 1$;
- $K_{\text{ExsA},\{\}} < 2 \vee 0 < K_{\text{ExsA},\{\text{ExsA},\text{ExsD}\}}$
- $K_{\text{ExsA},\{\}} > 0 \vee K_{\text{ExsA},\{\text{ExsA},\text{ExsD}\}} < 2$

Ainsi, pour $\Gamma_{5,2}$, la garde $\text{Increase}(\text{ExsD})$ du PGRN associé peut être simplifiée en :

$$\chi_{\text{ExsA}} \geq 1 \wedge \chi_{\text{ExsD}} = 0$$

car aucune interprétation v_x ne peut satisfaire

$$\llbracket \chi_{\text{ExsA}} < 1 \wedge \chi_{\text{ExsD}} < K_{\text{ExsD}}(\{\}) \rrbracket_{\mathcal{K}}$$

Au final, seuls sept instances de paramètres différentes sont compatibles avec ces contraintes, sur un total de 1296 instances correspondantes à la seule contrainte de domaine. Ces sept instances de paramètres définissent cinq dynamiques différentes. En effet, les instances de paramètres \mathcal{K} telles que

$$K_{\text{ExsA},\{\}} < 2 \wedge K_{\text{ExsA},\{\text{ExsA},\text{ExsD}\}} < 2$$

ne définissent par exemple que deux dynamiques différentes : l'une associant $K_{\text{ExsA},\{\}}$ à 0, l'autre à 1.

Connaissances biologiques

Dans le cas de l'inductibilité de la cytotoxicité de *Pseudomonas aeruginosa*, deux propriétés biologiques peuvent être exprimées :

- Pour une souche inductible (et dans un milieu pauvre en calcium), la cytotoxicité de la bactérie est toujours activée après un certain temps et ce caractère reste stable. Cette connaissance peut se traduire par la formule LTL $\varphi_1 \equiv \mathbf{G}(\text{inductible} \Rightarrow \mathbf{FG} \text{cytotoxique})$.
- Pour une souche non inductible, la bactérie ne peut pas sécréter de toxines. Ce qui se traduit par $\varphi_2 \equiv \mathbf{G}(\neg \text{inductible} \Rightarrow \mathbf{FG} \neg \text{cytotoxique})$.

Remarquons ici que nous considérons que l'acquisition ou la perte du caractère cytotoxique peut ne pas être immédiate ; d'autres interprétations sont possibles.

Que recouvrent les propositions *cytotoxique* et *inductible* ? La cytotoxicité correspond à la production de toxines, et donc aux états où $x_{\text{ToxT3SS}} = 1$ (autrement dit, *cytotoxique* coïncide avec l'égalité $x_{\text{ToxT3SS}} = 1$). D'après le seuil de l'interaction $\text{ExsA} \rightarrow \text{ToxT3SS}$ (identique dans les deux versions de graphes d'interaction), un état inductible correspond aux situations où $x_{\text{ExsA}} = 2$ (autrement dit, *inductible* coïncide avec l'égalité $x_{\text{ExsA}} = 2$) ; en-dessous de cette valeur, la bactérie est non inductible (car nous considérons dans ce modèle simplifié que le signal d'activation (milieu pauvre en calcium) est toujours présent, l'état inductible est donc ici assimilé à celui d'induit).

Le système doit donc vérifier la formule LTL φ suivante (correspondant à $\varphi_1 \wedge \varphi_2$) :

$$\mathbf{G}(\chi_{\text{ExsA}} = 2 \Rightarrow \mathbf{FG} \chi_{\text{ToxT3SS}} = 1) \wedge \mathbf{G}(\chi_{\text{ExsA}} < 2 \Rightarrow \mathbf{FG} \chi_{\text{ToxT3SS}} = 0) \quad (5.1)$$

Notons ici que nous nous éloignons de la formule utilisée par [Fil+06] car nous considérons explicitement que le signal est activé.

L'automate de Büchi présenté en figure 5.5, noté $B_{\neg\varphi}$, reconnaît la négation de la formule φ .

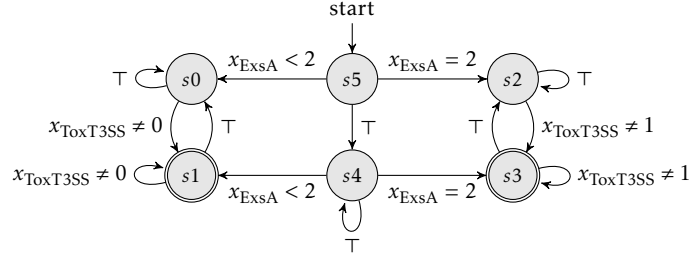


FIGURE 5.5 – $B_{\neg\varphi} = B(\neg\varphi)$ avec $\neg\varphi \equiv \mathbf{F}(\chi_{\text{ExsA}} = 2 \wedge \mathbf{GF} \chi_{\text{ToxT3SS}} \neq 1) \vee \mathbf{F}(\chi_{\text{ExsA}} < 2 \wedge \mathbf{GF} \chi_{\text{ToxT3SS}} \neq 0)$

Le Produit $\Pi_{5.2}$ entre le PGRN de $G_{5.2}$ (figure 5.4) et l'automate de Büchi $B_{\neg\varphi}$ (figure 5.5) contient 12 sommets et 128 transitions (les transitions contenant des gardes avec des disjonctions sont séparées en plusieurs transitions de même source et même cible).

Résultats

Comme nous ne considérons que les états sans toxine comme états initiaux du système (soit $\chi_{\text{ToxT3SS}} = 0$), six arbres d'exécution symbolique (correspondant aux six états possibles) peuvent être construits à partir du Produit $\Pi_{5.2}$.

Notons que les deux premiers arbres d'exécution symbolique construits possèdent respectivement dans leur totalité 22 et 24 nœuds, alors que les autres arbres en possèdent respectivement 13, 3, 3 et 3. Cette diminution de la taille des arbres est représentative de ce que l'on observe sur d'autres exemples. En effet, lorsqu'un arbre est construit, il peut réutiliser l'exploration des arbres précédents, en coupant les branches dont les conditions sur le chemin correspondent à des conditions de chemin acceptantes. Le nombre de nœuds de coupure augmente donc, et la taille de l'arbre diminue (de même que le temps d'exécution).

Les nœuds acceptants de l'ensemble des arbres d'exécution sont au nombre de trois, leurs conditions de chemin sont :

$$\begin{aligned} K_{\text{ExsA},\{\text{ExsA},\text{ExsD}\}} < 2 & \wedge K_{\text{ExsA},\{\}} = 0 \\ K_{\text{ExsA},\{\text{ExsA},\text{ExsD}\}} < 2 & \wedge K_{\text{ExsA},\{\}} > 0 \\ K_{\text{ExsA},\{\text{ExsA},\text{ExsD}\}} = 2 & \wedge K_{\text{ExsA},\{\}} = 2 \end{aligned}$$

La conjonction de la CPBF et des négations des conditions de chemin acceptantes nous

permet d'obtenir une seule solution, correspondant à l'instance de paramètres \mathcal{K} telle que :

- $\mathcal{K}(K_{\text{ExsA},\{\}}) = 1$;
- $\mathcal{K}(K_{\text{ExsA},\{\text{ExsA}\}}) = 2$;
- $\mathcal{K}(K_{\text{ExsA},\{\text{ExsD}\}}) = 0$;
- $\mathcal{K}(K_{\text{ExsA},\{\text{ExsA},\text{ExsD}\}}) = 2$;
- $\mathcal{K}(K_{\text{ExsD},\{\}}) = 0$;
- $\mathcal{K}(K_{\text{ExsD},\{\text{ExsA}\}}) = 1$;
- $\mathcal{K}(K_{\text{ToxT3SS},\{\}}) = 0$;
- $\mathcal{K}(K_{\text{ToxT3SS},\{\text{ExsA}\}}) = 1$.

La dynamique correspondante est présentée en figure 5.6.

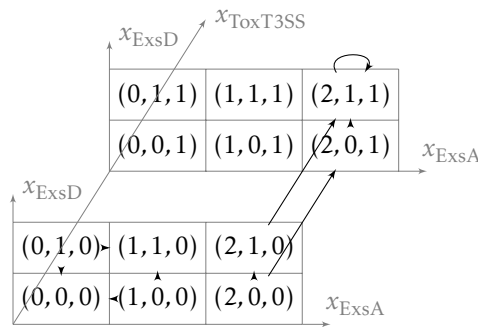


FIGURE 5.6 – Dynamique associée à $\Gamma_{5.2}$ vérifiant φ dans un système initialement exempt de toxines

En appliquant notre méthode sur le graphe d'interactions $\mathbf{G}_{5.3}$ (figure 5.3, 2ème variante du graphe d'interaction du même GRN), nous n'obtenons pas de solution : un seul graphe d'interaction est donc compatible avec la propriété biologique testée. Remarquons que nous divergeons ici du résultat obtenu dans [Fil+06], où une solution au graphe d'interactions $\mathbf{G}_{5.3}$ existe. Ceci est dû à la façon d'exprimer les propriétés biologiques en LTL et au fait que nous étudions la variation de x_{ToxT3SS} et non pas seulement celles de x_{ExsA} et x_{ExsD} . L'instanciation de paramètres proposée dans [Fil+06] peut conduire à un cycle infini d'augmentation/diminution de x_{ExsA} avec possibilité – mais pas obligation – d'augmentation de x_{ToxT3SS} . Insistons sur le fait que cette différence de résultats n'est pas due à l'outil utilisé (SPvTNI κ ou SMBioNet) mais uniquement aux formules LTL données en entrée. En testant les formules de [Fil+06] avec SPvTNI κ , nous retrouvons bien la solution trouvée par SMBioNet.

À l'aide de SPvTNI κ , nous avons donc déterminé qu'une des versions du graphe d'interactions admet une solution satisfaisant l'hypothèse du switch épigénétique du T3SS de la bactérie. L'augmentation (resp. diminution) de ExsA modifie le phénotype de la bactérie d'un état non-cytotoxique (resp. cytotoxique) à un état cytotoxique (resp. non cytotoxique). Ces deux états sont stables. D'autre part, la concentration de l'inhibiteur de ExsA (ExsD) ne semble pas intervenir dans le phénomène. Ces résultats sont conformes à ceux obtenus dans [Fil+06] avec l'outil

SMBioNet (à la différence du résultat obtenu avec le deuxième graphe d'interaction, comme expliqué précédemment).

Ce résultat amène une idée du protocole expérimental à entreprendre pour tester l'hypothèse du switch épigénétique :


1. Souche non inductible (pas ou très peu de production de toxines en déplétion calcique)
2. Pulse de ExsA
3. Test de détection de toxines après déplétion en calcium
4. Acquisition d'un phénotype inductible stable après un certain temps ?

Ainsi, les auteurs de [Fil+06] ont testé les conséquences d'un pulse de ExsA (pour l'amener à saturation, *i.e.* au-dessus du seuil de concentration représenté discrètement par $x_{\text{ExsA}} = 2$) *in vitro* pour une souche de *P. aeruginosa* non inductible et soumise à une déplétion calcique. Ils ont constaté que la souche produisait des toxines sur plusieurs générations (environ dix). Ils supposent donc que ce phénotype acquis est stable, ce qui confirme l'hypothèse épigénétique.

5.1.2 Version étendue

Version simple, selon le principe du rasoir d'Ockham

Le GRN étudié précédemment (figures 5.2 et 5.3) est une version réduite à l'essentiel des phénomènes mis en cause dans l'inductibilité de la cytotoxicité de *P. aeruginosa*. Afin de modéliser plus précisément le phénomène d'inductibilité, deux éléments peuvent être considérés : d'une part un graphe d'interactions plus complet, mettant en jeu d'autres gènes susceptible de modifier le comportement du réseau génétique par la création de complexes de protéines ; d'autre part, la prise en compte de la déplétion calcique.

Pour symboliser la déplétion calcique dans le graphe d'interactions, nous ajoutons un élément supplémentaire noté :  (et $\text{Ca}\setminus$ dans le texte). Pour distinguer ses effets des interactions entre les gènes, nous utilisons la représentation graphique \dashv .

Concernant le graphe d'interactions : ExsD inhibe l'activité de ExsA empêchant la production et la sécrétion des toxines, mais les protéines synthétisées à partir d'un autre gène, ExsC, peuvent elles-mêmes se lier à ExsD, empêchant la formation du complexe ExsD-ExsA qui est à l'origine de l'inhibition de ExsA par ExsD. Or ExsC n'est libre de se fixer à ExsD qu'à la condition qu'elle ne fasse pas elle-même partie d'un complexe avec les protéines d'un autre gène, ExsE. Cependant, pendant la sécrétion, ExsE est exportée de la bactérie à travers le T3SS, libérant ainsi ExsC. Par réaction en chaîne, ExsA est alors libérée de l'inhibition de ExsD et active la production de toxines. Nous ajoutons ainsi les gènes ExsC et ExsE dans le graphe d'interaction et la formation des complexes est représentée via des interactions négatives. Enfin, nous séparons le complexe représentant les gènes producteurs des toxines de ceux codant l'appareil de sécrétion (anciennement ToxT3SS, respectivement séparé en Tox et T3SS). Ces deux complexes sont en effet régulés par ExsA mais a priori à un niveau de concentration différent.

Il reste alors à déterminer les seuils des différentes interactions. Dans notre première approche présentée en figure 5.7, nous avons appliqué le principe du rasoir d’Ockham³ afin d’obtenir le modèle biologique le plus simple possible. Nous avons ainsi fixé à la même valeur les seuils de régulation des gènes se trouvant sur le même opéron⁴, soit :

- $t(\text{ExsA}, \text{ExsA}) = t(\text{ExsA}, \text{ExsC}) = t(\text{ExsA}, \text{ExsE})$;
- $t(\text{ExsA}, \text{T3SS}) = t(\text{ExsA}, \text{ExsD})$.

D’autre part, nous avons limité ExsA à quatre états différents, correspondant directement aux états biologiques suivant :

- *non exprimé*,
- *non inductible*,
- *inductible mais non induit*
- *induit*

avec *induit* correspondant à la valeur maximale que peut atteindre ExsA (soit la valeur du seuil de l’interaction $\text{ExsA} \rightarrow \text{Tox}$).

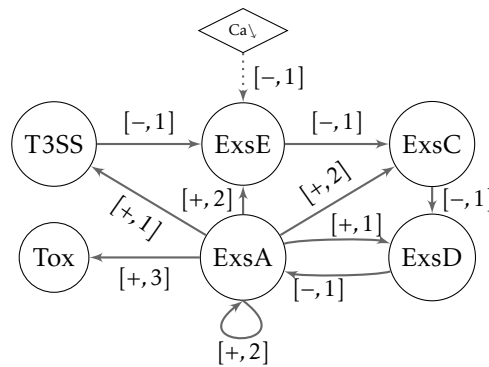


FIGURE 5.7 – Graphe d’interaction du GRN étendu

Après application de la contrainte CPBF il reste 16384 instances de paramètres différentes possibles. Plusieurs connaissances biologiques peuvent être prises en compte pour réduire le nombre d’instances de paramètres.

Nous pouvons utiliser les propriétés biologiques φ_1 et φ_2 déjà illustrées sur le GRN minimal en les adaptant légèrement. En effet, dans ce système, deux niveaux d’expression du gène ExsA représentent les états inductibles :

- si $x_{\text{ExsA}} = 3$ alors les toxines sont produites, *i.e.* le système est dans un état induit,
- si $x_{\text{ExsA}} = 2$ alors le système est dans un état inductible mais non induit.

3. Ce principe est résumé dans la citation latine *Entia non sunt multiplicanda praeter necessitatem*, soit « Les entités ne sont pas multipliées au-delà de ce qui est nécessaire ». Il est très utilisé dans le domaine de la modélisation biologique comme principe de base. Ici cela revient à utiliser un nombre minimum de valeurs de seuil différentes.

4. Un opéron est une unité d’ADN regroupant un ensemble de gènes sous influence du même régulateur, en général transcrits à partir d’un seuil de régulation commun.

Ainsi, les propriétés biologiques φ_1 et φ_2 doivent être réécrites de la manière suivante :

$$\varphi_1 \equiv \mathbf{G}((x_{\text{ExsA}} \geq 2 \wedge \text{Ca} \searrow = 1) \Rightarrow \mathbf{FG} x_{\text{Tox}} = 1), \quad (5.2)$$

$$\varphi_2 \equiv \mathbf{G}(x_{\text{ExsA}} < 2 \Rightarrow \mathbf{FG} x_{\text{Tox}} = 0). \quad (5.3)$$

Nous pouvons ajouter une propriété, φ_3 , représentant l'effet de l'absence de déplétion calcique, définie ainsi (où *depletionCalcium* correspond à un milieu appauvri en calcium, soit $\text{Ca} \searrow = 1$) :

$$\varphi_3 \equiv \mathbf{G}((\text{inductible} \wedge \neg \text{depletionCalcium}) \Rightarrow \mathbf{FG} \neg \text{cytotoxique}) \quad (5.4)$$

Elle se réécrit de la manière suivante :

$$\varphi_3 \equiv \mathbf{G}((x_{\text{ExsA}} \geq 2 \wedge \text{Ca} \searrow = 0) \Rightarrow \mathbf{FG} x_{\text{Tox}} = 0) \quad (5.5)$$

Enfin, nous avons par ailleurs des connaissances biologiques liées à la formation des complexes inhibiteurs, s'appliquant directement aux paramètres pour limiter les valeurs qu'ils peuvent prendre. Si ExsD inhibe ExsA alors il n'y a pas production de toxines donc, dans cette configuration, son niveau d'expression doit être strictement inférieur au seuil de production des toxines, soit :

$$K_{\text{ExsA},\{\text{ExsA},\text{ExsD}\}} \leq 2$$

ExsC et ExsD peuvent également former un complexe, signifiant que l'inhibition de ExsC sur ExsD est plus forte que l'activation de ExsA sur ExsD, soit :

$$K_{\text{ExsD},\{\text{ExsA},\text{ExsC}\}} = 0$$

De la même façon, l'inhibition de ExsE sur ExsC est plus forte que l'activation de ExsA sur ExsC, soit :

$$K_{\text{ExsC},\{\text{ExsA},\text{ExsE}\}} = 0$$

D'autre part, nous savons également que l'effet de $\text{Ca} \searrow$ et de T3SS sur ExsE est conjoint (car ExsE n'est exportée qu'à condition que l'appareil de sécrétion soit formé et sous réserve que le signal d'activation soit donné), aboutissant aux égalités suivantes :

$$K_{\text{ExsE},\{\}} = K_{\text{ExsE},\{\text{Ca} \searrow\}} = K_{\text{ExsE},\{\text{T3SS}\}} \quad (5.6)$$

$$K_{\text{ExsE},\{\text{ExsA}\}} = K_{\text{ExsE},\{\text{Ca} \searrow, \text{ExsA}\}} = K_{\text{ExsE},\{\text{ExsA}, \text{T3SS}\}} \quad (5.7)$$

En donnant en entrée à $\text{SP}_{\text{U}}\text{TNI}_{\text{K}}$ le graphe d'interactions étendu et les trois formules LTL, nous obtenons cinq jeux de paramètres compatibles. En prenant en compte les contraintes CPBF, une seule solution est conservée.

Ainsi, le GRN étendu, qui prend en compte la présence de calcium dans le milieu ainsi que

son appauvrissement de manière plus fine que le GRN réduit, est compatible avec l'hypothèse et les expériences sur le switch épigénétique du T3SS.

Variations avec modifications des seuils des interactions

Les seuils du GRN étendu ont été fixés en considérant que le modèle le plus simple devait être le plus probable. Cependant, en l'absence d'expériences biologiques corroborant ces valeurs, d'autres possibilités peuvent être envisagées. Nous avons ainsi testé avec SPuTNIK cinq autres configurations à partir du graphe d'interaction du GRN étendu, en conservant les égalités des seuils des régulations des gènes appartenant au même opéron et en gardant le seuil $t(\text{ExsA}, \text{Tox})$ maximal (en veillant à adapter les formules LTL en fonction de ce seuil).

Un seul des cinq modèles donne au moins une solution (une solution exactement). Il s'agit du graphe d'interaction dérivé du graphe présenté en figure 5.7 en apportant les modifications suivantes :

$$t(\text{ExsA}, \text{ExsA}) = t(\text{ExsA}, \text{ExsC}) = t(\text{ExsA}, \text{ExsE}) = 1 \quad (5.8)$$

$$t(\text{ExsA}, \text{Tox}) = 2 \quad (5.9)$$

Ce modèle est cependant moins probable que le premier (en figure 5.7) car ExsA, ExsC, ExsE d'une part et ExsD, T3SS d'autre part appartiennent à deux opérons différents, il est donc peu probable que les seuils de ces deux groupes de gènes régulés par ExsA soient identiques.

Variations avec modifications des propriétés LTL

Lors de l'étude du GRN minimal, nous avons vu que nous obtenions des résultats différents de [Fil+06] pour le graphe d'interaction $G_{5.3}$, dus à une traduction différente des propriétés biologiques en LTL. Nous proposons à présent de faire un comparatif en considérant le GRN étendu.

L'adaptation des formules de [Fil+06] au GRN étendu nous donne :

$$\varphi'_1 \equiv \mathbf{G}((\text{inductible} \wedge \text{depletionCalcium}) \Rightarrow \mathbf{XF} \text{induit}) \quad (5.10)$$

$$\varphi'_2 \equiv \mathbf{G}(\neg \text{inductible} \Rightarrow \neg \mathbf{F} \text{inductible}) \quad (5.11)$$

$$\varphi'_3 \equiv \mathbf{G}((\text{inductible} \wedge \neg \text{depletionCalcium}) \Rightarrow \mathbf{XF} \neg \text{induit}) \quad (5.12)$$

avec $\text{induit} \equiv \text{ExsA} = 3$.

Soit (pour le graphe d'interaction en figure 5.7) :

$$\varphi'_1 \equiv \mathbf{G}((x_{\text{ExsA}} \geq 2 \wedge \text{Ca} \leq 1) \Rightarrow \mathbf{XF} \text{ExsA} = 3) \quad (5.13)$$

$$\varphi'_2 \equiv \mathbf{G}(\neg x_{\text{ExsA}} \geq 2 \Rightarrow \neg \mathbf{F} x_{\text{ExsA}} \geq 2) \quad (5.14)$$

$$\varphi'_3 \equiv \mathbf{G}((x_{\text{ExsA}} \geq 2 \wedge \text{Ca} \leq 0) \Rightarrow \mathbf{XF} \text{ExsA} < 3) \quad (5.15)$$

SP_{UTNIK} donne deux solutions avec ces formules, y compris la solution obtenue précédemment avec $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$. Nous avons également testé les graphes d'interaction modifiés présentés ci-dessus : nous obtenons une solution avec le modèle tel que

$$t(\text{ExsA}, \text{ExsA}) = t(\text{ExsA}, \text{ExsC}) = t(\text{ExsA}, \text{ExsE}) = 1 \quad (5.16)$$

$$t(\text{ExsA}, \text{Tox}) = 2 \quad (5.17)$$

et 2 solutions avec un autre modèle (qui n'obtenait pas de solution avec les formules précédentes) obtenu à partir du graphe en figure 5.7 en apportant les modifications suivantes :

$$t(\text{ExsA}, \text{ExsA}) = t(\text{ExsA}, \text{ExsC}) = t(\text{ExsA}, \text{ExsE}) = 1 \quad (5.18)$$

$$t(\text{ExsA}, \text{T3SS}) = t(\text{ExsA}, \text{ExsD}) = 2 \quad (5.19)$$

$$t(\text{ExsA}, \text{Tox}) = 2 \quad (5.20)$$

Une modification légère dans l'interprétation des propriétés biologiques est donc susceptible de valider ou d'invalider un modèle.

5.2 Cycle de vie du phage λ

Dans cette étude de cas, nous nous intéressons au *cycle de vie du phage λ* , un exemple de régulation très étudié en biologie moléculaire depuis les années 50.

Un *bactériophage* (ou plus simplement *phage*) est un virus ciblant des bactéries qui, après avoir pénétré à l'intérieur d'une cellule cible, détourne les mécanismes et les ressources de son hôte à son profit pour se répliquer. Le phage λ cible un genre spécifique de bactéries, *Escherichia coli*. Après avoir pénétré à l'intérieur d'une cellule cible, le phage λ peut avoir deux comportements complètement différents (figure 5.8).

— *Comportement lytique.*

Dans ce cas, le phage λ se réplique rapidement et en grande quantité de manière autonome, puis s'extrait de son hôte en provoquant la destruction de celui-ci par lyse. Les phages libérés peuvent alors infecter d'autres bactéries *Escherichia coli*.

— *Comportement lysogénique.*

Dans ce cas, le phage λ intègre son propre génome dans celui de son hôte, son matériel génétique est ainsi transmis lors des futures divisions cellulaires de la bactérie. Ce cycle est dit *dormant*, il est sans dommage pour l'hôte et protège même son hôte d'une future infection du même virus.

Précisons que le comportement lysogénique, bien que très stable, n'est pas définitif : sous certaines conditions (d'origine environnementale), une bactérie infectée par le phage λ (par infection directe ou par transmission du génome infecté) ayant un comportement lysogénique peut acquérir un comportement lytique.

Pour plus de détails sur les mécanismes du cycle de reproduction du phage λ , le lecteur pourra par exemple consulter les ressources [Pta04; Opp+05].

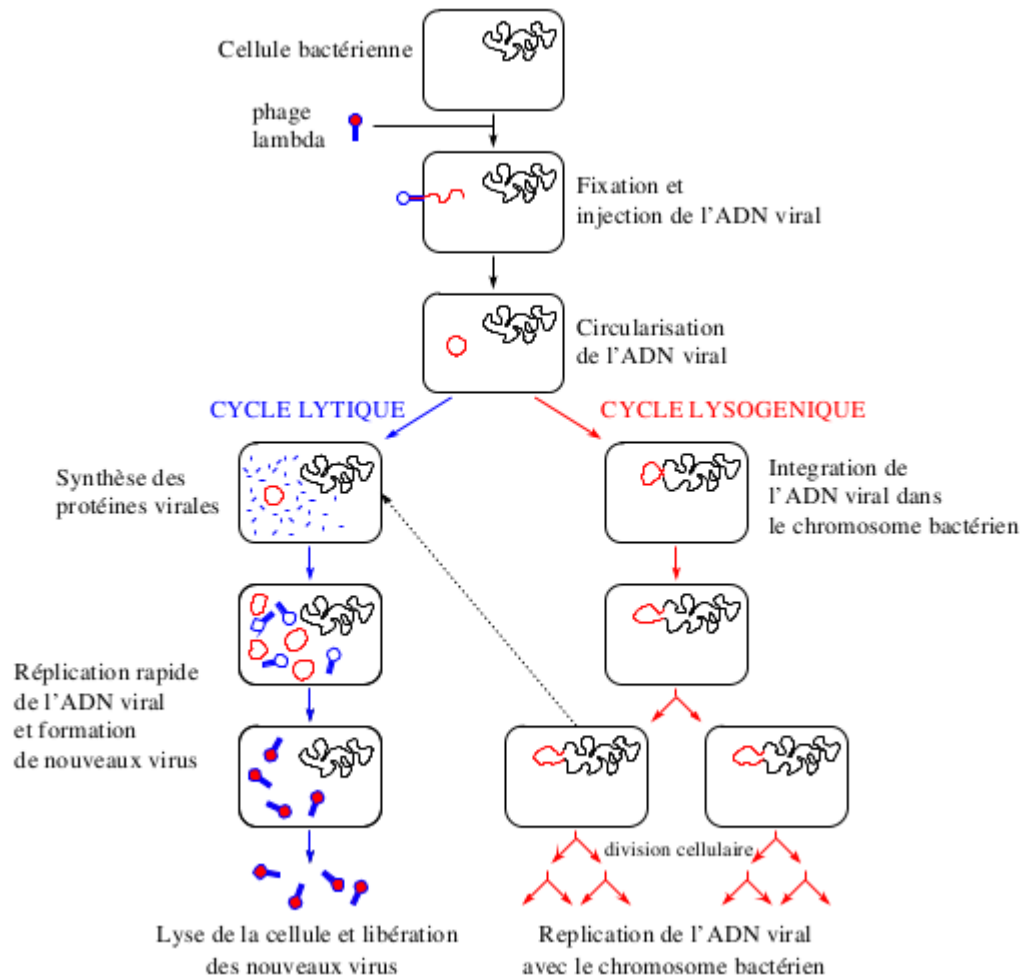


FIGURE 5.8 – Schéma du cycle de vie du phage λ tiré de [Ric06]

Nous nous intéressons au GRN contrôlant le cycle de reproduction du phage λ , i.e. au switch de l'expression génétique permettant l'apparition du comportement lysogénique ou lytique. Dans le cadre du modèle de Thomas, ce GRN a été étudié en particulier par THOMAS et al. eux-même dans nombre de leurs articles pour présenter et tester le modèle de Thomas [Tho71; Tho73; TGL76; Td90; TT95]. Le GRN présenté dans [TT95] a été repris par RICHARD et al. dans [RCB06], dans lequel ils présentent les résultats obtenus avec leur outil SMBioNet en traduisant les propriétés biologiques du système (présence des deux cycles) en CTL. KLARNER et al. ont également utilisé ce GRN pour illustrer leur outil Parsybone, dans l'article [Kla+12a]. Dans cet article, il présente les propriétés biologiques sous forme de *séries temporelles*, i.e. les modèles

solutions sont ceux dont la dynamique passe par deux états complètement déterminés (en terme de niveaux d'expression) caractéristiques des cycles lytique et lysogénique.

Cette étude de cas se fera en deux temps. Tout d'abord, les premières propriétés que nous vérifions sur le système sont une traduction directe du cycle de vie du phage λ ; dans un second temps, nous utilisons les séries temporelles (exprimées en LTL) reprises de [Kla+12a]. Nous utiliserons cependant le même GRN, présenté ci-dessous, commun à [TT95 ; RCB06 ; Kla+12a]. Ce GRN, décrit en figure 5.9, est composé de quatre gènes du phage λ (cI, cII, cro, N) et dix interactions. Le gène cI est le seul gène exprimé en phase lysogénique, il synthétise un répresseur (appelé protéine cI) des autres gènes viraux, empêchant ainsi la survenue du comportement lytique, tout en stimulant sa propre synthèse.

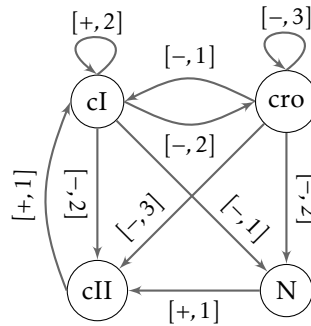


FIGURE 5.9 – Graphe d'interaction Γ_λ du cycle de vie du phage λ .

Ce GRN est lié à 24 paramètres, avec un total de 6879707136 instances de paramètres différentes possibles.

5.2.1 Version 1

Connaissances biologiques et contraintes

Lorsque le phage λ intègre une cellule cible, celle-ci ne contient pas de protéines virales du phage λ car, une fois infectée, soit la cellule est détruite suite au comportement lytique du phage λ , soit le phage λ la protège d'une infection dans le cas d'un comportement lysogénique. On peut ainsi considérer que l'initialisation du système correspond à l'état ⁵ $init = (0, 0, 0, 0)$, ce qui se traduit en terme de contraintes dans $Contraintes(\mathbb{X})$ par :

$$init \equiv (x_{cI} = 0 \wedge x_{cro} = 0 \wedge x_{cII} = 0 \wedge x_N = 0)$$

De plus, le comportement lysogénique est caractérisé par la présence en forte concentration de cI (de manière à s'autoréguler) et l'absence des autres protéines. Cette propriété correspond à l'état $lysogenic = (2, 0, 0, 0)$. Le cycle lytique quant à lui, est caractérisé par une forte

5. Rappelons que les composantes d'un état sont données dans l'ordre alphabétique des noms associés aux gènes, soit ici : $(x_{cI}, x_{cII}, x_{cro}, x_N)$.

concentration en *cro* (qui est un régulateur négatif de *cI*) et d'une absence des autres protéines. Ceci correspond à deux états $lytic_1 = (0, 0, 2, 0)$ et $lytic_2 = (0, 0, 3, 0)$ qui peuvent osciller. Nous pouvons aussi caractériser un état lytique par la condition $x_{cro} \geq 2$. En termes de contraintes dans $Contraintes(\mathbb{X})$ nous obtenons alors :

$$lysogenic \equiv (x_{cI} = 2 \wedge x_{cro} = 0 \wedge x_{cII} = 0 \wedge x_N = 0)$$

$$lytic_1 \equiv (x_{cI} = 0 \wedge x_{cro} = 0 \wedge x_{cII} = 2 \wedge x_N = 0) \quad lytic_2 \equiv (x_{cI} = 0 \wedge x_{cro} = 0 \wedge x_{cII} = 3 \wedge x_N = 0)$$

$$lytic \equiv lytic_1 \vee lytic_2 \equiv (x_{cI} = 0 \wedge x_{cro} \geq 2 \wedge x_{cII} = 0 \wedge x_N = 0)$$

Enfin, les deux types de comportement doivent pouvoir être atteints à partir de l'infection du phage, i.e. à partir de l'état *init*. On suppose par contre que les comportements lytiques et lysogéniques sont irréversibles : on ne prend pas en compte ici les conditions environnementales qui pourraient induire le passage d'un comportement lysogénique à un comportement lytique ; le passage de lytique à lysogénique est par contre de toute façon impossible puisque le comportement lytique mène à la destruction de la cellule. Ces propriétés peuvent se traduire en formules LTL :

- $\varphi_1 \equiv A : \neg(lytic \wedge \mathbf{F}(\neg(lytic)))$;
- $\varphi_2 \equiv A : \neg(lysogenic \wedge \mathbf{F}(\neg(lysogenic)))$;
- $\varphi_3 \equiv E : init \wedge \mathbf{F}(lytic)$;
- $\varphi_4 \equiv E : init \wedge \mathbf{F}(lysogenic)$.

Deux des formules sont à vérifier sur tous les chemins (type « A »), on peut donc former une seule formule à vérifier à partir de φ_1 et φ_2 , soit $\varphi_? = \varphi_1 \wedge \varphi_2$. Nous obtenons ainsi trois formules à vérifier sur le système.

Résultats

La contrainte Min/Max nous permet de fixer 8 paramètres sur les 24, correspondants à :

- $K_{cI,\{cro\}} = 0$;
- $K_{cI,\{cI,cII\}} = 2$;
- $K_{cro,\{\}} = 3$;
- $K_{cro,\{cI,cro\}} = 0$;
- $K_{N,\{\}} = 1$;
- $K_{N,\{cI,cro\}} = 0$;
- $K_{cII,\{cI,cro\}} = 0$;
- $K_{cII,\{N\}} = 1$.

Ceci nous permet de réduire le nombre d'instance de paramètres à 2985984 possibilités.

Les contraintes de définition et d'observation permettent de réduire ce nombre à 216.

Dans le détail, nous obtenons une vingtaine de conditions de chemins en vérifiant la formule $\varphi_?$. Ces contraintes peuvent être simplifiées en :

$$K_{cII}(\{cro\}) = 0 \wedge K_N(\{cro\}) = 0 \wedge K_{cII}(\{\}) = 0 \wedge K_{cro}(\{cro\}) < 3 \wedge K_{cro}(\{cI\}) = 0 \wedge K_{cII}(\{cI\}) = 0 \wedge K_{cI}(\{cI\}) = 2 \wedge K_N(\{cI\}) = 0 \wedge (K_{cI}(\{cII, cro\}) = 2 \vee (K_{cI}(\{cII\}) > 0 \wedge K_{cI}(\{\}) = 2) \vee K_{cI}(\{cII\}) = 2)$$

5.2.2 Version 2 : séries temporelle

Dans [Kla+12a], KLARNER et al. ont décrit les propriétés à vérifier sur le système sous la forme de *séries temporelles*. Les séries temporelles sont une séquence d'état spécifique à atteindre, de la forme $\theta = s_1, *, s_2, *, \dots, *, s_n$ avec s_i le i ème état observé et $*$ désigne une suite, éventuellement vide, d'états non spécifiés.

Les séries temporelles sont équivalentes aux formules LTL de la forme $\phi = s_1 \wedge \mathbf{F}(s_2 \wedge \mathbf{F}(\dots \wedge \mathbf{F}(s_n) \dots))$, composées uniquement d'opérateur \mathbf{F} et de \wedge . De plus, les états des séries temporelles sont complètement déterminés, chaque niveau d'expression est fixé à une valeur particulière. Par exemple, pour G_λ , chaque état est un quadruple $(x_{cI}, x_{cII}, x_{cro}, x_N) \in \{0, 1, 2\} \times \{0, 1\} \times \{0, 1, 2, 3\} \times \{0, 1\}$.

KLARNER et al. distinguent les états suivants :

- $init \equiv (x_{cI} = 0 \wedge x_{cro} = 0 \wedge x_{cII} = 0 \wedge x_N = 0)$;
- $lytic_1 \equiv (x_{cI} = 0 \wedge x_{cro} = 2 \wedge x_{cII} = 0 \wedge x_N = 1)$;
- $lytic_2 \equiv (x_{cI} = 0 \wedge x_{cro} = 2 \wedge x_{cII} = 0 \wedge x_N = 0)$;
- $lytic_3 \equiv (x_{cI} = 0 \wedge x_{cro} = 3 \wedge x_{cII} = 0 \wedge x_N = 0)$;
- $lysogenic_1 \equiv (x_{cI} = 2 \wedge x_{cro} = 1 \wedge x_{cII} = 0 \wedge x_N = 1)$;
- $lysogenic_2 \equiv (x_{cI} = 2 \wedge x_{cro} = 0 \wedge x_{cII} = 0 \wedge x_N = 0)$;

appartenant aux séries temporelles suivantes :

- $\theta_1 = init, *, lytic_1, *, lytic_2, *, lytic_3$
- $\theta_2 = init, *, lysogenic_1, *, lysogenic_2$

Ces deux séries temporelles correspondent à l'évolution en comportement lytique ou lysogénique.

Les équivalents en formules LTL sont :

- $\phi_1 \equiv E : init \wedge \mathbf{F}(lytic_1 \wedge \mathbf{F}(lytic_2 \wedge \mathbf{F}(lytic_3 \wedge \mathbf{F}(lytic_2))))$
- $\phi_2 \equiv E : init \wedge \mathbf{F}(lysogenic_1 \wedge \mathbf{F}(lysogenic_2))$

Une série temporelle exprimant l'atteignabilité de n états peut également être traduite en LTL sous la forme de n formules :

- $\phi_{11} \equiv E : init \wedge \mathbf{F}(lytic_1)$
- $\phi_{12} \equiv E : lytic_1 \wedge \mathbf{F}(lytic_2)$
- $\phi_{13} \equiv E : lytic_2 \wedge \mathbf{F}(lytic_3)$
- $\phi_{14} \equiv E : lytic_3 \wedge \mathbf{F}(lytic_2)$
- $\phi_{21} \equiv E : init \wedge \mathbf{F}(lysogenic_1)$
- $\phi_{22} \equiv E : lysogenic_1 \wedge \mathbf{F}(lysogenic_2)$.

Dans [Kla+12a] KLARNER et al. expriment les propriétés biologiques à l'aide de séries temporelles (*time series*), qui expriment l'existence d'un chemin qui passe successivement par les états mentionnés dans les séries temporelles. C'est pour cela que nous avons traduit ces propriétés en des formules de type $E\phi$, qui expriment que la propriété ϕ est vraie pour un chemin au

moins (et non pour tous les chemins comme pour la plupart des formules précédemment écrites, implicitement de forme $A\varphi$).

Afin de reproduire les mêmes expériences que KLARNER et al. dans [Kla+12a], nous ne prenons pas en compte la contrainte Min/Max pour l'ensemble des gènes (car non supporté dans [Kla+12a]) et nous relâchons la contrainte d'observation pour le cas spécifique où cI est un activateur de lui-même (comme dans [Kla+12a]).

Nous utilisons ensuite SP \cup TNI κ pour déterminer les instances de paramètre correspondantes aux dynamiques dans lesquelles apparaît un phénotype lytique ou lysogénique, en accord avec ϕ_1 et ϕ_2 (i.e. tous les modèles contenant au moins un chemin vérifiant ϕ_1 et au moins un chemin vérifiant ϕ_2).

Résultats

Les résultats obtenus sont en accord : parmi les 7 milliards de modèles possibles, nous obtenons le même nombre (8759) de modèles valides que dans [Kla+12a].

L'outil Parsybone [Str14] est optimisé pour traiter des séries temporelles, il traite ces deux séries en ⁶ 22 s. Chaque série peut être exprimée sous la forme d'une formule LTL avec l'opérateur F imbriqué, i.e. sous la forme $F(s_1 \wedge F(s_2 \dots \wedge F(s_k)))$. Avec SP \cup TNI κ , nous obtenons un temps d'exécution de 3 min 51 s (dont 34 s consacrées à l'énumération des solutions, très lente avec Z3). Parsybone peut également prendre en entrée un automate de Büchi, mais est plus lent dans ce cas. En lui fournissant les deux automates de Büchi correspondant aux deux séries temporelles, il a alors un temps d'exécution de ⁷ 9 min.

Cependant, contrairement à [Kla+12a], notre méthode n'est pas restreinte aux séries temporelles : nous pouvons considérer n'importe quelle forme de formule LTL et il n'y a pas besoin non plus de préciser toutes les composantes d'un état. Par exemple, on sait qu'un phage λ avec un comportement lytique ne peut pas devenir lysogénique (et inversement). Cette propriété ne peut pas être traduite en série temporelle, mais c'est possible en LTL :

- $\phi_3 \equiv \mathbf{G}(lysogenic_2 \Rightarrow \neg \mathbf{F}(lytic_3))$;
- $\phi_4 \equiv \mathbf{G}(lytic_3 \Rightarrow \neg \mathbf{F}(lysogenic_2))$.

En ajoutant ces formules aux précédentes, nous réduisons le nombre de solutions à 2390.

6. exécuté sur leur serveur, dont les caractéristiques sont inconnues (temps d'exécution donné par l'outil)

7. voir note de bas de page précédente

Deuxième partie

Analyse de la voie de signalisation Wnt/ β -caténine à l'aide du model-checking statistique HASL

Plan de la partie II

6	Préliminaires	109
6.1	Éléments de la théorie des probabilités	109
6.2	Les processus stochastiques à événements discrets	112
6.3	Réseaux de Petri stochastiques généralisés	118
6.4	Modélisation de systèmes de réactions biochimiques	123
6.5	Analyse des modèles	127
6.6	Model-checking statistique avec la logique HASL	128
7	Analyse de la voie de signalisation Wnt/β-caténine à l'aide du model-checking statistique HASL	137
7.1	Les voies de signalisation Wnt	138
7.2	Cas d'étude : la voie Wnt/ β -caténine	140
7.3	Construction d'une version stochastique du modèle <i>core</i> de Mazemondet.	142
7.4	Analyse du modèle Wnt <i>core</i>	144
7.5	Conclusion	152

Chapitre 6

Préliminaires

Dans la deuxième partie de ce manuscrit, nous aurons recours au model-checking statistique pour analyser un système biologique particulier : la voie de signalisation Wnt/ β -caténine.

L'objectif de ce chapitre est, dans un premier temps, d'effectuer un rappel rapide des définitions fondamentales dans le cadre de la théorie des probabilités et d'introduire la modélisation stochastiques de réseaux de réactions biochimiques, afin de présenter, dans un deuxième temps, le cadre formel sur lequel repose le model-checker statistique COSMOS.

6.1	Éléments de la théorie des probabilités	109
6.2	Les processus stochastiques à événements discrets	112
6.3	Réseaux de Petri stochastiques généralisés	118
6.4	Modélisation de systèmes de réactions biochimiques	123
6.4.1	Sémantique déterministe.	125
6.4.2	Sémantique stochastique.	125
6.4.3	Relation entre les modèles déterministes et les modèles stochastiques	126
6.5	Analyse des modèles	127
6.6	Model-checking statistique avec la logique HASL	128
6.6.1	1ère composante du HASL : le LHA synchronisé	128
6.6.2	2ème composante du HASL : expression	133
6.6.3	COSMOS	135

6.1 Éléments de la théorie des probabilités

Dans la suite, nous rappelons rapidement certaines notions de base de la théorie des probabilités, notamment la notion de σ -algèbre, d'espace de probabilité, de variable aléatoire et enfin de processus stochastique. Pour de plus amples détails, le lecteur intéressé est invité à consulter la littérature générale sur la théorie des probabilités (par exemple [MA02]).

Le calcul des probabilités s'intéresse à mesurer la possibilité qu'un événement a de se produire au cours d'une expérience aléatoire (i.e. dépendante du hasard). Étant donnée une expérience aléatoire, on note Ω l'univers des possibles (appelé aussi simplement *univers*), qui représente l'ensemble de ses issues (i.e. des résultats possibles de l'expérience).

Notons qu'un univers Ω peut être soit discret (i.e. $\Omega \subseteq \mathbb{Z}$), soit continu (i.e. $\Omega \subseteq \mathbb{R}$).

Un événement associé à un univers Ω est un sous-ensemble $A \subseteq \Omega$. La probabilité des événements d'un univers Ω est formellement définie à travers les notions de σ -algèbre, de mesure de probabilité et d'espace de probabilité données ci-dessous.

Définition 6.1.1 (Sigma-algèbre). Soit Ω un univers.

L'ensemble $\Sigma \subseteq 2^\Omega$ est une σ -algèbre sur Ω s'il satisfait les propriétés suivantes :

- $\Omega \in \Sigma$;
- Σ est stable par passage au complémentaire : i.e. si $A \in \Sigma$ alors $\Omega \setminus A \in \Sigma$;
- Σ est stable par union dénombrable : i.e. si $A_1, A_2, \dots, A_n \in \Sigma$ alors $A_1 \cup A_2 \cup \dots \cup A_n \in \Sigma$.

Exemple 21. Soit $\Omega = \mathbb{N}$ et $\mathbb{N}_{pair} \subseteq \mathbb{N}$ (respectivement $\mathbb{N}_{impair} \subseteq \mathbb{N}$) le sous-ensemble des nombres naturels pairs (respectivement impairs) alors $\Sigma = \{\emptyset, \mathbb{N}_{pair}, \mathbb{N}_{impair}, \mathbb{N}\}$ est une σ -algèbre de Ω .

Définition 6.1.2 (Espace probabilisable). Soit Ω un univers et Σ une σ -algèbre sur Ω .

Le couple (Ω, Σ) est un *espace probabilisable*.

Définition 6.1.3 (Événement). Soit (Ω, Σ) un espace probabilisable.

Tout $X \in \Sigma$ est un *événement*.

Définition 6.1.4 (Mesure de probabilité). Soit (Ω, Σ) un espace probabilisable.

Une fonction $\mathbb{P} : \Sigma \rightarrow [0, 1]$ est une *mesure de probabilité* si elle vérifie les conditions suivantes :

- $\mathbb{P}(\emptyset) = 0$ (\emptyset est dit *événement impossible*) ;
- Pour toute famille dénombrable d'événements deux-à-deux disjoints $A_i \in \Sigma$,

$$\mathbb{P}\left(\sum_{i=1}^{+\infty} A_k\right) = \sum_{i=1}^{+\infty} \mathbb{P}(A_k) ;$$
- $\mathbb{P}(\Omega) = 1$ (Ω est dit *événement certain*).

Exemple 22. Soit $(\Omega, \Sigma) = (\mathbb{N}, \{\emptyset, \mathbb{N}_{pair}, \mathbb{N}_{impair}, \mathbb{N}\})$ un espace probabilisable. Les fonctions $P_1, P_2 : \Sigma \rightarrow [0, 1]$ définies par $P_1(\mathbb{N}_{pair}) = P_1(\mathbb{N}_{impair}) = 0.5$, $P_2(\mathbb{N}_{pair}) = 0.3$, $P_2(\mathbb{N}_{impair}) = 0.7$, $P_1(\emptyset) = P_2(\emptyset) = 0$, $P_1(\mathbb{N}) = P_2(\mathbb{N}) = 1$ sont des mesures de probabilité sur l'espace probabilisable $(\mathbb{N}, \{\emptyset, \mathbb{N}_{pair}, \mathbb{N}_{impair}, \mathbb{N}\})$.

Pour A un événement de Σ , le nombre $\mathbb{P}(A)$ est la *probabilité de l'événement A* .

Définition 6.1.5 (Espace de probabilité). Un espace de probabilité est un triplet $(\Omega, \Sigma, \mathbb{P})$ avec :

- Ω un univers ;
- Σ une σ -algèbre sur Ω ;
- \mathbb{P} une mesure de probabilité sur (Ω, Σ) .

Définition 6.1.6 (Variable aléatoire). Soit $(\Omega, \Sigma, \mathbb{P})$ un espace de probabilité et (S, \mathcal{E}) un espace probabilisable.

Une variable aléatoire X est une application $X : \Omega \rightarrow S$.

L'ensemble S est l'espace des observations. La modélisation aléatoire consiste à interpréter des mesures concrètes comme étant les réalisations d'une variable aléatoire.

Définition 6.1.7 (Loi de probabilité). Soit $(\Omega, \Sigma, \mathbb{P})$ un espace de probabilité et (S, \mathcal{E}) un espace probabilisable. Soit X une variable aléatoire de Ω vers S .

La loi de probabilité de X sur (S, \mathcal{E}) , notée P_X , est l'image de \mathbb{P} par X , soit :

$$\forall A \in \mathcal{E} \quad P_X(A) = \mathbb{P}(X^{-1}(A)) = \mathbb{P}(X \in A)$$

Une variable aléatoire est dite discrète si S est un ensemble discret (par exemple $S \subseteq \mathbb{Z}$), ou continue si S est dense (par exemple $S \subseteq \mathbb{R}$). Dans la suite, on ne s'intéressera qu'aux variables aléatoires continues.

Une variable aléatoire continue X est caractérisée par sa *fonction de répartition*, notée $F_X : S \rightarrow [0, 1] = \mathbb{P}[X \leq x]$, ainsi que par sa *densité de probabilité*, notée $f_X : S \rightarrow [0, 1]$ et pour laquelle on a : $F_X(s) = \int_{-\infty}^s f_X(x) dx$. Étant donné X avec pour *densité de probabilité* f_X , on dit que X a une *distribution* f_X .

Par exemple, une variable aléatoire X a une *distribution exponentielle négative* de paramètre $\lambda \in \mathbb{R}^+$, notée $X \sim \text{Exp}(\lambda)$, si $f_X = \lambda \cdot e^{-\lambda x}$ et donc $F_X(x) = 1 - e^{-\lambda x}$.

Notation 6.1.1. Soit A un ensemble tel que $A \subseteq \mathbb{R}$. On note $dist(A) = \{f : A \rightarrow [0, 1]\}$ l'ensemble des distributions de probabilité sur A , sans restriction sur leur nature.

En connaissant la loi de probabilité de X , on peut déterminer la probabilité de tous les événements décrits par la variable X .

De plus, étant donné un espace de probabilité $(\Omega, \Sigma, \mathbb{P})$, la notion de probabilité d'un événement A conditionné par un événement B est décrite par la définition suivante.

Définition 6.1.8 (Probabilité conditionnelle). Soit $(\Omega, \Sigma, \mathbb{P})$ un espace de probabilité et A, B deux événements de Σ .

La probabilité de A conditionné par B , noté $\mathbb{P}(A | B)$ est définie par :

$$\mathbb{P}(A | B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

6.2 Les processus stochastiques à événements discrets

Dans cette partie de la thèse, nous nous intéressons aux systèmes modélisés par des processus stochastiques à événements discrets.

Un processus stochastique représente l'évolution d'une variable aléatoire en fonction du temps (vu comme discret ou continu), i.e. une famille de variables aléatoires $X(t)$ associées à toutes les valeurs $t \in T$ (avec T ensemble arbitraire représentant le temps, par exemple \mathbb{N} ou \mathbb{R}^+). L'ensemble des observations disponibles $x(t)$ constitue une réalisation du processus.

Définition 6.2.1 (Processus stochastique). Un processus stochastique est une famille de variables aléatoires notée $\{X_t\}_{t \in T}$, définie sur un espace de probabilité $(\Omega, \Sigma, \mathbb{P})$ et un ensemble S , telle que pour tout $t \in T$ on a $X_t : \Omega \rightarrow S$. Nous imposons que S est soit fini, soit infini et dénombrable.

L'indice $t \in T$ se rapporte au temps ; selon la nature de T on parle de processus stochastique à temps discret (si $T = \mathbb{N}$) ou à temps continu (si $T = \mathbb{R}^+$).

On se réfère à l'ensemble S comme étant l'*espace des états* d'un processus $\{X_t\}_{t \in T}$. Dans le contexte de cette partie, nous imposons que S soit fini ou alors infini et dénombrable.

La variable aléatoire X_n d'un processus stochastique représente l'état du système après que le n^e événement se soit produit (avec X_0 l'état initial), tandis que la variable T_n représente le temps écoulé entre la réalisation du n^e et du $(n+1)^e$ événement.

Sachant que nous nous intéressons à la modélisation des réseaux de réactions biochimiques (i.e. des systèmes caractérisés par un nombre fini $N \in \mathbb{N}$ d'espèces chimiques), nous introduisons formellement ci-dessous la notion de *modèle de populations stochastiques*.

Définition 6.2.2 (Modèle de populations stochastiques). Un modèle de populations stochastiques pour N espèces chimiques est un processus stochastique tel que $S \subseteq \mathbb{N}^N$.

Pour caractériser complètement un processus stochastique $\{X_t\}_{t \in T}$, il faut fournir pour chaque séquence d'états $s_0, \dots, s_n \in S$ et chaque séquence d'instant $t_0, \dots, t_n \in T$ (tel que $t_0 < \dots < t_n$), la fonction de répartition conjointe F_X (6.1) qui décrit les interdépendances entre toutes les variables aléatoires X_t du processus :

$$F_X(s_0, \dots, s_n, t_0, \dots, t_n) = \mathbb{P}[X_{t_0} \leq s_0, \dots, X_{t_n} \leq s_n] \quad (6.1)$$

La spécification d'un processus stochastique à travers la fonction (6.1) n'est pas aisée ; en pratique on a recours à des *modèles opérationnels*, ces derniers ayant une caractérisation très simple grâce à certaines sous-classes de processus stochastiques (notamment le processus Markovien). Dans le contexte de cette thèse, nous considérons le *modèle opérationnel* pour la classe très générique de processus stochastiques appelée *processus stochastiques à événements discrets* (abrégée en DESP pour *Discrete Event Stochastic Process*, définition 6.2.3).

Définition 6.2.3 (DESP). Un DESP est un octuplet $\mathcal{D} = \langle S, \pi_0, E, Ind, enabled, delay, choice, target \rangle$ où :

- $S = \{s_0, \dots, s_n, \dots\}$ est un ensemble (potentiellement infini mais dénombrable) d'états ;
- $\pi_0 \in dist(S)$ est la distribution de probabilité initiale sur les états ;
- E est un ensemble fini d'événements ;
- Ind est un ensemble de fonctions de S dans \mathbb{R} appelées indicateurs d'état (incluant la fonction constante) ;
- $enabled : S \rightarrow 2^E$ est une fonction renvoyant les événements franchissables dans chaque état ;
- $delay : S \times E \rightarrow dist(\mathbb{R}^+)$ est une fonction partielle définie pour les paires (s, e) telles que $e \in enabled(s)$ (avec $delay(s, e) = \infty$ si e ne peut pas être franchi dans s , i.e. $e \notin enabled(s)$) ;

- $choice: S \times 2^E \times \mathbb{R}^+ \rightarrow dist(E)$ est une fonction partielle définie pour les tuples (s, E', d) tels que $E' \subseteq enabled(s)$, les sorties possibles de la distribution $dist(E)$ sont restreintes sur E' ;
- $target: S \times E \times \mathbb{R}^+ \rightarrow S$ est une fonction partielle décrivant les changements d'états suite à un déclenchement d'événement, définie pour des tuples (s, e, d) tels que $e \in enabled(s)$.

Pour des raisons de simplicité, nous n'incluons pas ici la définition formelle de la sémantique opérationnelle du DESP. Pour cela, nous renvoyons le lecteur à [Bal+15].

Remarque. Remarquons qu'une transition $target(s, e, d) \in S$ dépend en particulier de l'instant d auquel se produit e . En effet, la définition 6.2.3 vise le cas le plus général des processus stochastiques sans se limiter à des sous-classes particulières, comme celle des processus Markoviens. Ainsi, dans le cas général, les transitions d'un processus stochastique dépendent aussi de la date de réalisation des événements (i.e. le comportement futur dépend du passé, représenté par la date de réalisation d'un événement). Par contre, pour représenter des processus markoviens (qui sont liés à une propriété d'*absence de mémoire* : leur comportement futur ne dépend pas du passé), il suffit que la fonction $target$ soit indépendante du temps.

Les états d'un DESP sont décrits par des *configurations* (définition 6.2.4) qui capturent toutes les informations nécessaires pour déterminer les évolutions futures possibles.

Définition 6.2.4 (Configuration). Une configuration d'un DESP est un triplet $c = (s, \tau, sched)$ avec $s \in S$ un état, $\tau \in \mathbb{R}^+$ un temps et $sched : E \rightarrow \mathbb{R}^+ \cup \{+\infty\}$ une fonction *horaire* associant chaque événement avec un temps, correspondant à la planification de son prochain déclenchement (un temps $+\infty$ signifiant que l'événement n'est pas encore prévu).

On remarque que dans une configuration $c = (s, \tau, sched)$ le *planificateur* $sched$ décrit l'état de la file d'attente des événements dans une configuration c . Ainsi, tous les événements (actuellement) permis $e \in enabled(s)$ ont un temps de déclenchement prévu et fini (i.e. $sched(e) < \infty$), alors que les événements non permis $e \notin enabled(s)$ sont associés à un temps de déclenchement infini (i.e. $sched(e) = \infty$).

Ainsi, dans l'algorithme de génération d'une trajectoire d'un DESP, un planificateur $sched$ donne le temps de déclenchement du prochain événement (cf. algorithme 3).

Notation 6.2.1. Dans la suite, nous noterons $Sched$ l'ensemble des fonctions de planification $sched$ pour les événements du DESP et $Conf = S \times \mathbb{R}^+ \times Sched$ l'ensemble des configurations possibles d'un DESP.

Étant donnée une configuration $c = (s, \tau, sched) \in Conf$, on note $c(s)$, $c(\tau)$ et $c(sched)$, respectivement l'état s , le temps τ et le calendrier $sched$ de la configuration c .

Étant donnée une configuration $c = (s, \tau, sched)$, on note δ_m le temps de déclenchement minimal d'un événement dans $c = (s, \tau, sched)$ et $E_{min} \subseteq enabled(s)$ l'ensemble des événements pouvant se produire dans ce délai minimal. Alors $choice(s, E_{min}, \delta_m)$ décrit comment résoudre au hasard le conflit entre les événements simultanés dans E_{min} : à savoir, le choix $(s, E_{min}, \delta_m)(e)$ est la probabilité que $e \in E_{min}$ sera sélectionné et par conséquent, la probabilité qu'il se produira dans un délai δ_m .

Évolution d'un DESP. L'évolution d'un DESP \mathcal{D} à partir d'une configuration $c = (s, \tau, sched)$ est une procédure itérative constituée des étapes suivantes :

1. Détermination de l'ensemble des événements E_{min} permis dans l'état s et pouvant se produire dans le délai minimal δ_m ;
2. Sélection du prochain événement à se produire $e_{next} \in E_{min}$, en résolvant le conflit entre les événements simultanés à travers le choix probabiliste selon la fonction $choice(s, E_{min}, \delta_m)$;
3. Détermination de la nouvelle configuration du processus résultant du déclenchement de l'événement choisi e_{next} , selon les sous-étapes suivantes :
 - (a) Détermination du nouvel état résultant du déclenchement de e_{next} , i.e. $s = target(s, e_{next}, \tau)$,
 - (b) Avancement du temps pour tenir compte du délai de déclenchement de e_{next} , soit $\tau = \tau + \delta_m$,
 - (c) Mise à jour du planificateur des événements $sched$ en fonction du nouvel état s .

Le fonctionnement de la procédure de détermination de l'évolution d'un DESP est résumé dans l'algorithme 3.

Un chemin (ou trajectoire) d'un DESP est une séquence de configurations $\sigma = c_1, c_2, c_3, \dots$ résultant de l'exécution de la procédure présentée dans l'algorithme 3. Nous formalisons cela dans la définition 6.2.5.

Définition 6.2.5 (Chemin d'un DESP). Pour un DESP $\mathcal{D} = \langle S, \pi_0, E, Ind, enabled, delay, choice, target \rangle$, nous définissons l'ensemble des chemins finis noté $Path^* \subseteq \bigcup_{n \in \mathbb{N}} Conf^n$, avec $Conf$ l'ensemble des configurations de \mathcal{D} . Nous notons $\sigma = (c_0, c_1, \dots, c_n) \in Path^*$, où $\pi_0(c_0(s)) > 0$ et $\forall i, 0 \leq i < N, \exists e \in E_{min}(c_i(s))$ tel que $c_{i+1}(s) = target(c_i(s), e, c_i(\tau))$.

Par extension, on note respectivement $Path^\omega$ l'ensemble des chemins infinis et $Path = Path^* \cup Path^\omega$ l'ensemble des chemins d'un DESP.

Algorithme 3 : Évolution d'un DESP

Données : Configuration initiale : $(s, \tau, sched)$

```

1 tant que  $Enabled(s) \neq \emptyset$  faire
2    $E_{min} = \min(\cup_{e \in enabled(s)} sched(e));$ 
3    $e_{next} = choice(s, E_{min}, \delta_m);$ 
4    $\delta_m = sched(e_{next});$ 
5    $s = target(s, e_{next}, \tau);$ 
6    $\tau = \tau + \delta_m;$ 
7   pour chaque  $e \notin enabled(s)$  faire
8      $sched(e) = +\infty;$ 
9   pour chaque  $e \in enabled(s)$  faire
10     $sched(e) = sample(delay(s, e));$ 

```

Un chemin d'un DESP \mathcal{D} représente en effet un événement (i.e. une réalisation) dans l'espace de probabilité des chemins (i.e. $Path$) de \mathcal{D} . La probabilité d'un chemin de \mathcal{D} est induite par \mathcal{D} même, et sa mesure est définie comme suit :

Définition 6.2.6 (Mesure de probabilité des chemins d'un DESP). Soit $\sigma \in Path^*$, un chemin fini d'un DESP \mathcal{D} , la probabilité de σ est définie inductivement par :

$$\mathbb{P}(\sigma) = \begin{cases} 1 & \text{si } \sigma = c_0 \\ \mathbb{P}(c_0, \dots, c_n) \cdot Pr(c_n, c_{n+1}) & \text{si } \sigma = (c_0, \dots, c_n, c_{n+1}) \end{cases}$$

où $Pr(c_n, c_{n+1})$ est la probabilité d'enchaînement de la configuration $c_n = (s_n, \tau_n, sched_n)$ et de la configuration $c_{n+1} = (s_{n+1}, \tau_{n+1}, sched_{n+1})$, elle est définie comme suit :

$$Pr(c_n, c_{n+1}) = \begin{cases} delay(s_n, e)(\delta_n) & \text{si } E_{min} = \{e\} \\ & \text{et } target(s_n, e, \tau_n) = s_{n+1} \\ & \text{et } sched_{n+1} = sched_n \xrightarrow{e} \\ choice(s_n, E_{min}, \delta_n)(e') \cdot delay(s_n, e')(\delta_n) & \text{si } |E_{min}| > 1 \text{ et } e' \in E_{min} \\ & \text{et } target(s_n, e', \tau_n) = s_{n+1} \\ & \text{et } sched_{n+1} = sched_n \xrightarrow{e'} \\ 0 & \text{autrement} \end{cases}$$

où E_{min} est l'ensemble des événements de délai minimal et actifs dans l'état s_n , $\delta_n = \tau_{n+1} - \tau_n$ et $sched_n \xrightarrow{e}$ correspond à la mise à jour de la fonction de planification $sched_n$ suite au déclenchement de e .

La définition 6.2.6 nous permet de calculer la probabilité des événements dans l'espace de probabilité des chemins d'un DESP. Néanmoins, pour obtenir une caractérisation formelle de l'espace de probabilité des chemins, nous avons besoin de déterminer des partitions de l'univers des chemins d'un DESP. Pour cela, nous introduisons la notion d'*ensemble cylindrique*, qui nous permet de répartir les chemins infinis d'un DESP dans une sigma-algèbre et ainsi d'obtenir la définition formelle de probabilité pour les événements (les chemins) d'un DESP.

Définition 6.2.7 (Ensemble cylindrique d'un chemin fini). Soit $\sigma = (c_0, c_1, \dots, c_n) \in Path^*$ un chemin fini d'un DESP \mathcal{D} .

Notons $Cyl(\sigma) \subseteq Path^\omega$ l'ensemble des chemins infinis dont σ est le préfixe :

$$Cyl(\sigma) = \{\sigma' \in Path^\omega \mid \sigma \text{ est un préfixe de } \sigma'\}$$

La définition de probabilité de l'événement correspondant à l'ensemble des chemins infinis de préfixe σ découle naturellement de la définition de la probabilité d'un chemin fini d'un DESP (définition 6.2.6).

Définition 6.2.8 (Probabilité d'un ensemble cylindrique). Soit $Cyl(\sigma)$ le cylindre associé au chemin $\sigma = (c_0, c_1, \dots, c_n) \in Path^*$ d'un DESP \mathcal{D} .

La probabilité de $Cyl(\sigma)$ est définie par :

$$Pr(Cyl(\sigma)) = \mathbb{P}(\sigma)$$

où \mathbb{P} est donnée par la définition 6.2.6.

Comme nous avons introduit la définition de probabilité d'un événement correspondant à un ensemble cylindrique, nous avons à présent en main tous les éléments nécessaires pour formuler la définition d'espace de probabilité des chemins d'un DESP (définition 6.2.9).

Définition 6.2.9 (Espace de probabilité des chemins d'un DESP). L'espace de probabilité des chemins d'un DESP \mathcal{D} est défini comme un triplet $(\Omega, \Sigma, \mathbb{P})$ tel que :

- $\Omega = Path^\omega(s_0)$ est l'ensemble des chemins infinis de \mathcal{D} commençant dans l'état initial de \mathcal{D} , i.e. s_0 ;
- $\Sigma_{Path(s_0)}$ est la plus petite sigma-algèbre qui contient $Cyl(\sigma)$ pour chaque chemin fini $\sigma \in Path^*(s_0)$;
- $\mathbb{P} : \Sigma_{Path(s_0)} \rightarrow [0, 1]$ est la mesure de probabilité sur les ensembles cylindriques définie comme étant : $\mathbb{P}(Cyl(\sigma)) = Pr(Cyl(\sigma))$ pour chaque $Cyl(\sigma) \in \Sigma_{Path(s_0)}$, où $Pr(Cyl(\sigma))$ est donnée par la définition 6.2.8.

La définition d'espace de probabilité des chemins d'un DESP (définition 6.2.9) nous permet de conclure qu'un DESP est bien un modèle dont les événements sont des ensembles de trajectoires infinies et que la probabilité de chacun de ces événements est mathématiquement bien définie. Dans la suite, nous allons introduire un langage formel permettant une caractérisation sophistiquée des événements d'un DESP : le langage HASL. Nous allons voir qu'il est possible de sélectionner des ensembles particuliers de trajectoires d'un DESP avec HASL et d'estimer leur mesure de probabilité via une procédure statistique, dont nous savons à présent qu'elle est bien définie.

6.3 Réseaux de Petri stochastiques généralisés

La classe de processus DESP est très générique et peu adaptée à une utilisation directe lors de la réalisation de méthodologies d'analyse/vérification, notamment dans les algorithmes de simulation et/ou de vérification. Pour cette raison, nous introduisons dans la suite une classe particulière des réseaux de Petri temporisés et stochastiques, appelée GSPN-DESP (GSPN étant l'abréviation de *Generalised Stochastic Petri Net*, soit *réseau de Petri stochastique généralisé* [Ajm+95]), qui représente un formalisme de haut niveau pour la représentation des DESP.

Un réseau de Petri est un graphe biparti orienté et étiqueté, dont les nœuds sont répartis en *places* et *transitions*. Des jetons, symbolisant les conditions de franchissement et d'affectation des transitions sur des variables du système, circulent de place en place par le biais des transitions. Un GSPN-DESP est un réseau de Petri temporisé, priorisé et dont les transitions sont réparties en deux types : *immédiates* et *temporisées*. Comme leur nom l'indique, les transitions *immédiates* sont déclenchées sans délai dès que leur condition de franchissement est satisfaite. Les transitions *temporisées* sont quant à elles déclenchées après un certain délai (*temps de déclenchement*) à partir du moment où elles sont franchissables ; le délai de franchissement est déterminé aléatoirement suivant une distribution dont la nature et le paramètre sont propres à chaque transition.

Ainsi, contrairement aux GSPN [Ajm+95], la nature des distributions associées aux transitions temporisées d'un GSPN-DESP n'est pas restreinte à la seule loi exponentielle, et de fait elles correspondent à celles décrites par la fonction *delay* d'un DESP (définition 6.2.3).

Un GSPN-DESP est un graphe biparti défini de la façon suivante :

Définition 6.3.1 (GSPN-DESP). Un modèle GSPN-DESP est un tuple $\mathcal{N} = \langle P, T, T^-, T^+, T^{<}, PAR, PRED, M_0, \Pi, W \rangle$ où :

- $P = \{p_1, \dots, p_n\}$ est un ensemble fini et non vide de places ;
- $T = \{t_1, \dots, t_m\}$ est un ensemble fini et non vide de transitions ;
- $T \cap P = \emptyset$;
- $T_i \subseteq T$ et $T_t \subseteq T$ sont respectivement l'ensemble des transitions immédiates et l'ensemble des transitions temporisées, avec $T = T_i \uplus T_t$;
- $T^-, T^+, T^{<} : P \times T \rightarrow \mathbb{N}$ sont respectivement les fonctions d'entrée, de sortie et d'inhibition, représentant les arcs étiquetés entre places et transitions dans le graphe biparti.
 - Si $T^-(p, t) \neq 0$ alors il existe un arc de p vers t étiqueté par $T^-(p, t)$,
 - Si $T^+(p, t) \neq 0$ alors il existe un arc de t vers p étiqueté par $T^+(p, t)$,
 - Si $T^{<}(p, t) \neq 0$ alors il existe un arc de p vers t étiqueté par $T^{<}(p, t)$.
- L'étiquette des arcs donnée par T^- et T^+ indique le nombre de jetons respectivement retirés et ajoutés aux places correspondantes si la transition est déclenchée.
- L'étiquette des arcs donnée par $T^{<}$ indique le nombre de jetons maximum que doit contenir la place pour que la transition correspondante soit déclenchée.
- PAR est un ensemble de paramètres à valeur dans \mathbb{N} ;
- $PRED$ est l'ensemble des prédicats restreignant les domaines de valeurs des paramètres ;
- $M_0 : P \rightarrow \mathbb{N}$ est la fonction de marquage initiale, qui associe à chaque place un entier naturel représentant le nombre de jetons s'y trouvant dans l'état initial ;
- $\Pi : T \rightarrow \mathbb{N}$ associe les transitions à un entier naturel représentant un niveau de priorité. Si dans un marquage donné une transition déclenchable a un niveau de priorité π_t , aucune transition avec une priorité inférieure à π_t n'est déclenchable. Soit $\pi_{max} = \max(\{\Pi(t) | t \in T\})$ le plus haut niveau de priorité des transitions du GSPN. Tous les niveaux de priorité inférieure existent : $\forall t_j \in T \Pi(t_j) > 0 \Rightarrow \exists t_k \in T \Pi(t_k) = \Pi(t_j) - 1$.
- $\forall t_t \in T_t \Pi(t_t) = 0 \wedge \forall t_i \in T_i \Pi(t_i) > 0$;
- $type : T_t \rightarrow dist(\mathbb{R}^+)$ est une fonction associant chaque transition avec le type de la distribution déterminant le délai des transitions temporisées ;
- $par : T_t \rightarrow \mathbb{R}^+$ est une fonction associant chaque transition avec les paramètres de la distribution associée ;

- $W : T \rightarrow \mathbb{R}^+$ est une fonction associant à chaque transition $t \in T$ un réel positif, noté w_t .
- Si $t \in T_i$, w_t est appelé *poinds de t* et il définit la règle de sélection utilisée pour choisir la transition à franchir parmi les transitions franchissables dans un marquage donné.
- Si $t \in T_t$, w_t est appelé *taux de t* et il correspond au taux de la distribution déterminant le délai de franchissement.

Définition 6.3.2 (Marquage). Un état i d'un GSPN-DESP correspond à un marquage $m_i \in \mathbb{N}^{|P|}$, i.e. un vecteur de taille $|P|$. Ce marquage est obtenu grâce à la *fonction de marquage* $M_i : P \rightarrow \mathbb{N} \cup PAR$, qui associe à chaque place $p \in P$ son nombre de jetons disponibles.

Remarque. Pour représenter un GSPN-DESP, nous utiliserons les conventions suivantes :

- les places P sont symbolisées par des cercles ;
- chaque jeton se trouvant dans une place est symbolisé par un point à l'intérieur du cercle de la place ;
- les transitions T sont symbolisées par des rectangles.
- Si la transition est immédiate, le rectangle est plein ; si elle est temporisée, le rectangle est évidé.
- Un arc d'inhibition est représenté par un arc terminé par un cercle.

Notation 6.3.1. Soit une transition $t \in T$.

$\bullet t = \{p \in P | T^-(p, t) \neq 0\}$ et $t^\bullet = \{p \in P | T^+(p, t) \neq 0\}$ représentent les ensembles de places reliées respectivement par un arc en direction de t (dites *places en entrée de t*) et en provenance de t (dites *places en sortie de t*).

Définition 6.3.3 (Transition franchissable). Soit une transition $t \in T$. t est dite franchissable dans le marquage M_i , soit $t \in \mathcal{E}_i$ avec \mathcal{E}_i représentant l'ensemble des transitions franchissables de \mathcal{N} dans l'état i , si et seulement si :

$$\forall p \in t \left(M_i(p) \geq T^-(p, t) \wedge M_i(p) < T^{<}(p, t) \right) \wedge \nexists t' \in \mathcal{E}_i \Pi(t') > \Pi(t)$$

i.e. si dans l'état i le nombre de jetons disponibles dans les places en entrée de t est supérieur

ou égal aux nombres de jetons demandés par la fonction d'entrée et inférieur au seuil d'inhibition, et s'il n'existe aucune transition franchissable de priorité plus élevée.

La stochasticité du GSPN provient de la situation de compétition qui se produit quand au moins deux transitions sont franchissables dans un marquage donné. Dans ce cas, la transition à déclencher est choisie grâce à la fonction W .

Définition 6.3.4 (Franchissement d'une transition). Soit $t \in T$ une transition franchissable dans le marquage M_i , i.e. $t \in \mathcal{E}_i$.

La probabilité que la transition t soit la prochaine transition déclenchée dans le marquage M_i est égale à :

$$P\{t|M_i\} = \frac{w_t}{q_i} \text{ avec } q_i = \sum_{t \in \mathcal{E}_i} w_t$$

Le franchissement de t modifie le marquage M_i en M_j , tel que :

$$\forall p \in \bullet t, M_j(p) = M_i(p) - T^-(p, t) \text{ et } \forall p \in t^\bullet, M_j(p) = M_i(p) + T^+(p, t)$$

Autrement dit, les jetons des places en entrée et en sortie de t sont respectivement diminués et augmentés de la valeur de l'étiquette de l'arc les reliant.

Le franchissement est effectué de manière atomique, i.e. les opérations liées à la modification du marquage sont indivisibles.

La propriété suivante met en relation les modèles GSPN-DESP et les processus stochastiques de type DESP.

Propriété 6.3.1. *Un modèle GSPN-DESP correspond à un DESP.*

Pour démontrer qu'un modèle GSPN-DESP \mathcal{N} est bien un DESP \mathcal{D} , il suffit de montrer les correspondances entre les éléments de \mathcal{D} et ceux de \mathcal{N} :

- L'ensemble des états de \mathcal{D} correspond à l'ensemble des marquages de \mathcal{N} ;
- L'ensemble des événements de \mathcal{D} correspond à l'ensemble des transitions de \mathcal{N} ;
- La distribution des délais de \mathcal{D} (fonction $delay()$) correspond à l'ensemble des distributions des transitions temporelles de \mathcal{N} (i.e. $type()$) ;
- La fonction de probabilité qui décrit le choix probabiliste entre les événements concurrents de \mathcal{D} (fonction $choice()$) est dérivée directement de la fonction des poids de transitions de \mathcal{N} (i.e. la fonction $W()$) ;

- La fonction qui décrit les événements activés dans un état de \mathcal{D} (fonction *enabled()*) est dérivée directement à partir de la règle de franchissement d’une transition dans un état de \mathcal{N} (cf. définition 6.3.3);
- La fonction qui décrit l’état atteignable à partir de la réalisation d’un événement de \mathcal{D} (fonction *enabled()*) est dérivée directement par application de la règle de franchissement d’une transition de \mathcal{N} (cf. définition 6.3.4);
- La distribution initiale π_0 de \mathcal{D} est définie par une *distribution de Dirac* centrée sur le marquage initial M_0 de \mathcal{N} (i.e. $\pi_0(M_0) = 1$).

GSPN-DESP et modélisation de réseaux biologiques. Dans notre contexte de modélisation de réseaux biologiques, les places du réseau de Petri représentent les espèces étudiées (des protéines par exemple) et les jetons contenus dans les places symbolisent la population de chaque espèce dans l’état (instant) courant. Les transitions correspondent quant à elles aux réactions chimiques se produisant entre les espèces, tandis que l’étiquette des arcs reliant les places aux transitions indique le nombre de réactifs consommés et que l’étiquette des arcs reliant les transitions aux places indique les produits générés.

Autrement dit, dans le contexte de la modélisation biologique, une transition d’un GSPN-DESP correspond à une réaction biochimique tandis que les étiquettes des arcs entrants et sortants de la transition correspondent aux *coefficients stœchiométriques* de la réaction.

Exemple 23 (Modèle-jouet d’expression génétique). La figure 6.1 contient un modèle-jouet représentant le processus de synthèse d’une protéine par un gène¹, appelés respectivement protéine A et gène A.

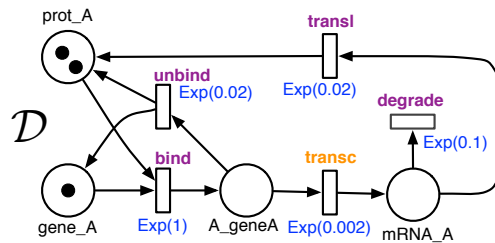


FIGURE 6.1 – GSPN-DESP « jouet » représentant les étapes simplifiées de l’expression génétique

Le réseau GSPN-DESP comporte quatre places :

- *gene_A* représente la « population » de séquences régulatrices² du gène A non occupées;
- *prot_A* représente la population de protéines A libres;
- *A_geneA* représente la population de protéines A fixées sur des séquences régulatrices du gène A (ce qui a pour effet d’enclencher la transcription);

1. Se reporter à la partie I pour la description du processus d’expression génétique, section 2.1 p.24.

2. Les séquences régulatrices sont des régions d’ADN sur lesquelles peuvent se fixer des activateurs du gène considéré, permettant de stimuler la transcription de celui-ci.

- $mRNA_A$ représente la population de molécules d'ARNm transcrites à partir du gène A et qui seront traduites en molécules de $prot_A$.

Notons que la protéine A synthétisée à partir du gène A joue, dans ce modèle, le rôle d'activateur de la synthèse du gène A : il s'agit d'un auto-activateur.

Au niveau de la dynamique, le modèle GSPN-DESP de la figure 6.1 comporte cinq transitions temporisées, toutes de type exponentiel :

- Les transitions $bind$ et $unbind$ représentent respectivement les réactions réversibles de complexation et de décomplexation des protéines A avec les séquences régulatrices du gène A ;
- La transition $transc$ représente l'achèvement de la phase de transcription du gène A , ce qui produit une molécule de $mRNA_A$;
- La transition $degrade$ correspond à la réaction de dégradation des molécules de $mRNA_A$;
- La transition $translate$ représente l'achèvement de la phase de traduction de $mRNA_A$, soit la fin de la production d'une nouvelle molécule de $prot_A$.

Nous reviendrons sur cet exemple dans la section 6.6, lors de la présentation du formalisme HASL pour l'expression et l'analyse de propriétés d'un modèle GSPN-DESP (en particulier dans les exemples 27 et 28).

6.4 Modélisation de systèmes de réactions biochimiques

Dans le contexte de cette thèse, nous représentons formellement un système biologique par un ensemble d'espèces chimiques, notées S_1, \dots, S_n , dont l'évolution est gouvernée par un ensemble d'équations chimiques, notés R_1, \dots, R_m .

Chaque équation chimique a la forme générale suivante :



dans laquelle $j \in \mathbb{N}$ est l'indice de la réaction, a_{ij} et b_{ij} dans \mathbb{N} sont respectivement les coefficients stœchiométriques des *réactifs* et des *produits* et $k_j \in \mathbb{R}^{++}$ est la constante de vitesse de la réaction R_j . Pour une réaction R_j , on appelle *réactifs* (respectivement *produits*) de R_j , les espèces S_i dont le coefficient stœchiométrique a_{ij} (respectivement b_{ij}) est strictement supérieur à 0.

Notation 6.4.1. La concentration d'une espèce S sera notée $[S]$.

Exemple 24. On considère deux espèces S_1 et S_2 présentes dans une cellule de volume fixé V , liées

par les réactions suivantes :



R_1 est une réaction de synthèse, elle représente la création d'une molécule de S_1 . La réaction R_2 est dite de complexation car elle représente la formation du complexe $S_1 S_2$ à partir de la combinaison d'une molécule de S_1 avec une molécule de S_2 . Enfin, la réaction R_3 , dite de décomplexation, est la réaction inverse de R_2 : elle représente la destruction du complexe $S_1 S_2$.

Nous classerons les réactions en fonction de leur *ordre de réaction*. L'*ordre d'une réaction* exprime le lien entre la vitesse de la réaction et la concentration de ses réactants. Pour les réactions élémentaires que nous considérons, l'*ordre d'une réaction* est égal à la somme des coefficients stœchiométriques des réactifs.

Exemple 25. Dans l'exemple 24, R_1 est d'ordre 0 (pas de réactif), R_2 d'ordre 2 (deux réactifs) et R_3 d'ordre 1 (un seul réactif).

On remarque que même si dans la forme générale (6.2) aucune contrainte n'est imposée sur les coefficients stœchiométriques d'une réaction (i.e. a_{ij}, b_{ij}), en pratique on ne considère que très rarement des réactions correspondant à la collision de grandes quantités de molécules (car improbables).

Cinétique des réactions : la loi d'action de masse. Pour pouvoir développer un *modèle de calcul* à partir d'un système d'équations chimiques, il faut associer les réactions à une loi cinétique. Une loi cinétique d'une réaction chimique donne une caractérisation de la vitesse de la réaction en fonction de différents facteurs. En particulier, la *loi d'action de masse* établie que la vitesse d'une réaction du type $R : s_1 S_1 + \dots + s_n S_n \rightarrow \dots$ d'ordre supérieur à 0, est proportionnelle au produit des concentrations de ses réactifs, soit $k[S_1]^{s_1} \dots [S_n]^{s_n}$, avec k le *coefficient de vitesse* de R . Pour une réaction d'ordre 0, la vitesse de la réaction est tout simplement égale au coefficient de vitesse k . Bien que la loi d'action de masse ne soit pas forcément toujours applicable pour bien représenter la dynamique d'une réaction biochimique, dans le contexte de cette thèse nous nous limitons à ce cadre ; les modèles de réactions biochimiques que nous considérons ici sont par défaut supposés être basés sur la loi d'action de masse.

Un système d'équations chimiques (6.2) est une forme générique de représentation d'un système biologique, qui peut donner lieu à différents types d'interprétation. En pratique, il existe deux approches communément utilisées dans la littérature afin d'interpréter un ensemble d'équations chimiques :

- une sémantique *déterministe continue*, à travers laquelle on obtient un modèle d'équations différentielles ordinaires (EDO);
- une sémantique *stochastique discrète*, à travers laquelle on obtient un processus stochastique.

Dans la suite, nous allons présenter succinctement ces deux sémantiques et les relations nous permettant de passer de l'une à l'autre.

6.4.1 Sémantique déterministe.

La sémantique déterministe (i.e. *modélisation déterministe*) d'un système d'équations chimiques s'appuie sur la cinétique chimique classique pour étudier la variation de la concentration des espèces du système, représentée par des variables réelles continues. Elles évoluent de manière déterministe selon un ensemble d'équations différentielles ordinaires (EDO) appelé *équation de vitesse*. La dynamique du modèle est décrite par l'*équation de vitesse* du système.

L'équation de vitesse. *L'équation de vitesse* est l'ensemble des équations différentielles exprimant la variation de la concentration de chaque espèce du système. D'après la loi d'action de masse, la variation de la concentration de chaque espèce est égale à la somme des vitesses des réactions dans lesquelles l'espèce est produite, à laquelle on retranche les vitesses des réactions dans lesquelles elle est consommée.

Exemple 26. *L'équation de vitesse correspondant aux réactions données en exemple 24 sont :*

$$\frac{d[S_1]}{dt} = k_1 + k_3[S_1 S_2] - k_2[S_1][S_2]$$

$$\frac{d[S_2]}{dt} = k_3[S_1 S_2] - k_2[S_1][S_2]$$

avec k_1 , k_2 et k_3 les coefficients de vitesse respectifs des réactions de même indice.

6.4.2 Sémantique stochastique.

Dans le cadre de la sémantique stochastique (i.e. *modélisation stochastique discrète*), les variables des équations chimiques représentent le nombre de molécules des espèces et sont par conséquent discrètes. De plus, l'évolution du système (i.e. les changements d'état des variables) au fil du temps est gouvernée par des lois stochastiques : un système d'équations chimiques est, dans ce cadre, associé à un processus stochastique à événements discrets (modèle de population stochastique, cf. définition 6.2.2).

La nature du processus stochastique dépend de certaines hypothèses concernant les réactions du système considéré. Dans les *conditions d'homogénéité* (température, pression et volume constants), il a été prouvé par Gillespie [Gil77] que le moment d'occurrence d'une réaction d'un système biochimique peut très bien être approché par une distribution exponentielle négative.

Dans ce cas, cela signifie que la dynamique du système peut être représentée par une chaîne de Markov à temps continu (CTMC pour *Continuous Time Markov Chain*), ce sur quoi repose la définition de l'algorithme de simulation stochastique dite SSA (*Stochastic Simulation Algorithm*). Cet algorithme s'appuie sur le calcul itératif de la propension des réactions après chaque occurrence de réaction, basée sur le calcul du nombre de combinaisons distinctes de molécules de réactifs pour chaque équation.

La généralisation de l'algorithme SSA au delà de l'hypothèse d'homogénéité du système nous amène à l'algorithme de simulation d'un processus stochastique « générique » (i.e. DESP), décrit précédemment dans l'algorithme 3.

6.4.3 Relation entre les modèles déterministes et les modèles stochastiques

Dans la littérature, il arrive fréquemment que les modèles biochimiques soient exprimés selon la sémantique déterministe-continue. Dans ce cas, les paramètres d'un modèle correspondent à des *concentrations* (nombre de molécules par unité de volume) et à des *constantes de vitesse* « continues » (exprimées par exemple en nombre de *Moles* par unité de temps). Cependant, si on se place dans l'hypothèse d'homogénéité du système, il existe une relation entre les concentrations *denses* et les populations *discrètes* (en terme de nombre de molécules), ainsi qu'entre les constantes cinétiques continues et leur valeur correspondante dans le cadre stochastique-discret (i.e. nombre de molécules produites ou transformées par unité de temps) [UW11].

Si on note kd_r la *constante de vitesse* « déterministe » d'une réaction chimique r et ks_r la *constante de vitesse* « stochastique » correspondante, la relation entre les deux constantes est définie par la formule suivante :

$$ks_r = \frac{kd_r}{V^{(K_r-1)}} \prod_{j=1}^{L_r} (l_{rj}!)$$

avec K_r la molécularité de la réaction r (pour les réactions simples, elle est égale à l'ordre), L_r le nombre de réactifs de r , l_{rj} le coefficient stœchiométrique de l'espèce j dans r et V le volume dans lequel les espèces sont contenues.

En fonction de l'ordre des réactions, on obtient les égalités suivantes :

- Ordre 0 : $ks_r = kd_r \times V$;
- Ordre 1 : $ks_r = kd_r$;
- Ordre 2 (réactifs d'espèces différentes) : $ks_r = \frac{kd_r}{V}$;
- Ordre 2 (deux molécules de la même espèce en réactifs) : $ks_r = \frac{2kd_r}{V}$.

Dans le chapitre 7, nous présenterons une étude de cas dans laquelle nous avons développé une version stochastique d'un modèle biologique (la voie de signalisation Wnt/ β -caténine) à partir d'un modèle déterministe-continu [Maz+12], en appliquant les règles de conversion décrites ci-dessus.

A travers la correspondance entre les paramètres cinétiques des deux types de modèles, il s'avère qu'il existe un lien entre les modèles déterministes-continus et les modèles stochastiques-

discrets ; en fait, il est même possible de démontrer que les modèles déterministes correspondent à la moyenne des modèles stochastiques, à condition de ne pas prendre en compte les variations dues aux fluctuations causées par des bruits éventuels [Wol+04].

Les modèles stochastiques permettent de prendre en compte les fluctuations liées au bruit interne ou externe au système (par exemple dans [MA99]). Par contre, le modèle déterministe considère les possibilités de collision en continu dans un intervalle de temps infinitésimal. Dans le cas de concentration faible des réactifs des réactions du système, il n'est plus assuré de pouvoir considérer cette continuité ; de plus, le bruit stochastique est moins négligeable [RWA02]. Dans le cas où l'on considère de faibles populations d'espèces, le modèle déterministe peut ainsi sembler moins pertinent que le modèle stochastique.

6.5 Analyse des modèles

La *biologie systémique* [Kit01] est un domaine de recherche fortement interdisciplinaire dont l'objectif est de capturer les comportements importants d'un système biologique à travers des modèles mathématiques, afin de découvrir des propriétés émergentes du système considéré.

L'analyse de la dynamique d'un modèle de populations stochastiques comporte des aspects différents, notamment :

- l'analyse du *comportement transitoire* ;
- l'analyse d'*atteignabilité* ;
- l'analyse du *comportement en équilibre*.

Tous ces types d'analyse peuvent être abordés soit à l'aide de *méthodes numériques* « exactes » (qui existent à la fois pour les CTMC [Ste94; Bai+03a] ainsi que pour les modèles *non-Markoviens* [Hor+12]), soit à l'aide des *méthodes statistiques*, i.e. des méthodes qui s'appuient sur la simulation stochastique [Ros02; LDB10].

Les méthodes numériques pour l'analyse des modèles stochastiques nécessitent de stocker dans la mémoire de l'ordinateur des structures de données proportionnelles à l'espace des états du modèle considéré. Le *model-checking probabiliste numérique*, dont le but est de réaliser le calcul exact de la probabilité de certaines propriétés exprimées en logique temporelle (par exemple en CSL, *Continuous Stochastic Logic* [Bai+03b], ou en l'une de ses extensions : asCSL, *action-state CSL*, qui utilise des expressions rationnelles, ou CSL^{TA} [DHS07], qui se base sur des automates) est ainsi limité par le problème de l'explosion de l'espace des états.

Le *model-checking probabiliste statistique* permet quant à lui ce passage à l'échelle, en réalisant l'estimation de mesures d'intérêt via l'échantillonnage de simulations ; les mesures d'intérêt étant exprimées à l'aide de la logique temporelle (par exemple en CSL - restreint à la sous-logique sans opérateurs de chemin imbriqués - ou en HASL, *Hybrid Automata Specification Logic* [Bal+15]).

Dans cette thèse, nous nous intéressons au *model-checking statistique* utilisant la logique HASL, qui permet d'exprimer et d'estimer des mesures de performance d'un modèle stochastique de type DESP, et donc également d'un GSPN-DESP.

6.6 Model-checking statistique avec la logique HASL

Le model-checking statistique HASL utilise la simulation stochastique pour échantillonner des trajectoires issues d'un DESP et applique des techniques statistiques « classiques » pour obtenir une estimation de l'intervalle de confiance des valeurs des mesures considérées.

Plus précisément, HASL (pour *Hybrid Automata Specification Logic*) [Bal+15] est un formalisme de spécification de propriétés qui permet l'analyse des performances d'un modèle DESP. L'idée de base sur laquelle s'appuie la méthode HASL est celle d'utiliser un automate (hybride) pour sélectionner les trajectoires d'un DESP qui sont conformes à des *caractéristiques dynamiques* particulières (par exemple des traces contenant N occurrences d'un événement e).

Une spécification HASL $\phi = (\mathcal{A}, Z)$ se compose de deux éléments : un automate hybride, noté \mathcal{A} , qui se synchronise avec le DESP tout en sélectionnant les « bonnes » trajectoires, ainsi qu'une expression, notée Z , exprimant la quantité à estimer par rapport aux trajectoires sélectionnées. Dans la suite, nous introduisons formellement les automates hybrides ainsi que la grammaire des expressions Z utilisés dans HASL.

6.6.1 1ère composante du HASL : le LHA synchronisé

Un automate hybride linéaire LHA (pour *Linear Hybrid Automata*) est une restriction des automates hybrides [Hen96] ; il s'agit d'un automate associé à un ensemble X de variables à valeur dans \mathbb{R} , appelées *data variables*, et dont l'évolution est décrite par des vecteurs de flux (qui expriment la valeur de la dérivée première de chaque variable) dans chaque place de l'automate.

Dans le contexte du HASL, un automate LHA est utilisé pour se synchroniser avec les trajectoires issues d'un modèle GSPN. Pour cela, un LHA utilise des *indicateurs* (références) du modèle GSPN-DESP, notamment les noms des événements et les noms des places du modèle GSPN-DESP dont le LHA exprime une propriété. Un automate LHA utilise deux types de transitions : les *transitions synchronisées*, qui sont associées à un ensemble d'événements du GSPN-DESP et sont franchies de manière synchronisée lors du déclenchement d'un événement du GSPN-DESP, ainsi que les *transitions autonomes* qui, à l'inverse, sont franchies dans l'automate sans synchronisation avec le déclenchement d'un événement du GSPN-DESP.

Notation 6.6.1 (Pseudo-événements). Les *pseudo-événements* liés aux transitions autonomes du LHA synchronisé sont notés \sharp .

Le but du LHA synchronisé est à la fois de décider si une exécution du GSPN-DESP est *acceptée* et également de conserver les valeurs des *data variables* tout au long de l'exécution. Les *data variables* sont initialement évaluées à une valeur nulle ; elles évoluent ensuite dans les places de l'automate en fonction de leur taux d'évolution (fonction $flow()$) et également après actualisation, suite au déclenchement d'un événement ou d'un pseudo-événement.

Un LHA synchronisé et défini comme suit (définition 6.6.1) :

Définition 6.6.1 (LHA). Un automate hybride linéaire (LHA) synchronisé est un octuplet $\mathcal{A} = \langle E, L, \Lambda, Init, Final, X, flow, \rightarrow \rangle$ avec :

- E : un alphabet fini d'événements (les événements du GSPN-DESP) ;
 - L : un ensemble fini de places ;
 - $\Lambda : L \rightarrow Prop$: une fonction d'étiquetage des places ;
 - $Init$: un sous-ensemble de L appelé *places initiales* ;
 - $Final$: un sous-ensemble de L appelé *places finales* ;
 - $X = (x_1, \dots, x_n)$: un n -uplet de data variables ;
 - $flow : L \mapsto Ind^n$: une fonction associant à chaque place un indicateur pour chaque data variable représentant le taux d'évolution de la variable dans la place.
- On note $flow_i$ la projection de $flow$ sur sa i ème composante.
- $\rightarrow \subseteq L \times ((Const \times 2^E) \uplus (IConst \times \{\#\})) \times Up \times L$: un ensemble de transitions, où la notation $l \xrightarrow{\gamma, E', U} l'$ signifie que $(l, \gamma, E', U, l') \in \rightarrow$, avec γ la garde de la transition et U la fonction d'actualisation des data variables après franchissement.

De plus, \mathcal{A} remplit les conditions suivantes :

1. **Déterminisme initial** : $\forall l \neq l' \in Init \ \Lambda(l) \wedge \Lambda(l') \Leftrightarrow \perp$;
2. **Déterminisme sur les événements** : $\forall E_1, E_2 \subseteq E$ tels que $E_1 \cap E_2 \neq \emptyset, \forall l, l', l'' \in L$, si $l'' \xrightarrow{\gamma, E_1, U} l$ et $l'' \xrightarrow{\gamma', E_2, U'} l'$ sont deux transitions distinctes, alors :
 - soit $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \perp$,
 - soit $\gamma \wedge \gamma' \Leftrightarrow \perp$;
3. **Déterminisme sur les pseudo-événements** : $\forall l, l', l'' \in L$, si $l'' \xrightarrow{\gamma, \#, U} l$ et $l'' \xrightarrow{\gamma', \#, U'} l'$ sont deux transitions distinctes, alors :
 - soit $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \perp$,
 - soit $\gamma \wedge \gamma' \Leftrightarrow \perp$;
4. **Synchronisation des boucles** : Pour toutes les séquences $l_0 \xrightarrow{\gamma_0, E_0, U_0} l_1 \xrightarrow{\gamma_1, E_1, U_1} \dots \xrightarrow{\gamma_{n-1}, E_{n-1}, U_{n-1}} l_n$ telles que $l_0 = l_n$, alors il existe $i \leq n$ tel que $E_i \neq \#$.

Définition 6.6.2 (Gardes). Une garde d'une transition d'un LHA exprime une contrainte correspondante aux conditions de franchissement de cette transition.

Formellement, une garde est une proposition booléenne de la forme suivante :

$$\sum_{1 \leq i \leq n} \alpha_i x_i + c < 0$$

avec $\alpha_i, c \in \text{Ind}$ et $\in \{=, <, \leq, >, \geq\}$.

Notons que la satisfaction d'une garde porte sur des intervalles de valeurs (i.e. des intervalles de temps car les variables x_i sont des horloges).

Notation 6.6.2. L'ensemble des gardes liées aux événements E seront notées Const . L'ensemble des gardes liées aux pseudo-événements seront notées IConst .

Une garde est dite *semi-fermée à gauche* si, quel que soit l'état du GESP-DSPN, le temps pendant lequel la contrainte est satisfaite est une union d'intervalles semi-fermée à gauche.

Propriété 6.6.1. *Toutes les contraintes dans IConst sont semi-fermées à gauche.*

Définition 6.6.3 (Actualisation des variables). Après franchissement d'une transition, les data variables sont actualisées selon une fonction linéaire sur les valeurs des variables. On note $U = (u_1, \dots, u_n)$ le n -uplet des fonctions d'actualisation où u_k est de la forme $x_k = \sum_{1 \leq i \leq n} \alpha_i x_i + c$, avec $\alpha_i, c \in \text{Ind}$ des indicateurs. L'ensemble des actualisations est noté Up .

Propriété 6.6.2. *Nous considérons des automates déterministes : les trois premières contraintes de la définition 6.6.1 assurent l'unicité de la synchronisation du DESP avec le LHA. Ceci nous assure que le système synchronisé est un processus stochastique.*

Propriété 6.6.3. *La quatrième contrainte de la définition 6.6.1 désactive la divergence de la synchronisation, i.e. la possibilité d'une infinité d'événements autonomes consécutifs sans synchronisation.*

Notation 6.6.3. Une évaluation ν associe chaque data variable à une valeur dans \mathbb{R} . La valeur de la data variable x_i dans ν est notée $\nu(x_i)$.

Notation 6.6.4. Soit une évaluation ν et un état s . Étant donnée une expression $exp = \sum_{1 \leq i \leq n} \alpha_i x_i + c$ liée aux variables et aux indicateurs, son interprétation en fonction de ν et s est définie par : $exp(s, \nu) = \sum_{1 \leq i \leq n} \alpha_i(s) \nu(x_i) + c(s)$.

Notation 6.6.5. Étant donné une actualisation $U = (u_1, \dots, u_n)$, $U(s, \nu)$ représente l'évaluation définie par $U(s, \nu)(x_k) = u_k(s, \nu)$ pour tout $1 \leq k \leq n$.

Notation 6.6.6. Soit $\gamma \equiv \text{exp} < 0$ une contrainte, nous notons $(s, \nu) \models \gamma$ si $\text{exp}(s, \nu) < 0$.

Notation 6.6.7. Soit φ une proposition d'état, nous notons $s \models \varphi$ si $\varphi(s) = \top$.

L'automate se synchronise avec le DESP avec une exécution infinie, jusqu'à atteindre une configuration finale.

Notation 6.6.8. Soit un DESP $\mathcal{D} = \langle S, \pi_0, E, \text{Ind}, \text{enabled}, \text{delay}, \text{choice}, \text{target} \rangle$ et un LHA $\mathcal{A} = \langle E, L, \Lambda, \text{Init}, \text{Final}, X, \text{flow}, \rightarrow \rangle$. Une configuration de \mathcal{A} synchronisé avec \mathcal{D} est un tuple (s_i, l, ν, τ) avec $s \in S$, $l \in L$, ν une évaluation des data variables et $\tau \in \mathbb{R}^+$ un temps.

Il existe deux sortes de configurations pour une synchronisation :

Configurations non finales (s_i, l, ν, τ) avec $l \notin \text{Final}$, $s_i \models \Lambda(l)$ où $\tau (\leq \tau_i)$ est le temps courant ;

Configurations finales (s_i, l, ν, τ) avec $l \in \text{Final}$, $s_i \models \Lambda(l)$ et $\tau \leq \tau_i$ ou la *configuration finale de rejet implicite* \perp .

Soit un chemin $\sigma = s_0 \xrightarrow{e_0, \tau_0} s_1 \xrightarrow{e_1, \tau_1} \dots$ d'un GSPN-DESP \mathcal{D} synchronisé avec un LHA $\mathcal{A} = \langle E, L, \Lambda, \text{Init}, \text{Final}, X, \text{flow}, \rightarrow \rangle$.

Définition 6.6.4 (Départ de la synchronisation entre un DESP et un LHA). Supposons qu'il existe une place $l_0 \in \text{Init}$ telle que $s_0 \models \Lambda(l_0)$. A cause du déterminisme initial, cette place est unique. La configuration initiale est alors : $(s_0, l_0, \nu_0, 0)$ avec l'évaluation initiale $\forall x_i \nu_0(x_i) = 0$.

Si à l'inverse l_0 n'existe pas, alors la synchronisation commence et s'arrête immédiatement en configuration \perp .

Définition 6.6.5 (Configuration entre deux transitions). Soit un chemin $\sigma = s_0 \xrightarrow{e_0, \tau_0} s_1 \xrightarrow{e_1, \tau_1} \dots$ d'un GSPN-DESP \mathcal{D} synchronisé avec un LHA $\mathcal{A} = \langle E, L, \Lambda, \text{Init}, \text{Final}, X, \text{flow}, \rightarrow \rangle$.

Étant donnée une configuration non finale (s_i, l, ν, τ) , un temps δ s'écoule avant qu'une transition (autonome ou synchronisée) soit franchie. Posons $0 \leq \delta \leq \tau_i - \tau$. La nouvelle configuration s'actualise en : $(s_i, l, \nu', \tau + \delta)$ avec $\forall 1 \leq k \leq n \nu'(x_k) = \nu(x_k) + \text{flow}_k(l)(s_i)\delta$.

Définition 6.6.6 (Déclenchements des transitions autonomes et synchronisées du LHA).

Une transition autonome $l \xrightarrow{\gamma, \sharp, U} l'$ est déclenchable après avoir laissé s'écouler δ si $(s_i, \nu') \models \gamma$ et si $s_i \models \Lambda(l')$. Dans ce cas, la transition potentiellement atteinte est $(s_i, l', U(s_i, \nu'), \tau + \delta)$.

Pour chaque transition autonome potentiellement déclenchable, son déclenchement au plus tôt³ est calculé et la transition autonome qui peut se déclencher au plus tôt est sélectionnée pour être franchie. Dû au déterminisme sur \sharp , il y a au plus une transition autonome déclenchable.

Si aucune transition autonome n'est déclenchable avec $\delta \in [0, \tau_i - \tau]$, on fixe $\delta = \tau_i - \tau$ et on considère les transitions synchronisées $l \xrightarrow{\gamma, E', U} l'$ telles que $e_i \in E'(s_i, \nu') \models \gamma$ et $(s_{i+1}, l', U(s_i, \nu'), \tau_i)$. Dû au déterminisme sur les événements, il y a au plus une transition synchronisée déclenchable.

Si aucune transition n'est franchissable, la configuration devient finale (\perp).

Exemple 27 (Automates de mesure utilisés sur un modèle-jouet d'expression génétique). En référence au modèle-jouet d'expression génétique présenté sous la forme d'un GSPN-DESP dans l'exemple 23 p.122, nous illustrons ci-dessous deux exemples d'automates hybrides LHA qui servent à mesurer certaines propriétés concernant la dynamique de la transcription génétique.

Les automates \mathcal{A}_1 (à gauche) et \mathcal{A}_2 (à droite) dans le tableau 6.1 sont destinés à se synchroniser avec le GSPN-DESP de la figure 6.1. \mathcal{A}_1 est utilisé pour faire la sélection des trajectoires d'une durée temporelle bornée et égale à T , tandis que \mathcal{A}_2 réalise la sélection des trajectoires contenant exactement N occurrences de l'événement *transc* (correspondant à la transcription génétique).

Notons que les deux automates \mathcal{A}_1 et \mathcal{A}_2 sont presque identiques, à l'exception de la garde de la transition qui amène à la place acceptante.

Décrivons le fonctionnement des deux automates. Les deux automates utilisent trois variables d'états :

- t est une horloge ;
- n_1 est un compteur des événements *transc* ;
- r enregistre la valeur de la population de la place *prot_A* (i.e. le nombre de protéines *A*).

3. qui existe car les contraintes portent sur des intervalles de temps semi-fermés à gauche

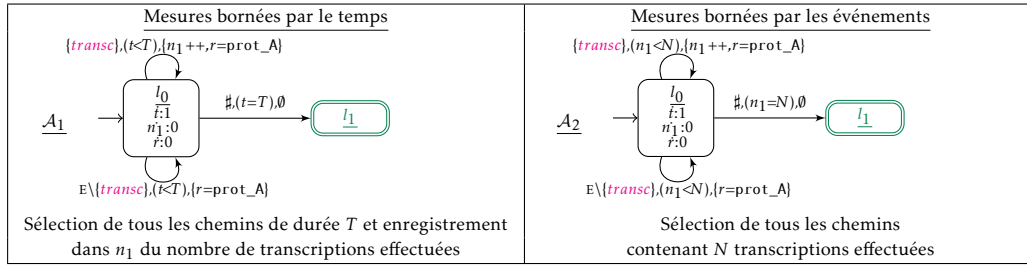


TABLEAU 6.1 – Exemples d’automates LHA pour sélectionner les chemins du modèle GSPN-DESP de la figure 6.1.

Notons que dans la place l_0 , seule la variable t a un taux différent de zéro, car t étant une horloge, elle a forcément un taux unitaire.

Par ailleurs, les deux transitions bouclant sur l_0 sont des transitions synchronisées tandis que la transition $l_0 \rightarrow l_1$ est une transition autonome.

Observons que parmi les transitions bouclant sur l_0 , une seule se synchronise avec le déclenchement de la transition $transc$ du GSPN-DESP ; dans ce cas, la variable n_1 est incrémentée et la variable r est mise à jour avec la valeur courante de la population de $prot_A$. Par contre, l’autre transition bouclant sur l_0 se synchronise avec les occurrences de tous les autres événements différents de $transc$. Dans ce cas, la variable n_1 n’est pas modifiée, mais r est mise à jour avec la valeur courante de $prot_A$.

Enfin, la garde sur la transition $l_0 \rightarrow l_1$ représente la condition d’acceptation d’une trajectoire du GSPN-DESP synchronisée avec l’automate. Pour \mathcal{A}_1 , toutes les trajectoires issues du modèle GSPN-DESP et synchronisées avec \mathcal{A}_1 sont acceptées, dès que la durée $t = T$ est atteinte. Alors que pour \mathcal{A}_2 , ce sont toutes les trajectoires issues du modèle GSPN-DESP et synchronisées avec \mathcal{A}_2 qui sont acceptées, en sachant que chaque trajectoire est acceptée dès que la N^e occurrence de l’événement $transc$ est observée.

Dans la suite (exemple 28), nous allons introduire des exemples d’expressions Z associées aux deux automates \mathcal{A}_1 et \mathcal{A}_2 présentés ici.

6.6.2 2ème composante du HASL : expression

La deuxième composante d’une formule HASL est une expression, notée Z , en lien avec l’automate. Z exprime la quantité à estimer en fonction des trajectoires acceptées par l’automate LHA ; sa grammaire est décrite dans l’équation 6.4.

$$\begin{aligned}
Z &::= E[Y] \mid Z + Z \mid Z \times Z \mid Pdist \\
Pdist &::= PDF(Y, s, l, h) \mid CDF(Y, s, l, h) \mid PROB() \\
Y &::= c \mid Y + Y \mid Y \times Y \mid Y/Y \mid last(y) \mid min(y) \mid max(y) \mid int(y) \mid avg(y) \\
y &::= c \mid x \mid y + y \mid y \times y \mid y/y
\end{aligned}
\tag{6.4}$$

Pour comprendre la sémantique des expressions Z associées à un automate \mathcal{A} , décrivons les éléments de la grammaire 6.4 :

- c représente une constante quelconque ;
- x représente une variable (*data variable*) de l'automate \mathcal{A} ;
- y représente une expression algébrique de variables de l'automate \mathcal{A} ;
- Y représente une expression algébrique construite par application des *opérateurs sur les chemins* (i.e. *path operators*) en particulier par rapport aux opérateurs chemins suivants :
 - $last(y)$: représente la valeur de y à la fin d'un chemin accepté par \mathcal{A} ,
 - $avg(y)$: représente la valeur moyenne de y tout au long d'un chemin accepté \mathcal{A} ,
 - $min(y)$ ($max(y)$) : représente la valeur minimale (maximale) de y tout au long d'un chemin accepté par \mathcal{A} ,
 - $int(y)$: représente la valeur de l'intégrale de y tout au long d'un chemin accepté par \mathcal{A} ;
- $E[Y]$ représente l'espérance de la variable aléatoire Y ;
- $PROB()$ représente une mesure de probabilité correspondant au nombre de chemins acceptés par \mathcal{A} par rapport au nombre total de chemins traités par \mathcal{A} pendant une simulation ;
- $PDF(Y, s, l, h)$ représente l'estimation de la PDF (pour *Probability Density Function*, soit la *densité de probabilité*) de la variable aléatoire Y tout en considérant l'intervalle $[l, h] \subset \mathbb{R}^+$ comme ensemble de support de la PDF et $s < h - l$ comme largeur de la granularité de la discrétisation du $[l, h]$. Pendant l'estimation de $PDF(Y, s, l, h)$, nous supposons que le domaine de Y est bien $[l, h]$, et $[l, h]$ est discrétisé dans $n = \lceil (h - l)/s \rceil$ sous-intervalles chacun de largeur s et, pour chaque sous-intervalle $i \leq n$, nous estimons la proportion de chemins acceptés par \mathcal{A} pour lesquels le valeur Y tombe dans le i^e sous-intervalle de $[l, h]$.
- $CDF(Y, s, l, h)$ représente l'estimation de la CDF (pour *Cumulative Distribution Function*, soit la *fonction de répartition de probabilité*) de la variable aléatoire Y . Même principe que pour $PDF(Y, s, l, h)$.

Exemple 28 (Expression Z pour les LHA). En relation avec les automates LHA \mathcal{A}_1 et \mathcal{A}_2 présentés dans l'exemple 27, nous présentons dans le tableau 6.2 trois exemples de formules HASL complètes, i.e. trois exemples de couples dont le premier élément est un automate LHA et le deuxième est une expression Z construite par application de la grammaire 6.4.

Formule HASL	Description
$\phi_1 \equiv (\mathcal{A}_1, E[\text{last}(n_1)])$	nombre moyen de <i>transcriptions</i> effectuées entre un temps T
$\phi'_1 \equiv (\mathcal{A}_2, E[\max(r) - \min(r)])$	<i>dévi</i> ation moyenne de la population de prot_A lors de N <i>transcriptions</i>
$\phi_2 \equiv (\mathcal{A}_2, PDF(\text{last}(t), s, l, h))$	PDF du délai pour effectuer N <i>transcriptions</i>

TABLEAU 6.2 – Exemples de formules HASL complètes en relation avec le modèle-jouet de l'expression génétique.

6.6.3 COSMOS

Au niveau outillage, la méthode HASL est implémentée dans l'outil COSMOS [Bal+11], un outil de model-checking statistique développé au sein de deux laboratoires de recherche : le Laboratoire en Spécification et Vérification (LSV) de l'École Normale Supérieure de Cachan et le Laboratoire d'Algorithmique Complexité et Logique (LACL) de l'Université de Paris-Est Créteil. L'outil a été développé en langage C et il est disponible en mode *open source* sur le site <http://www.lsv.ens-cachan.fr/software/cosmos/>.

La figure 6.2 montre, schématiquement, la structure de l'outil COSMOS, ses entrées ainsi que ses sorties.

En particulier, COSMOS prend en entrée un modèle GSPN (stocké dans un fichier textuel ayant l'extension .gspn) ainsi qu'un automate LHA (stocké dans un fichier textuel avec l'extension .lha). Le fichier .lha contient la description de l'automate ainsi que l'expression Z associée. De plus, COSMOS demande un certain nombre de paramètres d'entrée, en particulier le *niveau de confiance* ϵ et la largeur de l'intervalle de confiance δ .

En sortie, COSMOS fournit l'intervalle de confiance de la valeur moyenne \bar{Z} (estimée avec la confiance ϵ et avec la largeur $\leq \delta$) de l'expression Z associée à l'automate considéré.

Dans le contexte de cette thèse, nous avons utilisé COSMOS pour développer et analyser le modèle d'un système biologique (modèle de la voie Wnt/ β -caténine) décrit dans le chapitre suivant.

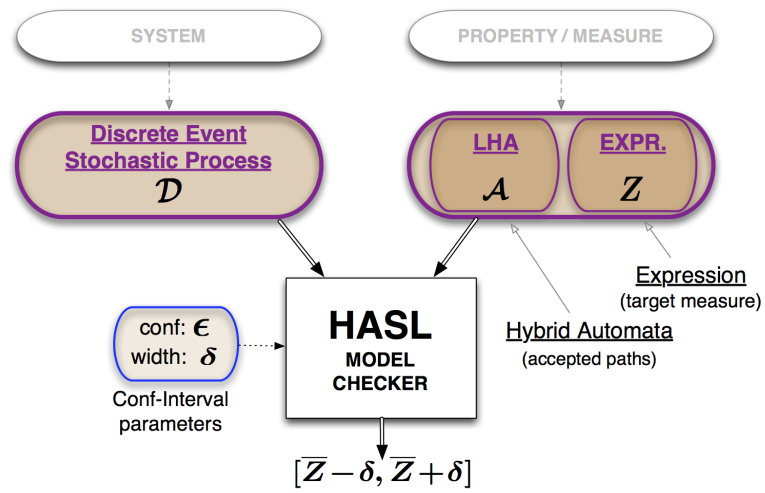


FIGURE 6.2 – Schéma des entrées et sorties de l'outil COSMOS

Analyse de la voie de signalisation Wnt/ β -caténine à l'aide du model-checking statistique HASL

Dans le contexte de la biologie cellulaire, les voies de signalisation sont constituées de chaînes de réactions chimiques qui coordonnent la réponse d'une cellule à des stimuli. Elles sont par exemple en jeu dans les phénomènes tels que l'apoptose, la prolifération ou la différenciation cellulaire [Ich+97 ; Hua+05 ; Che+08]. Afin de simplifier notre explication du contexte biologique, nous considérons ici uniquement le cas particulier qui nous intéresse dans notre étude, i.e. le cas où les stimuli sont externes et où les réactions engendrées sont internes à la cellule.

De manière plus détaillée, l'activation d'une voie de signalisation se fait par l'intermédiaire de la liaison d'une molécule extracellulaire, dite de *signallement* ou *ligand* (par exemple une protéine, une hormone, un neurotransmetteur, ...) avec un récepteur (une protéine) spécifique situé sur la membrane plasmique de la cellule. Le récepteur déclenche alors une cascade d'activation de réactions chimiques internes à la cellule. Ce mécanisme est désigné sous le terme de *transduction du signal*. En présence d'un signal spécifique, la cellule produit une réponse qui peut affecter sa forme ou son comportement, en particulier sa capacité à se reproduire ou l'activité de l'expression génétique.

Comme une même espèce chimique est souvent impliquée dans plusieurs voies de signalisation, il s'avère que les différentes voies de signalisation forment des réseaux de signalisation très complexes. L'enjeu est alors d'analyser le comportement, ou *dynamique*, de ces réseaux. Plus spécifiquement, il s'agit d'identifier aussi précisément que possible les réseaux (sous-réseaux, réseaux simplifiés) avec leurs constituants afin de produire une analyse fine du phénomène étudié, en anticipant les problèmes d'explosion combinatoire, inévitables en toute généralité.

Dans ce chapitre, nous nous intéressons à l'analyse d'une voie de signalisation connue sous le nom de *voie Wnt/ β -caténine*. En particulier, en nous appuyant sur la plateforme de modélisation COSMOS, nous présentons des modèles stochastiques de la voie Wnt/ β -caténine associés à des propriétés formelles exprimées à l'aide du langage de spécification HASL. Notre objectif est d'estimer avec précision, à partir d'un ensemble de mesures particulières, certaines caractéristiques de la dynamique de la voie Wnt/ β -caténine, qui sont notamment en lien avec la propagation du signal le long de cette voie.

7.1	Les voies de signalisation Wnt	138
7.2	Cas d'étude : la voie Wnt/β-caténine	140
7.2.1	Fonctionnement général de la voie Wnt/ β -caténine	140
7.2.2	Modèle <i>core</i> de la voie Wnt/ β -caténine de Mazemondet	141
7.3	Construction d'une version stochastique du modèle <i>core</i> de Mazemondet.	142
7.4	Analyse du modèle Wnt <i>core</i>	144
7.4.1	Observation de la dynamique de β_{nuc} sur une seule trajectoire stochastique	145
7.4.2	Spécification formelle de mesures à travers le langage HASL	146
7.4.3	Mesure formelle des pics du signal β_{nuc} en utilisant l'automate \mathcal{A}_{peaks}	149
7.5	Conclusion	152

7.1 Les voies de signalisation Wnt

Les voies de signalisation Wnt sont, comme leur nom l'indique, des voies activées par des protéines de la famille Wnt. Elles sont principalement connues pour leur rôle dans deux contextes différents : d'une part, la formation de cellules cancéreuses (carcinogénèse), d'autre part, le développement embryonnaire (embryogénèse).

Les différentes voies Wnt forment un réseau de signalisation complexe regroupant plusieurs voies qui se distinguent par leurs constituants ainsi que par leur fonction, i.e. le type de réponse cellulaire qu'elles déclenchent lorsqu'elles sont activées.

Il existe trois voies Wnt principales (cf. figure 7.1, les voies sont présentées de gauche à droite) :

- la *voie Wnt/ β -caténine* (désignée également sous le nom de *voie canonique Wnt*), qui a une action sur la régulation de la transcription génétique ;
- la voie de polarité planaire cellulaire, qui régule le cytosquelette responsable de la forme de la cellule ;
- la voie Wnt/calcium, qui régule le niveau de calcium dans la cellule.

Dans le contexte de cette thèse, nous nous limitons à l'étude de la voie Wnt/ β -caténine. Le travail présenté est issu d'une collaboration avec des neuropédiatres de l'hôpital Robert Debré (UMR 1141, Neuroprotection du cerveau en développement, dirigée par Pierre Gressens), qui ont pour objet d'intérêt la compréhension des dommages neurologiques observés chez les prématurés,

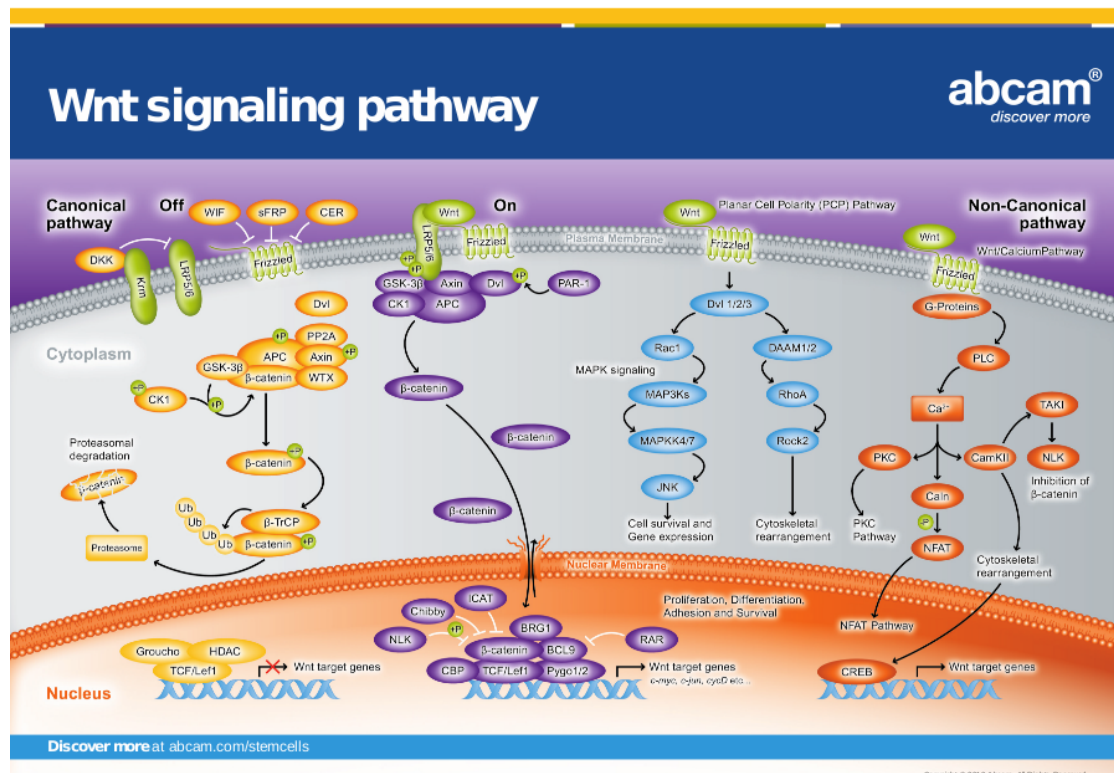


FIGURE 7.1 – Les trois voies Wnt principales : la voie Wnt/ β -caténine canonique (représentée inactivée et activée), la voie de polarité planaire cellulaire et la voie Wnt/calcium. Image provided courtesy of Abcam. Image copyright©2016 Abcam.

dans le but de proposer des thérapies adaptées. Ils s'intéressent en particulier aux microglies (un type de macrophage du cerveau), dont on a observé plusieurs phénotypes différents : un phénotype actif et toxique (pro-inflammatoire), un phénotype actif et protecteur (deux variantes : l'une anti-inflammatoire, l'autre immunorégulatrice) et un phénotype inactif. L'enjeu serait d'arriver à favoriser l'apparition du phénotype protecteur au détriment du phénotype toxique, causant des dommages au cerveau. Or, d'après des expériences réalisées *in vivo* et *in vitro* par l'équipe de Pierre Gressens, il apparaît que le phénotype toxique des microglies serait favorisé quand la voie Wnt est inactivée.

Dans [Maz+12] (dont nous avons utilisé le modèle de la voie Wnt/ β -caténine pour construire notre propre modèle comme nous le verrons dans la suite), la voie Wnt/ β -caténine est étudiée par rapport à son rôle dans la neurodégénérescence, i.e. le processus de perte progressive de la structure et des fonctions des cellules neuronales, qui est à la base de nombreuses maladies neurodégénératives telles que la maladie de Parkinson et la maladie d'Alzheimer. Leur étude porte sur l'influence de la voie Wnt/ β -caténine lors de la différenciation de cellules neuronales

humaines progénitrices¹ (de type ReNcell VM), qui ont la particularité de se différencier en neurones ou en cellules gliales.

7.2 Cas d'étude : la voie Wnt/ β -caténine

Dans la suite, nous allons décrire le fonctionnement général du mécanisme de signalisation de la voie Wnt/ β -caténine. Pour cela, nous devons considérer un certain nombre d'*acteurs* de la signalisation, i.e. des protéines qui sont impliquées dans la propagation du signal Wnt dans la cellule. De manière à éviter une description trop détaillée, on se limite ici à identifier les acteurs principaux, notamment : les protéines de la famille Wnt, leurs récepteurs Fz et co-récepteurs LRP5-6 localisés sur la membrane plasmique, les protéines de la famille Axin et la protéine β -caténine qui est l'actrice principale de la voie Wnt/ β -caténine.

7.2.1 Fonctionnement général de la voie Wnt/ β -caténine

Le comportement global de la voie Wnt/ β -caténine repose sur un mécanisme dont l'effet est de provoquer, ou au contraire d'empêcher, l'accumulation à l'intérieur de la cellule d'une protéine particulière : la protéine β -caténine.

En l'absence du signal Wnt, la voie Wnt/ β -caténine est *désactivée*, ce qui conduit à la destruction progressive de β -caténine causée par la formation d'un *complexe destructeur*. Ce complexe se forme dans le cytosol par complexation de différentes protéines, notamment l'Axin, l'APC (*Adenomatosis Polyposis Coli*) et le GSK3 (*Glycogen Synthase Kinase 3*). Le complexe Axin2/APC/GSK3 est dit *destructeur* car il provoque la dégradation de la β -caténine et empêche ainsi son accumulation dans le *cytosol* (et donc sa relocalisation dans le noyau).

A l'inverse, si la voie Wnt/ β -caténine est activée, i.e. si une protéine Wnt se lie au site extra-cellulaire d'un récepteur de la famille Frizzled (Fz) ou d'un co-récepteur comme par exemple la lipoprotéine liée au récepteur (noté LRP-5/6), la fonctionnalité du *complexe destructeur* Axin/APC/GSK3 est perturbée. En particulier, la présence de protéines Wnt liées aux récepteurs Fz et LRP5-6 provoque tout d'abord la *phosphorylation* des molécules d'Axin (suivie par leur dégradation), puis la diminution du nombre de complexes de destruction Axin/APC/GSK3 (qui n'arrivent plus à se former à cause du manque en Axin), et permet ainsi l'accumulation de β -caténine dans le cytosol. L'accumulation de la β -caténine dans le cytosol amène la relocalisation d'une partie des protéines β -caténine dans le noyau et ceci induit une régulation de la transcription génétique de certains gènes cibles, via les facteurs de transcription de la famille *T-Cell factor/ lymphoid enhancing factor* TCF/LEF. Ceci provoque en particulier la synthèse de nouvelles protéines d'Axin, donnant ainsi lieu à une boucle de rétroaction négative.

Dans la suite, nous allons considérer la version dite du *modèle de Mazemondet* (du nom d'un

1. Les cellules progénitrices sont à l'instar des cellules souches, des cellules qui tendent à se différencier en un type spécialisé de cellule.

de ses auteurs) pour étudier le comportement de la voie Wnt/ β -caténine que nous venons de décrire. Ce modèle a été décrit dans [Maz+12].

7.2.2 Modèle *core* de la voie Wnt/ β -caténine de Mazemondet

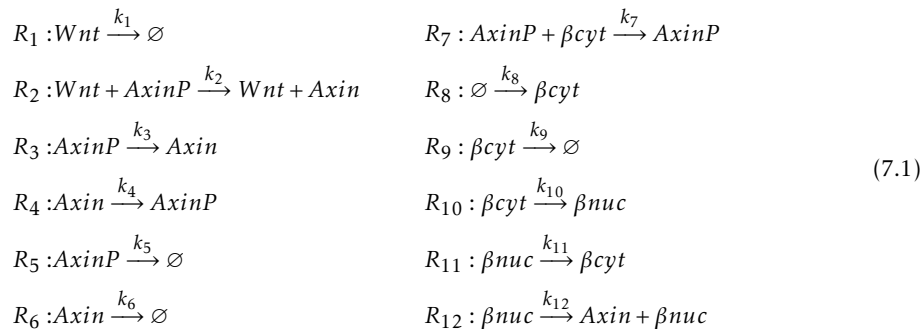
On considère ici le modèle de la voie Wnt/ β -caténine décrit dans [Maz+12]. Il s'agit d'une version abstraite (contenant les composants essentiels) du modèle de Lee [Lee+03].

Le modèle de Mazemondet, aussi appelé *core model*, se compose de trois espèces moléculaires de base :

- La protéine axine, qui peut être phosphorylée (notée $AxinP$) ou non (notée $Axin$);
- La protéine Wnt (Wnt);
- La β -caténine, qui peut être située soit dans le cytosol (β_{cyt}), soit dans le noyau (β_{nuc}) de la cellule.

La dynamique du modèle est décrite par douze réactions chimiques, qui correspondent aux équations (7.1). Les équations (7.1) représentent en particulier les événements suivants :

- Deux événements réversibles, à savoir la phosphorylation de l'Axin (réactions R_4 et R_3) et la relocalisation de la β -caténine de/vers le cytosol/noyau (réactions R_{10} et R_{11});
- La déphosphorylation de l'Axin à l'aide de Wnt (réaction R_2);
- La formation d'Axin régulée par la β -caténine nucléaire (réaction R_{12});
- La dégradation de la β -caténine cytosolique (i.e. β_{cyt}) via l'AxinP (réaction R_7);
- La formation (à taux constant) de la β -caténine cytosolique (réaction R_8);
- La dégradation de toutes les espèces : à savoir Wnt (réaction R_1), Axin sous ses deux formes (réactions R_5 ou R_6) et la β_{cyt} (réaction R_9).



Remarque. La version *core* décrite par (7.1) représente une abstraction du mécanisme de la voie Wnt/ β -caténine que nous avons illustré dans la section 7.2. En particulier, dans le modèle (7.1), nous représentons la dégradation du complexe destructeur (Axin/APC/GSK3) par la dégradation d'une seule de ses composantes : l'Axin. Cette abstraction est raisonnable car l'axine est le facteur limitant de la dégradation du complexe Axin/APC/GSK3 en raison de sa faible quantité par rapport aux deux autres composants (APC et GSK3). De la même façon, nous soulignons que

dans le modèle (7.1), on ne représente pas les réactions de liaison entre les protéines Wnt et leurs récepteurs (Fz et LPR5-6) sur la membrane plasmique.

7.3 Construction d'une version stochastique du modèle *core* de Mazemondet.

À partir des équations (7.1), nous avons développé un réseau de Petri stochastique GSPN (figure 7.2) qui nous permet d'étudier le comportement du modèle Wnt *core* tout en considérant différents modes de stimuli.

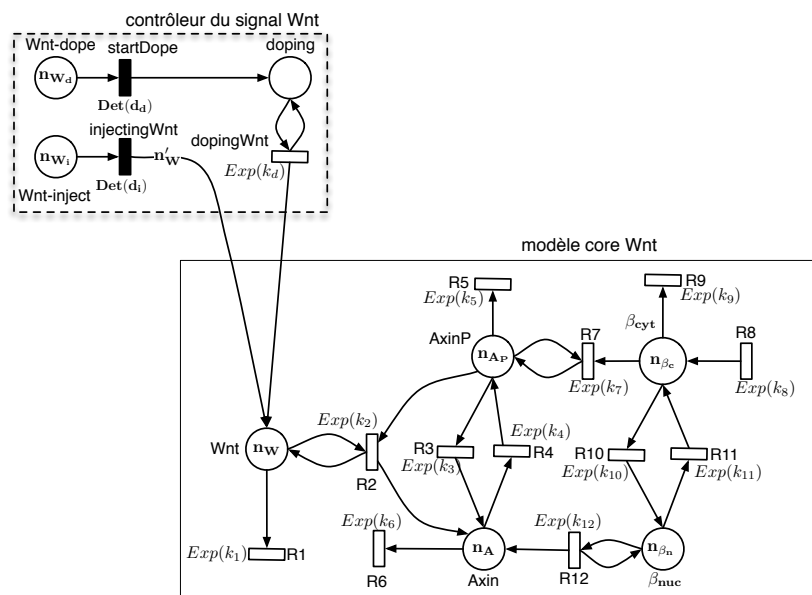


FIGURE 7.2 – Réseau de Petri GSPN dérivé des équations (7.1) de la voie Wnt/ β -caténine.

Le réseau de Petri de la figure 7.2 est constitué de deux parties : le modèle Wnt *core*, qui reproduit le comportement des équations (7.1), et le contrôleur du signal Wnt, qui nous permet de considérer plusieurs types de signal d'entrée ; ainsi, nous pouvons étudier la dynamique du modèle *core* en réponse à des conditions environnementales différentes.

Le réseau *core* est composé de 5 places, correspondant à chacune des espèces chimiques en présence, et de 12 transitions temporisées (appelées R_i avec $i \in \{1, 2, \dots, 12\}$), correspondant aux réactions chimiques décrites dans (7.1).

Chaque transition R_i est associée à une distribution exponentielle ayant pour taux la constante cinétique k_i de la réaction correspondante (se reporter au tableau 7.2 pour avoir les valeurs des constantes k_i). De plus, chaque transition R_i est de type *dépendant du marquage*, ce qui signifie que la distribution effective associée à la transition R_i est de type $Exp(k_i \cdot \prod_{p \in \bullet R_i} M(p))$, i.e. une

distribution exponentielle négative dont le taux est donné par le produit de la constante cinétique k_i et du marquage de chaque place en entrée de la transition R_i ($p \in \bullet R_i$), ce qui correspond bien à l'application de la loi d'*action de masse*.

Le marquage initial de chaque place du réseau du modèle *Wnt core* est noté par un ensemble de paramètres ($n_A, n_{A_p}, n_{\beta_c}, n_{\beta_n}$ et n_W), dont les valeurs sont données dans le tableau 7.1.

Le réseau contrôleur a été conçu pour pouvoir représenter facilement différents types d'expériences que les biologistes peuvent vouloir effectuer, notamment pour représenter des stratégies différentes de stimulation de la voie *Wnt*.

En particulier, le réseau contrôleur de la figure 7.2 est constitué de 3 places, notées *Wnt-dope*, *Wnt-inject*, et *doping* ainsi que de 3 transitions temporisées, notées *StartDope*, *injectWnt* et *dopingWnt*. Les transitions *StartDope* et *injectWnt* sont associées à un délai déterministe, donc associées à des *distributions de Dirac* centrées sur la valeur du paramètre d_i , pour la transition *injectWnt*, et d_d pour la transition *StartDope*. Par contre, la transition *dopingWnt* est associée à un délai exponentiel avec comme paramètre k_d .

Le modèle contrôleur dépend de six paramètres : n_{w_i} , et n_{w_d} qui correspondent au marquage des places respectives *Wnt-inject* et *Wnt-dope*. n'_w indique la multiplicité de l'arc de sortie de la transition *injectWnt* et correspond donc à l'amplitude (en terme de nombres de molécules) du signal *Wnt*, avec lequel le modèle *Wnt core* est stimulé. D'autre part, d_i et d_d représentent les délais au bout desquels soit une *impulsion*, soit une *diffusion constante* de *Wnt* est déclenchée. Enfin, k_d indique l'inverse du délai moyen dans lequel une molécule de *Wnt* est envoyée pour stimuler le modèle *core*.

Pour le bon fonctionnement du contrôleur, les places *Wnt-inject*, et respectivement *Wnt-dope*, doivent être mutuellement exclusives, i.e. leur marquage initial doit satisfaire la contrainte $n_{w_i} + n_{w_d} = 1$. Selon le marquage initial choisi, on obtient un signal *Wnt*, soit de type *impulsion immédiate* ($n_{w_i} = 1$), soit de type *diffusion constante* ($n_{w_d} = 1$). Plus précisément, avec $n_{w_i} = 1$ (donc $n_{w_d} = 0$), la transition *injectWnt* est activée et, avec un délai d_i , ajoute n'_w jetons (molécules) dans la place *Wnt* du réseau *core* (qui est sensé être vide initialement, i.e. $n_w = 0$). D'autre part, si $n_{w_d} = 1$ (donc $n_{w_i} = 0$), la transition *StartDope* sera franchie avec un délai d_d et va donc ajouter un jeton dans la place *doping*, ce qui déclenche la production constante (avec un délai qui suit une loi exponentielle avec un taux k_d) d'un jeton (d'une molécule) dans la place *Wnt* du réseau *core*, ce qui correspond à une possible stratégie d'administration lente et constante d'un médicament.

Configuration du *core model* de Mazemondet. Pour déterminer la configuration initiale du modèle, nous avons utilisé des données tirées de l'article [Maz+12]; celles-ci proviennent à l'origine de la littérature et, en particulier, d'expériences réalisées sur des cellules *oocytes* de la grenouille *Xenopus* publiées dans l'article Lee [Lee+03] ainsi que (en ce qui concerne les taux cinétiques des réactions R_{10} et R_{11}) de données publiées dans [Kri06].

Les tables 7.1 et 7.2 décrivent les valeurs des paramètres concernant, respectivement, les quantités initiales de chaque espèce et les constantes cinétiques de chaque réaction dans (7.1).

Ces valeurs sont données pour les deux formes de modèle dérivant des équations (7.1), i.e. pour le modèle déterministe continu ainsi que pour le modèle stochastique discret.

Les données originales proviennent de celles publiées dans [Lee+03] et [Kri06], où elles sont sous la forme *déterministes et continues*, donc les quantités initiales représentent des *concentrations* exprimées en *nano Moles* (nM) tandis que les constantes cinétiques représentent la *vitesse de transformation* exprimée en concentration par unité de temps. Les valeurs pour le modèle *stochastique discret* sont obtenues à partir des paramètres *déterministes continus*, en appliquant les règles de conversion décrites dans la section 6.4.3 de ce manuscrit (p.126).

	concentrations initiales (nM) [modèle déterministe]		population initiale (molécules) [modèle stochastique]	
	Ensemble A	Ensemble B	Ensemble A	Ensemble B
n β cyt	24.9	24.9	11145	12989
n β _{nuc}	24.9	24.9	4532	5282
nAxin	0.007	0.051	144	252
nAxinP	0.042	0	125	219
nWnt	100	1000	1000	1000

TABLEAU 7.1 – Conditions initiales du modèle Wnt *core* (7.1), version déterministe et version stochastique.

	constantes cinétiques (nM · min ⁻¹) [modèle déterministe]		constantes cinétiques (molécules · min ⁻¹) [modèle stochastique]	
	Ensemble A	Ensemble B	Ensemble A	Ensemble B
k ₁	6.65 · 10 ⁻³	0.27	0.6	0.27
k ₂	7	10	10	20
k ₃	0.3	0.0182	0.03	0.03
k ₄	3	1.619 · 10 ⁻³	0.03	0.03
k ₅	0.167	0.167	4.48 · 10 ⁻³	4.48 · 10 ⁻³
k ₆	0.367	0.0167	2.4 · 10 ⁻³	4.48 · 10 ⁻³
k ₇	4.17 · 10 ⁻⁴	52.5	3 · 10 ⁻⁴	2.1 · 10 ⁻⁴
k ₈	0.232	0.571	420	600
k ₉	9.98 · 10 ⁻⁵	1.13 · 10 ⁻⁴	1.13 · 10 ⁻⁴	1.13 · 10 ⁻⁴
k ₁₀	0.0549	0.0549	0.0549	0.0549
k ₁₁	0.135	0.135	0.135	0.135
k ₁₂	9.3 · 10 ⁻⁵	9.9 · 10 ⁻⁵	2 · 10 ⁻⁴	4 · 10 ⁻⁴

TABLEAU 7.2 – Constantes cinétiques du modèle Wnt *core* (7.1), version déterministe et version stochastique.

7.4 Analyse du modèle Wnt *core*

Avant de passer à l'analyse formelle du modèle Wnt *core* à l'aide de la méthode HASL, nous avons commencé par exécuter des expériences simples d'échantillonnage d'une trajectoire par

simulation du modèle GSPN de la voie Wnt. Ce genre d'observation permet d'avoir une première impression, même si elle n'est que « grossière », du comportement du modèle en fonction des différents jeux de paramètres considérés.

Pour la configuration de cette première série d'expériences, nous nous sommes inspirés des configurations considérées par MAZEMONDET et al. dans [Maz+12]. Ainsi, nous avons échantillonné des trajectoires correspondant à un temps d'observation de 24 heures et nous sommes concentrés sur l'observation de la dynamique de la population de la bêta-caténine dans le noyau (i.e. β_{nuc}).

7.4.1 Observation de la dynamique de β_{nuc} sur une seule trajectoire stochastique

Les figures 7.3 et 7.4 présentent la dynamique de la population de β_{nuc} en considérant différentes conditions initiales ainsi que plusieurs types de stimulation du modèle.

Plus précisément, la figure 7.3 montre la dynamique de β_{nuc} en présence d'un stimulus initial de Wnt (i.e. population initiale de 1000 molécules de Wnt) et en fonction du jeu de paramètres considéré (i.e. **l'ensemble A** ou **l'ensemble B**), correspondant aux différentes *conditions initiales* et *constantes cinétiques* décrites dans les tables 7.1 et 7.2.

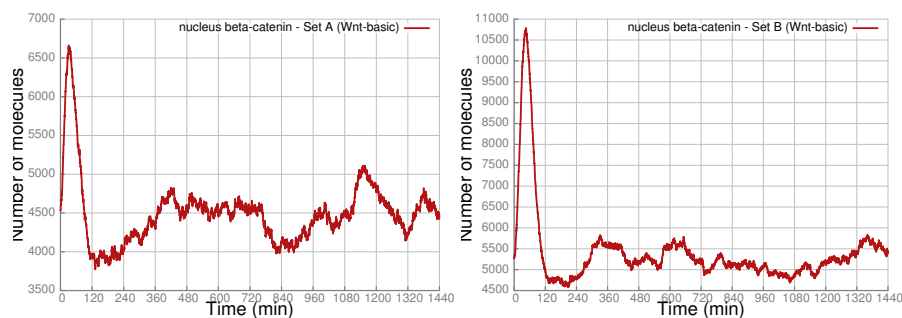


FIGURE 7.3 – Dynamiques de β_{nuc} sur 24h, sur une seule trajectoire stochastique du modèle Wnt *core* avec les paramètres de l'ensemble A (à gauche) et de l'ensemble B (à droite)

Les trajectoires dans la figure 7.3 indiquent qu'une stimulation du modèle à travers un signal initial Wnt induit une réponse *transitoire* (i.e. apparition d'un pic d'une certaine hauteur après un délai) au niveau de la bêta-caténine nucléaire. Les différences entre les deux jeux de paramètres (i.e. **l'ensemble A** et **l'ensemble B**) apparaissent au niveau de l'amplitude du pic de β_{nuc} (la hauteur du pic est beaucoup plus importante avec **l'ensemble B**) mais pas au niveau du délai de réponse.

La figure 7.4 montre des trajectoires issues de différents types de stimulation initiales. En particulier, la figure 7.4 à gauche représente la trajectoire en réponse à une stimulation intermittente avec une réinjection du signal Wnt après un délai de 150 min par rapport au signal

initial. Cette trajectoire indique qu'une deuxième stimulation avec un nouveau signal Wnt en entrée induit un deuxième pic transitoire de la bêta-caténine nucléaire.

Notons que le maximum du deuxième pic se produit à environ $t = 480$ minutes ; ceci signifie que le délai d'apparition du deuxième pic, dans cette trajectoire particulière, est décalé d'environ 30 minutes par rapport à l'instant de réinjection de Wnt (qui a lieu à $t = 450$ minutes). De plus, son amplitude est mineure comparé à l'amplitude du premier pic ; ceci semble suggérer que la réactivité du système à des signaux Wnt répétés et de même intensité diminue après le premier signal (comme on le verra dans la suite, cette intuition est bien confirmée par les mesures *formelles* effectuées à l'aide de propriétés HASL).

La trajectoire à droite de la figure 7.4 montre, quant à elle, la réponse de la bêta-caténine au stimuli de type *dopé*, ce qui correspond à la configuration du modèle de la figure 7.2 avec un marquage initiale $n_{W_d} = 1$ pour la place Wnt-dope, de sorte que la transition temporisée dopingWnt (dont la distribution est supposée être exponentielle de paramètre k_d) soit toujours activée et donc que la population Wnt (place Wnt dans la GSPN) reçoive en moyenne une nouvelle molécule (jeton) chaque $\frac{1}{k_d}$ unité de temps.

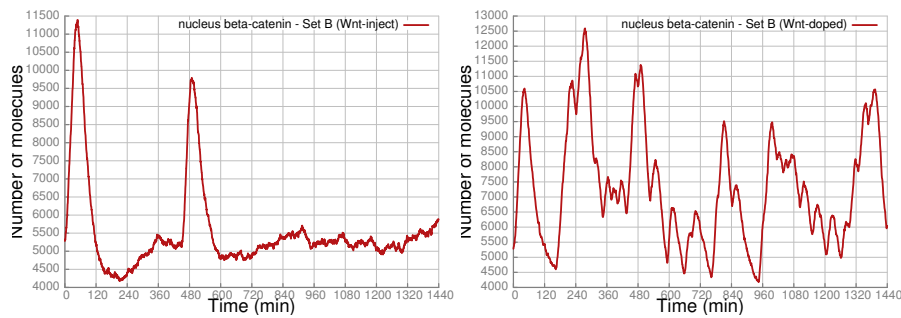


FIGURE 7.4 – Dynamiques de β_{nuc} sur une seule trajectoire stochastique du modèle Wnt *core* avec réinjection de Wnt à $t = 450$ minutes (à gauche) et avec le dopage de Wnt commencé à $t = 150$ minutes (à droite), en considérant les paramètres de l'ensemble B

Cette trajectoire montre un profil oscillatoire de la bêta-caténine nucléaire, avec des oscillations très irrégulières et affectées par un *bruit* important. Ce comportement oscillatoire est intuitivement raisonnable, sachant que dans le modèle *dopé*, le signal Wnt en entrée ne disparaît jamais complètement : il diminue à cause de la dégradation normale de Wnt, puis augmente grâce à la présence de la réaction de *doping* dopingWnt . L'irrégularité du signal observé (i.e. β_{nuc}) est donc en fait un effet de l'irrégularité du signal d'entrée Wnt.

7.4.2 Spécification formelle de mesures à travers le langage HASL

Dans cette partie, nous introduisons à travers le langage HASL les spécifications formelles qui nous permettent de mesurer avec une *confiance* arbitraire les caractéristiques principales de la dynamique du modèle de la voie Wnt, i.e. la présence de pics transitoires dans le signal de β_{nuc} , ainsi que leur hauteur et leur délai d'apparition. Plus précisément, nous allons présenter

un automate hybride, que nous noterons \mathcal{A}_{peaks} , conçu pour la détection de pics d'une espèce observée quelconque.

Cet automate permet de calculer la hauteur ainsi que le délai d'un pic détecté sur une trajectoire quelconque issue du modèle considéré. La hauteur et le délai des pics mesurés sur chaque trajectoire seront utilisés, en appliquant la méthode de model-checking statistique HASL, pour estimer l'intervalle de confiance de la valeur moyenne de la hauteur et du délai des pics de β_{nuc} .

L'automate hybride \mathcal{A}_{peaks}

La figure 7.5 présente l'automate \mathcal{A}_{peaks} , qui est un automate hybride de type LHA conçu pour la détection des pics maximums et minimums dans les traces analysées. L'automate \mathcal{A}_{peaks} utilise plusieurs variables de différents types, qui sont décrites dans la table 7.3.

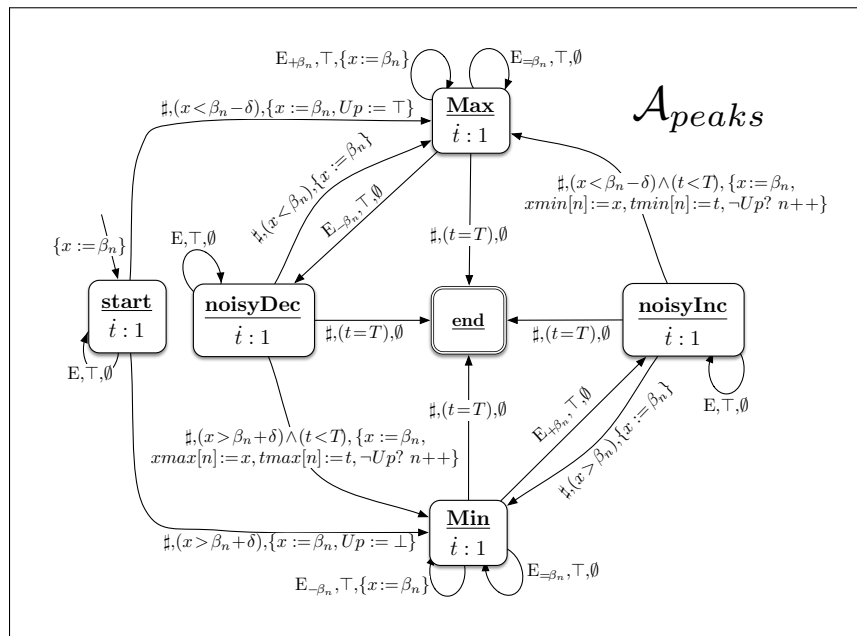


FIGURE 7.5 – $\mathcal{A}_{\beta_{peaks}}$: un LHA permettant de localiser les pics maximums et minimums de β_n (en considérant le niveau de bruit δ).

L'automate réalise la détection des maximums (ainsi que des minimums) locaux d'une espèce observée (dans notre cas la population de β_{nuc}) en se basant sur le principe suivant : une valeur est détectée comme un maximum (minimum) local si c'est un maximum local et si sa distance par rapport au minimum (maximum) qui la précède dépasse un seuil donné, noté δ , choisi par l'utilisateur. Le seuil δ sert à éliminer de la détection les pics correspondants à du bruit stochastique, i.e. des pics associés à de petites oscillations dues à la stochasticité du modèle et qui ne correspondent pas à de « vrais » pics maximaux (minimaux) de la dynamique de l'espèce

Variables de l'automate hybride \mathcal{A}_{peaks}		
nom	domaine	description
t	$\mathbb{R}_{\geq 0}$	temps écoulé depuis le début de la simulation
n	\mathbb{N}	nombre de pics (max ou min) détectés
up	bool	variable booléenne indiquant si le premier pic détecté est un max (<code>true</code>) ou un min (<code>false</code>)
x	\mathbb{N}	hauteur du dernier pic maximal (ou minimal) détecté
$xmax(xmin)$	$\mathbb{N}^{\mathbb{N}}$	tableau des hauteurs des pic max (min) détectés
$tmax(tmin)$	$\mathbb{R}^{\mathbb{N}}$	tableau des délais des pic max (min) détectés

TABLEAU 7.3 – Les variables de l'automate \mathcal{A}_{peaks} (figure 7.5) utilisées pour l'identification et la mesure des pics maximums et minimums dans une trace

observée.

Pour cette raison, l'automate \mathcal{A}_{peaks} utilise un paramètre (noté δ dans la figure 7.5) correspondant au seuil du bruit à ignorer. De plus, étant donné l'espèce observée (dans notre cas β_{nuc}), l'automate nécessite aussi que l'ensemble des événements E (i.e. les transitions) du modèle GSPN considéré soit partitionné dans trois sous-ensembles $E = E_{+\beta_{nuc}} \cup E_{-\beta_{nuc}} \cup E_{=\beta_{nuc}}$, où $E_{+\beta_{nuc}}$, $E_{-\beta_{nuc}}$ et $E_{=\beta_{nuc}}$ représentent respectivement les ensembles de réactions qui augmentent, diminuent ou n'ont pas d'effet sur la population de β_{nuc} . Par conséquent, par rapport au modèle GSPN de la figure 7.2, on obtient la partition d'événements suivante :

$$E_{+\beta_{nuc}} = \{R_{10}\}$$

$$E_{-\beta_{nuc}} = \{R_{11}\}$$

$$E_{=\beta_{nuc}} = \{R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{12}\}$$

Fonctionnement de l'automate \mathcal{A}_{peaks} . L'automate \mathcal{A}_{peaks} comporte une place initiale **start**, une place finale **end**, et quatre places *intermédiaires* **Min**, **noisyInc**, **Max** et **noisyDec**.

Au début de la synchronisation, tout en rentrant dans la place **start**, l'automate valorise la variable x avec la valeur de la population β_{nuc} au début de la simulation. À partir de **start**, l'analyse de la trajectoire simulée amène l'automate à rentrer soit dans la place **Min**, si la trajectoire simulée présente une diminution de β_{nuc} au-delà du seuil du bruit δ (par rapport à la valeur initiale stockée dans x), soit dans la place **Max**, si la trajectoire montre une augmentation de β_{nuc} au-delà de δ .

Une fois dans la place **Min** (respectivement **Max**), le comportement de l'automate dépend des événements observés. Si un événement $e \in E_{+\beta_{nuc}}$ (respectivement $e \in E_{-\beta_{nuc}}$) est observé, alors l'automate rentre dans la place **noisyInc** (respectivement **noisyDec**), ce qui signifie que la population de β_{nuc} a augmenté (respectivement diminué) bien que l'augmentation (respectivement la diminution) n'ait pas dépassée le seuil de bruit δ . Par contre, si pendant le stationnement dans la

place **Min** (respectivement **Max**), un événement $e \in E_{-\beta_{nuc}}$ ($e \in E_{+\beta_{nuc}}$) est observé, alors cela signifie que la valeur courante de β_{nuc} est devenue plus petite (respectivement plus grande) que le minimum (respectivement maximum) détecté précédemment ; la variable x est alors mise à jour avec la nouvelle valeur de β_{nuc} qui est potentiellement le nouveau minimum (respectivement maximum). Enfin si un événement $e \in E_{-\beta_{nuc}}$ survient alors que l'automate se trouve dans la place **Min** ou **Max**, cela n'a aucun effet sur l'état de l'automate.

D'autre part, dans la place **noisyInc** (**noisyDec**), l'analyse de la trajectoire présente deux alternatives possibles. Dans le premier cas (transition autonome **noisyInc** \rightarrow **Min**, respectivement **noisyDec** \rightarrow **Max**), l'automate revient en arrière et rentre dans **Min** (respectivement **Max**), si la population β_{nuc} , après avoir augmenté (respectivement diminué), diminue (augmente) à nouveau en-dessous (au-dessus) de x , et nécessite donc une mise à jour $x := \beta_{nuc}$. Dans le deuxième cas (transition autonome **noisyInc** \rightarrow **Max**, respectivement **noisyDec** \rightarrow **Min**), l'automate rentre dans la place **Max** (respectivement **Min**) dès que la population de β_{nuc} augmente (diminue) au-dessus (au-dessous) du seuil de bruit δ , ce qui correspond à la *fermeture* de la détection d'un minimum local pour β_{nuc} . Ainsi, lorsque l'arc **noisyInc** \rightarrow **Max** (**noisyDec** \rightarrow **Min**) est emprunté, on est certain que la variable x contient bien la valeur du dernier minimum (respectivement maximum) observé sur la trajectoire simulée. Sa valeur et son délai d'apparition sont stockés dans la n -ième position des deux tableaux respectifs $xmin[n] := x$ ($xmax[n] := x$) et $tmin[n] := t$ ($tmax[n] := t$).

L'analyse de la trajectoire simulée se termine en rentrant dans la place **End** (à partir de n'importe quelle place en entrée) dès que le temps de simulation est $t = T$. Tous les maximums et les minimums détectés sont stockés dans les variables de l'automate \mathcal{A}_{peaks} .

7.4.3 Mesure formelle des pics du signal β_{nuc} en utilisant l'automate \mathcal{A}_{peaks}

On utilise à présent l'automate \mathcal{A}_{peaks} pour réaliser la mesure formelle des pics du signal de sortie β_{nuc} pour le modèle de la voie Wnt. Pour cela, nous avons tout d'abord besoin de définir les expressions HASL qui caractérisent les mesure à effectuer sur l'ensemble des trajectoires sélectionnées par \mathcal{A}_{peaks} . Rappelons ici qu'une formule dans le langage HASL est constituée par un couple dont le premier élément est un automate hybride (dans notre cas \mathcal{A}_{peaks}), et dont le deuxième est une expression construite selon la grammaire décrite dans (6.4).

Les trois propriétés HASL complètes que nous avons considérées dans nos expériences sont les suivantes (également résumées dans la table 7.4) :

- $\phi_{xmax} \equiv (\mathcal{A}_{peaks}, E[last(xmax[1])])$: elle correspond à la valeur moyenne de la hauteur du premier pic maximal de β_{nuc} ;
- $\phi_{tmax} \equiv (\mathcal{A}_{peaks}, E[last(tmax[1])])$: elle correspond à la valeur moyenne du délai d'apparition du premier pic maximal de β_{nuc} ;
- $\phi_{PDFmax} \equiv (\mathcal{A}_{peaks}, PDF(last(tmax[1]), 1, 30, 80))$: elle correspond à la densité de probabilité du délai du premier pic maximal de β_{nuc} (calculé par rapport à l'intervalle $[30, 80]$ discrétisé dans le sous-intervalle de largeur 1).

propriété	déscription
$\phi_{x_{max}} \equiv (\mathcal{A}_{peaks}, E[last(x_{max}[1])])$	valeur moyenne de la hauteur du premier pic maximal de β_{nuc}
$\phi_{t_{max}} \equiv (\mathcal{A}_{peaks}, E[last(t_{max}[1])])$	valeur moyenne du délai d'apparition du premier pic maximal de β_{nuc}
$\phi_{PDF_{max}} \equiv (\mathcal{A}_{peaks}, PDF(last(t_{max}[1]), 1, 30, 80))$	densité de probabilité du délai du premier pic maximal de β_{nuc} (calculé par rapport à l'intervalle [30, 80] discrétisé dans le sous-intervalle de largeur 1).

TABEAU 7.4 – Propriétés HASL pour la mesure de pics du signal de sortie β_{nuc} pour le modèle GSPN de la voie Wnt.

Mesure de l'incidence du taux de dégradation de Wnt sur les pics de β_{nuc}

Dans les expériences suivantes, nous avons mesuré l'effet de la vitesse de dégradation du signal d'entrée Wnt sur les pics transitoires du signal de sortie β_{nuc} . Pour cela, nous avons évalué les propriétés $\phi_{x_{max}}$ et $\phi_{t_{max}}$ (décrites dans la table 7.4) en fonction d'instances différentes du modèle GSPN de la voie Wnt. Nous avons considéré des configurations du modèle avec réinjection de 1000 molécules de Wnt avec un délai $d = 450$ min, dont le profil de trajectoire stochastique (figure 7.4 à gauche) montre deux pics potentiellement de hauteur différente. De plus, afin d'évaluer l'effet de la vitesse de dégradation de Wnt, nous avons évalué les propriétés $\phi_{x_{max}}$ et $\phi_{t_{max}}$ avec des instances différentes du modèle avec réinjection : chaque instance est obtenue en faisant varier la valeur du paramètre k_1 (qui est le taux de la transition R_1 , la réaction de dégradation de Wnt) à partir de sa valeur initiale indiquée dans la table 7.2.

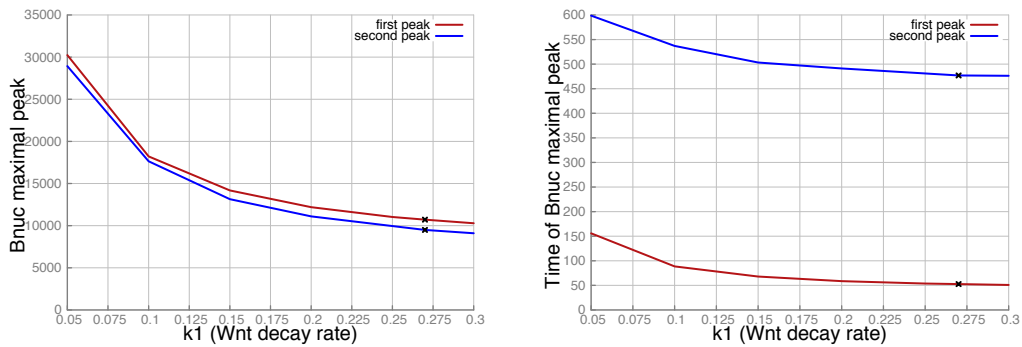


FIGURE 7.6 – Valeur moyenne (à gauche) et délai d'apparition (à droite) du premier et du second pic de β_{nuc} pour le modèle *Wnt-inject* en fonction du taux de dégradation de Wnt (les points mis en évidence correspondent à la valeur originale k_1 pris dans le tableau 7.2 de l'ensemble B).

Les résultats de l'évaluation de $\phi_{x_{max}}$ et de $\phi_{t_{max}}$ sont illustrés dans la figure 7.6. Celle-ci

contient deux graphes, où figurent la valeur moyenne (dans l'intervalle de confiance) de la hauteur (à gauche) et du délai d'apparition (à droite) des deux pics de β_{nuc} en fonction de la vitesse (k_1) de dégradation de Wnt. Ces résultats ont été calculés avec une confiance de 99.9% et un intervalle de confiance dont la largeur *relative* correspond à 1% du centre de l'intervalle. Les courbes de la figure 7.6 mettent en évidence les résultats intéressants suivants :

- La hauteur ainsi que le délai d'apparition des deux pics de β_{nuc} diminuent à mesure que la vitesse de dégradation de Wnt augmente ;
- Le deuxième pic de β_{nuc} atteint bien (en moyenne) une amplitude plus faible que le premier pic ;
- La différence d'amplitude entre le premier et le deuxième pic de β_{nuc} est à peu près constant et est d'environ 10%, sauf pour la valeur $k_1 = 0.1$ pour laquelle on mesure une différence d'environ 5%.

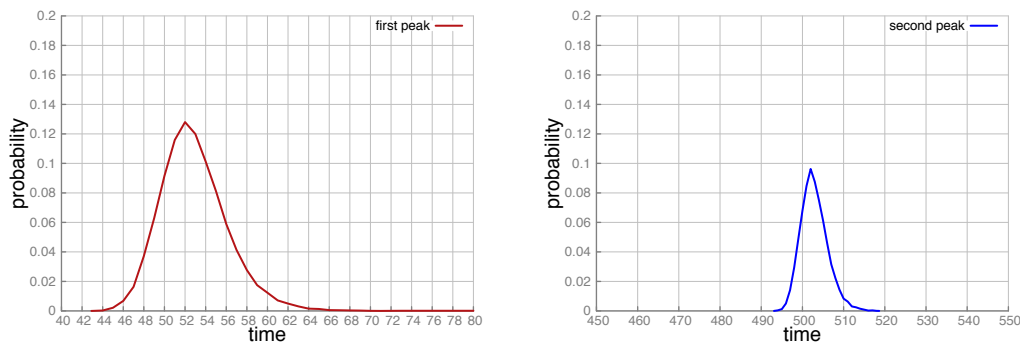


FIGURE 7.7 – La densité de probabilité du délai d'apparition du premier pic (à gauche) et du second pic (à droite) de β_{nuc} pour le modèle *Wnt-inject* (avec réinjection de 1000 molécules de Wnt à $t = 450$ minutes).

Mesure de la densité de probabilité du délai d'apparition des pics de β_{nuc} .

Dans ce jeu d'expériences, nous avons mesuré la densité de probabilité (PDF) du délai d'apparition des pics de β_{nuc} . Pour cela, nous avons évalué la propriété HASL ϕ_{PDFmax} décrite dans la table 7.4. Dans ce cas particulier, nous avons évalué ϕ_{PDFmax} dans le cadre d'une seule instance du modèle GSPN de la voie Wnt, qui correspond au jeu de paramètres de l'Ensemble B donné dans la table 7.2. Les résultats de l'évaluation de ϕ_{PDFmax} sont illustrés dans la figure 7.7, qui présente la courbe de la densité de probabilité du délai d'apparition du premier pic β_{nuc} (à gauche) et celle du deuxième pic (à droite). Notons que les deux PDF estimées atteignent leur amplitude maximale à respectivement $t \sim 52$ minutes pour le premier pic, et à $t \sim 502$ minutes pour le deuxième pic. Ces valeurs sont en accord avec les valeurs moyennes du délai d'apparition des pics mis en évidence par les points en noir dans le graphe de droite de la figure 7.6. De

plus, les maximums à $t \sim 52$ minutes (respectivement $t \sim 502$ minutes) indiquent que la vitesse de réaction du système (en terme du signal de sortie β_{nuc}) en réponse aux deux stimuli en entrée $Wnt=1000$ est constante, en sachant que la première stimulation démarre à $t = 0$ et la deuxième à $t = 450$ minutes. Par contre, les deux courbes de la figure 7.7 montrent un profil considérablement différent, avec une variation beaucoup plus importante de la PDF du deuxième pic par rapport à celle du premier pic (ce qui correspond à une cloche beaucoup plus étroite pour la courbe PDF du premier pic que pour celle du deuxième). Une interprétation possible pour justifier cette caractéristique est la suivante : si la réponse au premier stimulus Wnt montre une stochasticité relativement faible, la stochasticité lors de la réponse au deuxième pic augmente, intuitivement par *accumulation* (i.e. la stochasticité du deuxième pic est nécessairement affectée par celle du premier pic et est donc, naturellement, plus importante).

7.5 Conclusion

Dans ce chapitre, nous avons présenté une étude formelle d'un modèle de la voie de signalisation Wnt/ β -caténine par application de la méthode HASL. Nous avons montré que grâce au formalisme HASL nous pouvons développer des spécifications formelles de propriétés particulières, destinées à mesurer des aspects très pointus caractérisant la dynamique de la voie Wnt/ β -caténine.

En particulier, nous avons développé des automates LHA capables de détecter les pics transitoires du signal de sortie (i.e. β_{nuc}) et nous les avons utilisés pour mesurer la réponse du modèle à des stimulations transitoires en entrée de la voie (i.e. des pics de Wnt en entrée). Par application de la méthode du *model-checking* statistique du type HASL, nous avons pu estimer avec une précision arbitraire (confiance de 99.9%) des grandeurs caractéristiques de la dynamique de la voie Wnt/ β -caténine, typiquement la hauteur et le délai d'apparition des pics du signal de sortie. Cela nous a permis d'observer une certaine différence dans la réponse de la voie Wnt/ β -caténine face à des stimulus identiques répétés au cours du temps. En particulier, l'amplitude du signal de sortie (β_{nuc}) en réponse à un deuxième stimulus identique présente une atténuation de l'ordre de 10% par rapport à la hauteur du premier pic.

Bien que l'étude que nous avons présentée est une étude préparatoire, il démontre bien le potentiel de la méthode HASL. De fait, dans la plupart des travaux de modélisation stochastique des systèmes biologiques, l'analyse des modèles est faite à travers l'observation d'une trajectoire ou alors d'une petite quantité de trajectoires, stockées et analysées *offline*, souvent via le calcul de statistiques *à la main* sur l'échantillon des trajectoires simulées. Bien que raisonnable, cette approche ne se base sur aucun formalisme des propriétés et manque d'automatisation. Par contre, avec la méthode HASL, nous avons une méthode générale qui permet à l'utilisateur d'exprimer des propriétés très sophistiquées et d'estimer leur valeur à travers une procédure automatique (la génération de traces et le calcul des statistiques sont faits automatiquement par la méthode).

Conclusion

Dans ce manuscrit, nous avons présenté les travaux de recherche effectués au cours de ma thèse dans l'équipe LogiMAS du laboratoire MICS de CentraleSupélec. Cette thèse a comporté deux volets différents, traitant de l'utilisation de techniques de model-checking afin, d'une part, d'inférer les paramètres régissant la dynamique des réseaux de régulation génétique dans le cadre du modèle de Thomas et, d'autre part, d'analyser la dynamique de modèles stochastiques d'un modèle biologique particulier, la voie de signalisation Wnt/ β -caténine.

Dans la partie I, nous avons présenté notre approche visant à réduire les effets de l'explosion combinatoire lors de la recherche des paramètres du modèle de Thomas.

Notre méthode (chapitre 3) se base sur le model-checking LTL, auquel nous avons joint des techniques d'exécution symbolique et de résolution de contrainte.

Notre méthode consiste tout d'abord à construire à partir du graphe d'interaction d'un GRN un méta-modèle modélisant l'ensemble des dynamiques possibles du GRN en une seule structure, appelée PGRN. Sur le modèle du model-checking LTL, nous réalisons alors le produit du PGRN et de l'automate de Büchi correspondant à la formule LTL que doit satisfaire le système pour vérifier ou invalider les hypothèses biologiques en entrée. Ce produit est ensuite exécuté symboliquement, de manière à collecter les contraintes sur les paramètres permettant de satisfaire le modèle. Ainsi, nous nous éloignons du model-checking LTL classique et réduisons les effets de l'explosion combinatoire car les paramètres ne sont pas instanciés pendant l'exécution : toutes les dynamiques sont étudiées conjointement. Notre méthode s'appuie également sur la détection de différents cas de coupure de branche, permettant d'arrêter l'exploration des branches qui ne fourniraient pas de contraintes supplémentaires sur les paramètres.

Notre approche n'est pas uniquement théorique : nous l'avons outillée grâce à notre logiciel $SP_{\forall}TNI_{\kappa}$ (chapitre 4 et annexe A). $SP_{\forall}TNI_{\kappa}$ permet de créer (et modifier) facilement un modèle à étudier, de visualiser ses données et de restituer les résultats obtenus avec notre méthode.

En nous servant de notre outil $SP_{\forall}TNI_{\kappa}$, nous avons modélisé plusieurs études de cas classiques de la littérature, ainsi que des nouveaux cas en collaboration avec des biologistes (chapitre 5 et annexe B).

Nous avons montré que les temps de calcul obtenus avec $SP_{\forall}TNI_{\kappa}$ étaient meilleurs ou du moins du même ordre de grandeur que ceux obtenus avec d'autres outils pris dans l'état de

l'art. Dans le chapitre 4, nous avons présenté des pistes d'optimisation qui pourraient être implémentées pour améliorer encore l'efficacité de $SP_{\nu}TNI_k$, en particulier la parallélisation du parcours du Produit.

Néanmoins, l'intérêt de notre méthode repose également sur les possibilités spécifiques qu'elle offre : pas de restriction sur la longueur des chemins à tester, utilisation à loisir de contraintes statiques sur les paramètres et possibilité d'utiliser une interface utilisateur facilitant l'entrée du modèle et de ses données.

Dans la partie II, nous avons présenté un travail préliminaire effectué dans le cadre d'une collaboration avec des neuropédiatres de l'unité mixte de recherche 1141 (*Neuroprotection du cerveau en développement*) dirigée par le Dr. Pierre Gressens.

Ce travail a consisté à étudier la voie de signalisation Wnt/ β -caténine à l'aide du model-checker statistique COSMOS, basé sur la logique HASL, afin d'analyser les dynamiques de ce système biologique de manière formelle. Pour cela, nous avons développé deux modèles différents à partir de modèles trouvés dans la littérature (chapitre 7). Les résultats de notre étude a permis de mettre en évidence des comportements intéressants l'équipe du Dr. Pierre Gressens.

Les contributions de cette deuxième partie de la thèse portent sur les points suivants :

1. Le développement de deux versions différentes du modèle de la voie de signalisation Wnt/ β -caténine sous la forme de réseaux de Petri stochastiques : une version dite *modèle Wnt core*, caractérisée par un comportement *transitoire* en réponse à un signal transitoire, ainsi qu'une version dite *modèle Wnt oscillateur*, caractérisée par un régime oscillatoire permanent déclenché par des pulse transitoires du signal Wnt. Les modèles de réseaux de Petri développés ont été enrichis avec des modules (i.e. sous-réseaux) *de contrôle*, que nous avons développé spécialement pour pouvoir reproduire en toute facilité des conditions expérimentales différentes.
2. Le développement de *mesures cibles*, formellement spécifiées en utilisant le langage HASL, qui visent à la mise en œuvre d'un cadre d'analyse *fine* de la dynamique, à la fois du *modèle Wnt core* ainsi que du *modèle Wnt oscillateur*.
3. La réalisation de *l'analyse de sensibilité* pour les deux modèles à l'aide du model-checker statistique COSMOS.

En perspective, cette étude préliminaire pourrait être poursuivie dans plusieurs directions, notamment en faisant varier le type de signal Wnt en entrée : par exemple, en étudiant les effets induits par un signal Wnt soutenu, ainsi que l'effet de la synchronisation cellulaire sur la dynamique de β_{nuc} sur un modèle de population cellulaire dont les cellules évoluent de manière asynchrone.

Nous avons également commencé un travail d'estimation des paramètres à partir des premiers résultats fournis par l'équipe du Dr. Pierre Gressens, afin d'adapter notre modèle aux spécificités des cellules qu'ils étudient. Ce travail n'a pas pu être mené à terme lors de cette thèse à cause

des délais de réalisation des expériences biologiques, il serait intéressant qu'il soit repris lorsque les données seront disponibles.

Bibliographie

- [Ahm+09] Jamil AHMAD, Jérémie BOURDON, Damien EVEILLARD, Jonathan FROMENTIN, Olivier ROUX et Christine SINOQUET. “Temporal constraints of a gene regulatory network : Refining a qualitative simulation”. In : *Biosystems* 98.3 (2009), p. 149–159.
- [Ajm+95] M. AJMONE MARSAN, G. BALBO, G. CONTE, S. DONATELLI et G. FRANCESCHINIS. *Modeling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1995.
- [Alb+02] Bruce ALBERTS, Alexander JOHNSON, Julian LEWIS, Martin RAFF, Keith ROBERTS et Peter WALTER. *Molecular Biology of the Cell. 4th edition*. Garland Science, 2002. URL : <http://www.ncbi.nlm.nih.gov/books/NBK21054>.
- [Bai+03a] C. BAIER, B. HAVERKORT, H. HERMANNNS et J.-P. KATOEN. “Model-checking algorithms for CTMCs”. In : *IEEE Trans. on Software Eng.* 29.6 (2003).
- [Bai+03b] Christel BAIER, Boudewijn HAVERKORT, Holger HERMANNNS et Joost-Pieter KATOEN. “Model-checking algorithms for continuous-time Markov chains”. In : *Software Engineering, IEEE Transactions on* 29.6 (2003), p. 524–541.
- [Bal+10] Paolo BALLARINI, Hilal DJAFRI, Marie DUFLLOT, Serge HADDAD et Nihal PEKERGIN. *HASL : an Expressive Language for Statistical Verification of Stochastic Models*. Rapp. tech. TR-LACL-2010-8. LACL University of Paris-Est Créteil, 2010.
- [Bal+11] P. BALLARINI, H. DJAFRI, M. DUFLLOT, S. HADDAD et N. PEKERGIN. “COSMOS : a Statistical Model Checker for the Hybrid Automata Stochastic Logic”. In : *Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST’11)*. IEEE Computer Society Press, sept. 2011, 143–144.
- [Bal+12] Deepak BALASUBRAMANIAN, Lisa SCHNEPER, Hansi KUMARI et Kalai MATHEE. “A dynamic and intricate regulatory network determines *Pseudomonas aeruginosa* virulence”. In : *Nucleic acids research* (2012), gks1039.
- [Bal+14] Paolo BALLARINI, Emmanuelle GALLET, Pascale Le GALL et Matthieu MANCENY. “Formal Analysis of the Wnt/ β -catenin through Statistical Model Checking”. In : *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications - 6th International Symposium, ISO/LA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part II*. Sous la dir. de Tiziana MARGARIA et Bernhard STEFFEN. T. 8803. Lecture Notes in Computer Science. Springer, 2014, p. 193–207. ISBN : 978-3-662-45230-1. DOI : [10.1007/978-3-662-45231-8_14](https://doi.org/10.1007/978-3-662-45231-8_14). URL : http://dx.doi.org/10.1007/978-3-662-45231-8_14.
- [Bal+15] Paolo BALLARINI, Benoît BARBOT, Marie DUFLLOT, Serge HADDAD et Nihal PEKERGIN. “HASL : A new approach for performance evaluation and model checking from concepts to experimentation”. In : *Perform. Eval.* 90 (2015), p. 53–77. DOI : [10.1016/j.peva.2015.04.003](https://doi.org/10.1016/j.peva.2015.04.003). URL : <http://dx.doi.org/10.1016/j.peva.2015.04.003>.

- [Bar+12] J. BARNAT, L. BRIM, A. KREJCI, A. STRECK, D. SAFRÁNEK, M. VEJNAR et T. VEJPUSTEK. “On Parameter Synthesis by Parallel Model Checking”. In : *IEEE/ACM Trans. Comput. Biology Bioinform.* 9.3 (2012), p. 693–705.
- [Bat+05] Grégory BATT, Delphine ROPERS, Hidde DE JONG, Johannes GEISELMANN, Radu MATEESCU, Michel PAGE et Dominique SCHNEIDER. “Validation of qualitative models of genetic regulatory networks by model checking : analysis of the nutritional stress response in *Escherichia coli*”. In : *Bioinformatics* 21.suppl 1 (2005), p. i19–i28.
- [BCK08] Gilles BERNOT, Jean-Paul COMET et Zohra KHALIS. “Gene regulatory networks with multiplexes”. In : *European Simulation and Modelling Conference Proceedings*. 2008, p. 423–432.
- [Ber+04] G. BERNOT, J.-P. COMET, A. RICHARD et J. GUESPIN. “Application of formal methods to biological regulatory networks : extending Thomas’ asynchronous logical approach with temporal logic”. In : *Journal of Theoretical Biology* 229.3 (août 2004), p. 339–347. ISSN : 00225193. DOI : [10.1016/j.jtbi.2004.04.003](https://doi.org/10.1016/j.jtbi.2004.04.003).
- [Big+03] C. BIGOT, A. FAIVRE, J.-P. GALLOIS, A. LAPITRE, D. LUGATO, J.-Y. PIERRON et N. RAPIN. “Automatic test generation with Agatha”. In : *TACAS*. Sous la dir. de H. GARAVEL et J. HATCLIFF. T. 2619. Lecture Notes in Computer Science. Springer, 2003, p. 591–596.
- [BK08] Christel BAIER et Joost-Pieter KATOEN. *Principles of Model Checking*. T. 26202649. The MIT Press, 2008.
- [Bod11] Eric BODDEN. *LTL2BA4J Software*, <http://www.sable.mcgill.ca/~ebodde/rv/ltl2ba4j/>. RWTH Aachen University, 2011.
- [Bon+09] Nicola BONZANNI, K Anton FEENSTRA, Wan FOKKINK et Elzbieta KREPSKA. “What can formal methods bring to systems biology ?” In : *FM 2009 : Formal Methods*. Springer, 2009, p. 16–22.
- [Bri+15] Luboš BRIM, Milan ČEŠKA, Martin DEMKO, Samuel PASTVA et David ŠAFRÁNEK. “Parameter Synthesis by Parallel Coloured CTL Model Checking”. In : *Computational Methods in Systems Biology*. Springer. 2015, p. 251–263.
- [Buc60] J. R. BÜCHI. “On a Decision Method in a Restricted Second Order Arithmetic”. In : *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science*. Sous la dir. de PRESS. 1960, p. 1–11.
- [C+99] Ting CHEN, Hongyu L HE, George M CHURCH et al. “Modeling gene expression with differential equations.” In : *Pacific symposium on biocomputing*. T. 4. 29. World Scientific. 1999, p. 4.
- [CE81] Edmund M CLARKE et E Allen EMERSON. “Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic”. In : (1981).
- [CFS06] Laurence CALZONE, François FAGES et Sylvain SOLIMAN. “BIOCHAM : an environment for modeling biological systems and formalizing experimental knowledge”. In : *Bioinformatics* 22.14 (2006), p. 1805–1807. DOI : [10.1093/bioinformatics/btl172](https://doi.org/10.1093/bioinformatics/btl172). eprint : <http://bioinformatics.oxfordjournals.org/content/22/14/1805.full.pdf+html>. URL : <http://bioinformatics.oxfordjournals.org/content/22/14/1805.abstract>.
- [CFT10] F. CORBLIN, E. FANCHON et L. TRILLING. “Applications of a formal approach to decipher discrete genetic networks”. In : *BMC Bioinformatics* 11 (2010), p. 385.
- [CGP99] Edmund M CLARKE, Orna GRUMBERG et Doron PELED. *Model checking*. MIT press, 1999.

- [CH09] Federica CIOCCETTA et Jane HILLSTON. “Bio-PEPA : A framework for the modelling and analysis of biological systems”. In : *Theoretical Computer Science* 410.33 (2009), p. 3065–3084.
- [Che+08] Xi CHEN et al. “Integration of External Signaling Pathways with the Core Transcriptional Network in Embryonic Stem Cells”. In : *Cell* 133.6 (2008), p. 1106–1117. ISSN : 0092-8674. DOI : <http://dx.doi.org/10.1016/j.cell.2008.04.043>. URL : <http://www.sciencedirect.com/science/article/pii/S009286740800617X>.
- [CHO10] CHOCO TEAM. *CHOCO : an Open Source Java Constraint Programming Library*. Research report 10-02-INFO. École des Mines de Nantes, 2010. URL : <http://www.emn.fr/z-info/choco-solver/pdf/choco-presentation.pdf>.
- [Cim+02] Alessandro CIMATTI, Edmund CLARKE, Enrico GIUNCHIGLIA, Fausto GIUNCHIGLIA, Marco PISTORE, Marco ROVERI, Roberto SEBASTIANI et Armando TACHELLA. “Nusmv 2 : An opensource tool for symbolic model checking”. In : *Computer Aided Verification*. Springer, 2002, p. 359–364.
- [Cor+09] F. CORBLIN, S. TRIPODI, E. FANCHON, D. ROPERS et L. TRILLING. “A declarative constraint-based method for analyzing discrete genetic regulatory networks”. In : *BioSystems* 98 (2009), p. 91–104.
- [Cor+12] Fabien CORBLIN, Eric FANCHON, Laurent TRILLING, Claudine CHAOUÏYA et Denis THIEFFRY. “Automatic inference of regulatory and dynamical properties from incomplete gene interaction and expression data”. In : *Proceedings of the 9th international conference on Information Processing in Cells and Tissues*. IPCAT’12. Cambridge, UK : Springer-Verlag, 2012, p. 25–30. ISBN : 978-3-642-28791-6. DOI : [10.1007/978-3-642-28792-3_4](https://doi.org/10.1007/978-3-642-28792-3_4).
- [Cri58] Francis H CRICK. “On protein synthesis.” In : *Symposia of the Society for Experimental Biology*. T. 12. 1958, p. 138.
- [DB08] Leonardo DE MOURA et Nikolaj BJØRNER. “Z3 : An efficient SMT solver”. In : *Tools and Algorithms for the Construction and Analysis of Systems*. T. 4963. Lecture Notes in Computer Science. Springer, 2008, p. 337–340.
- [DHS07] S. DONATELLI, S. HADDAD et J. SPROSTON. “CSL^{TA} : an Expressive Logic for Continuous-Time Markov Chains”. In : *Quantitative Evaluation of Systems, 2007. QEST 2007. Fourth International Conference on the*. Sept. 2007, p. 31–40. DOI : [10.1109/QEST.2007.40](https://doi.org/10.1109/QEST.2007.40).
- [DL04] Vincent DANOS et Cosimo LANEVE. “Formal molecular biology”. In : *Theoretical Computer Science* 325.1 (2004), p. 69–110.
- [EC80] E Allen EMERSON et Edmund M CLARKE. *Characterizing correctness properties of parallel programs using fixpoints*. Springer, 1980.
- [EH86] E Allen EMERSON et Joseph Y HALPERN. “iSometimesj and inot neverj revisited : on branching versus linear time temporal logic”. In : *Journal of the ACM (JACM)* 33.1 (1986), p. 151–178.
- [Fau+06] Adrien FAURÉ, Aurélien NALDI, Claudine CHAOUÏYA et Denis THIEFFRY. “Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle”. In : *Bioinformatics* 22.14 (2006), e124–e131. DOI : [10.1093/bioinformatics/bt1210](https://doi.org/10.1093/bioinformatics/bt1210). eprint : <http://bioinformatics.oxfordjournals.org/content/22/14/e124.full.pdf+html>. URL : <http://bioinformatics.oxfordjournals.org/content/22/14/e124.abstract>.

- [Fil+06] D. FILOPON, A. MÉRIEU, G. BERNOT, J.-P. COMET, R. LEBERRE, B. GUERY, B. POLACK et J. GUESPIN. “Epigenetic acquisition of inducibility of type III cytotoxicity in *P. aeruginosa*”. In : *BMC Bioinformatics* 7 (2006), p. 272–282. DOI : [10.1186/1471-2105-7-272](https://doi.org/10.1186/1471-2105-7-272).
- [Fra97] Dara W FRANK. “The exoenzyme S regulon of *Pseudomonas aeruginosa*”. In : *Molecular microbiology* 26.4 (1997), p. 621–629.
- [Fro+07] Jonathan FROMENTIN, Jean-Paul COMET, Pascale LE GALL et Olivier ROUX. “Analysing Gene Regulatory Networks by both Constraint Programming and Model-Checking”. In : *EMBS 2007. 29th Annual International Conference of the IEEE*. Août 2007, p. 4595–4598. DOI : [10.1109/IEMBS.2007.4353363](https://doi.org/10.1109/IEMBS.2007.4353363).
- [Gal+13] Emmanuelle GALLET, Matthieu MANCENY, Pascale LE GALL et Paolo BALLARINI. “A Symbolic Approach Based on Model Checking and Constraint Solving Techniques for Reverse Engineering of Thomas Networks Parameters”. In : *Computational Methods in Systems Biology : 11th International Conference, CMSB 2013, Klosterneuburg, Austria, September 22-24, 2013, Proceedings*. T. 8130. Springer. 2013, p. 240.
- [Gal+14a] Emmanuelle GALLET, Matthieu MANCENY, Pascale LE GALL et Paolo BALLARINI. “Adapting LTL model checking for inferring biological parameters”. In : *Actes de la 13ème édition d’AFADL, atelier francophone sur les Approches Formelles dans l’Assistance au Développement de Logiciels, juin 2014*. Sous la dir. de Régine Laleau CATHERINE DUBOIS. AFADL, mai 2014.
- [Gal+14b] Emmanuelle GALLET, Matthieu MANCENY, Pascale LE GALL et Paolo BALLARINI. “An LTL Model Checking Approach for Biological Parameter Inference”. In : *Formal Methods and Software Engineering*. Sous la dir. de Stephan MERZ et Jun PANG. T. 8829. Lecture Notes in Computer Science. Springer International Publishing, nov. 2014, p. 155–170. ISBN : 978-3-319-11736-2. DOI : [10.1007/978-3-319-11737-9_11](https://doi.org/10.1007/978-3-319-11737-9_11).
- [Gal+15] Emmanuelle GALLET, Matthieu MANCENY, Pascale LE GALL et Paolo BALLARINI. “Étude de réseaux de Thomas par validation de propriétés LTL pour *Pseudomonas aeruginosa*”. In : *Technique et Science Informatiques* 34.5 (2015), p. 575–600. DOI : [10.3166/tsi.34.575-600](https://doi.org/10.3166/tsi.34.575-600). URL : <http://dx.doi.org/10.3166/tsi.34.575-600>.
- [Gas+06] Christophe GASTON, Pascale LE GALL, Nicolas RAPIN et Assia TOUIL. “Symbolic Execution Techniques for Test Purpose Definition”. In : *18th IFIP Int. Conf. TestCom*. T. 3964. Lecture Notes in Computer Science. Springer, 2006, p. 1–18.
- [Gil77] Daniel T GILLESPIE. “Exact stochastic simulation of coupled chemical reactions”. In : *The journal of physical chemistry* 81.25 (1977), p. 2340–2361.
- [GO01] P. GASTIN et D. ODDOUX. “Fast LTL to Büchi Automata Translation”. In : *Proceedings of the 13th International Conference on Computer Aided Verification (CAV’01)*. T. 2102. Lecture Notes in Computer Science. Paris, France : Springer, 2001, p. 53–65.
- [Gon+06] A Gonzalez GONZALEZ, Aurélien NALDI, Lucas SANCHEZ, Denis THIEFFRY et Claudine CHAOUÏYA. “GINSim : a software suite for the qualitative modelling, simulation and analysis of regulatory networks”. In : *Biosystems* 84.2 (2006), p. 91–100.
- [Goo65] Brian C. GOODWIN. “Oscillatory behavior in enzymatic control processes”. In : *Advances in Enzyme Regulation* 3 (1965), p. 425–437. ISSN : 0065-2571. DOI : [http://dx.doi.org/10.1016/0065-2571\(65\)90067-1](http://dx.doi.org/10.1016/0065-2571(65)90067-1). URL : <http://www.sciencedirect.com/science/article/pii/0065257165900671>.
- [Hen96] Thomas A. HENZINGER. “The Theory of Hybrid Automata”. In : IEEE Computer Society Press, 1996, p. 278–292.

- [HJ94] Hans HANSSON et Bengt JONSSON. “A logic for reasoning about time and reliability”. In : *Formal aspects of computing* 6.5 (1994), p. 512–535.
- [Hor+12] András HORVÁTH, Marco PAOLIERI, Lorenzo RIDI et Enrico VICARIO. “Transient analysis of non-Markovian models using stochastic state classes”. In : *Perform. Eval.* 69.7-8 (2012), p. 315–335. DOI : [10.1016/j.peva.2011.11.002](https://doi.org/10.1016/j.peva.2011.11.002). URL : <http://dx.doi.org/10.1016/j.peva.2011.11.002>.
- [Hua+05] Jianbin HUANG, Shian WU, Jose BARRERA, Krista MATTHEWS et Duojia PAN. “The Hippo signaling pathway coordinately regulates cell proliferation and apoptosis by inactivating Yorkie, the Drosophila Homolog of YAP”. In : *Cell* 122.3 (2005), p. 421–434.
- [Ich+97] Hidenori ICHIJO, Eisuke NISHIDA, Kenji IRIE, Peter ten DIJKE, Masao SAITOH, Tetsuo MORIGUCHI, Minoru TAKAGI, Kunihiro MATSUMOTO, Kohei MIYAZONO et Yukiko GOTOH. “Induction of Apoptosis by ASK1, a Mammalian MAPKKK That Activates SAPK/JNK and p38 Signaling Pathways”. In : *Science* 275.5296 (1997), p. 90–94. ISSN : 0036-8075. DOI : [10.1126/science.275.5296.90](https://doi.org/10.1126/science.275.5296.90). eprint : <http://science.sciencemag.org/content/275/5296/90.full.pdf>. URL : <http://science.sciencemag.org/content/275/5296/90>.
- [Ito+13] Sohei ITO, Takuma ICHINOSE, Masaya SHIMAKAWA, Naoko IZUMI, Shigeki HAGIHARA et Naoki YONEZAKI. “Modular analysis of gene networks by linear temporal logic”. In : *Journal of integrative bioinformatics* 10.2 (2013), p. 216.
- [Jac+05] François JACOB, David PERRIN, Carmen SÁNCHEZ et Jacques MONOD. “L’opéron : groupe de gènes à expression coordonnée par un opérateur [C. R. Acad. Sci. Paris 250 (1960) 1727–1729]”. In : *Comptes Rendus Biologies* 328.6 (2005). Retour sur l’opéron lacLac Operon revisited, p. 514–520. ISSN : 1631-0691. DOI : <http://dx.doi.org/10.1016/j.crvi.2005.04.005>. URL : <http://www.sciencedirect.com/science/article/pii/S1631069105000673>.
- [JM61] François JACOB et Jacques MONOD. “On the regulation of gene activity”. In : *Cold Spring Harbor Symposia on Quantitative Biology*. T. 26. Cold Spring Harbor Laboratory Press. 1961, p. 193–211.
- [Jon02] H. de JONG. “Modelling and simulation of genetic regulatory systems : A litterature review”. In : *J Comput Biol* 9.1 (2002), p. 67–103.
- [Kau69] S.A. KAUFFMAN. “Metabolic stability and epigenesis in randomly constructed genetic nets”. In : *Journal of Theoretical Biology* 22.3 (1969), p. 437–467.
- [Kha+09] Z. KHALIS, J.-P. COMET, A. RICHARD et G. BERNOT. “The SMBioNet Method for Discovering Models of Gene Regulatory Networks”. In : *Genes, Genomes and Genomics* 3(special issue 1) (2009), p. 15–22.
- [Kin75] J.-C. KING. “A new approach to program testing”. In : *Proceedings of the international conference on Reliable software, Los Angeles, California* 21-23 (avr. 1975), p. 228–233.
- [Kit01] H. KITANO. *Foundations of Systems Biology*. MIT Press, 2001.
- [KKK09] Hiroyuki KOBAYASHI, Osamu KOBAYASHI et Shin KAWAI. “Pathogenesis and clinical manifestations of chronic colonization by *Pseudomonas aeruginosa* and its biofilms in the airway tract”. In : *Journal of infection and chemotherapy* 15.3 (2009), p. 125–142.

- [Kla+12a] H. KLARNER, A. STRECK, D. ŠAFRÁNEK, J. KOLČÁK et H. SIEBERT. “Parameter identification and model ranking of Thomas networks”. In : *Proceedings of the 10th international conference on Computational Methods in Systems Biology*. CMSB’12. London, UK : Springer-Verlag, 2012, p. 207–226. ISBN : 978-3-642-33635-5. DOI : [10.1007/978-3-642-33636-2_13](https://doi.org/10.1007/978-3-642-33636-2_13).
- [Kla+12b] Hannes KLARNER, Adam STRECK, David ŠAFRÁNEK, Juraj KOLČÁK et Heike SIEBERT. *Parameter identification and model ranking of Thomas networks*. Rapp. tech. FIMU-RS-2012-03. Masaryk University, 2012.
- [Kri06] Mayr B KRIEGHOFF E Behrens J. “Nucleo-cytoplasmic distribution of beta-catenin is regulated by retention.” In : *Journal of Cell Science* 119 (2006), p. 1453–63.
- [LDB10] Axel LEGAY, Benoît DELAHAYE et Saddek BENSLEM. “Statistical Model Checking : An Overview”. English. In : *Runtime Verification*. Sous la dir. d’Howard BARRINGER, Ylies FALCONE, Bernd FINKBEINER, Klaus HAVELUND, Insup LEE, Gordon PACE, Grigore ROȘU, Oleg SOKOLSKY et Nikolai TILLMANN. T. 6418. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, p. 122–135. ISBN : 978-3-642-16611-2. DOI : [10.1007/978-3-642-16612-9_11](https://doi.org/10.1007/978-3-642-16612-9_11). URL : http://dx.doi.org/10.1007/978-3-642-16612-9_11.
- [Lee+03] Ethan LEE, Adrian SALIC, Roland KRUGER, Reinhart HEINRICH et Marc W KIRSCHNER. “The roles of APC and Axin derived from experimental and theoretical analysis of the Wnt pathway”. In : *PLoS Biol* 1.1 (2003), E10.
- [MA02] J. Susan MILTON et Jesse C. ARNOLD. *Introduction to Probability and Statistics : Principles and Applications for Engineering and the Computing Sciences*. 4th. McGraw-Hill Higher Education, 2002. ISBN : 007246836X.
- [MA99] Harley H McADAMS et Adam ARKIN. “It’s a noisy business ! Genetic regulation at the nanomolar scale”. In : *Trends in genetics* 15.2 (1999), p. 65–69.
- [Mat+07] D. MATEUS, J.-P. GALLOIS, J.-P. COMET et P. LE GALL. “Symbolic modeling of genetic regulatory networks”. Anglais. In : *Journal of Bioinformatics and Computational Biology* 5.2B (2007), p. 627–640.
- [Maz+12] Oriane MAZEMONDET, Mathias JOHN, Stefan LEYE, Arndt ROLFS et Adelinde M UHRMACHER. “Elucidating the sources of β -catenin dynamics in human neural progenitor cells”. In : *PLOS-One* 7.8 (2012), p. 1–12.
- [MS95] Harley H McADAMS et Lucy SHAPIRO. “Circuit simulation of genetic networks”. In : *Science* 269.5224 (1995), p. 650–656.
- [MTA99] Luis MENDOZA, Denis THIEFFRY et Elena R ALVAREZ-BUYLLA. “Genetic control of flower morphogenesis in *Arabidopsis thaliana* : a logical analysis.” In : *Bioinformatics* 15.7 (1999), p. 593–606.
- [Mur+96] Eric MURAILLE, Denis THIEFFRY, Oberdan LEO et Marcelle KAUFMAN. “Toxicity and Neuroendocrine Regulation of the Immune Response : A Model Analysis”. In : *Journal of Theoretical Biology* 183.3 (1996), p. 285–305. ISSN : 0022-5193. DOI : <http://dx.doi.org/10.1006/jtbi.1996.0221>. URL : <http://www.sciencedirect.com/science/article/pii/S0022519396902210>.
- [Opp+05] Amos B OPPENHEIM, Oren KOBILER, Joel STAVANS, Donald L COURT et Sankar ADHYA. “Switches in bacteriophage lambda development”. In : *Annu. Rev. Genet.* 39 (2005), p. 409–429.

- [PFL16] Charles PRUD'HOMME, Jean-Guillaume FAGES et Xavier LORCA. *Choco Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S. 2016. URL : <http://www.choco-solver.org>.
- [PMO95] Erik PLAHTÉ, Thomas MESTL et Stig W OMHOLT. "Feedback loops, stability and multistationarity in dynamical systems". In : *Journal of Biological Systems* 3.02 (1995), p. 409–413.
- [Pnu77] Amir PNUELI. "The temporal logic of programs". In : *Foundations of Computer Science, 1977., 18th Annual Symposium on*. IEEE. 1977, p. 46–57.
- [Pnu81] Amir PNUELI. "The temporal semantics of concurrent programs". In : *Theoretical computer science* 13.1 (1981), p. 45–60. DOI : [http://dx.doi.org/10.1016/0304-3975\(81\)90110-9](http://dx.doi.org/10.1016/0304-3975(81)90110-9). URL : <http://www.sciencedirect.com/science/article/pii/0304397581901109>.
- [Pta04] Mark PTASHNE. *A genetic switch : phage lambda revisited*. T. 3. Cold Spring Harbor Laboratory Press Cold Spring Harbor, NY : 2004.
- [PVP08] G.H. PERDEW, J.P. VANDEN HEUVEL et J.M. PETERS. *Regulation of Gene Expression*. Infectious disease. Humana Press, 2008. ISBN : 9781597452281.
- [QS82] Jean-Pierre QUEILLE et Joseph SIFAKIS. "Specification and verification of concurrent systems in CESAR". In : *International Symposium on Programming*. Springer. 1982, p. 337–351.
- [RC07] Adrien RICHARD et Jean-Paul COMET. "Necessary conditions for multistationarity in discrete dynamical systems". In : *Discrete Applied Mathematics* 155.18 (2007), p. 2403–2413.
- [RCB06] A. RICHARD, J.-P. COMET et G. BERNOT. "Modern Formal Methods and Applications". In : Springer, 2006. Chap. Formal methods for modeling biological regulatory networks.
- [Ric06] Adrien RICHARD. "Modèle formel pour les réseaux de régulation génétique et influence des circuits de rétroaction". Thèse de doct. Evry-Val d'Essonne, 2006.
- [Ric09] Adrien RICHARD. "Positive circuits and maximal number of fixed points in discrete dynamical systems". In : *Discrete Applied Mathematics* 157.15 (2009), p. 3281–3288.
- [Ric10a] A. RICHARD. *SMBioNet User manual*, <http://www.i3s.unice.fr/~richard/smbionet/>. 2010.
- [Ric10b] Adrien RICHARD. "Negative circuits and sustained oscillations in asynchronous automata networks". In : *Advances in Applied Mathematics* 44.4 (2010), p. 378–392. ISSN : 0196-8858. DOI : <http://dx.doi.org/10.1016/j.aam.2009.11.011>. URL : <http://www.sciencedirect.com/science/article/pii/S0196885809001304>.
- [Ros02] Sheldon M. ROSS. *Simulation*. Third Edition. Academic Press, Elsevier, 2002.
- [RRT08] Élisabeth REMY, Paul RUET et Denis THIEFFRY. "Graphic requirements for multistability and attractive cycles in a Boolean dynamical framework". In : *Advances in Applied Mathematics* 41.3 (2008), p. 335–350.
- [RWA02] Christopher V RAO, Denise M WOLF et Adam P ARKIN. "Control, exploitation and tolerance of intracellular noise". In : *Nature* 420.6912 (2002), p. 231–237.

- [SA14] Tariq SAEED et Jamil AHMAD. "A parallel approach for accelerated parameter identification of Gene Regulatory Networks". In : *International Work-Conference on Bioinformatics and Biomedical Engineering, IWBBIO 2014, Granada, Spain, April 7-9, 2014*. 2014, p. 1769–1779. URL : http://iwbbio.ugr.es/2014/papers/IWBBIO_2014_paper_196.pdf.
- [SA85] Madeline A SHEA et Gary K ACKERS. "The O R control system of bacteriophage lambda : A physical-chemical model for gene regulation". In : *Journal of molecular biology* 181.2 (1985), p. 211–230.
- [SBB00] Paul SMOLEN, Douglas A BAXTER et John H BYRNE. "Modeling transcriptional control in gene networks—methods, recent results, and future directions". In : *Bulletin of mathematical biology* 62.2 (2000), p. 247–292.
- [SF63] Motoyosi SUGITA et Nobuo FUKUDA. "Functional analysis of chemical systems in vivo using a logical circuit equivalent : III. Analysis using a digital circuit combined with an analogue computer". In : *Journal of theoretical biology* 5.3 (1963), p. 412–425.
- [Sno98] El Houssine SNOUSSI. "Necessary conditions for multistationarity and stable periodicity". In : *Journal of Biological Systems* 6.01 (1998), p. 3–9.
- [Sou03] Christophe SOULÉ. "Graphic requirements for multistationarity". In : *ComplexUs* 1.3 (2003), p. 123–133.
- [Sou06] Christophe SOULÉ. "Mathematical approaches to differentiation and gene regulation". In : *Comptes rendus biologiques* 329.1 (2006), p. 13–20.
- [SSP06] Zoltan SZALLASI, Jörg STELLING et Vipul PERIWAL. "System modeling in cellular biology". In : *From Concepts to* (2006).
- [ST93] E. SNOUSSI et R. THOMAS. "Logical identification of all steady states : the concept of feedback loop characteristic states". In : *Bull. Math. Biol.* 55(5) (1993), p. 973–991.
- [Ste94] William J. STEWART. *Introduction to the numerical solution of Markov chains*. Princeton, NJ : Princeton Univ. Press, 1994. XIX, 539. ISBN : 0691036993. URL : http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+152880593&sourceid=fbw_bibsonomy.
- [Str14] Adam STRECK. *Model building for Parsybone Version 2.1*. Discrete Biomathematics, FU Berlin. 2014.
- [Sug61] Motoyosi SUGITA. "Functional analysis of chemical systems in vivo using a logical circuit equivalent." In : *Journal of theoretical biology* 1 (1961), p. 415–430.
- [Sug63] Motoyosi SUGITA. "Functional analysis of chemical systems in vivo using a logical circuit equivalent. II. The idea of a molecular automaton". In : *Journal of Theoretical Biology* 4.2 (1963), p. 179–192. ISSN : 0022-5193. DOI : [http://dx.doi.org/10.1016/0022-5193\(63\)90027-4](http://dx.doi.org/10.1016/0022-5193(63)90027-4). URL : <http://www.sciencedirect.com/science/article/pii/0022519363900274>.
- [Tar72] Robert TARJAN. "Depth-first search and linear graph algorithms". In : *SIAM journal on computing* 1.2 (1972), p. 146–160.
- [TCT93] D. THIEFFRY, M. COLET et R. THOMAS. "Formalisation of Regulatory Networks : a Logical Method and Its Automation". In : *Math. Modelling and Sci. Computing* 2 (1993), p. 144–151.
- [Td90] R. THOMAS et R. D'ARI. *Biological Feedback*. CRC Press, 1990.

- [TGL76] René THOMAS, Anne-Marie GATHOYE et Lucie LAMBERT. "A complex control circuit. Regulation of immunity in temperate bacteriophages." In : *European journal of biochemistry/FEBS* 71.1 (1976), p. 211.
- [Tho71] René THOMAS. "Regulation of gene expression in bacteriophage lambda". In : *Current Topics in Microbiology and Immunology/Ergebnisse der Mikrobiologie und Immunitätsforschung*. Springer, 1971, p. 13–42.
- [Tho73] René THOMAS. "Boolean formalization of genetic control circuits". In : *Journal of theoretical biology* 42.3 (1973), p. 563–585.
- [Tho79] René THOMAS. "Kinetic logic : a boolean analysis of the dynamic behaviour of control circuits". In : *Kinetic Logic A Boolean Approach to the Analysis of Complex Regulatory Systems*. Springer, 1979, p. 107–126.
- [Tho81] René THOMAS. "On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations". In : *Numerical methods in the study of critical phenomena*. Springer, 1981, p. 180–193.
- [Tho91] René THOMAS. "Regulatory networks seen as asynchronous automata : A logical description". In : *Journal of Theoretical Biology* 153.1 (1991), p. 1–23. issn : 0022-5193. DOI : [http://dx.doi.org/10.1016/S0022-5193\(05\)80350-9](http://dx.doi.org/10.1016/S0022-5193(05)80350-9). URL : <http://www.sciencedirect.com/science/article/pii/S0022519305803509>.
- [Tho98] René THOMAS. "Laws for the dynamics of regulatory networks". In : *International Journal of Developmental Biology* 42 (1998), p. 479–485.
- [TT95] D. THIEFFRY et R. THOMAS. "Dynamical behaviour of biological regulatory networks - II. Immunity control in bacteriophage lambda." In : *Bull. Math. Biol.* 57.2 (1995), p. 277–97.
- [TTK95] R. THOMAS, D. THIEFFRY et M. KAUFMAN. "Dynamical behaviour of biological regulatory networks - I. Biological role of feedback loops an practical use of the concept of the loop-characteristic state." In : *Bull. Math. Biol.* 57.2 (1995), p. 247–76.
- [UW11] Mukhtar ULLAH et Olaf WOLKENHAUER. *Stochastic approaches for systems biology*. Springer Science & Business Media, 2011.
- [Val+99] Amy J VALLIS, Timothy L YAHR, Joseph T BARBIERI et Dara W FRANK. "Regulation of ExoS Production and Secretion by *Pseudomonas aeruginosa* in Response to Tissue Culture Conditions". In : *Infection and immunity* 67.2 (1999), p. 914–920.
- [Wat+13] J.D. WATSON, T.A. BAKER, S.P. BELL et R. LOSICK. *Molecular Biology of the Gene*. Always learning. Pearson, 2013. isbn : 9780321851499.
- [Wol+04] Olaf WOLKENHAUER, Mukhtar ULLAH, Walter KOLCH et Kwang-Hyun CHO. "Modeling and simulation of intracellular dynamics : choosing an appropriate framework". In : *NanoBioscience, IEEE Transactions on* 3.3 (2004), p. 200–207.
- [YY71] Gad YAGIL et Ezra YAGIL. "On the relation between effector concentration and the rate of induced enzyme synthesis". In : *Biophysical Journal* 11.1 (1971), p. 11–27.

Annexes

Mode d'emploi de SPuTNIk

SPuTNIk¹ est le nom du prototype que nous avons mis au point pour inférer les paramètres de modèles de Thomas (selon la méthode présentée dans la partie I de ce manuscrit, à partir de la p.9). Un bref aperçu de SPuTNIk est donné dans le chapitre 4 (p.71) ; dans cette annexe, nous proposons un pas-à-pas plus complet et illustré pour prendre en main notre outil.

A.1 Entrée des données

Les entrées possibles de SPuTNIk sont :

- un graphe d'interactions,
- une ou plusieurs formules LTL (facultatif),
- des contraintes sur les paramètres (facultatif).

Il existe deux modes d'utilisation de SPuTNIk : avec ou sans interface graphique. L'interface permet d'entrer, modifier, visualiser facilement les données, elle est donc conseillée pour mettre au point les modèles et tester l'outil. Elle permet d'importer et d'exporter des modèles dans un fichier.

Sans interface, SPuTNIk attend en entrée un fichier décrivant le modèle ; ce fichier peut avoir été exporté précédemment en utilisant l'interface graphique ou être écrit indépendamment de SPuTNIk à condition de respecter la syntaxe attendue (décrite dans cette annexe en section A.1.2). Dans ce mode, le modèle n'est pas modifiable.

A.1.1 Utilisation de l'interface

Cette interface a été ajoutée à l'outil pour faciliter l'entrée des données dans le format attendu. Elle repose sur les items suivants :

1. *Symbolic Parameters of Thomas' Networks Inference*

- entrée manuelle réduite au minimum (nom des gènes);
- entrée des interactions et des connaissances biologiques guidées (liste déroulantes, changement d'état des boutons suivant ce qui peut être ajouté);
- possibilité de retenir des propositions LTL pour les réutiliser.

Elle est composée de plusieurs écrans successifs, correspondant aux différents types d'entrées que peut traiter SPuTNIk, d'une fenêtre récapitulative avant de lancer l'exécution et, enfin, d'un pop-up affichant le résultat à la fin du traitement.

La fonction d'import/export permet d'utiliser un modèle créé précédemment avec SPuTNIk ou de manière indépendante; le modèle peut alors être modifié si besoin, voire réenregistrer.

Spécification du graphe d'interaction

La première étape consiste à entrer les noms des gènes G et les caractéristiques des interactions I du graphe d'interaction $\Gamma = (G, I)$. (figure A.1).

FIGURE A.1 – 1ère étape : formulaire pour décrire le graphe d'interaction

Le nom des gènes doit être ajouté dans le cartouche prévu à cet effet et validé en appuyant sur la touche « Entrée » ou sur le bouton **Ok**. Un panneau de suppression apparaît alors, et le gène est ajouté dans les listes déroulantes des interactions.

Pour supprimer un gène qui a été ajouté, cocher la case correspondante et cliquer sur **Delete**.

Une interaction (g, s, t, g') dans I est composée d'un gène régulateur g (**Regulator**) qui active ($s = +$ correspond au choix **Sign (+)**) ou inhibe ($s = -$ correspond à **Sign (-)**) un gène régulé g' (**Regulated**), à partir d'un seuil t (**Threshold**). Donner ces quatre informations grâce aux listes déroulantes (remplies automatiquement en utilisant les noms des gènes entrés précédemment) et en modifiant le seuil à la valeur voulue grâce aux boutons **<** (diminuer d'une unité) et **>**

(augmenter d'une unité). Une fois ajoutée grâce au bouton , l'interaction est ajoutée dans la liste située dans le cadre apparaissant en-dessous.

Pour supprimer une interaction, cocher la case correspondante et cliquer sur .

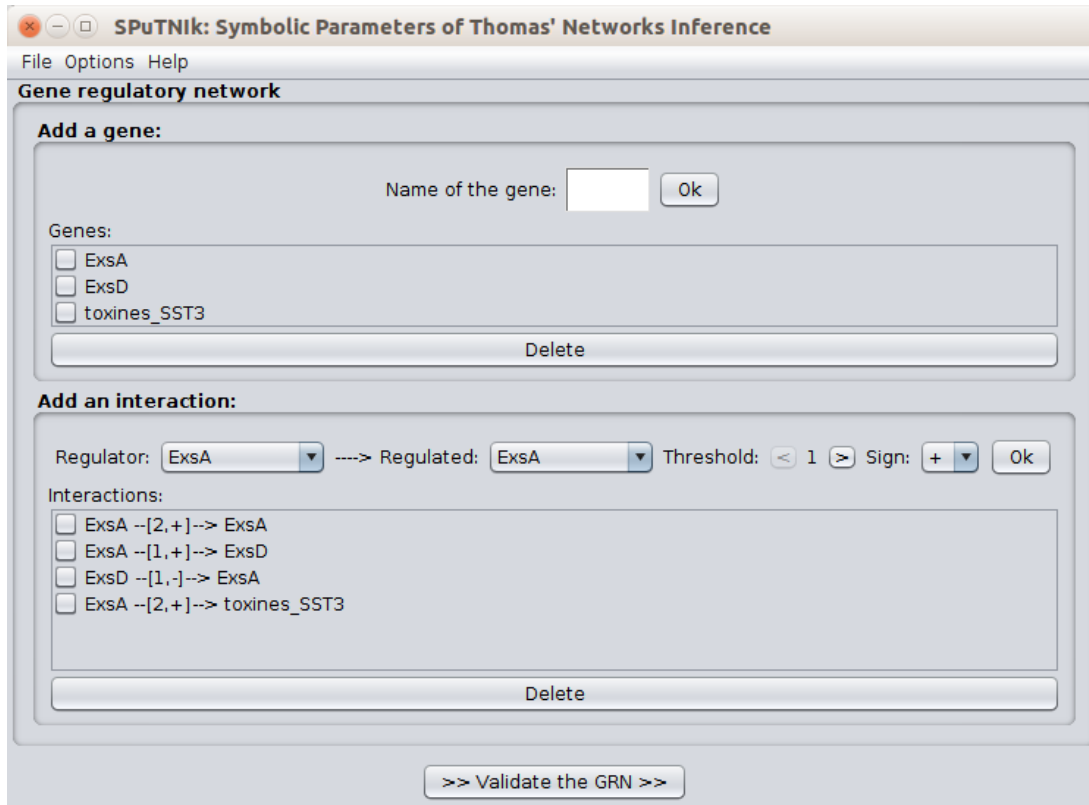


FIGURE A.2 – Graphe d'interaction complété

Pour passer à l'étape suivante, au moins une interaction doit avoir été ajoutée. Une fois toutes les interactions entrées (figure A.2), cliquer sur le bouton .

Ajout des formules LTL

Cette action a pour effet d'afficher la deuxième étape, i.e. l'entrée des formules LTL $(\varphi_i)_{i \in \mathbb{N}}$ correspondant aux propriétés biologiques attendues (figure A.3).

Cette étape est facultative : si aucune formule LTL n'est donnée, SPuTNI_K générera tous les jeux de paramètres \mathcal{K} possibles. Cela peut être utile dans le cas où des contraintes sur les paramètres (à valeur dans $Contrainte(\mathbb{K})$) sont ajoutées ou au minimum si la CPBF (cf. section 2.3.5, définition 2.3.9 p.40) est prise en compte.

Le panneau d'ajout des formules LTL contient plusieurs composants différents, décrits ci-dessous de haut en bas :

- des boutons correspondant aux états dans \mathbb{X} ;

FIGURE A.3 – 2ème étape : formulaire pour entrer des formules LTL

- un sélecteur d'entier ;
- des boutons associés aux opérateurs temporels, logiques et de comparaison ;
- des boutons permettant de réutiliser des formules enregistrées (à condition qu'il y en ait au moins une) ;
- le cartouche d'affichage de la formule LTL en cours d'entrée ;
- le choix de la portée de la formule (soit sur tous les chemins, soit sur au moins un chemin) ;
- un bouton d'enregistrement de la formule en cours d'entrée (pour la mettre de côté en vue de sa réutilisation) ;
- un bouton d'effacement de la formule en cours d'écriture ;
- un bouton de validation de la formule ;
- un panneau d'affichage et de suppression des formules déjà entrées (s'il y a au moins une formule entrée).

Par simplicité, les états ont la même notation que le gène associé (par exemple, l'état associé au gène *ExsA* est noté aussi *ExsA* et non x_{ExsA} comme c'était l'usage dans ce manuscrit). Les boutons correspondant sont automatiquement ajoutés en fonction des données de l'étape précédente.

Les boutons sont associés aux opérateurs selon la correspondance suivante :

- \boxed{G} , \boxed{F} , \boxed{X} , \boxed{U} et \boxed{R} correspondent aux opérateurs temporels respectifs :

- G, F, X, U et R;**
- , , , et symbolisent, dans l'ordre, les opérateurs logiques $\neg, \wedge, \vee, \Rightarrow$ et \Leftrightarrow ;
 - , , , , , sont associés aux opérateurs de comparaison $=, \neq, \leq, \geq, < \text{ et } >$;
 - et représentent les parenthèses (et).

L'entrée des formules se fait par l'utilisation des différents boutons. Dans le cas d'une formule longue ou de parties répétitives dans une ou plusieurs formules, le bouton offre la possibilité de retenir des bribes de formules pour les ajouter ensuite.

L'analyse de l'entrée se fait au fur et à mesure par le programme, grâce à la construction d'un arbre représentant les formules internes. Dès qu'une formule interne est jugée complète (opérateur et argument(s) présents, parenthèses fermées), la partie correspondante de l'arbre est dite résolue et l'élément englobant est analysé de façon récursive pour déterminer s'il peut ou non être résolu avec les informations disponibles.

Pour faciliter l'analyse, il n'est pas permis à l'utilisateur de modifier la formule à la main ni de revenir en arrière ; pour pouvoir enregistrer ou valider une formule, toutes les parenthèses doivent par ailleurs être présentes. En fonction de ce qui a été entré précédemment, les boutons sont ou non sélectionnables par l'utilisateur. Ce cadre permet de guider la saisie de l'utilisateur afin qu'il ne puisse entrer que des formules pouvant être associées à un automate de Büchi (cela ne garantit pas par contre qu'elles correspondent au sens voulu ni même qu'elles aient un sens).

La grammaire mise en place est la suivante :

$$\begin{aligned}
 \langle \text{Formule} \rangle &::= \langle \text{Unaire} \rangle " (" \langle \text{Formule} \rangle ") " \\
 &| " (" \langle \text{Formule} \rangle ") " \langle \text{Binaire} \rangle " (" \langle \text{Formule} \rangle ") " \\
 &| " (" \langle \text{Etat} \rangle \langle \text{Comparateur} \rangle \langle \text{Valeur} \rangle ") " \\
 &| " (" \langle \text{Formule} \rangle ") " \\
 \langle \text{Unaire} \rangle &::= "G" | "F" | "X" | "not" \\
 \langle \text{Binaire} \rangle &::= "U" | "R" | "and" | "or" | "->" | "<->" \\
 \langle \text{Etat} \rangle &::= \text{String} \\
 \langle \text{Comparateur} \rangle &::= "=" | "!=" | "<=" | ">=" | "<" | ">" \\
 \langle \text{Valeur} \rangle &::= \text{int}
 \end{aligned}$$

Comme on l'a vu, plusieurs formules LTL peuvent être ajoutées au modèle. S'il contient plusieurs formules, notées φ_A , dont « le comportement doit être vrai sur tous les chemins », SPvTNIk effectuera la conjonction de ces formules ($\varphi = \bigwedge \varphi_A$) au début du traitement pour pouvoir la traiter comme une formule unique φ (soit un seul automate de Büchi $B_{\neg\varphi}$, et donc un seul PGRN).

Par contre, toutes les formules dont « le comportement doit être vrai sur au moins un chemin »,

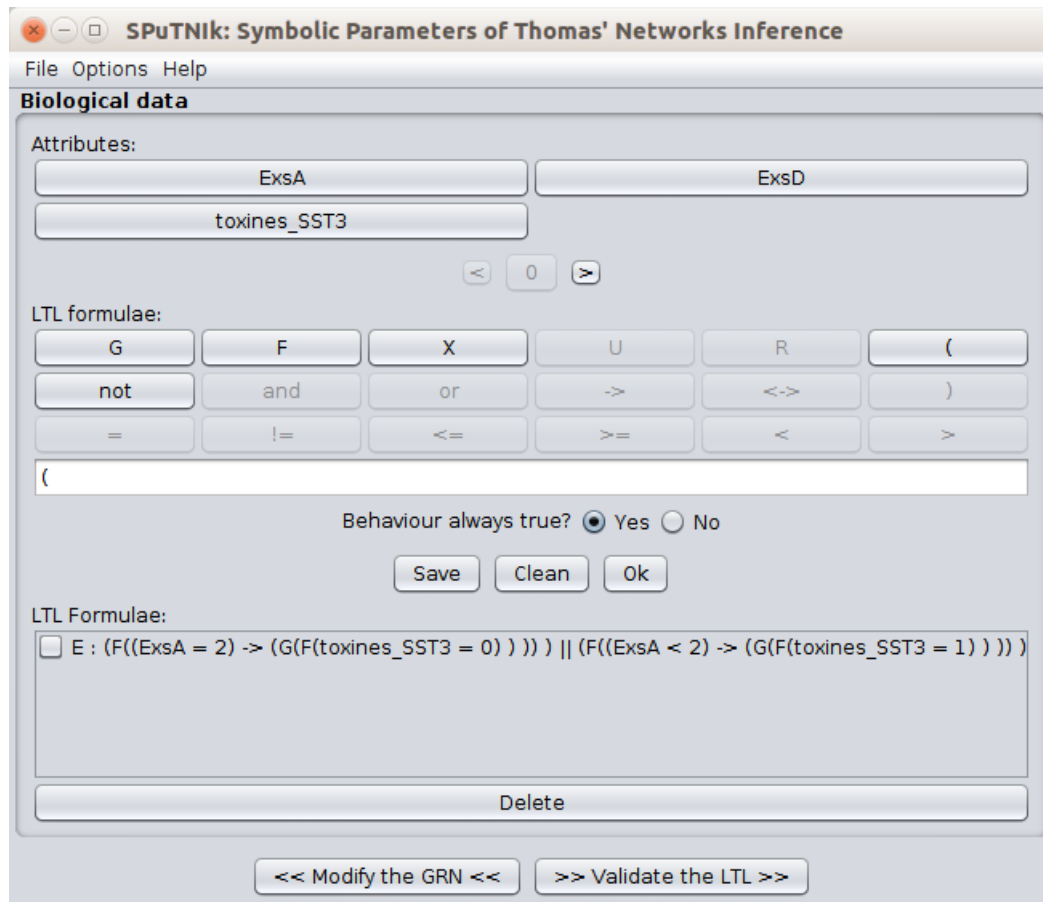


FIGURE A.4 – Connaissances biologiques complétées

notées φ_E , seront traitées séparément (soit un automate de Büchi B_φ par formule φ dans φ_E , et donc $|\varphi_E|$ PGRN à construire).

Le bouton **<< Modify the GRN <<** permet de retourner à l'étape précédente pour modifier le graphe d'interaction. Les formules LTL validées ne sont conservées que si aucun gène associé n'est supprimé. Ainsi, si on considère un graphe d'interaction initial $\Gamma = (G, I)$ et une formule LTL φ , si l'utilisateur revient à l'étape précédente et modifie les composantes de Γ de manière à obtenir un nouveau graphe d'interaction $\Gamma' = (G', I')$, la formule φ est conservée seulement si pour tout état x_g dans $VAR(\varphi)$ tel que x appartient à \mathbb{X} , g appartient à G' .

Quand toutes les formules LTL souhaitées ont été entrées et validées (figure A.4), cliquer sur **>> Validate the LTL >>**.

Réduction du domaine des paramètres

La troisième étape s'affiche alors (figure A.5). Cette étape est facultative, elle permet d'indiquer des contraintes sur les paramètres en utilisant les opérateurs de comparaison habituels.

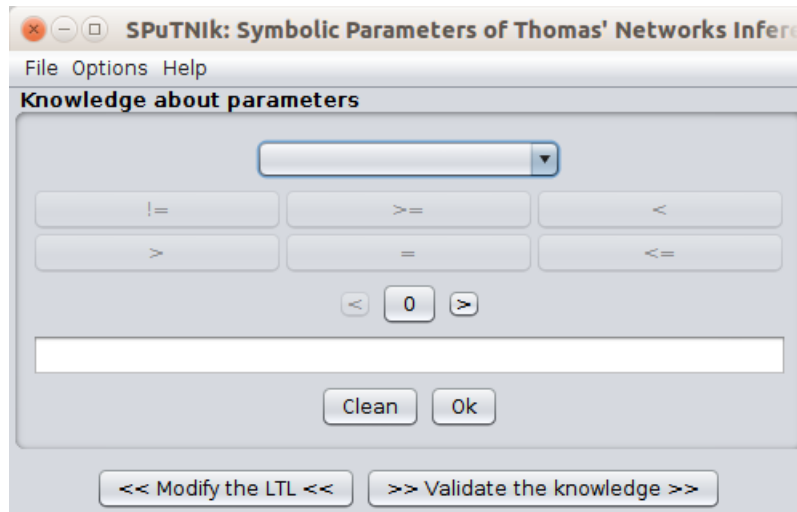


FIGURE A.5 – 3ème étape : ajout de connaissance sur les paramètres

Une liste déroulante préremplie contenant les noms des paramètres dans \mathbb{K} , permet de sélectionner le ou les paramètres que l'on veut contraindre (figure A.6). Les opérateurs sont sélectionnables après avoir choisi un opérateur (un paramètre ou un entier). Le deuxième opérateur peut être lui aussi un paramètre ou un entier.

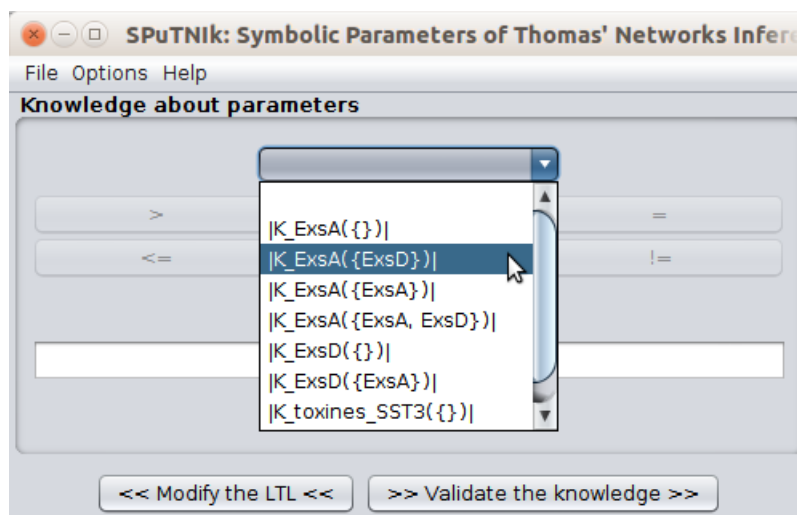


FIGURE A.6 – 3ème étape : formulaire pour ajouter des contraintes sur les paramètres

Comme précédemment, deux boutons permettent de revenir à l'étape précédente () ou de passer à l'étape suivante ().

Récapitulatif et lancement du traitement

L'interface obtenue après validation de l'étape 3 est présentée dans la figure A.7.

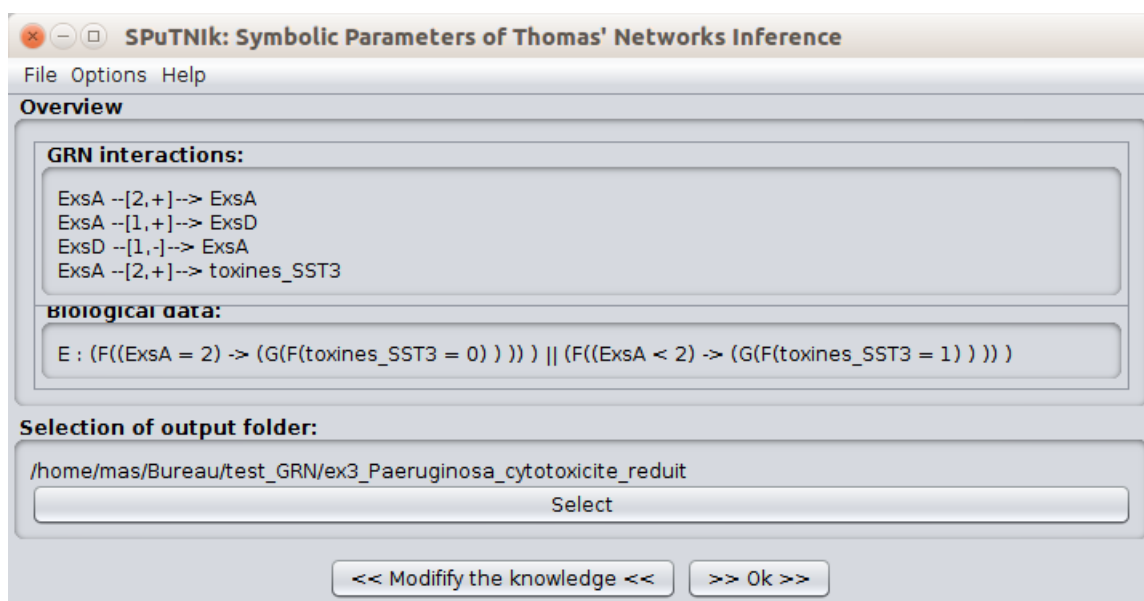


FIGURE A.7 – Etape finale : Récapitulatif du modèle et sélection du dossier de destination

Dans le haut du panneau, un récapitulatif des données entrées (graphe d'interaction et formules LTL) est présenté, pour une dernière vérification avant le lancement du traitement. Les formules LTL sont triées de manière à pouvoir différencier facilement le comportement associé (« vrai sur tous les chemins » ou « vrai sur au moins un chemins »).

Si une modification du modèle est voulu, cliquer sur le bouton pour revenir aux étapes précédentes.

Le deuxième cartouche indique le dossier de destination dans lequel les résultats de SPuTNIk seront écrits. Pour choisir un autre dossier, cliquer sur le bouton .

Enfin, cliquer sur le bouton pour lancer le traitement de SPuTNIk.

Traitement et résultats

L'IHM va alors se fermer pendant le traitement (cela permet de libérer des ressources, qui seraient inutilement consommées par un message du type « *En cours de traitement...* »). Les étapes du traitement peuvent être suivies via la console, celle-ci va afficher plus ou moins d'informations suivant le paramétrage de l'outil.

Quand le traitement sera terminé, une nouvelle fenêtre s'ouvrira (figure A.8), indiquant le nombre de solutions trouvées et le temps dépensé, et invitant à consulter le fichier de résultat. Fermer cette fenêtre terminera le programme.

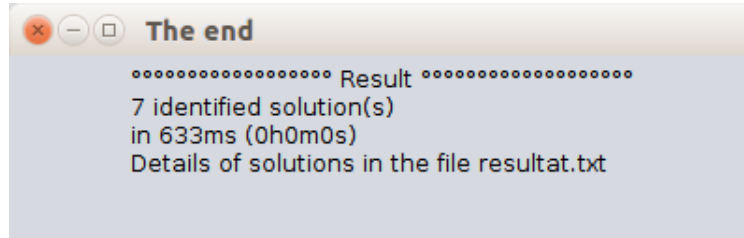


FIGURE A.8 – Affichage des résultats

A.1.2 En ligne de commande

SPVTNIK peut également être lancé sans utiliser l'interface, en ligne de commande. Les options suivantes sont utilisables :

- cteDefObs** (valeur par défaut : *true*)
Utilisation des contraintes de définition et d'observation ;
- cteInitAttributs** (*false*)
Utilisation d'une contrainte initiale sur les attributs ;
- cteMinMax** (*true*)
Utilisation de la contrainte min/max ;
- d** (*chemin de SPVTNIK*)
Dossier de destination des fichiers de sortie ;
- enumSol, -enumerate-Solutions** (*true*)
Énumération des solutions lors de la phase finale de résolution ;
- f** ()
Fichier du modèle ;
- h, -help, -?** ()
Affichage de l'aide ;
- i, -use-Interface** (*true*)
Utilisation de l'interface graphique ;
- m** (*1000*)
Modulo d'affichage du parcours des nœuds pendant le traitement ;
- maxMs** (*0*)
Temps d'exécution max (en ms) (0: pas de limite) ;

-maxNoeuds (0)

Nombre de nœuds max à parcourir (0 : pas de limite);

-l, -language (EN)

Interface en anglais (EN) ou en français (FR);

-seeSol, -see-Solutions (*false*)

Génération du graphe de parcours correspondant aux solutions (à condition que le nombre de solutions soit inférieur à 500);

-v, -version ()

Version courante de SPuTNIK;

-vb, -verbose (1)

Mode verbeux (trois valeurs au choix : 0, 1 ou 2 avec 0 représentant le mode non verbeux).

A.2 Fichiers de sortie

SPuTNIK va générer plusieurs fichiers de sortie, à la fois son exécution et à la fin du traitement. En voici la liste :

- **GRN.dot** : graphe du GRN au format dot
- **automateBuchi_i.dot** : graphe de l'automate de Büchi i^2 au format dot
- **APTS_i.dot** : graphe du Produit i au format dot
- **detail_lti.txt** : détail des nœuds parcourus lors de l'exécution symbolique de l'APTS de i et liste des contraintes acceptées
- **resultat.txt** : valeurs des paramètres des modèles solutions, nombre de solutions et temps de traitement

Ces fichiers sont générés dans le dossier indiqué lors de la validation des données (si l'interface a été utilisée) ou directement dans le dossier à partir duquel a été lancé l'application si le dossier de destination n'a pas été indiqué en ligne de commande (cas sans interface).

Les fichiers DOT peuvent être utilisés pour tracer les graphes correspondants, par exemple en utilisant Graphviz³ (logiciel de visualisation des graphes, en particulier de ce format).

2. i représente la formule LTL correspondante. Les formules du premier type sont regroupées en une seule formule avant le traitement; celles du deuxième type sont par contre traitées séparément.

3. Graphviz : <http://graphviz.org/>

Etudes de cas traitées avec SPuTNIk

Les études de cas rassemblées dans cette annexe sont des exemples hétérogènes tirés de la littérature permettant de présenter les caractéristiques de notre outil SPuTNIk. Le contexte de chaque étude n'est pas rappelé ici, le lecteur est invité à consulter leur source pour le connaître.

B.1 Le cycle cellulaire des mammifères

B.1.1 Version 1 tirée de [Fau+06]

Dans [Fau+06], FAURÉ et al. étudient un GRN contrôlant le cycle cellulaire des mammifères. Le réseau étudié est un réseau booléen, composé de 10 gènes liés par 35 interactions ; son graphe d'interaction est présenté en figure B.1.

Il y a 206 paramètres attachés à ce GRN, engendrant un nombre initial de solutions de l'ordre de 10^{62} .

En se basant sur la littérature, FAURÉ et al. ont spécifié des règles logiques sur les paramètres, permettant de restreindre leur valeur :

- $Rb := \overline{CycD} \wedge \overline{CycB} \wedge (p27 \vee (\overline{CycE} \wedge \overline{CycA}))$
- $E2F := \overline{Rb} \wedge \overline{CycB} \wedge (\overline{CycA} \vee p27)$
- $CycE := E2F \wedge \overline{Rb}$
- $CycA := \overline{Rb} \wedge \overline{Cdc20} \wedge \overline{Cdh1} \wedge \overline{UbcH10} \wedge (E2F \vee CycA)$
- $p27 := \overline{CycD} \wedge \overline{CycB} \wedge ((\overline{CycE} \wedge \overline{CycA}) \vee (p27 \wedge \overline{CycE} \wedge \overline{CycA}))$
- $Cdc20 := CycB$
- $Cdh1 := Cdc20 \vee (\overline{CycB} \wedge (\overline{CycA} \vee p27))$
- $UbcH10 := \overline{Cdh1} \vee (UbcH10 \wedge (Cdc20 \vee CycA \vee CycB))$
- $CycB := \overline{Cdc20} \wedge \overline{Cdh1}$

Nous avons utilisé cette spécification pour vérifier si nous obtenions des solutions avec

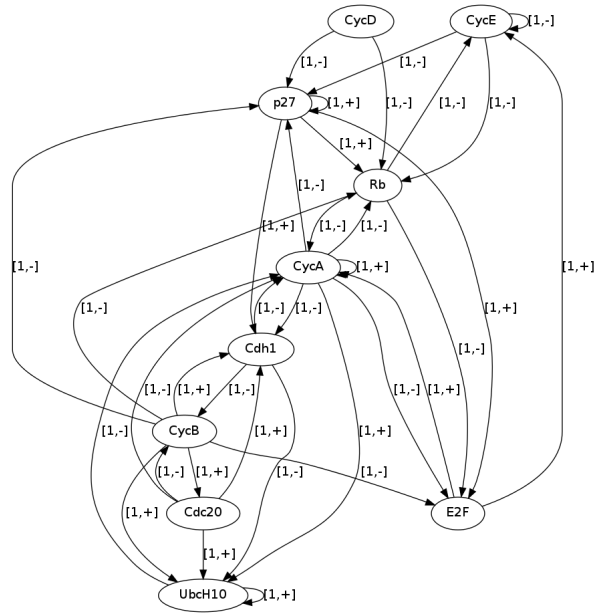


FIGURE B.1 – Graphe d’interaction Γ_{ccm1} lié au cycle cellulaire des mammifères.

SPuTNIk. Pour cela, nous avons utilisé la contrainte Min/Max ainsi que les contraintes de définition et d’observation, mis à part pour les gènes *CycE* et *Cdh1*, pour lesquels nous les avons désactivées car elles n’étaient pas compatibles avec la spécification sur les paramètres.

Nous obtenons finalement avec SPuTNIk une seule solution, qui permet de confirmer la validité du graphe d’interaction et des hypothèses tirées de la littérature qui ont été prises en compte dans la spécification sur les paramètres.

B.1.2 Version 2 tirée de [Kla+12b]

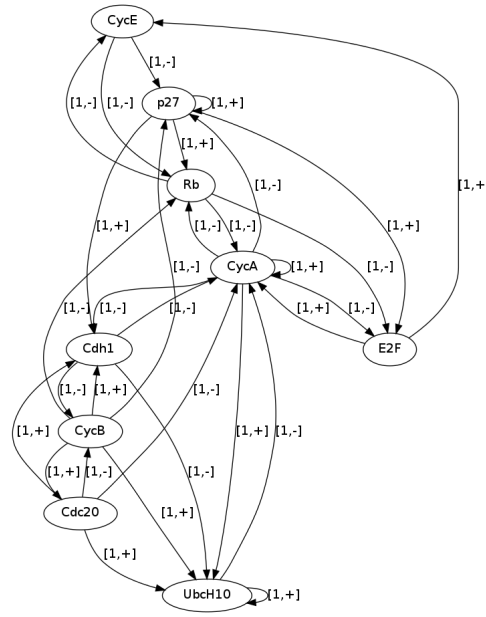
Dans [Kla+12b], KLARNER et al. ont repris l’étude de FAURÉ et al. dans [Fau+06], en simplifiant néanmoins le graphe d’interaction, en réduisant celui-ci à 9 gènes et 31 interactions (figure B.2). Le nombre de paramètres est ainsi réduit à 162 paramètres, correspondant à un espace de solutions initial de l’ordre de 6.10^{48} possibilités.

KLARNER et al. ont choisi de réduire l’espace des solutions en utilisant partiellement la spécification sur les paramètres de FAURÉ et al. ; les deux contraintes qu’ils ont gardées sont les suivantes :

- $CycA := \overline{Rb} \wedge \overline{Cdc20} \wedge \overline{Cdh1} \wedge \overline{UbcH10} \wedge (E2F \vee CycA)$
- $UbcH10 := \overline{Cdh1} \vee (UbcH10 \wedge (Cdc20 \vee CycA \vee CycB))$

Grâce à ces deux contraintes, ils affirment ainsi réduire le nombre de possibilités d’instanciation des paramètres à un peu moins de 700 millions.

KLARNER et al. utilisent les séries temporelles suivantes pour réaliser leur analyse du réseau :

FIGURE B.2 – Graphe d'interaction Γ_{ccm2} lié au cycle cellulaire des mammifères.

- $m1 : Rb = 0 \wedge E2F = 0 \wedge CycE = 0 \wedge CycA = 0 \wedge p27 = 0 \wedge Cdc20 = 1 \wedge Cdh1 = 1 \wedge UbcH10 = 1 \wedge CycB = 0$
- $m2 : Rb = 0 \wedge E2F = 1 \wedge CycE = 0 \wedge CycA = 0 \wedge p27 = 0 \wedge Cdc20 = 0 \wedge Cdh1 = 1 \wedge UbcH10 = 1 \wedge CycB = 0$
- $m3 : Rb = 0 \wedge E2F = 1 \wedge CycE = 1 \wedge CycA = 0 \wedge p27 = 0 \wedge Cdc20 = 0 \wedge Cdh1 = 1 \wedge UbcH10 = 0 \wedge CycB = 0$
- $m4 : Rb = 0 \wedge E2F = 1 \wedge CycE = 1 \wedge CycA = 1 \wedge p27 = 0 \wedge Cdc20 = 1 \wedge Cdh1 = 1 \wedge UbcH10 = 0 \wedge CycB = 0$
- $m5 : Rb = 0 \wedge E2F = 0 \wedge CycE = 1 \wedge CycA = 1 \wedge p27 = 0 \wedge Cdc20 = 0 \wedge Cdh1 = 0 \wedge UbcH10 = 0 \wedge CycB = 0$
- $m6 : Rb = 0 \wedge E2F = 0 \wedge CycE = 0 \wedge CycA = 1 \wedge p27 = 0 \wedge Cdc20 = 0 \wedge Cdh1 = 0 \wedge UbcH10 = 1 \wedge CycB = 1$
- $m7 : Rb = 0 \wedge E2F = 0 \wedge CycE = 0 \wedge CycA = 1 \wedge p27 = 0 \wedge Cdc20 = 1 \wedge Cdh1 = 0 \wedge UbcH10 = 1 \wedge CycB = 1$

La formule LTL que nous obtenons à partir de cette série temporelle est :

$$\phi : Em1 \wedge F(m2 \wedge F(m3 \wedge F(m4 \wedge F(m5 \wedge F(m6 \wedge F(m7 \wedge F(m1))))))))$$

Avec SPuTNIk, en utilisant la contrainte Min/Max ainsi que les deux contraintes sur les paramètres conservées par KLARNER et al., nous avons évalué que le nombre initial de paramètres

inconnus est égal à 52, soit un espace de solution initial de l'ordre de 5.10^{15} possibilités : nous sommes donc bien loin du nombre annoncé par KLARNER et al. (pour rappel, 7.10^8).

En exécutant cette étude de cas, SPuTNIk a dû s'arrêter avant d'avoir terminé à cause d'une saturation mémoire que nous n'avons pas pu résoudre sur la machine utilisée. SPuTNIk a tourné pendant environ 200h avant d'atteindre cet état, soit un temps inférieur à celui que Parsybone a utilisé (en sommant les temps utilisés par chacun des 8 processus indépendants) selon [Kla+12b]. De la mémoire supplémentaire serait donc nécessaire pour pouvoir réaliser cette étude de cas. Une autre solution serait d'effacer au cours du parcours les données de certains nœuds (appartenant à une branche entièrement traitée par exemple), avec le risque d'augmenter au final le nombre de nœuds parcourus (et donc le temps d'exécution) car certaines coupures ne se seraient plus réalisées en l'absence de ces nœuds.

Table des matières

Résumé	v
Remerciements	vii
Sommaire	ix
Introduction	1
Contexte	1
Organisation du manuscrit	4
Contributions	6
I Inférence de paramètres de modèles de Thomas par vérification de propriétés LTL	7
1 Préliminaires	11
1.1 Autour des ensembles	11
1.2 Contraintes sur les entiers	13
1.3 Éléments de la théorie des langages	18
1.4 Systèmes de transition	18
1.5 Logique Temporelle Linéaire	20
2 Présentation des réseaux de régulation génétique	23
2.1 Contexte : les réseaux de régulation génétique	24
2.2 État de l’art de la modélisation de GRN	26
2.2.1 Modélisation par système d’ODE	26
2.2.2 Modélisation discrète booléenne	27
2.2.3 Modélisation discrète multivaluée	28
2.3 Le modèle multivalué de R. Thomas	32
2.3.1 Graphe d’interactions	33
2.3.2 Espace des états	35
2.3.3 Dynamique	35
2.3.4 Paramètres du modèle de Thomas	37
2.3.5 Contraintes sur les paramètres	40
2.4 Sélection de modèles à partir d’observations temporelles	41

3	Méthode d'inférence de paramètres de Thomas basée sur le model-checking LTL et la résolution de contraintes	47
3.1	GRN Paramétré	49
3.1.1	Caractérisation des états par contraintes	49
3.1.2	GRN Paramétré	51
3.1.3	Dynamiques associées à un PGRN	54
3.2	Synchronisation d'un PGRN avec un automate de Büchi	58
3.3	Exécution symbolique du Produit	60
3.3.1	Vue d'ensemble	60
3.3.2	Construction des Arbres d'Exécution	62
3.3.3	Recherche des cycles acceptants	66
3.4	Inférence des paramètres	67
4	Implémentation	71
4.1	SPuTNIk	71
4.1.1	Entrées et sorties de SPuTNIk	72
4.1.2	Informations techniques	73
4.2	Algorithme de simplification du produit	73
4.3	Algorithme d'exécution des arbres	75
4.4	Pistes d'optimisation	79
4.4.1	Parallélisation du parcours	79
4.4.2	Autres pistes	83
5	Etudes de cas	85
5.1	Inductibilité de la cytotoxicité de <i>Pseudomonas aeruginosa</i>	86
5.1.1	Version minimale	88
5.1.2	Version étendue	94
5.2	Cycle de vie du phage λ	98
5.2.1	Version 1	100
5.2.2	Version 2 : séries temporelle	102
II Analyse de la voie de signalisation Wnt/β-caténine à l'aide du model-checking statistique HASL		105
6	Préliminaires	109
6.1	Éléments de la théorie des probabilités	109
6.2	Les processus stochastiques à événements discrets	112
6.3	Réseaux de Petri stochastiques généralisés	118
6.4	Modélisation de systèmes de réactions biochimiques	123
6.4.1	Sémantique déterministe.	125
6.4.2	Sémantique stochastique.	125
6.4.3	Relation entre les modèles déterministes et les modèles stochastiques	126
6.5	Analyse des modèles	127
6.6	Model-checking statistique avec la logique HASL	128
6.6.1	1ère composante du HASL : le LHA synchronisé	128
6.6.2	2ème composante du HASL : expression	133
6.6.3	COSMOS	135

7	Analyse de la voie de signalisation Wnt/β-caténine à l'aide du model-checking statistique HASL	137
7.1	Les voies de signalisation Wnt	138
7.2	Cas d'étude : la voie Wnt/ β -caténine	140
7.2.1	Fonctionnement général de la voie Wnt/ β -caténine	140
7.2.2	Modèle <i>core</i> de la voie Wnt/ β -caténine de Mazemondet	141
7.3	Construction d'une version stochastique du modèle <i>core</i> de Mazemondet.	142
7.4	Analyse du modèle Wnt <i>core</i>	144
7.4.1	Observation de la dynamique de β_{nuc} sur une seule trajectoire stochastique	145
7.4.2	Spécification formelle de mesures à travers le langage HASL	146
7.4.3	Mesure formelle des pics du signal β_{nuc} en utilisant l'automate \mathcal{A}_{peaks}	149
7.5	Conclusion	152
	Conclusion	155
	Bibliographie	161
	Annexes	173
A	Mode d'emploi de SPuTNIk	173
A.1	Entrée des données	173
A.1.1	Utilisation de l'interface	173
A.1.2	En ligne de commande	181
A.2	Fichiers de sortie	182
B	Etudes de cas traitées avec SPuTNIk	183
B.1	Le cycle cellulaire des mammifères	183
B.1.1	Version 1 tirée de [Fau+06]	183
B.1.2	Version 2 tirée de [Kla+12b]	184
	Table des matières	187

Titre : Techniques de model-checking pour l'inférence de paramètres et l'analyse de réseaux biologiques

Mots clefs : analyse formelle de réseaux biologiques, réseau de régulation génétique, modèle discret de Thomas, inférence de paramètres, model-checking LTL, voie Wnt/bêta-caténine, modélisation stochastique, model-checking HASL.

Résumé : Dans ce mémoire, nous présentons l'utilisation de techniques de model-checking pour l'inférence de paramètres de *réseaux de régulation génétique* (GRN) et l'analyse formelle d'une voie de signalisation.

Le cœur du mémoire est décrit dans la première partie, dans laquelle nous proposons une approche pour inférer les paramètres biologiques régissant les dynamiques de modèles discrets de GRN. Les GRN sont encodés sous la forme d'un méta-modèle, appelé *GRN paramétré*, de telle façon qu'une instance de paramètres définit un modèle discret du GRN initial. Sous réserve que les propriétés biologiques d'intérêt s'expriment sous la forme de formules LTL, les techniques de model-checking LTL sont combinées à celles d'exécution symbolique et de résolution de contraintes afin de sélectionner les modèles satisfaisant ces propriétés. L'enjeu est de contourner l'explosion combinatoire en terme de taille et de nombre de modèles discrets. Nous avons implémenté notre méthode en

Java, dans un outil appelé SP_UTNI_K.

La seconde partie décrit une collaboration avec des neuro-pédiatres, qui ont pour objectif de comprendre l'apparition du phénotype protecteur ou toxique des microglies (un type de macrophage du cerveau) chez les prématurés. Cette partie exploite un autre versant du model-checking, celui du model-checking statistique, afin d'étudier un type de réseau biologique particulier : la voie de signalisation Wnt/ β -caténine, qui permet la transmission d'un signal de l'extérieur à l'intérieur des cellules via une cascade de réactions biochimiques. Nous présentons ici l'apport du model-checker stochastique COSMOS, utilisant la *logique stochastique à automate hybride* (HASL), un formalisme très expressif nous permettant une analyse formelle sophistiquée des dynamiques de la voie Wnt/ β -caténine, modélisée sous la forme d'un processus stochastique à événements discrets.

Title : Model checking techniques for parameter inference and analysis of biological networks

Keywords : formal analysis of biological networks, genetic regulatory network, Thomas discrete modelling, parameters inference, LTL model checking, Wnt/ β -catenin pathway, stochastic modelling, HASL model checking.

Abstract : In this thesis, we present the use of model checking techniques for inference of parameters of *Gene Regulatory Networks* (GRNs) and formal analysis of a signalling pathway. In the first and main part, we provide an approach to infer biological parameters governing the dynamics of discrete models of GRNs. GRNs are encoded in the form of a meta-model, called *Parametric GRN*, such that a parameter instance defines a discrete model of the original GRN. Provided that targeted biological properties are expressed in the form of LTL formulas, LTL model-checking techniques are combined with symbolic execution and constraint solving techniques to select discrete models satisfying these properties. The challenge is to prevent combinatorial explosion in terms of size and number of discrete models. Our method is implemented in

Java, in a tool called SP_UTNI_K.

The second part describes a work performed in collaboration with child neurologists, who aim to understand the occurrence of toxic or protective phenotype of microglia (a type of macrophage in the brain) in the case of preemies. We use an other type of model-checking, the statistical model-checking, to study a particular type of biological network: the Wnt/ β -catenin pathway that transmits an external signal into the cells via a cascade of biochemical reactions. Here we present the benefit of the stochastic model checker COSMOS, using the *Hybrid Automata Stochastic Logic* (HASL), that is an very expressive formalism allowing a sophisticated formal analysis of the dynamics of the Wnt/ β -catenin pathway, modelled as a discrete event stochastic process.

