Université de la Méditerranée – Aix-Marseille II

Faculté des sciences de Luminy

Thèse

pour obtenir le grade de

Docteur de l'université d'Aix-Marseille II

Discipline: sciences de la vie, spécialité bioinformatique

Modélisation logique de la différenciation des lymphocytes T auxiliaires

Présentée et soutenue publiquement par

Aurélien NALDI

le 13 novembre 2009

Jury

Prof. Denis Thieffry	Université de la Méditerranée, Marseille	Directeur
Dr. Claudine Chaouiya	Université de la Méditerranée, Marseille et Instituto Gulbenkian de Ciência, Oeiras (Portugal)	Co-directrice
Prof. Alexander Bockmayr	Freie Universität Berlin (Allemagne)	Rapporteur
Prof. Hidde de Jong	Équipe INRIA Ibis, Grenoble	Rapporteur
Prof. Ioannis Xenarios	Swiss Institute of Bioinformatics, Lausanne (Suisse)	Examinateur
Prof. Pascal Rihet	Université de la Méditerranée, Marseille	Examinateur
Prof. Jorge Carneiro	Instituto Gulbenkian de Ciência, Oeiras (Portugal)	Invité

Remerciements

Je voudrais tout d'abord remercier mes directeurs de thèse Denis Thieffry et Claudine Chaouiya. En premier lieu pour avoir su me convaincre qu'une thèse était une façon agréable d'occuper le temps (dans l'ensemble ils avaient raison), mais surtout pour avoir su être toujours présents (des tâtonnements des débuts aux affres de la rédaction) tout en me laissant une grande liberté. Vos relectures attentives et votre gentillesse ont largement participé à rendre ces quelques années passionnantes.

D'autres membres du TAGC sont impliqués dans ce travail, en particulier Adrien Fauré qui a essuyé les plâtres de GINsim et réclamé de nouvelles fonctions, ainsi qu'Abibatou Mbodj, qui semble bien décidée à assurer la relève. Dans un autre registre, merci à Fabrice Lopez et à Duncan Berenguier d'avoir plongé dans le code de GINsim sans sourciller et implicitement accepté d'en assurer le service après vente.

Je voudrais également remercier Luca et Élodie, les autres membres de l'équipe, ainsi que l'ensemble du TAGC pour l'ambiance chaleureuse qui y règne. Cet environnement interdisciplinaire a été fort enrichissant, les séminaires de labo m'ont permis de découvrir divers aspects de la partie bio de "bioinformatique".

Merci également à Elisabeth Remy et Jorge Carneiro pour des discussions fructueuses dans différents domaines, ainsi qu'à Laurent Tichit, dont le bureau a été récemment converti en refuge pour "rédactant".

Merci à mes rapporteurs, Alexander Bockmayr et Hidde de Jong, ainsi qu'à Ioannis Xenarios et Pascal Rihet pour avoir accepté d'évaluer ce travail. J'attends vos commentaires avec impatience.

Merci également aux anciens de l'association resus et à tous ceux qui m'ont transmis leur passion et leur motivation, je n'aurais pas suivi ce parcours sans vous.

Enfin, merci à mes parents, frère et soeurs ainsi qu'à Natacha et à mes proches, pour leur soutien, mais surtout pour veiller à me maintenir (parfois) les pieds sur terre et m'aider à m'évader de mes habituelles considérations immatérielles !

Table des matières

Table des figures

Première partie

Introduction

Chapitre 1

Système immunitaire et lymphocytes T auxiliaires

1.1 Le système immunitaire des vertébrés

Le système immunitaire d'un organisme rassemble les éléments et mécanismes le protégeant des maladies dues aux pathogènes (bactéries, virus, parasites) ou aux cellules cancéreuses. Les mécanismes de défense sont généralement divisés en mécanismes spécifiques (réponse adaptative) et non-spécifiques (réponse innée). Ces types de réponses utilisent des mécanismes très différents mais chacun repose sur la capacité à distinguer, de façon plus ou moins sophistiquée, le soi des intrus potentiels. La figure 1.1 présente les principaux acteurs cellulaires de ce système.

Les défenses non-spécifiques existent, sous des formes variées, chez tous les organismes. Elles vont de simples barrières physiques empêchant l'intrusion à des mécanismes biochimiques et cellulaires dédiés à l'élimination de pathogènes.

Les barrières physiques sont notre première protection, que ce soit les membranes cellulaires, qui ne sont perméables qu'à certains produits, ou les barrières plus élaborées des organismes multicellulaires. Les plus communes sont passives : peau, coquille, exosquelette... D'autres barrières comme la toux, les larmes ou l'urine permettent d'expulser des déchets et agents irritants. Ces barrières sont perméables par essence, l'organisme ayant besoin d'échanger avec l'extérieur pour survivre. De nombreux pathogènes échappent donc à ces filtres rudimentaires.

À leur entrée dans un organisme, certains pathogènes endommagent des cellules, lesquelles sécrètent alors des substances chimiques (histamines, sérotonine, ...) déclenchant l'inflammation, point de départ de la seconde ligne de défense. Ces substances recrutent des agents réparateurs des tissus endommagés et stimulent la réponse immunitaire innée en provoquant la vasodilatation des vaisseaux sanguins et en attirant les phagocytes. Les phagocytes (cellules mangeuses) sont un sous-ensemble des leucocytes (globules blancs), composé en particulier des macrophages, neutrophiles et cellules dendritiques. Ces cellules disposent de récepteurs membranaires permettant de reconnaître un grand nombre de pathogènes. Ces récepteurs reconnaissent des motifs structuraux présents chez de nombreux pathogènes mais absents des cellules de l'organisme. Lorsqu'un phagocyte reconnaît un pathogène, il l'engloutit et l'enferme dans un lysosome avant de le digérer.

Les phagocytes jouent ensuite le rôle de cellules présentatrices d'antigène (APC pour Antigen Presenting Cell) : les peptides de pathogènes phagocytés sont présentés sur la membrane cellulaire des phagocytes, associés à un complexe protéique membranaire : le complexe majeur d'histocompatibilité (CMH, Major Histocompatibility Complex ou MHC en an-

glais). Cette présentation permet l'activation de la réponse spécifique. Les lymphocytes sont munis de récepteurs spécifiques capables de reconnaître un peptide particulier (antigène) associé au CMH. Cette reconnaissance d'antigène déclenche l'activation du lymphocyte. En dehors de leur grande spécificité, les réponses adaptatives sont caractérisées par une mémoire des invasions et se déclenchent donc plus rapidement lors d'une seconde invasion similaire. Plus généralement, les populations de cellules participant à la réponse spécifique fluctuent en fonction de l'environnement.

1.2 CMH et Lymphocytes

Nous venons de voir que le complexe majeur d'histocompatibilité (CMH) et les lymphocytes jouent, ensemble, un rôle important dans la mise en place de la réponse immunitaire. Pour mieux comprendre ce rôle, il faut détailler les spécificités des différentes formes de CMH et de lymphocytes.

Le CMH est une région génomique présente (sous diverses formes) chez la plupart des vertébrés. Cette région est très fortement diversifiée d'un individu à l'autre : elle comporte un grand nombre de gènes dont certains ont jusqu'à quelques centaines d'allèles. Les gènes de cette région codent pour deux variantes du complexe membranaire : le CMH de classe I et le CMH de classe II, tous deux associés à de petits peptides. Le CMH de classe I est présent dans la membrane de la plupart des cellules de l'organisme, associé à des peptides produits par la cellule. Sa grande variabilité d'un individu à l'autre est essentielle pour en faire un bon marqueur du soi et permet au système immunitaire de détecter et détruire des intrus aux caractéristiques très proches des cellules de notre corps. Cette reconnaissance du soi est notamment à l'origine des rejets de greffe. Le CMH de classe II n'est présent que chez certaines cellules spécialisées et sert de support pour la présentation d'antigène évoquée précédemment : il est associé à des peptides issus de pathogènes.

Les lymphocytes regroupent différents leucocytes, intimement liés au CMH (voir figure 1.1 pour une vue d'ensemble des principaux types de leucocytes). On distingue trois grands types de lymphocytes : d'une part les cellules NK (Natural Killer), qui interviennent dans la réponse innée, et d'autre part les lymphocytes B et T qui jouent un rôle prépondérant dans les réponses adaptatives.

Les lymphocytes NK sont des cellules tueuses : elles détruisent les cellules qu'elles rencontrent, à moins que celles-ci ne puissent prouver leur appartenance au soi, en particulier par l'expression de CMH de classe I. Plus précisément, la décision de lyser une cellule ou non dépend de la balance entre les signaux activateurs et inhibiteurs reçus par la cellule NK. Par exemple, l'interféron γ (IFNg) active les cellules NK, la plupart des cellules renforcent leur protection en augmentant le nombre de CMH sur leur membrane. Certaines cellules infectées ou déficientes échouent à ce test et sont donc détruites.

Les lymphocytes B et T disposent de récepteurs qui ne réagissent qu'en présence d'un antigène particulier. Bien que différents, les récepteurs spécifiques des lymphocytes B et T partagent certaines propriétés : ils sont composés d'une partie constante et d'une partie variable, responsable de la reconnaissance de l'antigène. Cette spécificité est acquise durant leur maturation (dans la moelle osseuse pour les lymphocytes B, dans le thymus pour les lymphocytes T). Durant cette maturation, un mécanisme complexe de recombinaison génétique permet de former aléatoirement une grande collection de récepteurs à partir

d'un ensemble de segments géniques. Ces recombinaisons sont suivies d'étapes de sélection permettant d'éliminer les lymphocytes spécifiques de peptides du soi, afin d'éviter les phénomènes d'auto-immunité.

Les récepteurs des lymphocytes B permettent de reconnaître un antigène isolé. Celui-ci est alors internalisé, lysé et présenté sur la membrane du lymphocyte, associé à un CMH de classe II : les lymphocytes B sont, comme les phagocytes, des cellules présentatrices d'antigène (APC). Lorsqu'ils sont confrontés à leur antigène, ils subissent une ultime étape d'activation pour devenir soit des plasmocytes, soit des cellules B à mémoire. Les plasmocytes sont capables de secréter des anticorps solubles. Similaires à leurs récepteurs membranaires spécifiques, ceux-ci se fixent directement au pathogène. La partie constante des anticorps sert alors à recruter d'autres acteurs du système immunitaire (macrophages, cellules NK, ...), afin de faciliter la destruction et l'élimination du pathogène. Les plasmocytes meurent (par apoptose) lorsque la cause de leur activation disparaît. Les cellules B à mémoire ont une plus grande espérance de vie et permettent une réaction plus rapide lors d'une nouvelle confrontation au même antigène.

Les cellules T possèdent également un récepteur spécifique, appelé TCR (T Cell Receptor), capable de reconnaître un antigène associé à un CMH. Le TCR fonctionne avec un co-récepteur CD8 ou CD4 (CD pour "Cluster de différenciation"), permettant de s'associer respectivement au CMH de classe I ou II. Durant leur maturation, les lymphocytes se spécialisent et expriment l'un ou l'autre de ces récepteurs, définissant deux grands groupes de lymphocytes T : CD4+ et CD8+. Les cellules T cytotoxiques (Tc) expriment le co-récepteur CD8, qui fonctionne avec les CMH de classe I. Ces cellules tueuses reconnaissent et détruisent les cellules infectées ou cancéreuses car celles-ci présentent des CMH de classe I associés à des antigènes du non-soi. Les cellules T auxiliaires (Th pour T Helper cells) par contre expriment le co-récepteur CD4, qui fonctionne avec les CMH de classe II. Ces cellules CD4+ se lient aux APC et jouent un rôle capital dans la régulation de l'action des cellules effectrices du système immunitaire. C'est à ce sous-ensemble de lymphocytes T que je m'intéresserai tout particulièrement ici. Comme pour les lymphocytes B, certains lymphocytes T (CD4+ comme CD8+) se spécialisent en cellules à mémoire.

Si l'on étudie de plus près les lymphocytes T auxiliaires (Th, CD4+), on peut à nouveau distinguer plusieurs sous-populations spécialisées. Cette nouvelle spécialisation est déclenchée lors de la première rencontre entre une cellule Th naïve (Th0) et son antigène. La cellule peut alors devenir Th1 ou Th2. Les Th1 sécrètent de l'IFNg et activent donc la réponse immunitaire cellulaire (cellules NK). Les Th2 produisent de l'interleukine 4 (IL4), qui active la réponse humorale (lymphocytes B, producteurs d'anticorps solubles). Cette dichotomie Th1/Th2 a récemment été revue, suite à la découverte de nouvelles sous-populations : les Th17, mis en évidence par Harrington et al. [47] et les cellules T régulatrices (Treg), découvertes par Sakaguchi et al. [96]. Les Th17 (initialement considérés comme des Th1) activent la réponse inflammatoire, par la sécrétion d'IL17. Les cellules régulatrices (Treg) jouent un rôle capital dans l'arrêt de la réponse immunitaire, par des mécanismes partiellement décryptés. La figure 1.2 présente les différents types de cellules Th.

La mise en place, le déroulement et l'arrêt de la réponse immunitaire reposent en grande partie sur l'évolution des populations de cellules T auxiliaires, et donc sur le contrôle de leur spécialisation, prolifération et mort cellulaire. Ces différentes sous-populations interagissent entre elles et avec les autres cellules du système immunitaire par la sécrétion de cytokines (interleukines et interféron).

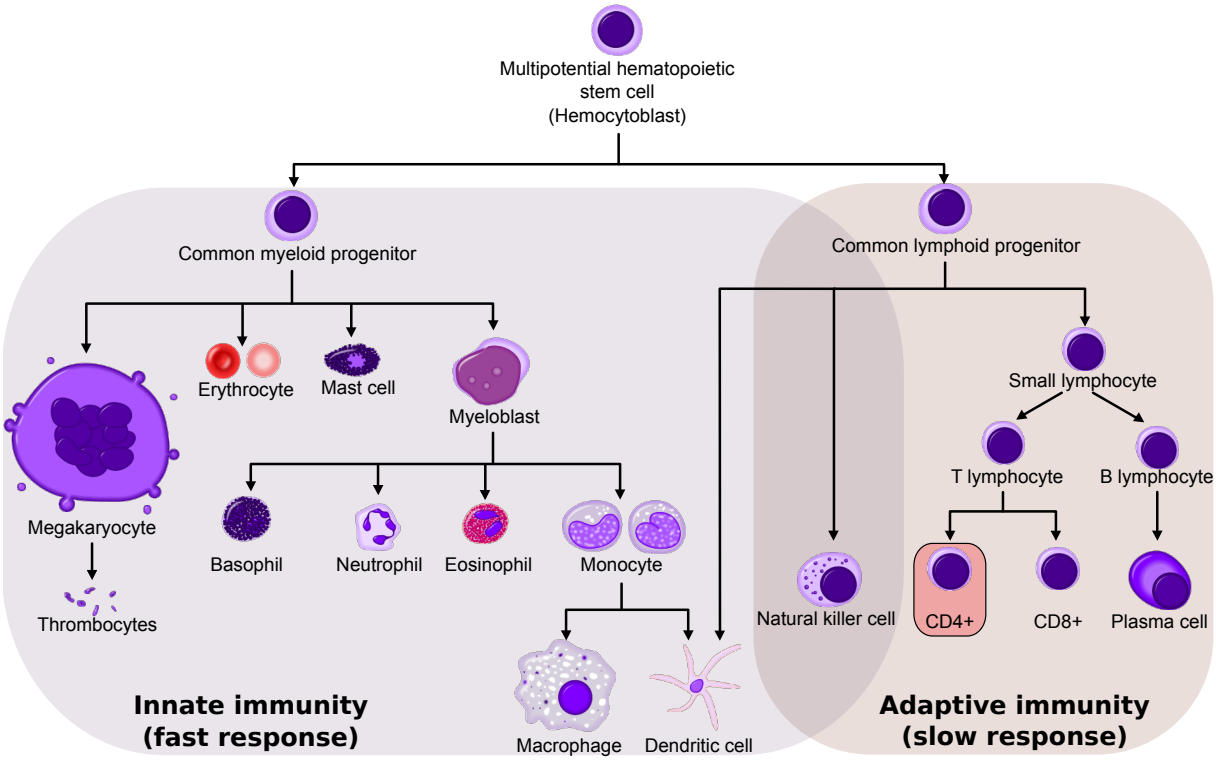


Figure 1.1: Les différentes lignées de leucocytes dérivent des cellules souches hématopoïétiques par différenciations successives (l'hématopoïèse). Elles interviennent soit dans la réponse innée soit dans la réponse adaptative. Dans la suite je m'intéresserai principalement aux sous-types de lymphocytes T auxiliaires (CD4+) [Basé sur Wikimedia Commons].

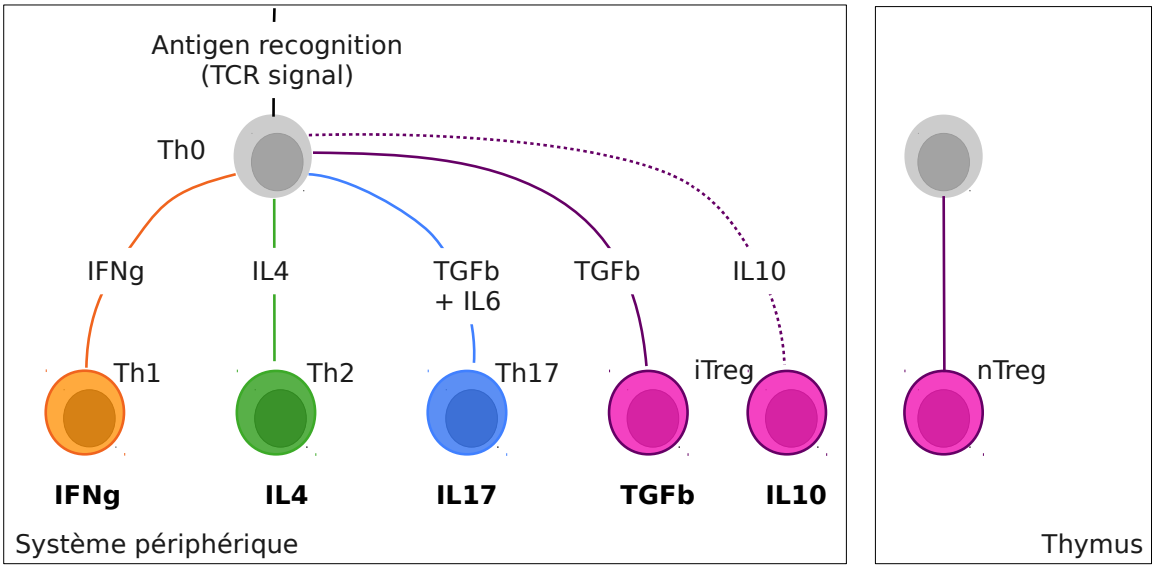


Figure 1.2: Les lymphocytes T auxiliaires se spécialisent en sous-types Th1, Th2, Th17 ou Treg lorsqu'ils rencontrent leur antigène. Cette spécialisation dépend fortement des cytokines présentes. Cependant, les Treg "naturels" semblent se différencier plus précocement dans le thymus [Basé sur [120]].

1.3 Pathologies liées à des dysfonctionnements des Th

De nombreux dérèglements du système immunitaire sont liés à des dysfonctionnements des lymphocytes T auxiliaires. On peut les classer en trois grandes catégories : hypersensibilité, maladies auto-immunes et immunodépression.

Les hypersensibilités sont des réactions excessives du système immunitaire. Il en existe 4 grands types, dont le plus connu est sans doute les allergies. Dans le cas d'allergies, la réaction est provoquée par un intrus spécifique : l'allergène. Celui-ci déclenche une forte excitation des lymphocytes Th2, qui produisent alors de grandes quantités d'IL4. Ceci entraîne une production excessive d'anticorps par les lymphocytes B. Cet excès d'anticorps stimule le système immunitaire, provoquant en particulier une forte vasodilatation.

Dans le cas de maladies auto-immunes, le système immunitaire ne parvient pas à reconnaître correctement le soi et s'attaque donc à des cellules saines de l'organisme. En réalité, il est normal que certaines cellules du système immunitaire reconnaissent des pathogènes du soi, mais des mécanismes de tolérance les empêchent normalement de s'y attaquer de manière trop agressive. Une certaine dose d'auto-immunité est même souhaitable dans la lutte contre les cancers. Les cellules T régulatrices (Treg) sont connues pour limiter les réponses auto-immunes agressives. L'auto-immunité naturelle implique donc un équilibre entre cellules effectrices (Th1, Th2 et Th17) et cellules régulatrices (Treg). Des dérèglements de cet équilibre peuvent être à l'origine de maladies auto-immunes, qui peuvent être localisées (diabète de type I) ou généralisées (lupus, polyarthrite rhumatoïde).

D'autres dérèglements du système empêchent au contraire le système immunitaire de répondre à des invasions, on parle alors d'immunodépression (ou immunodéficience). Celle-ci peut être innée dans le cas de certaines maladies rares, généralement d'origine génétique (cas des enfants-bulle). Il existe aussi des immunodéficiences acquises, en particulier chez les malades du SIDA. En effet, le HIV infecte les lymphocytes T auxiliaires, causant un fort ralentissement de l'ensemble du système immunitaire. On peut noter aussi que certains traitements médicaux ont des effets immunodépresseur, que ce soit leur but principal (dans le traitement de maladies auto-immunes ou contre le rejet de greffe) ou un effet secondaire (radiothérapies et chimiothérapies utilisées contre le cancer et bloquant la prolifération cellulaire). Dans une moindre mesure, les anti-inflammatoires nous soulagent en atténuant une partie de la réponse.

1.4 Pourquoi modéliser dans ce contexte ?

La réponse immunitaire fait intervenir un grand nombre de types cellulaires plus ou moins spécialisés. Ceux-ci proviennent de différenciations successives à partir des cellules souches hématopoïétiques. Une bonne compréhension de ces différenciations, ainsi que de l'activation, la prolifération et la mort des cellules du système immunitaire est essentielle pour l'étude de la réponse immunitaire dans son ensemble. Ces mécanismes sont fortement contrôlés et mènent à un délicat équilibre entre les différentes populations, chacune sensible à des stimuli différents et produisant des cytokines différentes. Une grande partie des mécanismes sous-jacents a été étudiée expérimentalement, mais ces données ne peuvent être reliées au comportement dynamique du système sans être intégrées dans une description formelle des nombreux acteurs du système et de leurs relations.

C’est cette vue plus abstraite que les techniques de modélisation permettent de construire. Dans un premier temps, il s’agit de synthétiser les données disponibles, en assemblant dans un même modèle des informations connues sur les liens entre les composants du système étudié. L’assemblage de ces informations peut mettre en évidence des lacunes dans nos connaissances. Ensuite, en comparant les prédictions du modèle avec des observations expérimentales, on peut s’assurer de la cohérence du modèle. Enfin, un modèle reproduisant des observations connues peut servir de base pour la construction de nouvelles expériences, en permettant de les concevoir *in silico*, afin de sélectionner les plus informatives.

Chapitre 2

Modélisation de systèmes biologiques

2.1 Méthodes de modélisation

La modélisation de systèmes biologiques fait intervenir de nombreuses méthodes, qui sont choisies en fonction du type de mécanismes que l'on souhaite décrire, du type de données disponibles et des contraintes techniques (pour une revue, voir [28, 36, 102]). Nous nous intéressons ici à la régulation de la différenciation cellulaire, je me concentre ici sur les méthodes utilisées pour modéliser des réseaux de régulation. Dans un premier temps, ces réseaux sont souvent décrits sous forme de *cartes d'interactions*. Ces cartes fournissent une vue statique des relations entre les composants et ne constituent donc pas des modèles dynamiques.

Une carte d'interactions est un graphe dans lequel les noeuds représentent les composants du système et les arcs représentent des relations entre ces composants. C'est essentiellement une notation graphique servant à donner un aperçu visuel du système dans son ensemble. Plusieurs notations existent, notamment les cartes d'interactions moléculaires proposées par Kohn [61] et les diagrammes de processus proposés par Kitano [58]. Ces deux types de cartes se concentrent sur une description précise au niveau réactionnel, les rôles des participants aux réactions étant symbolisés par différents types d'arcs.

Pour intégrer des informations sur le comportement dynamique du système, on utilise des techniques issues de l'étude des systèmes dynamiques et de l'automatique (théorie du contrôle). Les composants du système sont alors représentés par des *variables dynamiques*, dont la valeur peut évoluer au cours du temps. Un *état* du système est un vecteur donnant la valeur de chacune de ces variables. Ces états peuvent être placés dans un espace abstrait dont les coordonnées sont les valeurs des variables : l'*espace des phases*. Cet espace a donc autant de dimensions que le système a de variables. Les méthodes quantitatives visent une représentation et une évaluation précises de l'état du système, alors que les méthodes qualitatives se concentrent sur des propriétés plus abstraites, comme la présence d'une protéine ou le changement de forme d'une cellule. On distingue trois grandes classes de systèmes dynamiques : les systèmes à temps continu, les systèmes à temps discret et les systèmes à événements discrets.

Les systèmes à temps continu permettent d'étudier l'évolution du système au cours du temps, souvent à l'aide de simulations. En général, ces modèles sont formulés sous forme d'Équations Différentielles Ordinaires (EDO, ou ODE en anglais) et utilisent des variables continues (concentrations des composants du modèle). Dans ce type de formalisme, l'évolution dynamique de la concentration de chaque composant est donnée par une équation

différentielle, permettant de calculer sa dérivée (c'est-à-dire sa vitesse de synthèse ou de dégradation), en fonction des concentrations courantes des composants du système. Par exemple, on peut décrire l'évolution de la concentration d'une protéine X (notée [X]), dont la synthèse est activée par un facteur de transcription A, par une formule du type : $\frac{d[X]}{dt} = k_1[A] - k_2[X]$, ou k_1 est le taux de synthèse et k_2 le taux de dégradation. Les fonctions utilisées en pratique sont rarement de simples fonctions linéaires, et leur complexité augmente avec le nombre de régulateurs. Le principal problème ici est le choix du type de fonction et des valeurs des paramètres k_1 et k_2 utilisés. Ce choix peut se faire à partir de données biologiques précises (mesures des vitesses de réaction, de synthèse, de transport...) qui ne sont que rarement disponibles. En pratique, les paramètres manquants sont généralement obtenus par exploration. On sélectionne alors, pour chaque paramètre, la valeur pour laquelle la dynamique obtenue est la plus proche de celle observée expérimentalement. Cette exploration des paramètres est souvent très coûteuse. Par ailleurs, comme elle vise à minimiser la distance avec un comportement canonique, il existe un risque de sur-apprentissage. L'exploration des paramètres est un champ de recherche à part entière, qui déborde du cadre de ce document (pour plus de détails, voir par exemple [74]).

Les systèmes à temps discret, reposent sur un découpage du temps en "tranches" égales. $X(t+1)$, la concentration de X à l'instant $t+1$, peut alors être calculée en fonction de sa concentration et de sa dérivée discrète à l'instant t : $X(t+1) = X(t) + k_1A(t) - k_2X(t)$. Ce type de modèle fournit une approximation des modèles à temps continu. Dans certains cas, on peut faire correspondre l'échelle de temps discrète utilisée avec celle utilisée pour des mesures à intervalles de temps réguliers.

Dans les systèmes à événements discrets, l'évolution du système est définie par une succession d'événements ponctuels. Ce type de modèle est principalement utilisé pour la simulation de systèmes artificiels. Dans le cadre de l'étude de systèmes biologiques, les événements correspondent par exemple au début ou à la fin de la synthèse d'une protéine, ou encore au dépassement d'un seuil de concentration. On ne s'intéresse pas directement aux valeurs de concentrations mais à la succession de ces événements. Certains systèmes à événements discrets reposent sur une discrétisation des variables dynamiques. Leur espace des phases est alors fini si chacune de ces variables est bornée. De tels modèles abstraits permettent l'étude de systèmes de taille relativement grande (trop grands pour être abordés par une approche continue). Les réseaux de Petri standards sont une classe courante de tels systèmes (voir [77] et section 2.3).

Il existe également des systèmes dynamiques hybrides, dans lesquels une partie du système étudié est décrite comme système à événements discrets (typiquement le contrôle) et le reste comme système continu (le système contrôlé). Ce type d'approche est notamment utilisée dans les réseaux de Petri hybrides (section 2.3.2).

Les approches que nous venons d'introduire sont déterministes, c'est-à-dire qu'en partant d'un état donné, on arrive toujours au même résultat. Ce type d'approche n'est pas adapté à la prise en compte des incertitudes (en particulier sur les mesures et les valeurs des paramètres) et des fluctuations souvent rencontrées dans l'étude des systèmes biologiques. En effet, certains processus sont soumis à des bruits importants, qu'ils soient externes (environnemental, mesures) ou internes. La prise en compte du bruit est particulièrement importante pour la modélisation de processus faisant intervenir de faibles nombres de molécules : il n'est alors plus pertinent de considérer des taux de réaction moyens. Il est alors possible d'utiliser des variantes stochastiques des formalismes précédemment évoqués.

Ainsi, les *Équations Différentielles à coefficients aléatoires* permettent de prendre en compte

les incertitudes sur les paramètres. Les *Équations Différentielles Stochastiques* quant à elles intègrent un terme supplémentaire pour le bruit. Pour les bruits internes, ce terme dépend des variables dynamiques. Il est parfois plus pratique d'utiliser l'*équation maîtresse* (master equation). Cette équation donne la probabilité $P(n, t + dt)$ d'être dans un état n à l'instant $t + dt$ en fonction des $P(n, t)$ et des probabilités de transition d'un état à un autre : $P(n, t + dt) = P(n, t)a(n, n)dt + \sum_{n'} P(n', t)a(n', n)dt$, où $a(n', n)$ est la probabilité de passer de l'état n' à l'état n. La matrice des $a(n', n)$ peut être vue comme la matrice de transition d'une chaîne de Markov. Il existe également des variantes stochastiques de réseaux de Petri.

Ces systèmes stochastiques sont souvent difficiles à étudier analytiquement, et on recourt souvent à des simulations numériques. En particulier, les méthodes de Monte-Carlo consistent à déterminer les probabilités des différents comportements (dont on peut ensuite déduire un comportement moyen), en réalisant un grand nombre de simulations. Un exemple classique, adapté à la simulation de systèmes (bio)chimiques faisant intervenir de petits nombres de molécules, est l'algorithme de Gillespie. Dans ce cadre, chaque étape de la simulation consiste à tirer aléatoirement (avec des probabilités dépendant de paramètres fixés et de l'état courant du système) la prochaine réaction et le temps au bout duquel elle survient. C'est une méthode quantitative, discrète, utilisant comme variables les nombres de molécules de chaque espèce chimique.

La suite de ce document porte sur deux types de systèmes à événements discrets : le formalisme logique, autour duquel est centré ce travail (voir section 2.2) et les réseaux de Petri (section 2.3). Ces formalismes permettent une interprétation non-déterministe sans pour autant utiliser de méthode stochastique.

2.2 Le formalisme logique

2.2.1 Historique et motivation

L'utilisation d'un formalisme discret (Booléen) pour décrire et étudier les systèmes de régulation génétique a initialement été proposée bien avant que la taille des réseaux étudiés n'explose. En particulier, Stuart Kauffman a proposé en 1969 un formalisme de modélisation des réseaux génétiques basé sur l'algèbre Booléenne. Il s'est particulièrement intéressé à l'étude des propriétés de réseaux Booléens aléatoires (Random Boolean Network or RBN) [54]. Dans ces réseaux, le degré entrant des composants est fixé et les mises-à-jour sont synchrones. René Thomas a introduit en 1973 un formalisme proche. Ce formalisme ne contraint pas le type de mise-à-jour (asynchrone, synchrone, ...) ni le degré entrant des composants du modèle [116].

Lorsque l'on considère la régulation d'un gène par un facteur de transcription, on observe généralement que l'effet est négligeable jusqu'à un certain seuil de concentration puis devient rapidement maximal. Ce genre d'effet est bien représenté par une sigmoïde de forte pente et peut être approximé par une fonction de Heaviside (step function). Cette approximation discrète d'un processus continu, illustrée dans la figure 2.1, est à la base du formalisme logique : on considère que la concentration du facteur de transcription est soit en dessous, soit au dessus de son seuil d'activité et on peut donc la décrire par une variable Booléenne, la valeur 0 (faux) dénotant une concentration trop faible et la valeur 1 (vrai) indiquant qu'elle est suffisante.

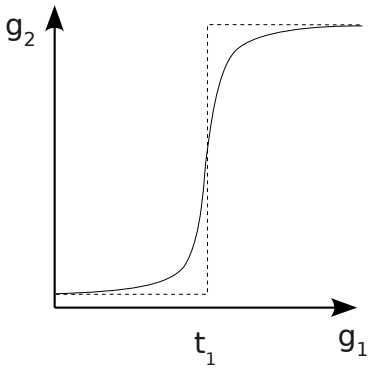


Figure 2.1: Approximation Booléenne d'une fonction sigmoïde. Cette courbe donne le niveau d'activité de g_2 en fonction de celui de son activateur, g_1. L'augmentation du niveau de g_1 n'a pas d'impact significatif sur celui de g_2 jusqu'à ce que g_1 atteigne le seuil t_1. Le niveau de g_2 augmente alors rapidement avant d'atteindre un plateau. Ce type d'effet peut être approximé par une fonction à seuil (en pointillés), et donc par une description Booléenne.

Les facteurs de transcription peuvent avoir plusieurs cibles, sur lesquelles ils agissent à des seuils différents (ou avoir des effets différents sur une même cible en fonction de leur concentration). On peut alors utiliser une variable discrète multi-valuée, dont les valeurs successives (0,1,2,...) indiquent la zone dans laquelle se trouve la concentration [118, 117, et références incluses]. En pratique, on limite autant que possible le nombre de seuils.

2.2.2 Formalisme logique étendu

Ce formalisme qualitatif se base sur la définition de deux graphes : le *graphe de régulation* décrit les interactions entre les différents composants du système, tandis que le *graphe de transitions d'états* représente l'évolution dynamique du système [18].

2.2.2.1 Graphe de régulation

Les interactions entre les composants du système sont représentées par un graphe appelé *graphe de régulation,* qui constitue un modèle abstrait décrivant notre compréhension du système biologique. Afin de simplifier la lecture, je présente ici le formalisme logique tel que nous l'utilisons aujourd'hui.

Comme dans le cas d'une carte d'interactions, les noeuds du graphe de régulation représentent les acteurs du système (composants de régulation) et les arcs représentent les interactions entre ces composants. Dans les premiers modèles, les composants représentaient des gènes et les interactions des régulations transcriptionnelles. C'est toujours majoritairement le cas, mais de plus en plus de modèles intègrent d'autres types de composants (protéines, complexes, variables abstraites) et d'interactions (modifications post-transcriptionnelles en particulier). Afin de pouvoir modéliser la dynamique du système, on associe à chaque composant un niveau d'activité et une fonction logique définissant son évolution en fonction de l'activité de ses régulateurs. Dans le cas multi-valué, ce graphe est formalisé comme suit.

Definition Un graphe de régulation est un multi-graphe orienté $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \Gamma, \Theta, \mathcal{K})$ où,

- $\mathcal{G} = \{g_1, \ldots, g_N\}$ est l'ensemble des noeuds, représentant les *composants de régulation*.

- $\mathcal{M}ax : \mathcal{G} \to \mathbb{N}^*$ associe un *niveau maximal* $\mathcal{M}ax(g_i) = \mathcal{M}ax_i$ au noeud g_i. Ainsi le niveau courant de g_i, noté x_i, prend ses valeurs dans $\mathcal{D}_i = \{0, \ldots, \mathcal{M}ax_i\}$.
- Γ est l'ensemble des arcs, définis par des paires ordonnées d'éléments de $\mathcal{G}$ représentant les régulations. Lorsque $\mathcal{M}ax_i > 1$, g_i peut avoir différents effets sur g_j, en fonction de son niveau courant x_i. Dans ce cas, l'arc reliant g_i à g_j est un multi-arc dénotant plusieurs interactions. La multiplicité de l'arc (g_i, g_j) (*i.e.* le nombre d'interactions qu'il contient), est notée $m_{i,j}$ ($1 \leq m_{i,j} \leq \mathcal{M}ax_i$). Les boucles (même multiples) sont autorisées : un arc (g_i, g_i) indique une autorégulation de g_i.
  Pour chaque $g_j \in \mathcal{G}$, $Reg(j)$ est l'ensemble de ses régulateurs : $g_i \in Reg(j)$ si et seulement si $(g_i, g_j) \in \Gamma$.
- Θ est une application associant un seuil à chaque élément de Γ. Plus précisément, $\theta_{i,j,k}$ est associé à la $k^{ième}$ interaction entre g_i et g_j (noté $(g_i, g_j, \theta_{i,j,k}), k \in \{1, \ldots, m_{i,j}\}$), avec $1 \leq \theta_{i,j,1} < \cdots < \theta_{i,j,m_{i,j}} \leq \mathcal{M}ax_i$. L'interaction $(g_i, g_j, \theta_{i,j,k})$ est *active*, lorsque x_i, le niveau de sa source g_i, est supérieur (ou égal) à son seuil $\theta_{i,j,k}$ et inférieur à $\theta_{i,j,k+1}$, le seuil de l'interaction suivante (par convention, $\theta_{i,j,m_{i,j}+1} = \mathcal{M}ax_i + 1$).
- $\mathcal{K} = (\mathcal{K}_1, \ldots, \mathcal{K}_N)$ définit les *règles logiques* associées aux noeuds et spécifiant leur comportement : chaque $\mathcal{K}_i$ est une fonction multi-valuée donnant la valeur cible de g_i en fonction de l'état du système :

$$\mathcal{K}_i : \prod_{g_j \in \mathcal{G}} \mathcal{D}_j \mapsto \{0, \ldots, \mathcal{M}ax_i\}.$$

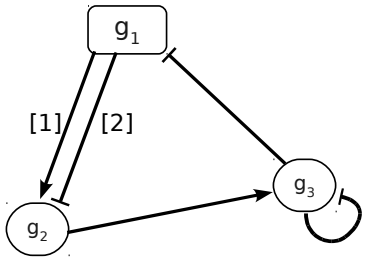
En réalité, seuls les régulateurs de g_i jouent un rôle dans $\mathcal{K}_i$, qui peut donc être définie sur l'ensemble $\prod_{g_j \in Reg(i)} \mathcal{D}_j$.

Pour alléger les notations, nous utiliserons i au lieu de g_i lorsqu'aucune confusion n'est possible.

Dans le cas Booléen, cette définition est simplifiée car les niveaux maximaux, seuils et multiplicités valent toujours 1 et les fonctions logiques sont des fonctions Booléennes.

Cohérence Il est important de noter que les interactions et leurs seuils peuvent être déduits à partir des fonctions logiques. En effet, si $\mathcal{K}_i(x) \neq \mathcal{K}_i(y)$ avec $x_j = y_j + 1$ et $x_k = y_k \, \forall k \neq j$, on peut déduire que j a un effet sur i au seuil x_j. Les interactions sont donc définies de façon redondante dans le graphe de régulation lui-même et dans les fonctions logiques. Il est alors important de s'assurer de la cohérence entre les deux définitions. On considère que le graphe de régulation est cohérent avec les fonctions logiques lorsque ces dernières permettent de retrouver le même ensemble d'interactions que celles définies dans le graphe de régulation. Pour simplifier la présentation, nous supposons ici que la structure du graphe donnée explicitement est toujours cohérente avec les fonctions logiques. Notre logiciel de modélisation (GINsim) peut prévenir ou détecter ces situations (voir section 8.5).

Signe des interactions On associe un signe (positif, négatif ou dual) à chaque interaction. Ce signe est lui aussi implicitement défini par les fonctions logiques. Reprenons le cas précédent : $\mathcal{K}_i(x) \neq \mathcal{K}_i(y)$ avec $x_j = y_j + 1$ et $x_k = y_k \, \forall k \neq j$. On considère que l'effet de j sur i est positif (resp. négatif) si $\mathcal{K}_i(x) > \mathcal{K}_i(y)$ (resp. $\mathcal{K}_i(x) < \mathcal{K}_i(y)$). Une interaction est considérée comme positive (resp. négative) si elle a au moins un effet positif et aucun effet négatif (resp. au moins un effet négatif et aucun effet positif). Une interaction peut également être duale lorsqu'elle a les deux types d'effets, en fonction de l'état des co-régulateurs. Voir section 8.5 pour l'aide apportée par notre logiciel de modélisation.



$\mathcal{G} = \{g_1, g_2, g_3\}$
$\mathcal{M}ax_1 = 2$; $\mathcal{M}ax_2 = \mathcal{M}ax_3 = 1$
$\mathcal{D}_1 = \{0, 1, 2\}$; $\mathcal{D}_2 = \mathcal{D}_3 = \{0, 1\}$
$Reg(3) = \{g_2, g_3\}$
$\theta_{1,2,1} = 1$, $\theta_{1,2,2} = 2$

Figure 2.2: Graphe de Régulation. A gauche : représentation graphique d'un graphe de régulation. Les arcs à bouts plats représentent des inhibitions et les autres arcs des activations (ceci est uniquement une convention graphique, le signe d'un arc est déduit de la fonction logique associée à sa cible). Le noeud rectangulaire g_1 est ternaire, alors que les autres sont Booléens. Toutes les interactions ont un seuil de 1, sauf (g_1, g_2), qui est duale avec un seuil 1 et un seuil 2. A droite : illustration des notations de la définition donnée dans la section 2.2.2.1.

2.2.2.2 Graphe de transitions d'états et simulation

Le comportement dynamique du système est représenté par un *graphe de transitions d'états*. Dans ce graphe :

- chaque noeud correspond à un *état* du système, c'est-à-dire un vecteur donnant le niveau de chaque composant (un entier compris entre 0 et le niveau maximal du composant) ;
- les arcs représentent des *transitions d'états*. Ces transitions correspondent à des changements de niveau des composants du système, dictés par les fonctions logiques du graphe de régulation.

Le graphe de transitions d'états est calculé à partir du graphe de régulation, en partant d'un état initial donné (ou d'un ensemble d'états initiaux). Chaque étape de ce calcul consiste à déterminer les états successeurs de l'état courant, d'après les fonctions logiques et le type de mise-à-jour choisi. On commence par déterminer les composants amenés à changer de niveau. Dans l'état x, le composant i tend vers $\mathcal{K}_i(x)$ par paliers de 1 : le niveau de i peut passer de x_i à $y_i = x_i + \Delta_i(x)$, avec $\Delta_i(x) = \begin{cases} \frac{\mathcal{K}_i(x) - x_i}{|\mathcal{K}_i(x) - x_i|} & si\, \mathcal{K}_i(x) \neq x_i, \\ 0 & sinon. \end{cases}$

Si $\mathcal{K}_i(x) = x_i$ pour chacun des composants, alors l'état x est un état stable. Dans les autres cas, au moins un composant est appelé à changer de niveau. Dans les cas où plusieurs changements sont en concurrence, il faut choisir une stratégie de mise-à-jour pour la détermination d'un ou plusieurs états successeurs. Plusieurs stratégies principales sont utilisées :

- mise-à-jour synchrone : on suppose que tous les changements possibles se produisent simultanément, un état donné aura donc un successeur unique prenant en compte tous ces changements. Dans ce cas, une transition va d'un état x à un état $y \neq x$ si et seulement si, pour chaque composant i, $y_i = x_i + \Delta_i(x)$.
- mise-à-jour asynchrone : les changements ne sont pas simultanés mais concurrents. Une transition d'un état à l'état suivant correspond donc au changement de niveau d'un seul composant du système. Un état successeur est créé pour chaque changement possible. Dans ce cas, une transition relie un état x à un état y s'il existe un composant i tel que $y_i = x_i + \Delta_i(x)$, avec $\Delta_i(x) \neq 0$ et $\forall j \neq i$, $y_j = x_j$.
- mise-à-jour séquentielle : pour déterminer le successeur d'un état, tous les composants sont mis à jour dans un ordre déterminé. La valeur cible du k^{e} composant prend alors en compte les éventuels changements appliqués aux $k-1$ composants précédents. Dans le cas où l'ordre de mise-à-jour correspond à l'ordre des composants, le successeur de l'état x est x^N, avec $x^0 = x$, $x_k^{k+1} = x_k^k + \Delta_k(x^k)$ et $x_i^{k+1} = x_i^k\ \forall i \neq k$.

Les mises-à-jour synchrone et séquentielle sont déterministes alors que l'asynchrone est généralement non-déterministe, car un état peut avoir plusieurs successeurs (autant que de composants dans le graphe de régulation). Le graphe de transitions d'états asynchrone contient généralement des comportements alternatifs, il est donc plus grand et plus complexe à étudier que les graphes déterministes.

À ces méthodes de mise-à-jour, on peut ajouter deux approches plus générales permettant de former des groupes de composants : la méthode bloc-séquentielle, dans laquelle chaque groupe est mis à jour de façon synchrone, les blocs étant considérés séquentiellement ; et l'utilisation de classe de priorités permettant de mélanger synchrone et asynchrone, présentée dans la section 7.1.

Il est également possible d'affiner la dynamique obtenue, par exemple par l'ajout d'information sur les vitesses relatives des transitions. Siebert et Bockmayr [107, 108] utilisent des automates temporisés pour analyser plus finement le comportement temporel de tels systèmes. Les états du système prennent alors en compte une horloge pour chaque composant dont le niveau cible est différent du niveau courant.

Parmi les propriétés du graphe de transitions d'états, on s'intéresse particulièrement à la présence d'attracteurs, c'est-à-dire d'états stables (états n'ayant aucun successeur), ou de composantes fortement connexes terminales (cycles ou ensembles de cycles entremêlés desquels le système ne peut s'échapper).

Les états stables sont les mêmes, quelle que soit la politique de mise-à-jour. En effet, ce sont des états dans lesquels aucun composant n'est amené à changer de niveau. Dans le cas synchrone, les attracteurs cycliques sont toujours des cycles simples. Ces attracteurs ne sont conservés à l'identique pour une mise-à-jour asynchrone que si chacune des transitions du cycle ne concerne qu'un seul composant. Dans les autres cas, il est possible (mais non garanti) que le graphe de transitions d'états asynchrone contienne un attracteur similaire. La mise-à-jour synchrone a l'avantage de la simplicité mais peut générer des attracteurs cycliques de manière artificielle. En effet, elle se base sur la supposition que lorsque plusieurs transitions sont en concurrence, elles se produisent au même instant. Dans certaines conditions, elle peut également supprimer des attracteurs complexes présents dans le cas asynchrone. La mise-à-jour séquentielle est également déterministe, mais elle nécessite d'ordonner les composants. La mise-à-jour asynchrone, bien que plus complexe à analyser, est généralement préférable car plus réaliste biologiquement. Pour plus d'informations sur l'impact du choix de la méthode de mise-à-jour sur les propriétés dynamiques voir [4] (article axé autour des méthodes séquentielle et bloc-séquentielle). Le successeur d'un état d'après la méthode séquentielle est atteignable à partir de cet état dans le graphe asynchrone (par un chemin de longueur maximale N), alors que le successeur d'un état d'après la méthode synchrone ne l'est pas forcément (voir figure 2.3). Le graphe asynchrone contient tous les comportements possibles pour tous les ordres sur les composants en séquentiel, ainsi que des comportements supplémentaires.

2.2.2.3 Variations des notations

Bien que les grands principes de ce formalisme soient inchangés depuis sa création, la notation que nous utilisons a évolué (par exemple par rapport à [18]).

Dans le cas des multi-arcs, nous utilisons aujourd'hui pour chaque interaction un seuil maximal implicite qui est le seuil de l'interaction suivante, ou encore le niveau maximal du com-

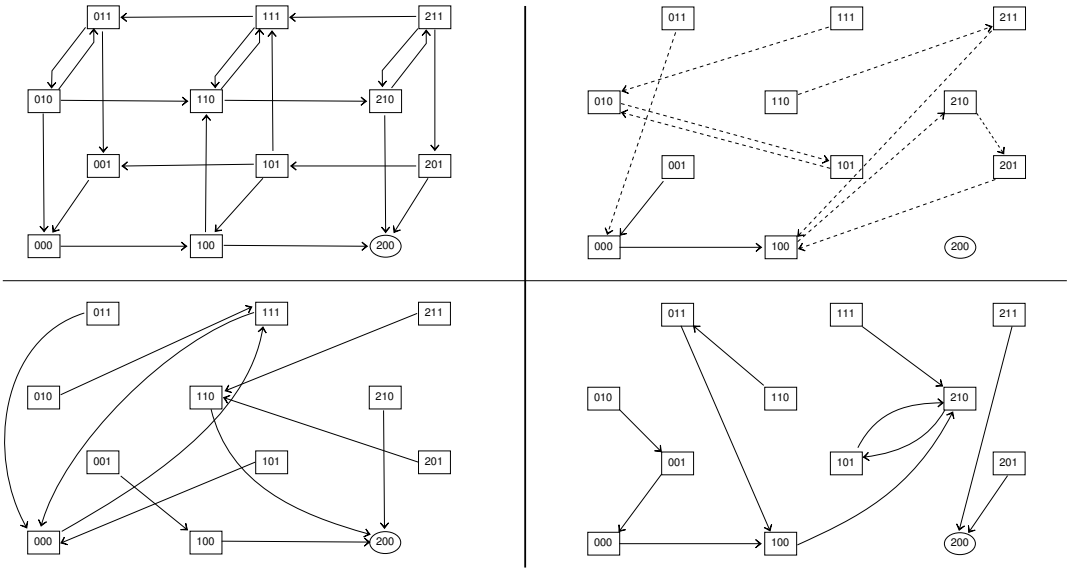


Figure 2.3: Cette figure montre les graphes de transitions d'états (GTE) asynchrone (en haut à gauche), synchrone (en haut à droite), et séquentiels (en bas) obtenus pour le graphe de régulation de la figure 2.2. Les noeuds correspondent aux états du système, sous la forme de vecteurs donnant le niveau de chaque composant. Par exemple l'état 201 correspond à un fort niveau de g_1, absence de g_2 et présence de g_3. Dans le cas asynchrone, chaque transition correspond au changement de niveau d'un seul composant et certains états ont plusieurs successeurs. Dans le graphe synchrone par contre, la plupart des transitions correspondent à des changements de niveau synchronisés de plusieurs composants (transitions en pointillés). Seuls deux états ont un seul successeur dans le graphe asynchrone (000 et 001), leurs transitions sortantes sont alors conservées dans le graphe synchrone. Ce système génère un seul état stable (200), conservé pour chaque type de mise-à-jour. Alors que dans le graphe asynchrone cet état est l'unique attracteur, le graphe synchrone contient deux attracteurs cycliques supplémentaires et l'état stable n'est pas atteignable à partir des autres états.
Les GTE du bas sont obtenus en séquentiel pour les ordres 1,2,3 (à gauche) et 3,2,1 (à droite). Là encore l'état stable est conservé. Chaque état a un seul successeur qui est atteignable dans le GTE asynchrone en suivant un chemin dans lequel les niveaux des composants changent dans l'ordre spécifié.

	Fonctions logiques	paramètres logiques	contributions positives
g_1	$!g_3$	$K_1\{\emptyset\} = 1$ $K_1\{g_3\} = 0$	$K_1\{g_3\} = 1$ $K_1\{\emptyset\} = 0$
g_2	$g_{1,1}$	$K_2\{\emptyset\} = 0$ $K_2\{g_{1,1}\} = 1$ $K_2\{g_{1,2}\} = 0$	$K_2\{g_{1,2}\} = 0$ $K_2\{g_{1,1}, g_{1,2}\} = 1$ $K_2\{\emptyset\} = 0$
g_3	$g_2 \,\&\, !g_3$	$K_3\{\emptyset\} = 0$ $K_3\{g_2\} = 1$ $K_3\{g_3\} = 0$ $K_3\{g_2, g_3\} = 0$	$K_3\{g_3\} = 0$ $K_3\{g_2, g_3\} = 1$ $K_3\{\emptyset\} = 0$ $K_3\{g_2\} = 0$

Table 2.1: Cette table donne une paramétrisation du graphe de régulation de la figure 2.2, sous la forme de fonctions logiques et de liste de paramètres logiques (en utilisant ou non la notion de contribution positive).

posant pour la dernière interaction. Ce seuil était précédemment défini explicitement, permettant la présence d'intervalles de valeurs sans interactions associées. Lorsque le niveau du régulateur se situe dans l'un de ces intervalles, il n'a aucun effet sur sa cible (comme pour le niveau 0). De tels comportements peuvent toujours être définis à l'aide d'une interaction supplémentaire et d'ajustements de la fonction logique. La nouvelle notation a l'avantage d'être plus légère dans la plupart des cas et d'interdire les chevauchements d'intervalles.

Depuis peu, nous utilisons explicitement des formules logiques (définies sous la forme de termes logiques). Leur utilisation est motivée en particulier par le travail de thèse d'Adrien Fauré (voir Annexe A.4). Jusqu'à présent, les règles logiques étaient généralement données sous forme de *paramètres logiques*. Un paramètre logique est une liste d'interactions entrantes actives (c'est-à-dire telles que le niveau d'activité de leur source se situe dans l'intervalle d'activité de l'interaction) associées à une valeur. Les interactions non-listées étant implicitement inactives, un paramètre correspond à une et une seule combinaison des régulateurs agissant sur le composant considéré. L'utilisation de paramètres logiques a l'avantage de donner une représentation unique, équivalente à une table de vérité. L'utilisation de fonctions logiques est plus complexe. En effet, il existe plusieurs fonctions correspondant à la même table de vérité et l'équivalence entre ces fonctions n'est pas toujours intuitive (mais peut être testée algorithmiquement). Malheureusement, l'utilisation de paramètres logiques s'avère rapidement ingérable lorsque le nombre de régulateurs augmente : un noeud avec n régulateurs a 2^n paramètres logiques dans le cas Booléen.

La définition originelle des paramètres logiques de E.H. Snoussi et R. Thomas utilise des listes de "contributions positives" et non d'interactions entrantes actives. Une interaction positive est donc listée dans un paramètre logique si elle est active alors qu'une interaction négative ne l'est que si elle est inactive. Le passage d'une notation à l'autre nécessite un peu d'exercice mais elles sont équivalentes (voir table 2.1).

Il est également possible de définir des fonctions logiques implicitement en fonction de la structure du graphe de régulation. Par exemple, on peut considérer qu'un composant est activé lorsqu'aucun de ses inhibiteurs n'est présent et qu'au moins un activateur l'est (pour plus de détails et des exemples d'application de ce type de règles, voir [65, 72, 26]). Ce genre d'approche a l'avantage de tirer parti des signes associés aux interactions et de garantir la cohérence des fonctions logiques avec la structure du graphe. La définition de certaines fonctions, plus complexes, nécessite alors l'introduction de composants artificiels visant à intégrer une combinaison particulière de régulateurs.

Nous construisons généralement le graphe de régulation à partir de données de la littérature. Une approche complémentaire consiste à observer la dynamique et à inférer le graphe de régulation pouvant l'expliquer. Dans le cas (théorique) où l'on dispose du graphe de transitions d'états complet (synchrone ou asynchrone), les transitions observées dans ce graphe donnent les tables de vérité de fonctions logiques compatibles avec ce comportement dynamique (dans le cas multi-valué, ces fonctions ne sont pas toujours uniques). Une fois ces fonctions logiques connues, il est possible de déduire la structure du graphe comme mentionné dans le paragraphe sur la cohérence du graphe de régulation (section 2.2.2.1). Ce type d'approche a été principalement utilisé afin de déterminer des liens entre un comportement dynamique et la structure du graphe de régulation [88, 91, 90, 105]. Des méthodes basées sur ce principe mais adaptées à des données incomplètes ou bruitées peuvent servir à l'inférence de réseaux de régulation à partir de données d'expression dynamiques [66, 84].

2.2.3 Rôle des circuits

Au cours des années 80, R. Thomas a formulé deux conjectures reliant la structure du graphe de régulation aux propriétés dynamiques observables dans le graphe de transitions d'états [115] :

- la présence d'un circuit positif dans le graphe de régulation est nécessaire pour la multistationnarité (présence de plusieurs attracteurs) ;
- la présence d'un circuit négatif dans le graphe de régulation est nécessaire pour des oscillations entretenues.

Un circuit de régulation est un chemin fermé dans le graphe de régulation passant une seule fois par chacun de ses membres. Le signe d'un tel circuit est le produit des signes des interactions entre ces membres : il est négatif pour un nombre impair d'interactions négatives, positif autrement. Ce signe est celui de l'effet que chaque membre du circuit exerce sur lui-même au travers du circuit. Ces conjectures, formulées pour l'approche logique, ont d'abord été démontrées dans le cadre différentiel [112]. Elles ont depuis été également démontrées dans le cadre logique [91, 109, 88, 89, 106]. Voir également [114] pour une revue. Dans le cas multi-valué, la définition d'un circuit doit être plus stricte afin de prendre en compte le seuil de chaque interaction impliquée. Si certaines de ces interactions sont multiples (c'est-à-dire ont plusieurs seuils), on considère alors autant de circuits que de combinaisons de seuils.

La présence d'au moins un circuit est une condition nécessaire, mais non suffisante, pour la propriété dynamique correspondante. Dans le cas d'un circuit isolé (c'est-à-dire tel que chacun de ses membres a pour seul régulateur son prédécesseur dans le circuit), il est prouvé que la propriété dynamique correspondante est toujours vérifiée [86]. Chaque composant a alors un seul effet sur une seule cible et une description Booléenne est donc suffisante. Commençons par fixer le niveau d'un des membres du circuit, on peut alors déduire le niveau de ses autres membres en remontant le long du circuit. En effet, chaque membre du circuit a pour seul régulateur le membre précédant dont nous venons de fixer le niveau. Le niveau du membre considéré est le même que celui du membre précédent si l'effet de celui-ci est positif, le niveau opposé sinon. Lorsqu'on atteint le membre de départ, comparons son nouveau niveau avec celui que nous avions fixé plus tôt. Dans le cas d'un circuit positif, ces deux niveaux sont identiques et on obtient une configuration stable. Un circuit positif a donc deux états stables, dans lesquels les valeurs de chacun des membres sont inversées. Dans le cas d'un circuit négatif, les deux niveaux sont différents, on peut alors

refaire un tour du circuit en changeant le niveau de chaque membre, et recommencer. Un circuit négatif isolé n'a pas de configuration stable et génère des oscillations.

Par contre, lorsque les membres du circuit sont régulés par d'autres composants du modèle, la seule présence du circuit ne suffit plus toujours à générer le comportement attendu. Par exemple, il suffit que la valeur d'un des membres du circuit soit fixée par les autres composants pour que la multi-stationnarité ou les oscillations disparaissent.

Les réseaux de régulation étudiés en pratique contiennent généralement de nombreux circuits entremêlés. Certains circuits génèrent le comportement attendu seulement dans une zone de l'espace des états [89]. Dans le cas général, il reste difficile de déduire directement des informations sur la dynamique à partir de la structure du graphe de régulation.

2.2.4 Applications

Le formalisme logique a été introduit dans le cadre de l'étude de la régulation du bactériophage lambda, en particulier pour décrire le contrôle de la décision lyse/lysogénie [113]. Ce formalisme a depuis été appliqué à de nombreux autres systèmes biologiques. Bien adapté aux systèmes de différenciation, il a été utilisé, entre autres, pour l'étude de la morphogenèse florale chez Arabidopsis Thaliana [71, 73, 33], mais aussi pour l'analyse de plusieurs processus développementaux de la drosophile, en particulier la segmentation [98, 99, 100], la formation du disque imaginal d'aile [44, 45] et le développement des organes sensoriels [42].

Certaines de ces applications concernent la formation de frontières de part et d'autres desquelles les cellules se maintiennent dans des états différents. Ces frontières sont créées et maintenues par des mécanismes de signalisation inter-cellulaires. Ils ont donc nécessité l'utilisation de modèles multi-cellulaires [44, 45, 100]. Ceux-ci ont été obtenus par la duplication d'un réseau intra-cellulaire et l'ajout de connections entre les copies, afin de prendre en compte la signalisation inter-cellulaire (ici par sécrétion de protéines qui diffusent et se fixent sur des récepteurs membranaires).

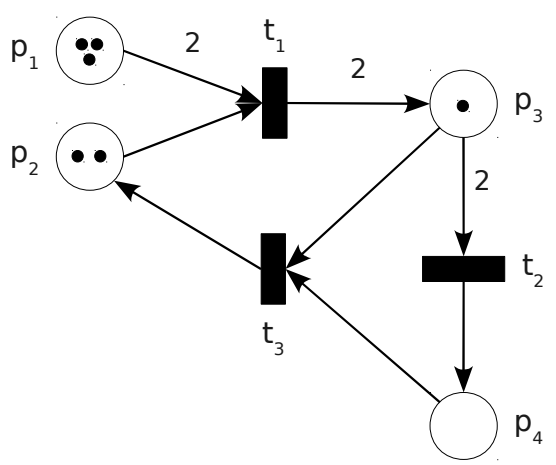
Plusieurs modèles logiques ont également été récemment proposés pour l'étude du cycle cellulaire chez différents organismes [50, 65, 34, 26, 25, 35, 51], où encore sur le contrôle de la détection et de la réparation des dommages à l'ADN [1].

Dans le cadre du système que nous étudions ici, l'approche logique a déjà été utilisée pour modéliser différents éléments du contrôle de la réponse immunitaire [56, 55, 76, 70, 95]. En particulier, des modèles récents se concentrant sur l'activation [95] et la différenciation [70] des lymphocytes T sont présentés dans le chapitre 4.

2.3 Réseaux de Petri

2.3.1 Réseaux de Petri standards

Introduit par C. A. Petri, le formalisme des réseaux de Petri (RdP) est couramment utilisé pour la représentation de systèmes discrets concurrents [77]. Un réseau de Petri est un graphe biparti orienté pondéré $R = (S, T, F, M_0, W)$:



PRE

	t_1	t_2	t_3
p_1	2	0	0
p_2	1	0	0
p_3	0	2	1
p_4	0	0	1

POST

	t_1	t_2	t_3
p_1	0	0	0
p_2	0	0	1
p_3	2	0	0
p_4	0	1	0

Figure 2.4: Exemple de réseau de Petri. La partie gauche montre la représentation graphique d'un réseau de Petri comportant 4 places (p_1 à p_4) et trois transitions (t_1 à t_3). Les poids différents de 1 sont indiqués au dessus des arcs concernés. Les points noirs dans les places représentent les jetons. La partie droite donne les matrices PRE et $POST$ correspondantes. Pour le marquage $(3, 2, 1, 0)$ montré ici, seule la transition t_1 est habilitée.

- $S \bigcup T$ est l'ensemble fini des noeuds, partitionné en deux sous-ensembles correspondant à deux types de noeuds différents : S pour les *places* et T pour les *transitions* ($S \neq \emptyset$, $T \neq \emptyset$ et $S \bigcap T = \emptyset$).
- $F \subseteq (S \times T) \bigcup (T \times S)$ est l'ensemble des arcs, reliant des noeuds de types différents.
- $M_0 \in \mathbb{N}^{card(S)}$ est le *marquage* initial du réseau. Ce vecteur donne le nombre de *jetons* initialement présents dans chaque place.
- $W : (S \times T) \bigcup (T \times S) \rightarrow \mathbb{N}$ donne le poids de chaque arc. $W(a) = 0$ si et seulement si $a \notin F$.

Les réseaux de Petri peuvent être représentés graphiquement ou sous forme matricielle (voir figure 2.4). Dans la représentation graphique, les places sont représentées par des cercles, les transitions par des rectangles et les jetons par des points à l'intérieur des places. W peut être représenté par les deux matrices PRE et $POST$. Ces matrices ont autant de lignes et de colonnes que le réseau a de places et de transitions. Les éléments de ces matrices correspondent aux poids des arcs : $PRE_{p,t} = W(p, t)$ et $POST_{p,t} = W(t, p)$. La matrice d'incidence d'un RdP est la matrice C, définie comme la différence entre les matrices $POST$ et PRE, donnant ainsi le bilan en termes de jetons du tir de chaque transition : pour chaque place p et chaque transition t, $C_{p,t} = POST_{p,t} - PRE_{p,t}$.

2.3.1.1 Dynamique

L'état du système est représenté par le marquage du réseau : $M \in \mathbb{N}^{card(S)}$. Ce vecteur indique le nombre de jetons présents dans chaque place.

Les transitions correspondent aux événements pouvant modifier ce marquage. Une transition t est habilitée lorsque le nombre de jetons présents dans chaque place d'entrée p (*i.e.* p telle que $(p, t) \in F$) est supérieur ou égal à $W(p, t)$, poids de l'arc reliant p à t. La transition t peut alors se déclencher. Son déclenchement consomme $W(p, t)$ jetons dans ses places d'entrée p et produit $W(p', t)$ dans ses places de sortie p' ($(t, p') \in F$). Le marquage des places d'entrée et de sortie de la transition est donc généralement modifié par son déclenchement.

Sous forme matricielle, t est habilitée pour le marquage M si $M \geq PRE_t$ (ou PRE_t est la

t-ième colonne de la matrice PRE). Le déclenchement de la transition t conduit alors au marquage $M' = M + C_t$.

Le comportement dynamique d'un réseau de Petri peut être représenté sous forme d'un graphe de marquage. Les noeuds de ce graphe correspondent aux marquages du réseau atteignables à partir du marquage initial M_0. Un arc relie deux marquages M et M' s'il existe une transition t, habilitée pour le marquage M, telle que $M' = M + C_t$.

2.3.1.2 Quelques définitions

Le déclenchement successif d'une série de transitions est appelé séquence de tirs. Il peut être décrit par son vecteur de tirs $S \in \mathbb{N}^{card(T)}$, donnant le nombre de déclenchements de chaque transition du réseau au cours de la séquence. Le marquage atteint à partir de M à la suite d'une séquence de tirs satisfaisant S est donné par $M' = M + C.S$. Cela suppose que chaque transition est habilitée au cours de la séquence.

Un T-invariant $\mathcal{T}_i$ est une séquence de tirs ne modifiant pas le marquage du réseau. Elle correspond à un vecteur S tel que $C.S = 0$.

Un P-invariant $\mathcal{P}_i$ est un ensemble de places telles que la somme pondérée de leur marquage est constante. Il est donné par les composantes non-nulles d'un vecteur y tel que $C^T.y = 0$, où C^T est la transposée de la matrice d'incidence C.

Un marquage pour lequel aucune transition n'est habilitée est dit *mort*. Un tel marquage n'a aucun arc sortant dans le graphe de marquage. Un réseau est dit sans inter-blocage s'il ne permet d'atteindre aucun marquage mort.

Une transition est dite vivante si, pour tous les marquages atteignables à partir de M_0, il existe une séquence de tirs habilitée contenant cette transition. Un réseau est vivant si toutes ses transitions sont vivantes. Le graphe de marquage d'un RdP contenant au moins une transition vivante ne contient aucun marquage mort. Par contre, un RdP sans aucune transition vivante ne génère pas forcement de marquage mort. Afin d'affiner ces définitions, il existe différents degrés de vivacité, allant de L-0 (mort) à L-4 (vivant), voir [77].

Une place p est bornée s'il existe un entier k tel que tout marquage atteignable M vérifie $M(p) \leq k$. Un réseau est dit borné si toutes ses places vérifient cette propriété.

2.3.2 Extensions

De nombreuses extensions permettent d'étendre le pouvoir expressif du formalisme.

Parfois, une transition t crée autant de jetons qu'elle en consomme dans une place p ($PRE_{p,t} = POST_{p,t} \neq 0$, on a alors $C_{p,t} = 0$). Les deux arcs (p,t) et (t,p) peuvent alors être remplacés par un arc bidirectionnel de même poids appelé arc test. Cet arc indique que le déclenchement de la transition t requiert la présence de $PRE_{p,t}$ jetons dans la place p mais qu'il ne modifie pas son marquage.

Dans un RdP standard, le marquage du réseau est discret et le déclenchement d'une transition le modifie de manière ponctuelle. Les RdP continus utilisent des transitions qui modifient le marquage des places incidentes de façon continue [2]. Dans ces réseaux, les poids des arcs et les éléments du vecteur de marquage sont des valeurs réelles positives. Chaque transition a pour taux de déclenchement une valeur réelle qui peut dépendre du marquage.

Le comportement dynamique d'un tel réseau correspond à celui d'un système d'Équations Différentielles Ordinaires. Les RdP hybrides permettent de mélanger transitions continues et transitions classiques au sein d'un même réseau.

Lorsque plusieurs transitions sont habilitées, la concurrence est généralement gérée de manière similaire à la mise à jour asynchrone utilisée pour le formalisme logique : le graphe de marquage contient alors plusieurs successeurs, chacun correspondant au déclenchement d'une seule transition. Ce comportement peut être affiné par l'introduction de contraintes temporelles, aléatoires ou non, sur les transitions (voir [6] et références incluses). Parmi ces extensions, les RdP stochastiques distribuent exponentiellement ces probabilités, ils peuvent donc être traduits en chaînes de Markov à temps continu.

Les RdP colorés associent une information (couleur) aux jetons et des expressions aux arcs [62]. Ces expressions contraignent les valeurs des jetons présents dans les places d'entrée et définissent les valeurs de ceux déposés dans les places de sortie. L'utilisation de ces expressions permet de définir des réseaux plus compacts que les RdP standards. Il est possible de "déplier" un RdP coloré en RdP standard, ce qui permet de profiter des méthodes d'analyse basées sur leur représentation matricielle.

2.3.3 Applications en biologie

Pour une présentation générale de l'utilisation des réseaux de Petri pour la modélisation de systèmes biologiques, voir [46, 20].

Les réseaux de Petri représentent naturellement les réactions chimiques (c'était d'ailleurs une des motivations de C.A. Petri[1]). Les places représentent alors les espèces chimiques, les transitions représentent les réactions et les poids des arcs donnent les coefficients stoechiométriques. Reddy et al. [85] ont été parmi les premiers à utiliser des réseaux de Petri pour la modélisation de processus biochimiques. De nombreuses autres études ont suivi. Les RdP ont également été appliqués à l'étude de voies de transduction de signaux [94]. Les P-invariants de ces réseaux correspondent à des lois de conservation dans les réactions sous-jacentes. Les T-invariants correspondent aux modes élémentaires utilisés dans l'étude de réseaux métaboliques : ils peuvent être interprétés comme un ensemble de réactions permettant au système de se maintenir autour d'un état stable, par exemple pour l'homéostasie.

Les réseaux de Petri ont également été utilisés pour la représentation de réseaux de régulation. En particulier, nous avons introduit une méthode formelle de traduction de réseaux logiques en RdP (voir [21, 19] et section 8.3). Le graphe de marquage du RdP obtenu est isomorphe au graphe de transitions d'états du modèle logique de départ. Cette traduction permet d'utiliser les outils existants pour l'analyse de RdP, en particulier pour l'analyse d'atteignabilité de marquages morts, qui correspondent aux états stables dans le graphe de transitions d'états. Ces outils ont, en particulier, servi à l'analyse d'un modèle multicellulaire de la segmentation chez l'embryon de drosophile [100]. Cette traduction en RdP permet également de définir des modèles intégrant des voies métaboliques et leur régulation [111].

Les applications précédemment citées utilisent toutes des RdP standards et une approche qualitative. D'autres travaux reposent sur l'utilisation d'extensions quantitatives. Les RdP stochastiques ont, par exemple, été utilisés pour modéliser le cycle cellulaire [75]. D'autres systèmes ont été étudiés à l'aide de RdP hybrides fonctionnels [78].

[1] www.scholarpedia.org/article/Petri_net

Chapitre 3

Outils informatiques

3.1 Programmation : approches et langages

Bien que ce document n'ait pas vocation à être une introduction à la programmation, je souhaite présenter quelques notions importantes pour la compréhension de mon travail de thèse. Pour une introduction plus complète, voir par exemple [39, 37].

3.1.1 Collections et structures de données

En informatique, une structure de données est une méthode d'organisation de données permettant leur traitement. Il existe souvent plusieurs structures permettant de réaliser la même tâche. Le choix des structures de données utilisées est souvent un compromis entre performance (rapidité, occupation mémoire) et simplicité.

La plus simple des structures de données est l'enregistrement. Celui-ci permet de regrouper plusieurs valeurs dans une même variable. Par exemple un composant de régulation g dans un modèle logique est représenté par son identifiant, un nom long, un niveau d'activité maximal et une fonction logique. On a donc un enregistrement contenant ces quatre éléments, auxquels on accède à travers la variable représentant l'enregistrement : $g.id$, $g.name$, $g.maxlevel$ et $g.function$.

L'ensemble des composants d'un graphe de régulation est alors une collection de tels enregistrements. La plus simple des collections est le tableau. C'est une zone mémoire de taille fixe dans laquelle on peut stocker une série d'éléments. Il a l'avantage de permettre un accès rapide à n'importe quel élément via sa position dans le tableau. Par contre l'insertion (et dans certains cas la suppression) d'un élément au milieu d'un tableau nécessite de déplacer tous les éléments suivants. De plus, un tableau ayant une taille fixe, il faut souvent en créer un nouveau afin de le faire "grandir". Une série d'éléments peut également être stockée dans une liste chaînée. Dans ce type de liste, chaque élément est un enregistrement contenant l'élément lui-même ainsi qu'un lien vers l'élément suivant. Les insertions/suppressions d'éléments et l'extension d'une liste chaînée sont des opérations rapides mais l'accès au i^{ieme} élément nécessite le parcours de la liste.

En pratique, on souhaite pouvoir retrouver le composant à partir de son identifiant sans parcourir cette liste. On utilise pour cela une table de correspondance ("map" dans la nomenclature java). Cette structure de données permet de stocker un ensemble de *clés* uniques auxquelles on associe une *valeur*. Ici la clé est l'identifiant du composant et la valeur est

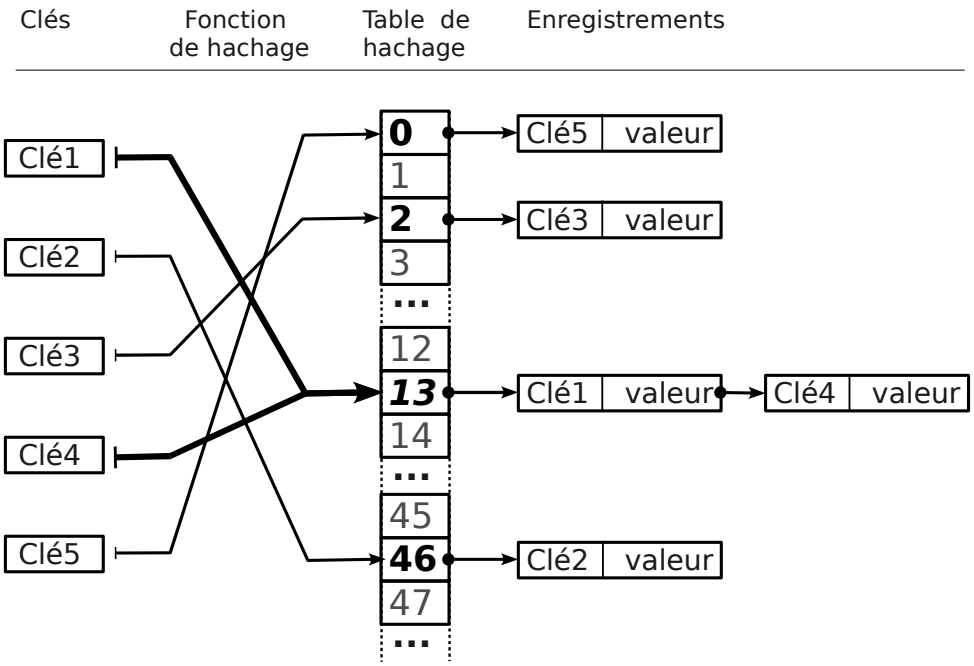


Figure 3.1: Exemple de table de hachage. La fonction de hachage permet de déterminer la position de l'enregistrement associé à une clé. Cet exemple montre une collision (en rouge) résolue par l'utilisation d'une liste chaînée [Basé sur Wikimedia Commons].

l'enregistrement complet. Il existe plusieurs façons d'implémenter une telle table, la plus courante étant la *table de hachage*. Cette technique consiste à stocker les entrées dans un tableau, leur position dans le tableau dépendant d'une *fonction de hachage* appliquée à la clé. La fonction de hachage prend un élément en entrée et calcule une *somme de contrôle* (hashcode). Les clés ayant le même hashcode sont ensuite stockées (par exemple) dans une liste chaînée (voir figure 3.1). Il est donc important de bien choisir la fonction de hachage et la taille du tableau afin d'éviter au maximum les collisions.

3.1.2 Programmation orientée objet

Il arrive couramment que des traitements spécialisés soient associés à une structure de données particulière. La Programmation Orientée Objet (POO) permet de clarifier cette association. Une *classe* regroupe un ensemble d'*attributs* (l'équivalent d'un enregistrement) et de *méthodes* (les traitements associés). La classe est la description abstraite d'un type d'objet. L'objet lui-même est une *instance* de la classe avec des valeurs propres pour les attributs. Le *constructeur* est une méthode particulière de la classe permettant de créer de nouveaux objets.

Il est alors possible de créer un objet et d'interagir avec lui au travers des méthodes de sa classe :

```
// création d'une instance de la classe HashMap
map = new HashMap()

// ajout de nouveaux enregistrements (clé,valeur)
// à l'aide de la méthode add
map.add(cle1, valeur1)
map.add(cle2, valeur2)
```

Le principal attrait de la POO est l'*héritage* qui permet de créer une classe par extension d'une classe existante. Une classe hérite des attributs et méthodes de sa classe mère. Elle peut en ajouter de nouveaux ou *surcharger* des méthodes de sa classe mère. Ceci permet la création d'une hiérarchie d'objets plus ou moins spécialisés partageant du code au travers de leurs classes parentes communes.

3.1.3 Langages

Au cours de ce travail, j'ai essentiellement utilisé les langages Java[1] et Python[2].

Le langage Java est un langage orienté objet introduit officiellement en 1995 et développé principalement au sein de Sun Microsystems. Il est fortement inspiré du C++ mais intègre une bibliothèque standard bien plus complète. Il a surtout été conçu pour s'exécuter de manière identique quelle que soit la plateforme. Un langage natif est compilé en un fichier binaire contenant des instructions pouvant être exécutées directement par le processeur. Le format de ce fichier et les instructions disponibles varient en fonction du type de processeur et du système d'exploitation. Pour palier ce problème, les langages interprétés utilisent un format intermédiaire : le *bytecode*. Celui-ci ne peut pas être exécuté directement, il nécessite l'utilisation d'un interpréteur, ici la *machine virtuelle* Java. Celle-ci traduit les instructions contenues dans le bytecode Java en instructions exécutables par le processeur du système utilisé. Le même fichier peut donc être utilisé sur tous les systèmes disposant de la machine virtuelle Java (alors qu'un programme natif doit être recompilé pour chaque système). On peut noter que certains compilateurs de code natif (en particulier gcc) utilisent une méthode similaire comme étape intermédiaire. Les versions récentes de la machine virtuelle disposent d'un compilateur "Just In Time" (JIT) qui optimise le bytecode pendant l'exécution, ce qui permet d'obtenir des performances proches de celles d'un programme natif. Java se charge également de la gestion de la mémoire : un ramasse-miettes (garbage-collector) libère les objets qui ne sont plus utilisés.

Le langage Python est un langage multi-paradigme introduit en 1990 par Guido van Rossum. Longtemps marginal, son développement s'est accéléré à partir de 2000. Comme Java, Python est un langage multi-plateforme utilisant un ramasse-miettes. Il nécessite l'utilisation d'une machine virtuelle, l'interpréteur Python. L'interpréteur Python officiel (CPython) ne dispose pas de mécanismes d'optimisation JIT. On peut signaler l'existence d'autres interpréteurs Python, en particulier IronPython[3] et Jython[4]. Ce dernier utilise la machine virtuelle Java, il profite donc de son compilateur JIT et permet de mélanger les deux langages. Comme Java, Python est un langage fortement typé, c'est-à-dire qu'une variable a un type (la classe dont elle est une instance) bien défini, il est impossible de l'utiliser comme une variable d'un type différent. Par contre, Java utilise un typage statique (le type d'une variable est déclaré à sa création et ne peut pas changer) alors que Python utilise un typage dynamique : le type d'une variable peut changer au cours de l'exécution du programme. Python intègre également des fonctionnalités de programmation fonctionnelle (compréhension de liste et fonctions anonymes), ainsi que des facilités de programmation (générateurs, décorateurs). Il est souvent utilisé comme langage de script et son interpréteur peut être lancé en mode interactif.

[1] java.sun.com
[2] www.python.org
[3] www.ironpython.com
[4] www.jython.org

Ces deux langages disposent également d'un très bon support de l'*introspection*, permettant d'interroger à l'exécution le type d'un objet, ce qui simplifie grandement la mise en place d'une architecture modulaire.

3.2 Arbres et Diagrammes de décision

Le formalisme logique est appliqué à des modèles de réseaux de régulation de taille de plus en plus importante (voir par exemple [95, 100, 97] et chapitre 9). Il devient alors fréquent que certains composants possèdent de nombreux régulateurs, ce qui provoque une forte augmentation de la taille des listes de paramètres qui leur sont associées. Ces listes deviennent alors difficiles à manier (un composant soumis à r régulations a 2^r paramètres logiques). Nous avons donc cherché une représentation plus efficace et nous nous sommes tournés vers les fonctions logiques et les structures de données utilisées pour leur représentation en informatique, en particulier les arbres et les diagrammes de décision.

3.2.1 Arbres

Un arbre est un graphe connexe acyclique. Je décrirai ici uniquement les arbres orientés et enracinés, communément utilisés pour représenter des structures hiérarchiques. On y distingue deux types de noeuds : les feuilles, qui n'ont aucun arc sortant, et les noeuds internes qui ont au moins un arc sortant. Dans un arbre, chaque noeud (feuille ou noeud interne) a un seul arc entrant, sauf la racine qui n'en a aucun et à partir de laquelle tous les autres noeuds peuvent être atteints.

Les noeuds cibles des arcs sortants d'un noeud sont ses fils, ce noeud est leur (unique) parent. La racine est donc l'ancêtre commun de tous les noeuds de l'arbre. Un arbre est donc défini par sa racine, chaque noeud d'un arbre étant la racine d'un sous arbre.

La représentation graphique des arbres varie selon le domaine, en particulier pour la position de la racine. Par exemple, les arbres de classification, sont généralement dessinés avec la racine en bas comme de vrais arbres, alors que les informaticiens placent généralement la racine en haut, les arcs allant tous vers le bas. Ils sont également parfois dessinés horizontalement, généralement avec la racine à gauche, ou encore disposés circulairement. Les arbres représentés dans ce document auront tous la racine en haut.

On peut parcourir les arbres à partir de leur racine et visiter chaque noeud. Ce parcours peut se faire en largeur ou en profondeur. Dans un parcours en profondeur, on visite un noeud en visitant ses fils les uns après les autres, alors qu'en largeur on visite les "frères" d'un noeud avant de visiter les fils de ce dernier.

L'utilisation d'arbres est très courante en algorithmique, je me concentrerai ici sur leur utilisation pour représenter des fonctions logiques. Une présentation plus générale et d'autres exemples d'utilisation se trouvent dans de nombreux ouvrages de référence, notamment [24, 92].

3.2.2 Représentation de fonctions sous forme d'arbres

Une opération arithmétique ou une fonction logique peut se représenter naturellement sous forme d'arbre. Dans ces arbres, les noeuds internes représentent les opérateurs, tandis que

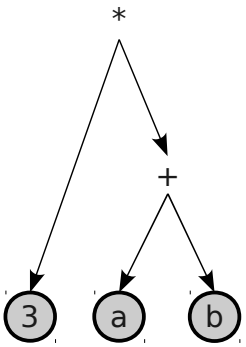


Figure 3.2: Arbre représentant l'opération $3 * (a + b)$

les feuilles représentent les termes (constantes ou variables). Ainsi, l'opération $3*(a+b)$ peut être représentée par un arbre ayant pour racine un noeud de multiplication, lequel a pour fils la feuille 3 et un noeud d'addition ayant à son tour pour fils les feuilles a et b (figure 3.2). La fonction peut alors être évaluée en parcourant l'arbre et en assignant des valeurs aux noeuds internes en fonction des valeurs de leurs fils. Par exemple, un noeud d'addition a pour valeur la somme des valeurs de ses fils. L'évaluation d'une fonction nécessite donc un parcours complet de l'arbre la représentant. Dans le cas de fonctions logiques, ce parcours peut ne pas couvrir la totalité de l'arbre, en effet si l'un des fils d'un noeud "ET" est faux, ce noeud est également faux, quelle que soit la valeur de ses autres fils.

3.2.3 Arbres de décision

Les arbres de décision sont utilisés pour la représentation de *problèmes de décision*, qui dépendent de la valeur de *variables de décision*. Les feuilles de ces arbres représentent les résultats possibles et les noeuds internes sont associés aux variables. L'évaluation d'un tel arbre se fait en parcourant un chemin de la racine à une feuille : on descend dans l'arbre en suivant, pour chaque noeud interne, le chemin correspondant à la valeur de la variable associée. La feuille ainsi atteinte donne le résultat recherché (voir figure 3.3). Dans la représentation graphique utilisée ici, le fils le plus à gauche correspond à la valeur minimale de la variable correspondante et le fils le plus à droite à sa valeur maximale.

Les arbres de décision les plus couramment utilisés sont des arbres binaires, dans lesquels chaque noeud interne a deux fils, correspondant aux valeurs 0 (faux) et 1 (vrai) de la variable associée. Des variantes multi-valuées existent.

3.2.4 Diagrammes de décision

Tout comme un arbre de décision, un diagramme de décision est un graphe connexe acyclique orienté et enraciné. Cependant, les noeuds d'un diagramme peuvent avoir plusieurs parents, ce qui permet une représentation plus compacte en évitant la duplication de sous-diagrammes identiques. Un diagramme de décision se parcourt cependant comme un arbre de décision, en partant de la racine et en suivant le chemin correspondant aux valeurs des variables de décision.

Le terme "diagramme de décision" fait habituellement référence à des diagrammes binaires réduits et ordonnés (ROBDD pour Reduced Ordered Binary Decision Diagram) [12, 121]. Les

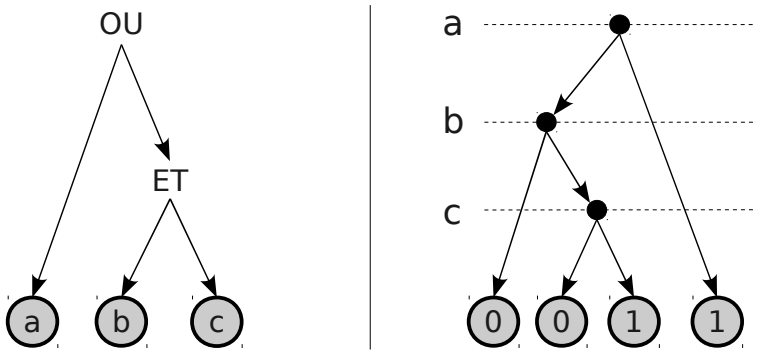


Figure 3.3: L'arbre d'opération (à gauche) représente la fonction $a \vee (b \wedge c)$. L'arbre de décision (à droite) représente sa table de vérité. Ces deux types d'arbres peuvent être utilisés pour évaluer la valeur d'une fonction logique mais cette évaluation nécessite (dans le pire des cas) le parcours complet de l'arbre d'opération alors qu'il suffit de parcourir une branche de l'arbre de décision. L'utilisation d'arbre de décision repose sur l'énumération des valeurs possible de chaque variable, ils sont donc généralement plus grands et ne peuvent s'appliquer qu'aux cas où chaque variable de décision peut prendre un nombre fini de valeurs.

diagrammes binaires sont utilisés pour la représentation de fonctions Booléennes. Ces diagrammes sont dits ordonnés car un ordre sur les variables de décision est imposé, de telle sorte qu'un noeud interne ne peut avoir comme fils que des feuilles ou des noeuds internes d'ordre supérieur. Ces diagrammes sont réduits en appliquant les deux règles suivantes :

1. Identification des sous-diagrammes identiques. Deux sous-diagrammes sont identiques si leurs racines sont des noeuds de même ordre et si leurs fils sont identiques deux à deux. Deux feuilles sont identiques si elles ont la même valeur.
   Une fois détectés, les doublons sont supprimés : un seul sous-diagramme est conservé et tous les prédécesseurs pointent sur lui. Ceci permet de réduire l'empreinte mémoire de la structure et simplifie l'application pratique de la règle suivante. Cette étape correspond à une factorisation de la formule logique.
2. Un noeud dont les fils sont tous identiques n'apporte pas d'information. En effet, pendant le parcours, le fils choisi pour ce noeud n'aura aucun impact sur la feuille atteinte. Ce noeud peut donc être remplacé par son "unique" fils.

Ces règles de réduction et l'ordre sur les variables de décision confèrent aux ROBDD quelques propriétés intéressantes. Tout d'abord, ils fournissent une représentation canonique des fonctions logiques et peuvent donc être utilisés pour tester l'équivalence de deux fonctions, à condition d'utiliser le même ordre sur les variables. De plus, l'ordre impose une limite sur la profondeur du diagramme, en effet aucun chemin ne peut passer par plus de noeuds internes qu'il n'y a de variables de décision (l'ordre implique qu'une même variable ne peut être prise en compte plus d'une fois par branche de l'arbre). L'évaluation nécessite donc au plus autant de tests qu'il y a de variables. Enfin, chaque noeud interne permet d'atteindre plusieurs feuilles différentes : si cela n'était pas le cas, cette partie du diagramme aurait été simplifiée et remplacée par son unique feuille. L'évaluation ne demande donc jamais de test inutile.

Nous avons donc une représentation canonique optimale pour un ordre donné mais pas pour autant optimale dans le cas général. En particulier, le choix de l'ordre peut avoir une influence sur la taille du diagramme (voir figure 3.4). Le choix d'un "bon" ordre est un problème complexe [110, 10]. Plusieurs heuristiques permettent de déterminer un bon ordre pour un diagramme donné [11, 31], mais plusieurs diagrammes auront des ordres optimaux différents, ce qui complique la combinaison de ces diagrammes par la suite. En pratique,

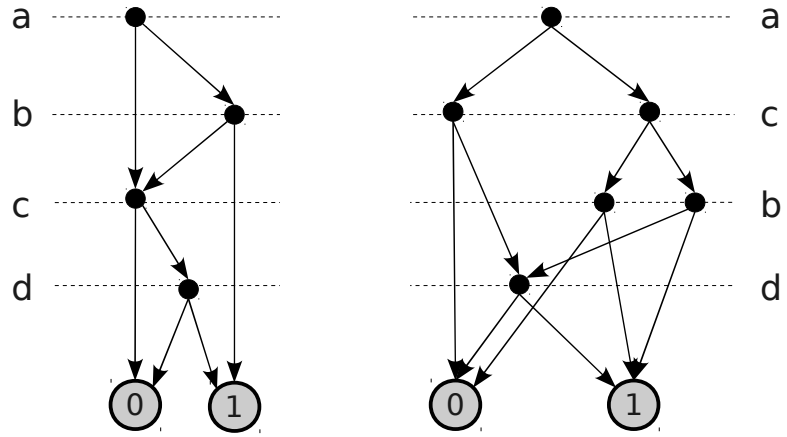


Figure 3.4: Impact de l'ordre des variables dans un BDD. Les deux BDD montrés ici représentent la même fonction pour deux ordres différents. Le BDD de gauche contient 4 noeuds internes alors que celui de droite en contient 6.

plusieurs implémentations modifient dynamiquement l'ordre en inversant localement la position de certaines variables [93]. Nous recherchons une structure de données permettant de représenter efficacement les fonctions logiques associées aux composants de nos modèles. Nous souhaitons également développer des algorithmes nécessitant de manipuler, comparer ou combiner ces fonctions (voir chapitres 6 et 7). Dans cette optique, les avantages des diagrammes de décision compensent largement leurs inconvénients.

Importante pour l'efficacité de cette représentation, la réduction d'arbres de décision en diagrammes peut cependant rendre moins lisible leur représentation graphique. Certains diagrammes représentés dans ce document ne seront donc pas complètement réduits.

Le terme générique xDD désigne l'ensemble des structures de données utilisant des principes similaires. Il existe en particulier une variante multi-valuée : les OMDD (Ordered Multilevel Decision Diagram [53]). Les variables déterminant le choix sont alors multi-valuées : chaque variable de décision V_i a un nombre spécifique de valeurs possibles. Un noeud interne du diagramme a donc autant de fils que la variable de décision qu'il représente a de valeurs possibles. On peut également mentionner les IDD (Interval Decision Diagram), adaptation des MDD aux cas où les valeurs consécutives des variables de décision conduisent aux mêmes noeuds et peuvent donc être regroupées en intervalles.

3.2.5 Implémentation

Les BDD sont populaires en informatique, il en existe donc plusieurs implémentations, notamment BuDDy[5], CUDD[6] et JavaBDD[7]. JavaBDD est une bibliothèque Java fournissant une API commune à plusieurs implémentations natives, mais aussi un port Java de BuDDy. Je ne détaillerai pas ici leurs fonctionnalités avancées, comme le réordonnement dynamique des variables, mais uniquement quelques principes généraux. Les variantes telles que MDD ou IDD sont moins populaires (il n'existe pas de bibliothèque java pour ces structures de

[5] buddy.sourceforge.net
[6] vlsi.colorado.edu/~fabio/CUDD
[7] javabdd.sourceforge.net

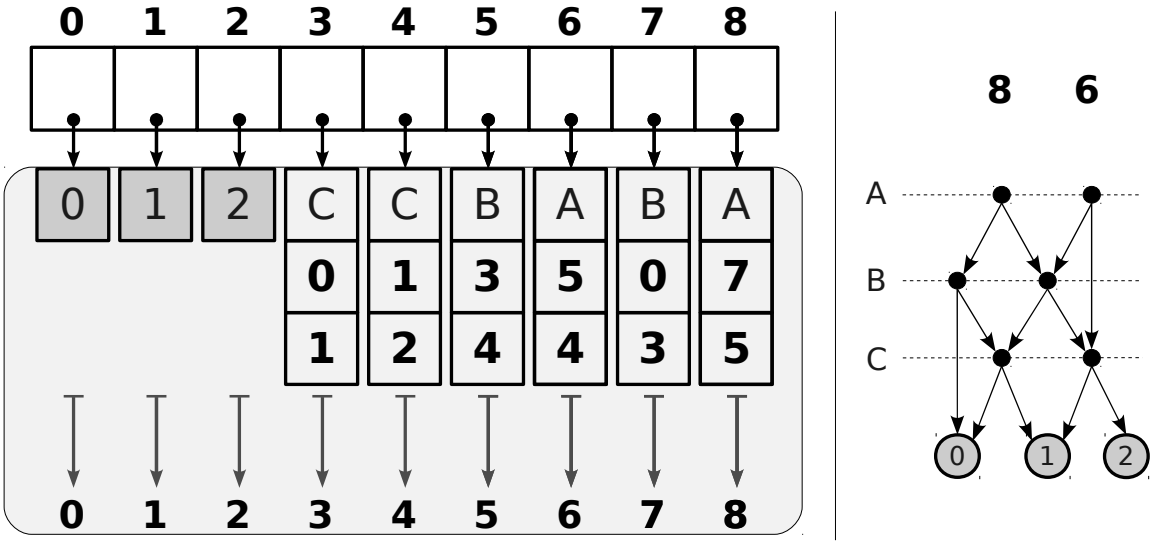


Figure 3.5: Représentation interne des MDD. Un noeud est identifié par sa position dans un tableau. A cette position, on retrouve une description du noeud : valeur pour les feuilles (0,1 et 2), rang et liste de fils pour les noeuds internes (3 à 8). Afin, d'éviter les duplications, il faut pouvoir déterminer rapidement si un noeud est déjà dans le tableau. Au lieu de parcourir le tableau, on utilise pour cela une fonction de hachage (symbolisée par le grand rectangle gris). Dans BuDDy, tout ceci est représenté au sein d'un seul grand tableau, utilisant des séries de 5 blocs successifs : [type (vide,interne,feuille)/suivant dans le hachage/rang ou valeur/fils 0/fils 1]. Pour optimiser encore l'occupation mémoire, BuDDy ajoute également un compteur d'utilisation, permettant de supprimer les noeuds qui ne sont plus utilisés.

données), cependant, les principes présentés ici peuvent s'appliquer aussi bien aux BDD qu'aux MDD, moyennant quelques ajustements.

Notre but est de manipuler des diagrammes ordonnés réduits. Pour cela deux approches sont possibles : appliquer les règles de réduction a posteriori ou s'assurer pendant la construction des diagrammes qu'ils restent bien réduits. La seconde méthode est plus robuste et permet d'éviter complètement le stockage de doublons.

Les bibliothèques existantes utilisent généralement une "usine" (factory) à laquelle toutes les demandes de création de noeuds sont adressées. Lorsque tous les fils du noeud demandé sont identiques, celle-ci ne crée pas de nouveau noeud et retourne directement cet unique fils. La détection de sous-diagrammes identiques est plus complexe. Pour cela, il faut que l'usine garde en mémoire l'ensemble des noeuds précédemment créés et teste si le noeud demandé existe déjà, auquel cas elle retournera le noeud existant au lieu d'en créer un nouveau. Si chaque noeud dispose d'un identifiant unique, un nouveau noeud interne est défini par son rang et les identifiants de ses fils. En les utilisant comme graines pour un code de hachage, il est possible de retrouver rapidement un éventuel noeud identique déjà existant. Une représentation simplifiée (et adaptée aux MDD) de la structure utilisée par BuDDy (et sa réimplémentation en java incluse dans JavaBDD) est donnée dans la figure 3.5.

L'usine dispose également de méthodes permettant la combinaison de diagrammes, principalement par des opérations Booléennes ET, OU, NON et dérivées.

Par exemple, un ET entre deux diagrammes est une opération récursive définie comme suit :

- si l'une des racines est une feuille 0, le résultat est cette feuille ;
- si l'une des racines est une feuille 1, le résultat est l'autre racine ;

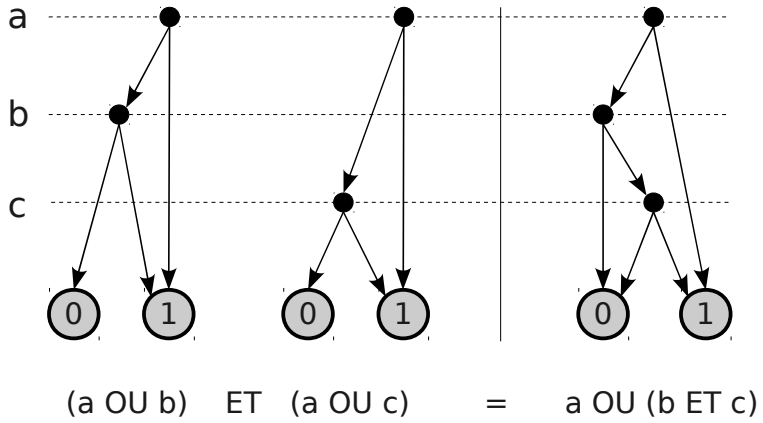


Figure 3.6: Exemple de combinaison de diagrammes de décision

- sinon, les deux racines sont des noeuds internes :
  - s'ils sont de même rang, le résultat est un noeud interne de même rang ayant pour fils les résultats d'un ET entre les fils des noeuds de départ ;
  - sinon, le résultat est un noeud interne dont le rang est le plus petit des rangs des noeuds d'origine et dont les fils sont les résultats d'un ET entre les fils de ce noeud et le second noeud.

3.2.6 Applications en biologie

Les BDD sont utilisés en électronique (en particulier pour la conception de circuits intégrés), ainsi qu'en model-checking symbolique [22, 13]. Ils sont donc utilisés indirectement par les analyses de modèles biologiques utilisant des outils de model-checking [3, 16, 9, 7]. D'autres types de diagrammes de décision ont été utilisés dans ce cadre, en particulier les IDD qui permettent de prendre en compte de larges intervalles de valeurs [103].

Les diagrammes de décision utilisés en model-checking servent à représenter efficacement des régions de l'espace des états. Cette représentation a été appliquée explicitement à des modèles discrets, afin de rechercher les attracteurs dans le graphe de transitions d'états [38]. Je présente ici plusieurs méthodes reposant sur l'utilisation de MDD pour représenter les fonctions logiques associées aux composants du graphe de régulation [81, 80].

Chapitre 4

Modélisation des Lymphocytes T

4.1 État de l'art

Le système immunitaire adaptatif fait l'objet de nombreuses études, y compris à l'aide de modèles mathématiques. En particulier, plusieurs modèles ont été proposés pour les différentes étapes du développement des lymphocytes T. Ces modèles concernent, par exemple, les mécanismes de recombinaison conduisant à la génération d'un large répertoire de TCR [52, 104], ainsi que les étapes de sélection des variants générés [101, 17]. D'autres modèles se concentrent sur l'activation des lymphocytes T [41, 57], la mise en place du signal TCR [43, 95], ou plus particulièrement sur le rôle qu'y joue la synapse immune [63]. Concernant la différenciation des cellules T auxiliaires, la différentiation Th1/Th2 a également fait l'objet de plusieurs études [8, 49, 69, 124, 70]. La réponse immunitaire implique des interactions entre de nombreux types cellulaires (voir par exemple [15] pour une revue sur les interactions entre lymphocytes CD4+ et CD8+) et peut également être étudiée à l'échelle de la population cellulaire [64, 14].

Ces modèles utilisent un large éventail de méthodes de modélisation : cartes de régulation [40], modèles logiques [76, 57, 70, 95], équations différentielles [8, 64, 101, 124] ainsi que des méthodes stochastiques, en particulier les chaînes de Markov [104] et des simulations de Monte Carlo [63, 52].

Pour ce travail, je me suis basé sur les deux modèles logiques introduits ci-dessous. Ces modèles sont disponibles sur le dépôt de modèles du site de GINsim[1].

4.2 Différenciation Th1/Th2

Le travail que je présente ici porte sur la différenciation des lymphocytes T auxiliaires. Comme nous venons de le voir, il existe plusieurs modèles prenant en compte la différenciation Th1/Th2. Le modèle logique proposé par Mendoza [70] prend en compte de nombreux acteurs de cette différenciation. Il comporte 17 composants : Tbet, GATA3 (les facteurs de transcription respectivement spécifiques des lignées Th1 et Th2) et 5 voies de signalisation activant ou réprimant ces facteurs : IFNb, IFNg, IL4, IL12 et IL18. Ce modèle a quatre états stables correspondant aux types cellulaires connus : Th0 (naïf), Th1, et Th2, ainsi qu'à Th1*, une variante de Th1 avec une plus forte expression de Tbet et IFNg. Une traduction en

[1] gin.univ-mrs.fr/GINsim/model_repository.html

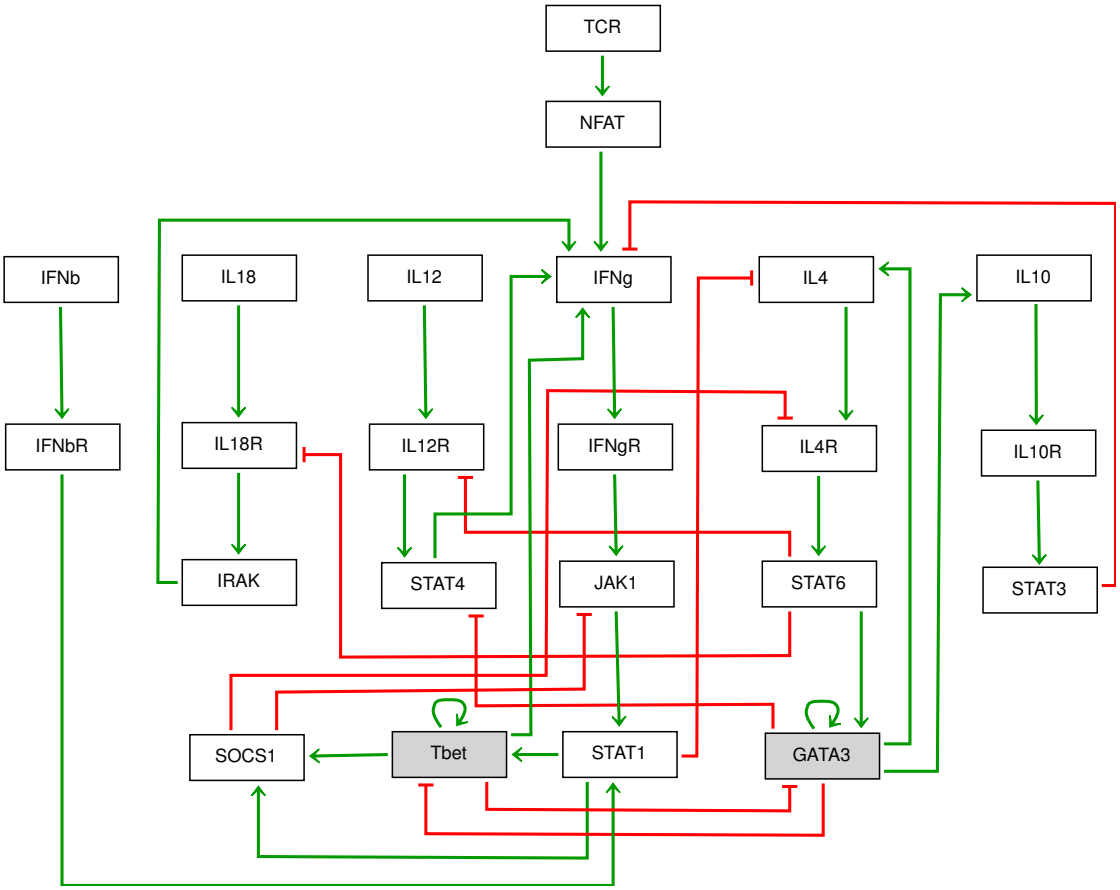


Figure 4.1: Modèle de la différenciation Th1/Th2 [72]. Ce modèle, comportant 23 composants, rend compte de la différenciation des Th0 en Th1 et Th2.

Réseau de Petri a été proposée [87]. Une version révisée, comportant 23 composants, a ensuite été présentée par Mendoza et Xenarios [72]. Cette nouvelle version, ajoute quelques composants : NFAT (Nuclear Factor of Activated T-cells, activé par le TCR) ainsi que la voie de signalisation d'IL10. Ce modèle Booléen conserve les états stables de la version précédente, à l'exception de Th1*.

4.3 Signalisation TCR

La différenciation Th1/Th2 se produit lors de l'activation des Th0, déclenchée par leur première rencontre avec leur antigène. La fixation de celui-ci sur le récepteur spécifique des cellules T (TCR) active plusieurs voies de signalisation, menant en particulier à l'activation d'un certain nombre de facteurs de transcription. Parmi ces facteurs, NFAT, NFkB et AP1 (notamment) sont déterminants pour la survie et l'activité des cellules T.

Les voies de signalisation activées par le TCR sont largement étudiées. Saez-Rodriguez et al. [95] ont récemment proposé un modèle logique de ces voies de signalisation intégrant 94 composants (ce modèle fait suite à un premier modèle faisant intervenir 48 composants Klamt et al. [59]). Il est centré sur les lymphocytes T CD8+, mais l'essentiel du système est commun avec les CD4+ (les lymphocytes T auxiliaires qui nous intéressent ici). La principale différence est le co-récepteur du TCR : CD4 est spécifique du CMH de classe II alors que CD8 est spécifique du CMH de classe I (voir section 1.2). Ce modèle détaille en particulier

les événements cachés derrière l'activation de NFAT par le TCR dans la seconde version du modèle de différenciation Th1/Th2 de Mendoza *et al*.

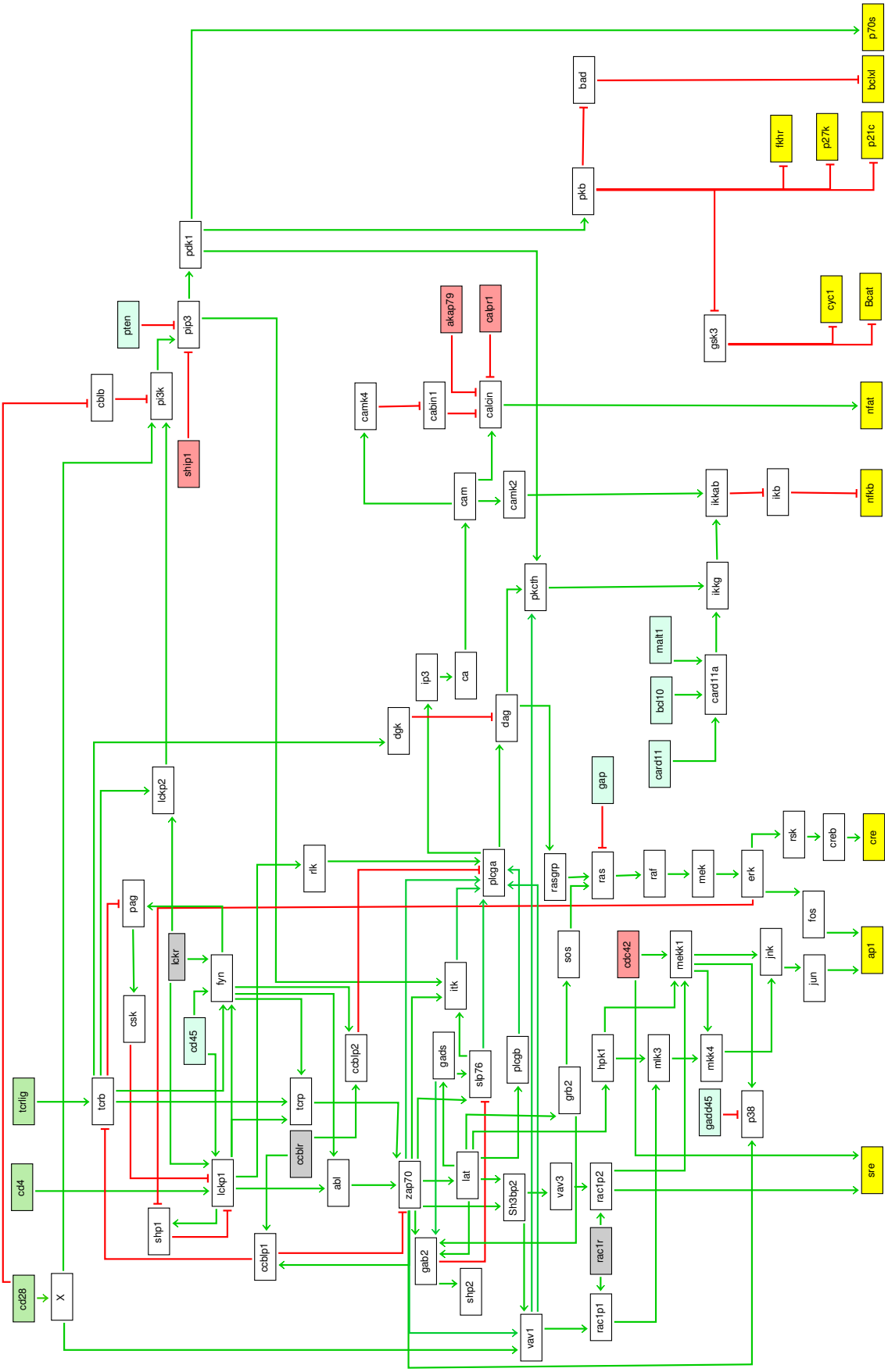


Figure 4.2: Modèle de la signalisation TCR Saez-Rodriguez et al. [95]. Ce modèle à 94 variables rend compte de plusieurs voies de signalisations activées suite à l'activation du récepteur TCR et de ses co-récepteurs.

Chapitre 5

Objectifs

Les lymphocytes T auxiliaires (Th) jouent un rôle prépondérant dans la régulation de la réponse immunitaire spécifique. En effet, les deux sous-types Th1 et Th2 activent différentes composantes de la réponse immunitaire. La différenciation Th1/Th2 est donc capitale pour le bon équilibre de la réponse et a fait l'objet de nombreuses études. La découverte récente de deux nouveaux sous-types de cellules T auxiliaires, les cellules régulatrices (Treg) et les Th17, remet en question cette dichotomie bien acceptée jusqu'à présent.

Nous souhaitons compléter les modèles existants afin d'y intégrer les résultats expérimentaux récents concernant ces deux lignées supplémentaires. Le contrôle de ce mécanisme de différenciation fait intervenir de nombreux acteurs et les données disponibles sont essentiellement d'ordre qualitatif. Une approche qualitative semble donc appropriée et permet de réutiliser les modèles présentés dans les sections 4.2 et 4.3. Comme les composants et interactions déjà présents dans ces modèles, ceux que nous souhaitons ajouter sont issus de la littérature.

La taille du système étudié rend son analyse ardue, même à l'aide de méthodes qualitatives. Il est donc important de développer les méthodes de modélisation afin de permettre l'analyse de modèles comprenant un grand nombre de composants. Pour cela, nous explorons une voie analytique d'une part et des méthodes de réduction d'autre part. La voie analytique permet de tirer certaines informations dynamiques sans recourir aux simulations. Les méthodes de réduction permettent de réduire la taille de l'espace des états à explorer.

Deuxième partie

Résultats

Chapitre 6

Analyse dynamique de graphes de régulation

6.1 Introduction

Lors de la construction et de l'analyse de modèles représentant des systèmes biologiques, on se heurte souvent à une explosion combinatoire liée au grand nombre de composants impliqués dans ces réseaux. La construction du graphe de transitions d'états (GTE) complet devient rapidement impossible car elle implique de stocker l'intégralité de ce graphe. En effet, dans le cas Booléen, la taille du graphe de transitions d'états double pour chaque composant ajouté au modèle (elle est donc multipliée par 1024 lorsque l'on ajoute 10 composants).

Dans l'état actuel, notre logiciel de modélisation (GINsim, voir chapitre 8) ne permet de calculer le GTE complet que pour des modèles comprenant une vingtaine de composants. Cette limite peut probablement être repoussée autour de 30 à 40 composants (40 composants donnent 10^{12} états : une simple liste occupant un octet pour chaque état occupe 1To d'espace mémoire). Un graphe de cette taille est de toute façon complexe à étudier. En outre, nous souhaitons analyser des graphes de régulation comprenant bien plus de 40 composants (voir chapitres 4 et 9). Nous cherchons donc des méthodes d'analyse du graphe de régulation, afin de tirer des informations sur la dynamique sans recourir aux simulations.

Comme évoqué dans l'introduction, nous utilisons des diagrammes de décision multi-valués (MDD) pour représenter les fonctions logiques associées aux composants du graphe de régulation. Pour chaque composant i du graphe de régulation, la fonction logique $\mathcal{K}_i$, donnant le niveau d'activité cible de i en fonction de l'état de ses régulateurs, est représentée par un MDD. Les variables de décision associées aux noeuds internes de ce diagramme sont les niveaux des régulateurs de i, tandis que les feuilles du MDD donnent le niveau cible de i.

Dans un premier temps, l'utilisation de MDD a permis d'améliorer les performances des simulations (de la construction des GTE). En effet, pour déterminer $\mathcal{K}_i(x)$ (la valeur cible du composant i à l'état x), il suffit de suivre dans ce diagramme le chemin correspondant à l'état x. Par construction, ce chemin contient au maximum autant de noeuds internes que i a de régulateurs.

L'utilisation de MDD a ensuite permis le développement de méthodes d'analyse du graphe de régulation reposant sur des manipulations des fonctions logiques. Les deux algorithmes présentés dans les sections suivantes permettent, respectivement, de déterminer tous les états stables et d'analyser les circuits de régulation. Ces travaux ont été présentés lors de deux conférences, l'une francophone (Journées Ouvertes en Biologie, Informatique et

Mathématiques[1]) et l'autre internationale (Computational Methods in Systems Biology[2]). La publication associée à CMSB [81] est insérée dans la section 6.4.

6.2 Détermination des états stables

Nous nous intéressons tout particulièrement aux attracteurs, qui correspondent aux composantes fortement connexes terminales du graphe de transitions d'états. Parmi ces attracteurs, les plus simples sont les états stables. Un état stable est un état tel que la valeur cible de chaque composant est identique à sa valeur courante. En utilisant cette définition et en manipulant les diagrammes de décision représentant les fonctions logiques, nous avons proposé un algorithme permettant de déterminer tous les états stables d'un modèle logique sans construire le graphe de transitions d'états ni énumérer tous les états possibles.

Cette méthode repose sur l'obtention, à partir des fonctions logiques donnant la valeur cible des composants du modèle, de nouvelles fonctions logiques donnant les conditions de stabilité de ces composants. Un composant i est stable lorsque son niveau cible $\mathcal{K}_i(x)$ est identique à son niveau courant x_i. Le MDD représentant les conditions de stabilité de i est obtenu à partir du MDD représentant sa fonction logique auquel on intègre la variable de décision représentant le niveau de i. Ce nouveau MDD a une feuille 1, correspondant aux situations dans lesquelles niveau cible et niveau courant sont identiques (stabilité), et une feuille 0 dans les autres situations (instabilité). Une fois les conditions de stabilité spécifiques à chaque composant obtenues, on définit une condition globale par un nouveau MDD obtenu en combinant, par une conjonction (ET logique), les MDD donnant les conditions de stabilité des composants. Dans ce MDD, la feuille 1 correspond aux cas dans lesquels, pour chaque composant, niveau cible et niveau courant sont identiques. Les états stables sont donc donnés par tous les chemins menant à la feuille 1 dans ce MDD.

6.3 Analyse des circuits

Les conjectures de R. Thomas associent circuit positif et existence de plusieurs attracteurs d'une part, et circuit négatif et oscillations entretenues d'autre part. Nous avons également vu qu'un circuit isolé (correctement paramétrisé) génère toujours le comportement dynamique attendu. Bien que ces conjectures fournissent une piste pour la déduction de propriétés dynamiques à partir de la structure du graphe de régulation, elles ne donnent que peu d'information pour les réseaux de régulation comprenant plusieurs circuits entremêlés (voir section 2.2.3 et figure 6.1). Nous proposons ici une approche complémentaire visant à déterminer les circuits fonctionnels d'un modèle donné, c'est-à-dire ceux susceptibles de générer les comportements dynamiques associés aux circuits de régulation.

En effet, lorsqu'un circuit est soumis à l'influence de régulateurs extérieurs, ceux-ci peuvent bloquer le niveau d'activité de ses membres et donc le fonctionnement du circuit. Dans le cas Booléen, un circuit de régulation se comporte comme un circuit isolé lorsqu'un changement du niveau de n'importe lequel de ses membres a un impact sur le membre suivant. Dans le cas multi-valué, il faut prendre en compte le seuil de chaque interaction composant le circuit. Nous avons proposé une méthode permettant de déterminer le contexte de

[1] crfb.univ-mrs.fr/jobim2007
[2] conferences.inf.ed.ac.uk/cmsb07

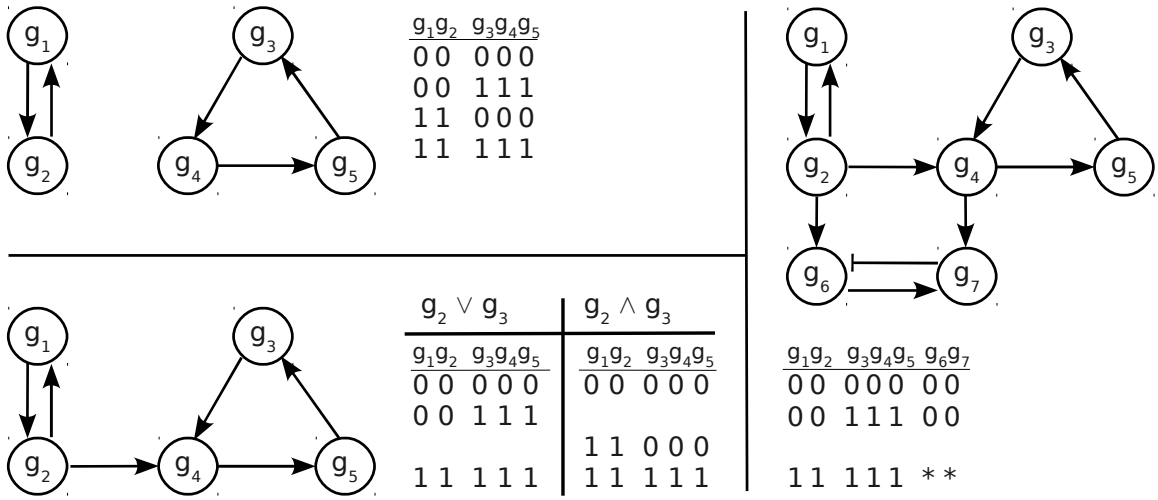


Figure 6.1: Quelques exemples de relations entre circuits et attracteurs pour des modèles simples. Chaque sous-figure contient un graphe de régulation et une table listant ses attracteurs :
En haut à gauche : deux circuits positifs isolés donnent quatre états stables.
En bas à gauche : si l'on couple ces deux circuits, l'un des deux n'est plus isolé, conduisant à la perte d'un des quatre états stables. Dans ce modèle, g_4 a deux fonctions logiques compatibles avec la structure du graphe : $g_2 \vee g_3$ et $g_2 \wedge g_3$. Le premier circuit (isolé) génère deux états stables : l'un en présence de g_1 et g_2, l'autre en leur absence. Si la fonction logique associée à g_4 est$g_2 \vee g_3$, le second circuit est fonctionnel en absence de g_2, on a donc trois états stables : un avec g_1 et g_2 et deux en absence de g_2. Pour la fonction $g_2 \wedge g_3$, le second circuit est fonctionnel en présence de g_2, on a donc deux états stables avec g_1 et g_2 et un seul en leur absence.
À droite : ajout d'un circuit négatif fonctionnel en présence de g_2 et g_4. L'état stable correspondant devient alors un attracteur cyclique. Les deux autres états stables sont conservés (une seule des paramétrisations possbles est considérée ici).
Ces exemples restent simples à analyser, en partie car les circuits peuvent être hiérarchisés. Dans l'exemple de droite, le premier circuit est isolé, le contexte de fonctionnalité du second dépend du premier et enfin le troisième circuit est fonctionnel en fonction des deux précédents.

fonctionnalité d'un circuit de régulation, c'est-à-dire la région de l'espace des états vérifiant cette propriété. Pour cela, on commence par déterminer, pour chaque interaction du circuit, la région dans laquelle cette interaction est fonctionnelle. L'intersection des contextes de fonctionnalité de toutes les interactions du circuit est le contexte de fonctionnalité de celui-ci. Un circuit est dit fonctionnel si ce contexte n'est pas vide. Voir [81] (reproduit dans la section 6.4) pour une définition formelle.

6.4 Article présenté à CMSB 2007

Decision Diagrams for the Representation and Analysis of Logical Models of Genetic Networks

Aurélien Naldi, Denis Thieffry, and Claudine Chaouiya

TAGC, INSERM ERM206, Université de la Méditerranée
Marseilles, France

Abstract. The complexity of biological regulatory networks calls for the development of proper mathematical methods to model their structures and to obtain insight in their dynamical behaviours. One qualitative approach consists in modelling regulatory networks in terms of logical equations (using either Boolean or multi-valued discretisation).

In this paper, we propose a novel implementation of the generalised logical formalism by means of Multi-valued Decision Diagrams. We show that the use of this representation enables the development of efficient algorithms for the analysis of specific dynamical properties of the regulatory graphs. In particular, we address the question of determining conditions insuring the functionality of feedback circuits, as well as the identification of stable states. Finally, we apply these algorithms to logical models of T cell activation and differentiation.

Keywords: Regulatory networks, logical modelling, decision diagrams, regulatory circuits, stable states.

1 Introduction

Modelling is a crucial step towards a functional understanding of the complex interaction networks that govern fundamental cellular processes. In this respect, in order to overcome the lack of quantitative data, logical approaches have been successfully applied to a wide variety of genetic networks involved in cell differentiation and pattern formation (for an introduction to logical modelling of genetic networks, see [12,3]). However, when facing very large regulatory networks, even logical abstraction leads to hard combinatorial problems. In other contexts, decision diagrams have been successfully applied to similar combinatorial problems, in particular for symbolic model-checking (*e.g.* [2]). Here, we show how decision diagrams can be used to represent sophisticated logical rules and enable the development of efficient algorithms to determine the conditions insuring specific dynamical roles for the different regulatory circuits, as well as to identify all the stable states of large and complex systems, without explicitly constructing the state transition graph and independently of any initial conditions.

Finally, we apply these algorithms to: (i) a model of Th1/2 differentiation [8] and (ii) a model of the TCR signalling pathway [7].

M. Calder and S. Gilmore (Eds.): CMSB 2007, LNBI 4695, pp. 233–247, 2007.

2 Logical Modelling of Gene Regulatory Networks

Our modelling approach leans on the generalised logical formalism initially developed by R. Thomas and collaborators [13,3]. In this context, a regulatory network and its dynamics are both represented in terms of oriented graphs.

2.1 Regulatory Graphs

A regulatory network is defined as a labeled directed graph $\mathcal{R} = (\mathcal{G}, \mathcal{A}, \mathcal{K})$ called *regulatory graph*, where:

- $\mathcal{G} = \{g_1, \ldots, g_n\}$ is the set of nodes of the regulatory graph, representing genes (or, more generally, regulatory components). Each $g_i \in \mathcal{G}$ is associated with its maximum *expression level* Max_i ($Max_i \in \mathbb{N}^*$) and its current expression level x_i ($x_i \in [0, Max_i]$).
- $\mathcal{A}$ is the set of arcs. An arc (g_i, g_j) specifies that the gene g_i regulates the gene g_j (when there is no possible confusion, we often write i for g_i). A regulatory graph may contain self-loops (*e.g.* a self-regulated gene g_i with an arc (i, i)).

  For each gene g_j, $Reg(j)$ denotes the set of its regulators: $i \in Reg(j)$ if and only if $(i, j) \in \mathcal{A}$.

  If $Max_i > 1$, g_i may have different effects onto a gene g_j, depending on the actual activity level of g_i. Thus the arc (i, j) may be indeed a multi-arc encompassing different *interactions*. The multiplicity of the arc (i, j) (*i.e.* the number of its constitutive interactions), is denoted $m_{i,j}$ ($1 \leq m_{i,j} \leq Max_i$). A *threshold* $\theta_{i,j,k}$ (integer taking its values in $[1, m_{i,j}]$) is associated to the k^{th} interaction (denoted $(i, j, k), k \in [1, m_{i,j}]$), with $1 \leq \theta_{i,j,1} < \ldots < \theta_{i,j,m_{i,j}} \leq Max_i$. The k^{th} interaction is *active*, when the level of its source g_i lays between the threshold of this interaction and that of the next interaction: $\theta_{i,j,k} \leq x_i < \theta_{i,j,k+1}$ (by convention, $\theta_{i,j,m_{i,j}+1} = Max_i + 1$).
- $\mathcal{K} = (\mathcal{K}_1, \ldots, \mathcal{K}_n)$ defines the dynamics of the system: each $\mathcal{K}_i$ is a multi-valued logical function defining the evolution of the variable x_i, depending on the incoming active interactions of g_i:

$$\mathcal{K}_i : \left(\prod_{j \in Reg(i)} [0, m_{j,i}] \right) \rightarrow [0, Max_i] \,.$$

  For example, if g_2 and g_3 are regulators of g_1 ($Reg(1) = \{2, 3\}$), $\mathcal{K}_1(0, 1)$ is the focal value of g_1 when no interaction from g_2 is active (*i.e.* $x_2 < \theta_{2,1,1}$) and the first interaction from g_3 is active ($\theta_{3,1,1} \leq x_3 < \theta_{3,1,2}$).

Note that the biologists often consider *different types* of interactions: activations (*resp.* repressions) have a positive (*resp.* negative) effect on their targets. However the actual effect of an interaction often depends on the presence of co-factors; its sign may even change depending on the context. In any case, the signs of interactions can be derived from the logical functions.

2.2 Dynamics of a Regulatory Graph

A state x of the regulatory graph is a n-tuple $(x_1, \ldots, x_n)$ of the expression levels of the genes: $x \in \prod_{i=0}^{n}[0, Max_i]$. Given a state and for each gene g_i, it is then possible to determine the set of interactions operating onto g_i (the active interactions). We thus define the functions $\mathcal{K}'_i(x)$s, which follow from the logical functions $\mathcal{K}_i$s and directly depend on the current state x of the system:

$$\begin{aligned} \mathcal{K}'_i : \prod_{j=1}^{n}[0, Max_j] &\longrightarrow [0, Max_i] \\ x &\longrightarrow \mathcal{K}_i\left(\sum_{k=1}^{m_{j,i}} k.\mathbf{1}_{[\theta_{j,i,k},\theta_{j,i,k+1}[}(x_j)\right)_{j\in Reg(i)} . \end{aligned}$$

where $\mathbf{1}$ denotes the indicator function.

In the following, for simplicity, $\mathcal{K}'_i$ will be denoted $\mathcal{K}_i$ (omitting the prime sign).

Given the current state x, the level of each g_i tends toward the *focal value* given by $\mathcal{K}_i(x)$. If this is greater (*resp.* lower) than x_i (the current value of g_i), there is a call to increase (*resp.* decrease) by one the value of g_i.

The dynamics of the regulatory graph can then be represented by a *state transition graph*, where nodes represent states (giving the levels of the regulatory components) and arcs represent transitions between states (*i.e.* changes of the values of some components). This state transition graph is computed by means of the $\mathcal{K}_i$s, which indicate the transitions leading from the current state to its following states (here, we consider an asynchronous updating, where each transition corresponds to a change of a unique variable, see [3] for further details). When facing large regulatory networks (*i.e.* dozens or hundreds of components), combinatorial problems impede the full computation of state transition graphs. In this paper, we assess the use of Decision Diagrams to handle this combinatorial problem for complex multi-valued logical models.

3 Regulatory Graphs and Multi-valued Decision Diagrams

Multi-valued logical functions have a finite number of possible values, depending on a set of *decision variables*, which also take a finite range of values. Such functions can be represented using efficient data structures (see [1] for further details).

Decision Diagrams are particularly promising in our context. Indeed, Garg et al. have already used BDD to represent the whole state transition graph of Boolean models of biological regulatory networks (their approach is contrasted with ours in the conclusion).

In the following section, we recall the definitions at the basis of our novel implementation of logical functions.

3.1 Decision Diagrams

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be represented as a *binary decision tree* where non-terminal nodes are labelled by a decision variable and where terminal nodes are labelled either 1 (true) or 0 (false). The edge from a decision node to its left child (*resp.* right child) corresponds to an assignment of the variable to 0 (*resp.* to 1). Given a state $x \in \{0,1\}^n$ (defining the values of n decision variables), a unique path from the root to a terminal node (a leaf) is defined. Along this path, the child chosen for each non-terminal node is labelled with the value of the corresponding variable in state x. The terminal node reached through this path gives the value of $f(x)$.

Reduced Binary Decision Diagrams (RBDDs) have been introduced to improve this representation, which requires exponential space ($2^{n+1} - 1$ nodes). RBDDs are obtained applying two reduction rules: (i) merge isomorphic subgraphs and (ii) bypass nodes whose children are the roots of isomorphic subdiagrams. The resulting structure is then a rooted directed acyclic graph. Bryant further extended this representation by the use of a fixed variable ordering which leads to canonical representations of logical functions. The resulting graphs are called Reduced *Ordered* BDDs (ROBDDs), commonly referred to as BDDs. Note that the size of BDDs may depend on the variable ordering (see Figure 1 for an illustration of the impact of the ordering).

BDDs have been generalised to the multi-valued case: a discrete multi-valued function can be represented by a *Multi-valued Decision Diagram* (MDD), where decision nodes may have as many children as the number of their possible values and the terminal nodes are labelled by the values of the function (see Figure 1 for an illustration) [6]. The ordering and reduction rules defined for BDDs apply also to this multi-valued generalisation.

3.2 Use of MDDs to Represent Logical Functions

The functions $\mathcal{K}_i$s, which take their values in $[0, Max_i]$, can be represented as MDDs, with the regulators of g_i as decision variables. During the simulation (*i.e.* the construction of the state transition graph), the focal value of a gene g_i is obtained in $O(\#Reg(i))$ in the worst case, traversing the corresponding MDD. This data structure thus definitely improved the performance of GINsim, our software implementing the logical formalism [5].

Moreover the representation of the $\mathcal{K}_i$s by means of MDDs greatly facilitates the analysis of specific dynamical properties as showed in the following sections.

In the sequel, for simplicity, diagrams will be named after the multi-valued functions they represent. For a given regulatory graph, an arbitrary ordering on the set of variables is used consistently for all the related MDDs.

4 Analysis of Regulatory Circuits

In what follows, we consider *elementary circuits* (circuits, for short), *i.e.* finite closed paths in the regulatory graph, where all the nodes are distinct.

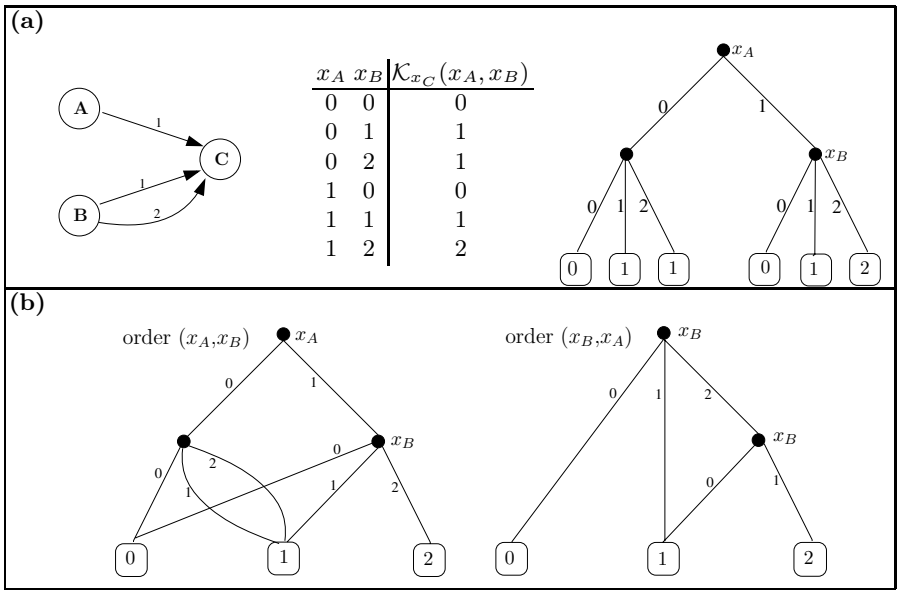
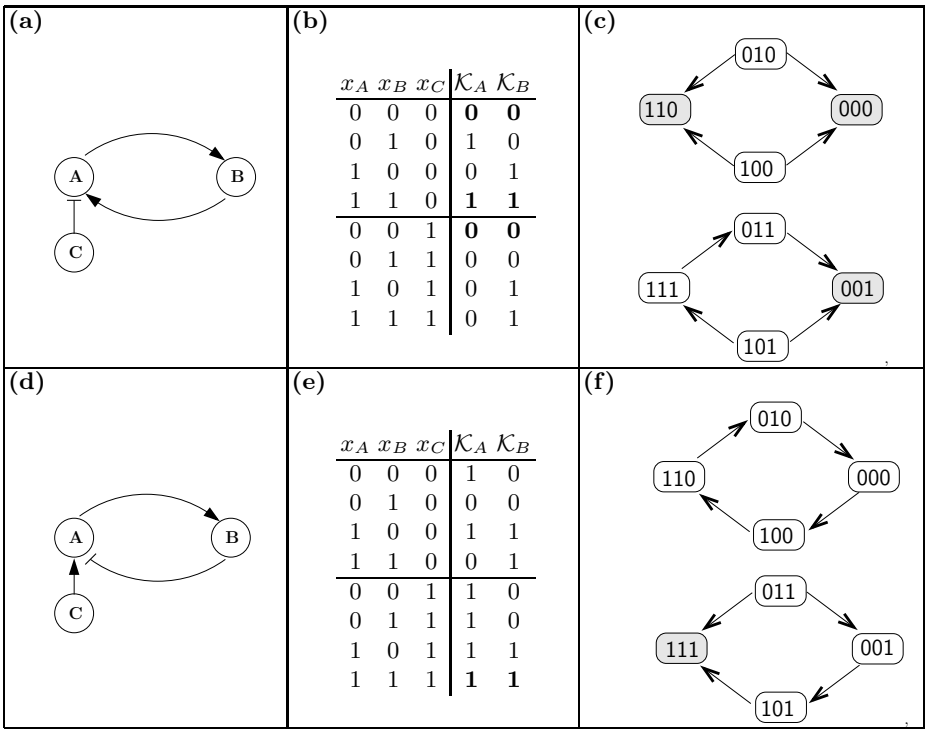


Fig. 1. Example of a simple logical regulatory graph with its MDD representation: **(a)** The regulatory graph, the table defining the function $\mathcal{K}_C$ together with its decision tree representation. **(b)** The reduced MDDs (considering the two different ordering of x_A and x_B) representing $\mathcal{K}_C$, with x_A and x_B as decision variables and the values of $\mathcal{K}_C$ labelling the leaves.

It is well established that complex dynamical behaviours of regulatory networks are related to their topological structures. In particular, the roles of regulatory circuits have been emphasised by R. Thomas, who proposed that negative circuits are required to observe oscillatory behaviours, whereas positive circuits are necessary for multistationarity (the sign of a circuit is given by the product of the signs of its interactions) [11]. Proofs of these conditions have been presented within different modelling formalisms [10,9]. But the sole presence of a circuit in a network does not guarantee the appearance of the corresponding dynamical behaviour. The circuit must be *functional* [11]. Figure 2 illustrates how the regulators of one of its components can prevent a circuit from generating the expected behaviour. We thus define the *context of functionality* of a circuit as a set of constraints on the values of the external inputs acting on that circuit. Our definition of functionality context may serve as a basis to formally prove the relationship between the functionality of a circuit and the corresponding dynamical properties.

In the multi-valued case, circuits containing multiple interactions can be splitted into multiple "elementary circuits" and considered separately. In the sequel we restrict ourselves to the case of single interactions to simplify the notation: the threshold of an interaction can be thus denoted $\theta_{i,j}$ (instead of $\theta_{i,j,k}$).



x_A	x_B	x_C	$\mathcal{K}_A$	$\mathcal{K}_B$
0	0	0	**0**	**0**
0	1	0	1	0
1	0	0	0	1
1	1	0	**1**	**1**
0	0	1	**0**	**0**
0	1	1	0	0
1	0	1	0	1
1	1	1	0	1

x_A	x_B	x_C	$\mathcal{K}_A$	$\mathcal{K}_B$
0	0	0	1	0
0	1	0	0	0
1	0	0	1	1
1	1	0	0	1
0	0	1	1	0
0	1	1	1	0
1	0	1	1	1
1	1	1	**1**	**1**

Fig. 2. Illustration of the influence of external inputs on the dynamics of a regulatory circuit. In the regulatory graphs of panels **(a)** and **(d)**, activations are depicted by normal arrows whereas inhibitions are depicted by blunt arrows. **(a)** A simple positive circuit affected by a negative input. **(b)** Values for $\mathcal{K}_A$ and $\mathcal{K}_B$: two situations arise, depending on the value of x_C (component A regulated by both B and C). **(c)** The two possible behaviours, depending on x_C: for $x_C = 0$ there are two stable states (top), while a unique stable state exists for $x_C = 1$ (bottom). **(d,e,f)** A simple negative circuit affected by a positive input; oscillations only appear in the absence of the input.

4.1 Functionality Context and Sign of an Interaction

In general, we say that an interaction (i, j) is functional when it affects the focal value of its target: $\mathcal{K}_j(x_1, \ldots, \theta_{i,j} - 1, \ldots, x_n) \neq \mathcal{K}_j(x_1, \ldots, \theta_{i,j}, \ldots, x_n)$. In the context of a circuit, this change must further affect the activity of the next interaction in the circuit (the threshold of the next interaction must be crossed to reach the focal value). This additional constraint is only relevant for multi-valued genes, as it is always satisfied in the Boolean case.

In the following, we define the functionality of the interaction (i, j) and its context, that is the set of constraints upon the regulators of j (target of the considered interaction).

Definition 1. *Let consider an interaction (i, j), member of a circuit $\mathcal{C}$ with a threshold $\theta_{i,j}$. Let (j, k) be the next interaction in $\mathcal{C}$, with a threshold $\theta_{j,k}$. The*

interaction (i,j) is said to be functional in $\mathcal{C}$ if and only if there exists a variable assignment for all regulators of g_j except g_i such that:

$$\mathcal{K}_j(x_1,\ldots,x_{i-1},\theta_{i,j}-1,\ldots,x_n) < \theta_{j,k} \leq \mathcal{K}_j(x_1,\ldots,x_{i-1},\theta_{i,j},\ldots,x_n), \quad (1)$$

$$\text{or} \quad \mathcal{K}_j(x_1,\ldots,x_{i-1},\theta_{i,j},\ldots,x_n) < \theta_{j,k} \leq \mathcal{K}_j(x_1,\ldots,x_{i-1},\theta_{i,j}-1,\ldots,x_n). \quad (2)$$

The functionality context of the interaction (i,j) in $\mathcal{C}$ is defined as the subset of $\Pi_{k=1}^{n}[0,Max_k]$ of n-tuples such that the values of the regulators of g_j let the interaction (i,j) functional (i.e. Equation (1) or (2) satisfied). The interaction is thus functional if its context is not empty.

Definition 1 establishes that the interaction (i,j) is functional provided its activity affects the activity of the following interaction of the circuit (going out g_j). This depends on the values of $\mathcal{K}_j$, considering values $\theta_{i,j}-1$ and $\theta_{i,j}$ for g_i and all possible values of other regulators of g_j.

We can then define the sign of an interaction, as 0 when it is not functional, or 1 (*resp.* -1) when it is functional and leads to an increase (*resp.* a decrease) of the focal value of its target.

Definition 2. *Let us consider consecutive interactions (i,j) and (j,k) in a circuit $\mathcal{C}$. Given a variable assignment for all regulators of g_j (except g_i), the sign of (i,j) in $\mathcal{C}$ is given by $\Gamma_{i,j}$, defined as follows:*

$$\Gamma_{i,j}(x) = \begin{cases} 1 & \text{if Equation (1) holds,} \\ -1 & \text{if Equation (2) holds,} \\ 0 & \text{otherwise,} \end{cases}$$

where x is a state ($x \in \Pi_{k=1}^{n}[0,Max_k]$), for which the values corresponding to the regulators of g_j (except that of g_i) equal the given assignment.

We say that (i,j) has a positive effect when $\Gamma_{i,j}(x)=1$, a negative effect when $\Gamma_{i,j}(x)=-1$ and no effect otherwise.

The construction of the MDD representing $\Gamma_{i,j}$ for a given interaction (i,j) is illustrated in Figure 3**(a-b)**. The algorithm is given as supplementary material. The leaves of the MDD give the sign of (i,j), depending on the path followed to reach them. This path defines the conditions on the values taken by the regulators of g_j.

Remark 1. It may happen that the sign of an interaction (i,j) is *context dependent*, that is, for an assignment of the regulators of g_j (except g_i) the sign of the interaction is positive and for another assignment, it is negative. To simplify the explanations in the next section (that defines the functionality of a whole circuit), we exclude such a case, which is infrequent in genetic regulatory networks.

4.2 Functionality Context and Sign of a Regulatory Circuit

We now consider the case of a whole elementary circuit. Definition 3 formalises the functionality context of a circuit, as well as its sign, depending on its constitutive interactions.

First, we will consider the case of circuits which do not contain smaller circuit(s), or shortcuts. A circuit $\mathcal{C} = (c_1, c_2, \ldots, c_r)$ (with $r + 1 = 1$) contains a shortcut if there exists c_i which regulates c_k, with $k \neq i + 1$ ($c_i \in \mathcal{C}$ and $c_k \in \mathcal{C}$). The simplest example of such a shortcut is an auto-regulation of a member of the circuit. Then, we extend the definition of the functionality to the general case (with, for simplicity, the restriction that no interaction of the circuit has a context dependent sign).

Definition 3. *The functionality context of a circuit with no shortcut is defined as the intersection of the functionality contexts of its constitutive interactions; the circuit is thus functional when this intersection is not empty (implying that all its interactions are functional). The sign of the circuit $\mathcal{C} = (c_1, c_2, \ldots, c_r)$ is defined as the product of the signs of its interactions:*

$$\Gamma_{\mathcal{C}}(x) = \prod_{i=1}^{i \leq r} \Gamma_{c_i, c_{i+1}}(x).$$

Definition 4. *Let consider a circuit $\mathcal{C} = (c_1, c_2, \ldots, c_r)$ which contains shortcut(s). Its functionality context $\Gamma_{\mathcal{C}} \subseteq \Pi_{k=1}^{n}[0, Max_k]$ is defined as the intersection of the contexts of its interactions further restricted to insure that, for all c_i acting on a component c_k of $\mathcal{C}$ different from c_{i+1}, and an assignment $\bar{y}$ of all variables but c_i:*

$$(y_1, \ldots, \theta_{i,i+1} - 1, \ldots y_n) \in \Gamma_{\mathcal{C}} \Longleftrightarrow (y_1, \ldots, \theta_{i,i+1}, \ldots y_n) \in \Gamma_{\mathcal{C}}.$$

When the above condition does not hold, all the tuples $(y_1, \ldots, x_i, \ldots y_n)$ are removed from $\Gamma_{\mathcal{C}}$ (for all possible values x_i of c_i). The circuit $\mathcal{C}$ is functional if its functionality context $\Gamma_{\mathcal{C}}$ is not empty. Its sign is defined as the product of the signs of its interactions.

The above definition guarantees that, given a fixed assignment of all other variables compatible with $\Gamma_{\mathcal{C}}$ (the functionality context of $\mathcal{C}$), both states where c_i level is less or equal to $\theta_{i,i+1}$ (the threshold of the interaction of $\mathcal{C}$ going from c_i to c_{i+1}) are in $\Gamma_{\mathcal{C}}$ (and this insures the functionality of this interaction).

The determination of the function Γ_C requires two operations: (i) the determination of all the $\Gamma_{i,j}$, giving the signs of single interactions; and (ii) the computation of their product.

Assume that the diagrams giving the signs of the interactions composing the circuit have been determined. The MDD giving the global sign of the circuit is built as a product of the signs of the individual interactions. This product is performed by means of the combination of MDDs Γ_1 and Γ_2, applying the following rules (see Figure 3**(c)** for an illustration):

- if $\Gamma_1$and Γ_2 are reduced to single nodes, the product is a node, its value being the product of those of $\Gamma_1$and Γ_2;
- else if Γ_1 (resp Γ_2) is a single node with value 1, the result is Γ_2 (*resp.* Γ_1);

- else if Γ_1 and Γ_2 roots are internal nodes with the same order (hence corresponding to the same decision variable), the result is a root of this order, and its children are recursive combinations of those of Γ_1 and Γ_2 roots;
- otherwise, if Γ_2 is a single node with value (-1) or if the root of Γ_1 has an order less than that of the root of Γ_2 (or symmetrically), the result is a root such that:
  - its order is the order of the root of Γ_1;
  - its children are recursive combinations of Γ_2 with those of Γ_1.

Finally, in the case of shortcuts in the circuit, as the sign of the circuit $\mathcal{C}$ may depend on the levels of members of $\mathcal{C}$, this dependancy is properly removed while ensuring that every member of the circuit can cross its threshold. Further details can be found in the supplementary material.

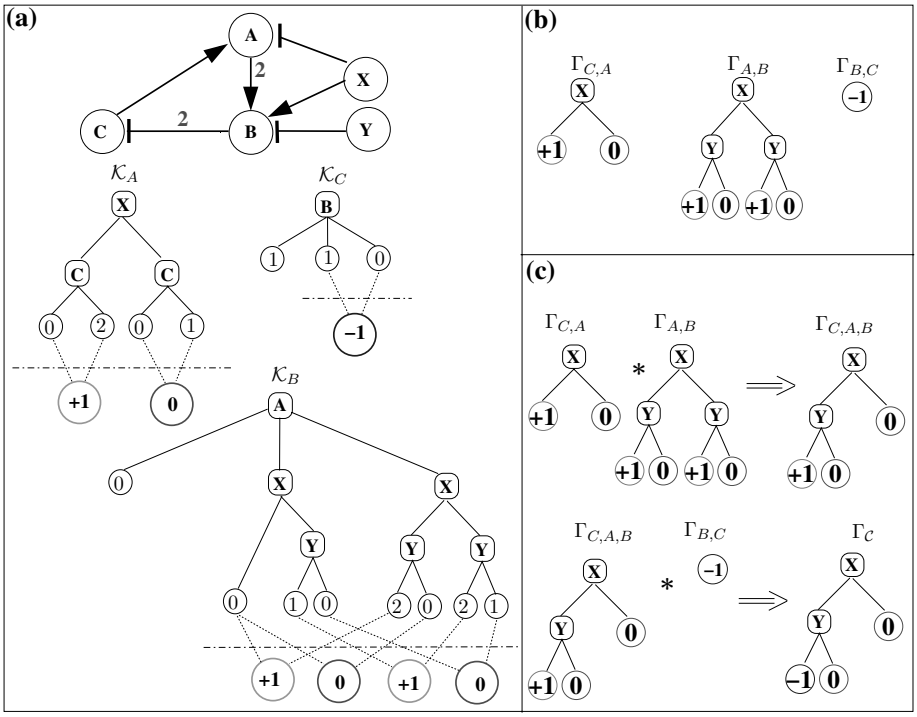


Fig. 3. Determination of the functionality context (and sign) of a circuit: **(a)** A regulatory network encompassing a circuit and decision diagrams of the logical functions $\mathcal{K}$s for the three members of the circuit. Below the logical functions $\mathcal{K}$s, pairwise comparisons of the leaves delineate the signs of the interactions targeting the members of the circuit. **(b)** MDDs giving the signs Γs of the interactions. **(c)** Combinations of the MDDs to determine $\Gamma_{\mathcal{C}}$, the sign of the circuit. The paths in $\Gamma_{\mathcal{C}}$ leading to non-zero leaves give the functionality context of the circuit. For sake of clarity, the diagrams are not fully reduced.

5 Efficient Determination of Logical Stable States

In this section, we show how the MDD representation of the logical functions $\mathcal{K}_i$s can be used to determine all the logical stable states of a parameterised regulatory graph. A stable state x is such that the focal value of each gene is identical to its current value:

$$x \in \Pi_{i=1}^{n}[0, Max_i] \text{ is stable } \textit{iff } \mathcal{K}_i(x) = x_i, \ \forall i \in \{1, \ldots n\}. \tag{3}$$

The algorithm to determine the stable states encompasses two main steps. First, for each gene g_i, a MDD $\mathcal{S}_i$ is constructed, which gives the logical stability condition depending on its value x_i and on those of its regulators (box (b) in Figure 4). Second, the resulting MDDs are combined as described in 4.2.

For a gene g_i, the first step amounts to transform the MDD representing the logical function $\mathcal{K}_i$. The decision variable x_i is properly added and the leaves values are set to 0 for a change (a decrease or an increase), or to 1 for no change. The resulting MDD implements a logical function with value 1 (true) when the node is stable, 0 (false) otherwise:

$$\mathcal{S}_i : \prod_{j \in Reg(i)} [0, Max_j] \rightarrow \{0, 1\},$$

with

$$\mathcal{S}_i = \begin{cases} 0 \text{ if } \mathcal{K}_i(x) \neq x_i, \\ 1 \text{ if } \mathcal{K}_i(x) = x_i. \end{cases}$$

To simplify the notation, $S_i(x)$ and $K_i(x)$ are considered beyond the sole regulators of i, to encompass all components of x. The diagrams $\mathcal{S}_i$ are then pairwise combined (the combination of $\mathcal{S}_i$ and $\mathcal{S}_j$ is the representation of the logical stability condition for both g_i and g_j). As for the determination of the sign of a circuit, each combination amounts to the product of the corresponding MDDs. Ultimately, the diagram $\mathcal{S}_{1\ldots n}$ defines the stability condition for the whole set of genes, hence the paths leading to 1-leaves give the stable states (see Figure 4).

6 Application to Regulatory Networks Controlling T Cell Activation and Differentiation

We have applied our novel analysis tools to two logical models recently published: (i) the first (multi-valued) model accounts for the differentiation of naive T-helper lymphocytes into two subtypes, called Th1 and Th2, controlling cellular and humoral immune responses, respectively [8]; (ii) the second, a Boolean model, integrates the information available on T cell receptor (TCR) signalling, taking into account several co-acting signals [7]. These two models are closely related, as NFAT, one of the outputs of the TCR signalling pathway, is one of the activators of the IFN-γ pathway.

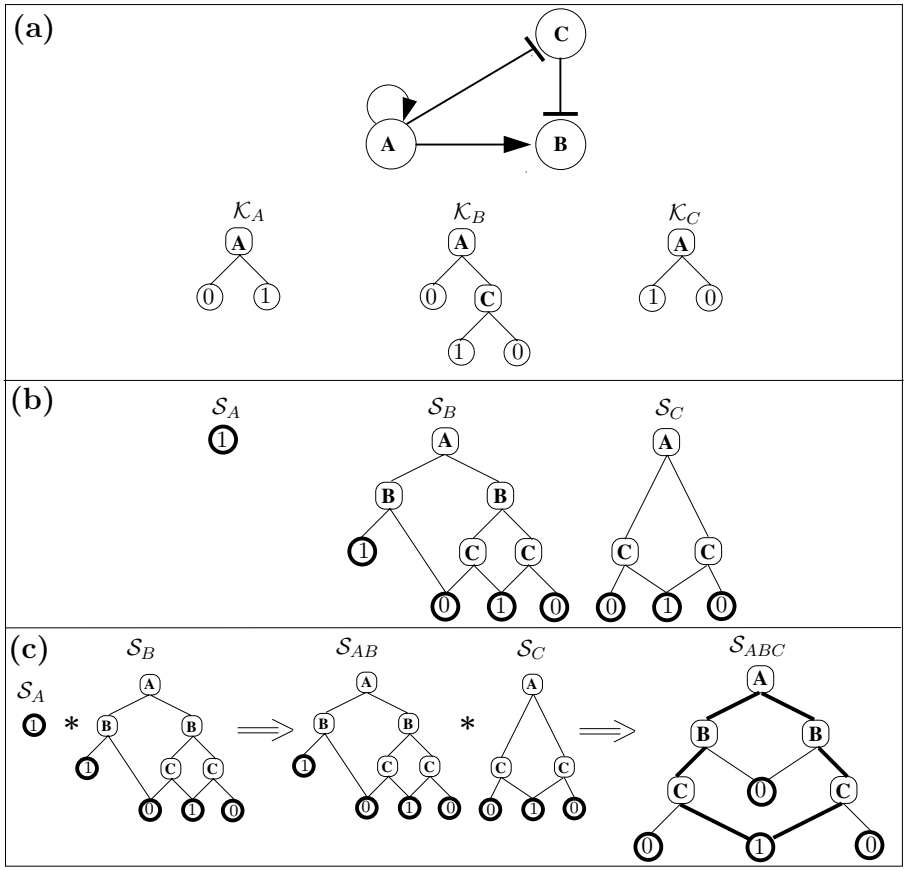


Fig. 4. Illustration of the stable state determination: **(a)** a simple regulatory graph and the associated logical functions; **(b)** the diagrams $\mathcal{S}$ giving the constraints for each gene to be stable; **(c)** the combination of the diagrams to obtain $\mathcal{S}_{ABC}$. For $\mathcal{S}_{ABC}$, two paths lead to the 1-leaf (they are depicted in bold), indicating that the regulatory graph has two stable states: $(x_A, x_B, x_C) = (0, 0, 1)$ and $(1, 1, 0)$. Note that, for clearness, the diagrams are not fully reduced.

6.1 T Cell Differentiation

The regulatory graph is shown in Figure 5. The graph encompasses 17 regulatory components, including five cytokines or intercellular signalling molecules, the interferons beta and gamma, and the interleukines 4, 12 and 18, the corresponding receptors, five mediatory molecules, SOCS1, IRAK, and STAT1, 4 and 6, as well as two transcription factors, Tbet and GATA3. All components but four are modelled by Boolean variables. The cascade involving IFN-γ, its receptor, STAT1 and Tbet are modelled by ternary variables as cells presenting two different levels of activation of the IFN-γ pathway have been

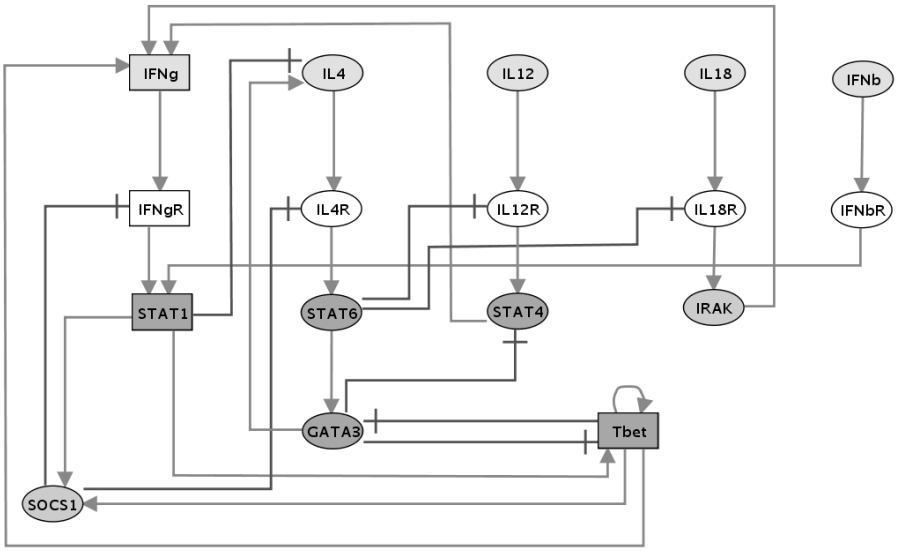


Fig. 5. The network controlling Th1/2 differentiation. All regulatory components but four are modelled by Boolean variables. The remaining four (rectangular nodes) are modelled by ternary variables. The first two nodes layers denote cytokines and their receptors. Normal arrows represent activations, whereas blunt arrows represent inhibitory effects. Note that the original model of Mendoza encompasses two variants, including an auto-activation for GATA3 (murine cells) or not (human cells).

experimentally observed. The experimental information in support of this graph, as well as the delineation of the logical parameters can be found in the original article published by Mendoza [8]. The logical model can be downloaded in a GINsim dedicated XML format from the address http://gin.univ-mrs.fr/GINsim/.

The algorithm presented in Section 5 takes less than a second to identify the four stable states reported by Mendoza:

- a *Th0* state without any active component, corresponding to the naive, undifferentiated cell;
- a *Th1* state where IFN-γ, IFN-γR, STAT1, SOCS1 and Tbet are expressed;
- a *Th1** state, similar to the previous one, but with higher expression levels for IFN-γ, and T-bet (the expression of SOCS1 prevents IFN-γR and STAT1 from showing a higher expression level);
- a *Th2* state showing expression of IL4, IL4R, STAT6 and GATA3.

Turning to the circuit functionality analysis, the algorithm described in Section 4 leads to the identification of five functional circuits among the 22 circuits found in the regulatory graph for the human model variant shown in Figure 5. All of these functional circuits are positive, which is consistent with the fact that this network is predominantly involved in the control of cell differentiation.

- The auto-regulation of T-bet is functional in the absence of both GATA3 and STAT1, or yet in the presence of a medium level of STAT1. Note that this circuit involves two different levels of Tbet and can thus enable the presence of up to three stable states, each characterized by one of the possible expression levels of Tbet.
- The (GATA3, IL4, IL4R, STAT6) circuit is functional in the absence of STAT1, SOCS1 and T-bet. This circuit ensures a coupling between the expressions of GATA3, IL4, IL4R and STAT6. In the murine cells, this circuit is not functional, replaced by the GATA3 auto-activation (functional in the absence of Tbet and STAT6).
- The (GATA3, T-bet) circuit is functional in the presence of STAT1 and STAT6. This cross-inhibitory circuit ensures the exclusive expressions of the transcriptional regulators Tbet and GATA3, characteristic of Th1 and Th2 responses, respectively. In the absence of the activators of Tbet and GATA3 (STAT1 and STAT6), the cell remains trapped in the naive state.
- The last two functional circuits involve IFN-γ, IFN-γR, STAT1, SOCS1, STAT6, and STAT4, plus a few additional components. Their contexts of functionality are relatively restrictive (absence of Tbet and IFNbR, and presence of IL12 and IL4, plus the absence of GATA3 in one of the two cases).

On the basis of the results of the circuit functionality analysis, it is possible to delineate specific perturbations affecting the stable state configuration.

6.2 T Cell Activation

The model recently published by Klamt et al. for T cell activation encompasses 40 regulatory components, hierarchically organised, from cell membrane receptors (TCR, CD8 and CD45) to transcription factors (CRE, AP1, NF-κB, NFAT) [7]. After adding three functional auto-activations on the input nodes (TCR, CD8 and CD45), our stable state identification tool identifies seven alternative stable states, each corresponding to a specific input configuration. Strikingly, the input configuration with all receptors permanently activated does not lead to any stable state, but rather to a complex periodic behaviour. However, in normal physiological situations, one should expect only transient activations of these receptors, thus ultimately leading to a unique stable state, corresponding to the resting situation.

In this respect, cross-talks between the signalling cascades may play an important role. Here, our circuit functionality analysis tool can be useful. Apart from the auto-activations purposively added on each of the three receptors, its application to this system leads to the identification of only one negative functional circuit out of nine: (ZAP70, cCBL). Under full stimulation (*i.e.* in the presence of all three inputs), this circuit enables an oscillatory behaviour of the involved regulatory components. These oscillations propagate downstream, leading to cyclic expression of the transcription factors (outputs of this model). In physiological situations, these oscillations must abort following the natural receptor inactivation.

7 Conclusion

In this paper, we have shown how Multi-valued Decision Diagrams (MDDs) can be used to encode the logical functions governing the behaviours of individual nodes in qualitative models of complex regulatory networks. Leaning on this encoding, we have further delineated two algorithms, one to efficiently determine all stable states of a logical model, the other to compute the functionality context of each regulatory circuit, in terms of conditions on the values of the variables acting on this circuit.

As mentionned above, Garg *et al.* have already represented Boolean state transition graphs in terms of BDD. They considered the particular case of networks where genes are expressed provided all their inhibitors are absent and at least one of their activators are present [4]. Based on their BDD representation, the authors propose an efficient method to determine stable states, and even more complex attractors. In contrast, our proposal refer to multi-valued logical networks where the logical functions can be more subtle. More importantly, our approach is based on a Decision Diagram representation of the transition functions for each node. Stable states, as well as feedback circuit functionality, are then determined through proper combinations of these diagrams.

On the basis of a prototype implementation of these algorithms, we are presently analysing a series of regulatory models (Boolean or multi-valued) involved in cell differentiation and pattern formation, encompassing dozens of regulatory components, involved in hundreds of regulatory circuits. For each of these systems, the delineation of all stable states and the computation of feedback circuit functionality domains took less than a second on a standard computer.

In section 6, we have briefly presented the results obtained for two recently published logical models: (i) a multi-valued model of the network controlling T-helper lymphocytes differentiation; (ii) a Boolean model encompassing some of the main signalling cascades controlling the activation of T cells.

At this point, we believe that it should be possible to further improve the performance of our algorithms. In particular, a proper ordering of decision variables can have a significant impact on the overall sizes of the MDDs (note that a common ordering of the variables must be defined to ensure a coherent combination of the MDDs). Similarly, we observed that the order of consecutive MDD combinations (*e.g.* in the course of the identification of all stable states) has a strong effect on the overall performance.

Finally, this MDD representation opens interesting prospects for the modelling of combinations of mutations or other perturbations through a rewriting of the MDDs describing the wild-type model.

Supplementary Material

Further details on the algorithms, as well as on the T-helper cell differentiation and activation models are available at the following URL:
http://gin.univ-mrs.fr/GINsim/publications/naldi2007.html.

Acknowledgements

We acknowledge financial support from the European Commission (contract LSHG-CT-2004-512143), the French Research Ministry through the ANR project JC05-53969 and A. Naldi PhD grant.

References

1. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Comput. 35, 677–691 (1986)
2. Burch, J.R., Clarke, E.M., Long, D.E., MacMillan, K.L., Dill, D.L.: Symbolic Model Checking for Sequential Circuit Verification. IEEE Trans. Comput.-Aided Design Integrated Circuits 13, 401–424 (1994)
3. Chaouiya, C., Remy, E., Mossé, B., Thieffry, D.: Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework. Lect. Notes Comp. Inf. Sci. 294, 119–126 (2003)
4. Garg, A., Xenarios, I., Mendoza, L., DeMicheli, G.: An Efficient Method for Dynamic Analysis of Gene Regulatory Networks and in-silico Gene Perturbation Experiments. Lect. Notes Comp. Sci. 4453, 62–76 (2007)
5. González, A., Naldi, A., Sanchez, L., Thieffry, D., Chaouiya, C.: GINsim: A software suite for the qualitative modelling, simulation and analysis of regulatory networks. Biosystems 84, 91–100 (2006)
6. Kam, T., Villa, T., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: Multi-valued decision diagrams: Theory and applications. Int. J. Multiple-Valued Logic 4, 9–12 (1998)
7. Klamt, S., Saez-Rodriguez, J., Lindquist, J.A., Simeoni, L., Gilles, E.D.: A methodology for the structural and functional analysis of signaling and regulatory networks. BMC Bioinformatics 7(56) (2006)
8. Mendoza, L.: A network model for the control of the differentiation process in Th cells. Biosystems 84, 101–114 (2006)
9. Remy, E., Ruet, P., Mendoza, L., Thieffry, D., Chaouiya, C.: From logical regulatory graphs to standard petri nets: Dynamical roles and functionality of feedback circuits. Lect. Notes Comp. Sci. 4230, 55–72 (2006)
10. Soulé, C: Graphic requirements for multistationarity. ComPlexUs 1, 123–133 (2003)
11. Thomas, R.: On the relation between the logical structure of systems and their ability to generate multiple steady states of sustained oscillations. Springer Series Synergetics 9, 180–193 (1988)
12. Thomas, R.: Regulatory networks seen as asynchronous automata: A logical description. J. Theor. Biol. 153, 1–23 (1991)
13. Thomas, R., Thieffry, D., Kaufman, M.: Dynamical behaviour of biological regulatory networks–I. biological role of feedback loops and practical use of the concept of the loop-characteristic state. Bull. Math. Biol. 57, 247–276 (1995)

6.4.1 Annexe : algorithme pour l'analyse des circuits

We present the construction of the MDD $\Gamma_{i,j}$, which gives the sign of the interaction (i,j) in a circuit $\mathcal{C}$ (assuming a following interaction (j,k)).

The pseudo-code is displayed using the following notations :

- $\mathcal{K}$, $\mathcal{K}_1$, $\mathcal{K}_2$, Γ are nodes of decision diagrams (*i.e.* roots of (sub-)diagrams) ;
- $\mathcal{K}$.order denotes the rank of the corresponding decision variable in the variable ordering ;
- $\mathcal{K}$.child[i] denotes the i^{th} child of $\mathcal{K}$ ($\mathcal{K}$.child[0] being the leftmost child) ;
- $\mathcal{K}$.nbChild denotes the number of children of $\mathcal{K}$ (number of values of the decision variable) ;
- when $\mathcal{K}$ is a leaf, $\mathcal{K}$.value denotes its value ;
- (0), (-1), (1), (2), etc. denote leaves with the corresponding value ;
- $\theta_1$is the threshold of the current interaction ($\theta_{i,j}$) and θ_2 that of the next interaction ($\theta_{j,k}$).

The MDD representation of $\Gamma_{i,j}$ is obtained by a depth-first exploration of the MDD representing $\mathcal{K}_j$. The *LOOKUP* function takes as inputs $\mathcal{K}_j$, the order of g_i, $\theta_{i,j}$ and $\theta_{j,k}$, and computes $\Gamma_{i,j}$. This function explores $\mathcal{K}_j$ recursively, building $\Gamma_{i,j}$ as a copy of the explored part until :

- a node having the order of g_i is found, then the exploration must continue comparing its children labelled $\theta_{i,j}-1$ and $\theta_{i,j}$; this is achieved by the *COMPARE* function, which takes as inputs the two children and $\theta_{j,k}$;
- a leaf or a node with a higher order is encountered, then the interaction has no effect, and a 0 leaf is added to $\Gamma_{i,j}$.

The function *COMPARE* returns a MDD resulting from the combination of the left and right children around the threshold $\theta_{i,j}$, which leaves indicate the effect of the interaction.

Pseudo-code of the construction of the MDD representing Γ, sign of a given interaction. The function *LOOKUP* returns the MDD recursively constructed during the traversing of $\mathcal{K}$, the MDD of the logical function attached to the target of the interaction.

```
// Main function: constructs Γ while recursively searching
//       a node of order n in K.
// If found, call COMPARE to explore children around θ1.
LOOKUP (K, n, θ1, θ2) {
   if (K is a leaf) or (K.order > n)
      return (0) // 0-valued leaf
   if K.order = n
      return COMPARE(K.child[θ1-1], K.child[θ1], θ2)
   Γ ← new node
   Γ.order ← K.order
   for i ← 0...K.nbChild
      Γ.child[i] ← LOOKUP(K.child[i], n, θ1, θ2)
   return Γ }
// constructs Γ while exploring K1 and K2
// and comparing their leaves with θ2.
COMPARE (K1, K2, θ2){
   if (K1 is a leaf) and (K2 is a leaf)
      if (K1.value < θ2) and (K2.value ≥ θ2)
         return (+1)
      if (K1.value ≥ θ2) and (K2.value < θ2)
         return (-1)
      return (0)
   if (K1 is a leaf) or (K1.order > K2.order)
      Γ ← new node
```

```
        Γ.order ← K2.order
        for i ← 0..K2.nbChild
            Γ.child[i] ← COMPARE(K1, K2.child[i], θ2)
        return Γ
    if (K2 is a leaf) or (K1.order < K2.order)
        Γ ← new node
        Γ.order ← K1.order
        for i ← 0...K1.nbChild
            Γ.child[i] ← COMPARE(K1.child[i], K2, θ2)
        return Γ
    // K1.order = K2.order
    Γ ← new node
    Γ.order ← K1.order
    for i← 0...K1.nbChild
        Γ.child[i] ← COMPARE(K1.child[i], K2.child[i], θ2)
    return Γ
}
```

6.5 Optimisations de l'implémentation

Les algorithmes présentés ici utilisent des MDD au lieu de BDD et recourent à des manipulations non-classiques de ces diagrammes. Ceci nous empêche d'utiliser les bibliothèques existantes. GINsim intègre donc une implémentation spécifique des MDD. Celle-ci ne bénéficie pas encore de toutes les optimisations possibles (en particulier le réordonnement dynamique des variables de décision).

Bien que relativement naïve, l'implémentation actuelle intègre tout de même quelques optimisations. En particulier, les MDD sont rapides à manipuler tant que leur profondeur reste faible. Il est donc souhaitable de limiter cette profondeur aussi longtemps que possible. Pour cela, les MDD à combiner lors de la recherche d'états stables sont triés en fonction du nombre de "nouveaux régulateurs" qu'ils apportent. Bien que le simple tri utilisé pour cela ne soit pas optimal, le gain de performance apporté en pratique est significatif (passage de plusieurs minutes à moins d'une seconde pour le modèle de signalisation TCR à 94 variables présenté dans la section 4.3). Dans le cas de l'analyse des circuits, les contextes de fonctionnalité des interactions sont conservés en mémoire afin d'éviter de les recalculer pour les interactions intervenant dans plusieurs circuits.

Chapitre 7

Méthodes de réduction

7.1 Classes de priorités : réduction de la dynamique

Les travaux présentés précédemment permettent de déterminer certaines propriétés dynamiques de systèmes comportant un nombre relativement grand de composants. Cependant, des résultats plus précis sont souvent nécessaires, il faut alors recourir aux simulations. La construction du graphe de transitions d'états (GTE) complet étant rapidement impossible (et souvent inutile), il est possible de n'en construire qu'une partie en partant de quelques états initiaux d'intérêt. Dans certains cas particulièrement défavorables, la simulation couvre tout de même une grande partie du GTE complet (voir le modèle multicellulaire du réseau "segment polarity" [100]). Il est également possible de limiter la profondeur de simulation ou le nombre total d'états créés, mais ces contraintes entrainent la perte de larges régions de la dynamique potentiellement intéressantes.

On se heurte ici aux limites de la mise-à-jour asynchrone qui, en générant tous les comportements possibles, conduit souvent à un grand nombre de trajectoires irréalistes ou redondantes. Cette caractéristique peut être très coûteuse dans le cas de très grands GTE. Il est toujours possible d'utiliser une mise-à-jour synchrone, générant au plus un unique successeur par état. Malheureusement, ce type de mise-à-jour introduit souvent des comportements artefactuels. Nous avons donc cherché un moyen de restreindre le nombre de successeurs des états sans pour autant tomber dans les excès de la mise-à-jour synchrone.

Pour cela, nous avons proposé l'introduction d'une méthode de mise-à-jour configurable dans laquelle les composants sont répartis entre classes de priorité. Chacune de ces classes a un rang (plusieurs classes peuvent partager le même rang) et une méthode de mise-à-jour interne (synchrone ou asynchrone). La simulation considère alors uniquement les transitions les plus prioritaires (c'est-à-dire telles qu'aucune transition n'est possible sur les composants associés aux classes de meilleur rang).

Cette méthode permet de prendre en compte des informations qualitatives sur les vitesses relatives des processus impliqués dans le modèle. Par exemple, il est bien connu que la synthèse *de novo* de protéines prend plus de temps que la plupart des modifications post-traductionnelles (de l'ordre de l'heure versus de la seconde). Tant que l'on utilise uniquement des classes asynchrones, le GTE obtenu est un sous-graphe du GTE asynchrone. Dans le cas général, les états stables sont les mêmes que pour les autres méthodes, mais la dynamique obtenue peut varier significativement par rapport à celles obtenues en synchrone ou asynchrone (voir figure 2.3 pour un exemple de variation entre les autres méthodes de mise-à-jour).

Plus de détails sur la définition de classes de priorités, ainsi qu'un exemple d'application au cycle cellulaire, sont disponibles dans [34] (inclus en annexe A.2).

7.2 Réduction de modèles

Afin de permettre l'analyse de grands graphes de régulation, il est également possible de réduire le nombre de composants. Lors de la construction du modèle, le modélisateur peut choisir un niveau de détail adapté. Il est courant de regrouper plusieurs acteurs en un seul composant ou de masquer des intermédiaires non-essentiels pour les analyses que l'on souhaite effectuer.

Ces réductions manuelles ont plusieurs inconvénients. D'une part, elles risquent de modifier le comportement du modèle et constituent une source d'erreur dans la construction du modèle. D'autre part, elles nuisent à la lisibilité du modèle : celui-ci ne reflète plus directement les données utilisées pour sa construction comme le fait un modèle "complet". Il est possible de maintenir plusieurs versions (complète et réduites) du modèle mais cela implique un travail supplémentaire et un risque d'incohérence entre ces versions.

L'automatisation de certaines réductions peut palier à ces inconvénients : un modèle réduit est déterminé à partir du modèle complet, il n'y a donc plus de problème de lisibilité ou d'incohérence. D'autre part, il est possible de déterminer l'impact de cette réduction sur le comportement du modèle.

Nous avons proposé une méthode de réduction des modèles logiques permettant de "masquer" un composant dans le graphe de régulation. Elle peut être appliquée de manière itérative afin de masquer plusieurs composants. Cette méthode masque un composant en reportant l'effet de ses régulateurs sur ses cibles, pour lesquelles de nouvelles fonctions logiques sont définies. On s'interdit de masquer les composants auto-régulés afin d'éviter la suppression de circuits de régulation. En effet, les circuits de régulation jouent un rôle important dans la configuration des attracteurs du modèle que nous souhaitons préserver.

En se reposant sur cette contrainte, nous avons pu montrer qu'un modèle réduit à l'aide de cette méthode a les mêmes états stables que le modèle d'origine. Nous avons également évalué l'impact de cette réduction sur la dynamique asynchrone. Les attracteurs complexes asynchrones sont préservés. Certaines propriétés dynamiques peuvent cependant être perturbées. En effet, cette réduction revient à considérer les composants masqués comme étant plus rapides que les autres. Tout comme l'utilisation de classes de priorité, cette réduction supprime les trajectoires incompatibles avec cette contrainte. La perte de ces trajectoires modifie l'atteignabilité de certains états et peut également conduire à l'apparition de nouveaux attracteurs cycliques à partir de cycles transients.

Le détail de la méthode et de son impact sur le comportement dynamique sont décrits dans le papier suivant, présenté à CMSB 2009[1] [80].

[1] cmsb09.cs.unibo.it

7.3 Article présenté à CMSB 2009

Les annexes sont insérées à la suite de l'article.

A Reduction of Logical Regulatory Graphs Preserving Essential Dynamical Properties

Aurélien Naldi[1], Elisabeth Remy[2], Denis Thieffry[1,3], and Claudine Chaouiya[1,4]

[1] TAGC Inserm U928 - Université de la Méditerranée, Marseille, France
[2] IML UMR 6206, Marseille, France
[3] CONTRAINTES, INRIA Paris Rocquencourt, France
[4] IGC, Instituto Gulbenkian de Ciência, Oeiras, Portugal

Abstract. To cope with the increasing complexity of regulatory networks, we define a reduction method for multi-valued logical models.

Starting with a detailed model, this method enables the computation of a reduced model by iteratively "hiding" regulatory components. To keep a consistent behaviour, the logical rules associated with the targets of each hidden node are actualised to account for the (indirect) effects of its regulators.

The construction of reduced models ensures the preservation of a number of dynamical properties of the original model. In particular, stable states and more complex attractors are conserved. More generally, we focus on the relationship between the attractor configuration of the original model and that of the reduced model, along with the issue of attractor reachability.

The power of the reduction method is illustrated by its application to a multi-valued model of the segment-polarity network Controlling segmentation in the fly Drosophila melanogaster.

Keywords: Regulatory networks, logical modelling, model reduction, decision diagrams, regulatory circuits, stable states, complex attractors, Drosophila development, segmentation.

1 Introduction

Biological data generation and integration efforts result in the delineation of ever more comprehensive and complex regulatory networks involved in the control of numerous processes. Consequently, current modelling and analysis approaches are reaching their limits in terms of the number and variety of components and interactions that can be efficiently considered. This is true for quantitative frameworks (*e.g.*, differential or stochastic models), as well as for qualitative approaches. Indeed, although logical modelling enables to handle networks comprising relatively large numbers of components (see *e.g.* [1,2]), the size of the state space grows exponentially with the number of regulatory nodes.

P. Degano and R. Gorrieri (Eds.): CMSB 2009, LNBI 5688, pp. 266–280, 2009.

One way to handle this problem consists in developing compositional approaches to compute the dynamical properties of comprehensive networks, relying on the knowledge of the properties of simpler sub-systems or modules. A complementary approach consists in reducing large systems, by focusing on the most relevant components and redefining their interactions in order to preserve relevant dynamical properties (*e.g.* stable states).

Most often, modellers intuitively and manually reduce regulatory networks to address specific questions. Such empirical reductions have several drawbacks: (i) the process is error prone and limited to relatively simple cases; (ii) the maintenance of different versions of a model (complete and reduced) is cumbersome; (iii) storing the sole reduced model leads to the loss of relevant biological information.

These considerations led us to develop a reliable, automated reduction method in the context of a logical modelling framework. In this respect, we lean on the software *GINsim*, which facilitates the definition of comprehensive logical regulatory graphs, as well as the analysis of their dynamical properties [3,4]. Established on firm mathematical bases, our reduction method allows the user to select nodes to be made implicit and to perform dynamical analyses on reduced model versions, which preserve relevant topological and dynamical properties.

The paper is organised as follows. Section 2 recalls the definitions of logical regulatory graphs and of the associated state transition graphs. Next, the reduction method is defined in Section 3. Relationships between the dynamical behaviour of the original model and that of the reduced model are delineated in Section 4. A multi-valued logical model of the segment-polarity network is then used to demonstrate the power of the proposed reduction method in Section 5. The paper ends with conclusions and further prospects.

All models presented in this paper can be opened, edited, simulated, and analysed with *GINsim*, which implements the logical formalism and the reduction method presented here.

2 Logical Modelling of Regulatory Networks

Our modelling approach leans on the generalised logical formalism initially developed by R. Thomas *et al.* [5,6,3]. In this context, a regulatory network and its dynamics are both represented in terms of oriented graphs.

2.1 Regulatory Graphs

Definition 1. A logical regulatory graph (LRG) *is a directed labelled multigraph* $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \Gamma, \Theta, \mathcal{K})$ *where,*

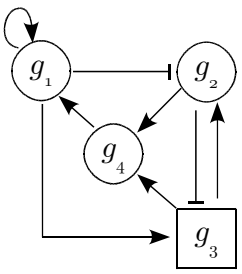
- $\mathcal{G} = \{g_1, \ldots, g_N\}$ *is the set of nodes, representing* regulatory components.
- $\mathcal{M}ax : \mathcal{G} \rightarrow \mathbb{N}^*$ *associates a maximum level* $\mathcal{M}ax(g_i) = \mathcal{M}ax_i$ *to node* g_i. *The* current level *of* g_i, *denoted* x_i, *takes its values in* $\mathcal{D}_i = \{0, \ldots, \mathcal{M}ax_i\}$.
- Γ *is the set of arcs, defined as a finite multiset of ordered pairs of elements of* $\mathcal{G}$ *representing* regulatory interactions. *If* $\mathcal{M}ax_i > 1$, g_i *may have different*

effects onto a component g_j, depending on level x_i. Hence, the arc connecting g_i to g_j may be a multi-arc encompassing different interactions. The multiplicity of the arc (g_i, g_j) (i.e. the number of its constitutive interactions), is denoted $m_{i,j}$ ($1 \leq m_{i,j} \leq \mathcal{M}ax_i$). Loops (even multi-loops) are allowed: an arc (g_i, g_i) denotes an autoregulation of g_i.
For each $g_j \in \mathcal{G}$, $Reg(j)$ denotes the set of its regulators: $g_i \in Reg(j)$ if and only if $(g_i, g_j) \in \Gamma$.

- *Θ is a labelling function, which associates a* threshold *to each element of Γ. More precisely, $\theta_{i,j,k}$ is associated to the k^{th} interaction between g_i and g_j (denoted $(g_i, g_j, \theta_{i,j,k}), k \in \{1, \ldots, m_{i,j}\}$), with $1 \leq \theta_{i,j,1} < \cdots < \theta_{i,j,m_{i,j}} \leq \mathcal{M}ax_i$. This interaction is* active*, when x_i, the level of its source g_i, lays between the threshold of this interaction and that of the next interaction: $\theta_{i,j,k} \leq x_i < \theta_{i,j,k+1}$ (by convention, $\theta_{i,j,m_{i,j}+1} = \mathcal{M}ax_i + 1$).*
- *$\mathcal{K} = (\mathcal{K}_1, \ldots, \mathcal{K}_N)$ defines the* logical rules *attached to the nodes specifying their behaviours: each $\mathcal{K}_i$ is a multi-valued logical function that gives the target value of g_i, depending on the levels of the regulators acting on g_i:*

$$\mathcal{K}_i : \left(\prod_{g_j \in \mathcal{G}} \mathcal{D}_j \right) \mapsto \{0, \ldots, \mathcal{M}ax_i\}.$$

The logical function $\mathcal{K}_i$ can be equivalently defined on the set $\prod_{g_j \in Reg(i)} \mathcal{D}_j$, giving the target value of g_i depending on the current levels of its regulators. Figure 1 illustrates this definition of a logical regulatory graph. In the following, when no confusion is possible, we will use i to denote g_i.



$\mathcal{G} = \{g_1, g_2, g_3, g_4\}$
$\mathcal{M}ax_1 = \mathcal{M}ax_2 = \mathcal{M}ax_4 = 1$
$\mathcal{M}ax_3 = 2$
$\mathcal{D}_1 = \mathcal{D}_2 = \mathcal{D}_4 = \{0, 1\}, \mathcal{D}_3 = \{0, 1, 2\}$
$Reg(2) = \{g_1, g_3\}$
$\theta_{3,2,1} = 2$

Fig. 1. Example of logical regulatory graph. Left: graphical representation of a LRG. Blunt arrows depict inhibitions while normal arrows depict activations (this is only a graphical convention, since the logical functions encode the regulatory effects). The rectangular node g_3 is ternary, whereas the others nodes are Boolean. The thresholds of all interactions are set to 1, except that of (g_3, g_2), which is set to 2. Right: illustration of the notations of Definition 1. Examples of logical functions $\mathcal{K}_i$ are displayed in Figure 2 for the same model.

2.2 State Transition Graphs

We represent the dynamical behaviour of a LRG in terms of a state transition graph, defined as follows.

Definition 2. *Given a LRG* $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \Gamma, \Theta, \mathcal{K})$*, its associated full state transition graph (STG)* $\mathcal{E} = (\mathcal{S}, \mathcal{T})$ *is a directed graph, where:*

- $\mathcal{S} = \Pi_{i \in \mathcal{G}} \mathcal{D}_i$ *is the state space, a state of the system being a vector* $x = (x_i)_{i=1,\ldots,N}$*, with* $x_i \in \mathcal{D}_i$*,* $\forall i \in \mathcal{G}$*,*
- $\mathcal{T} \subset \mathcal{S}^2$ *is the set of transitions defined as follows:* $(x, y) \in \mathcal{T}$ *if and only if* $\exists i \in \mathcal{G}$ *such that:*
  $x_i \neq \mathcal{K}_i(x)$*,*
  $y = x + \Delta_i(x).e^i$*, where* $\Delta_i(x) = \frac{\mathcal{K}_i(x) - x_i}{|\mathcal{K}_i(x) - x_i|}$ *and* e^i *is the canonical vector in* $\mathcal{S}$ *(*$e^i_i = 1$ *and* $e^i_j = 0$*,* $\forall j \in \mathcal{G}, j \neq i$*).*

Here $\Delta_i(x)$ gives the sign of the update of i (increase or decrease). One can also consider a state transition graph related to an initial (set of) condition(s). It is then a subgraph of the full STG.

When analysing the behaviour of a LRG, we mainly focus on attractors, which represent asymptotic dynamical properties. Given a STG, attractors are its terminal strongly connected components, classified as:

- *stable states*: reduced to a unique terminal node,
- *cyclic attractors*: terminal elementary (oriented) cycles,
- *complex attractors*: other terminal strongly connected components (*i.e.* involving intertwined cycles).

Cyclic and complex attractors will be called *non-trivial* attractors.

In what follows, LRGs are assumed consistent, *i.e.* all interactions are effective and autoregulations functional, meaning that all interactions have a dynamical role and could be recovered from the logical functions $\mathcal{K}$ (see further explanations in the Appendix A).

3 Logical Regulatory Graph Reduction

This section presents the principles underlying the reduction of a regulatory graph and then defines the new model, called *reduced model*. In what follows, we consider a reduction consisting of the *removal* of a single regulatory component (making it implicit). The generalisation to a reduction encompassing a set of nodes is obtained by iterating the corresponding one-node reductions. However, the ordering of a sequence of one-node reductions may have an impact on the resulting reduced model (see Appendix B).

Here, we aim at defining a reduction method, which preserves, as much as possible, the dynamical properties of the original model. The underlying principle is already intuitively applied by modellers when they make regulatory nodes implicit in their networks.

The removal of a node r basically consists in connecting directly its regulators to its targets, which logical functions are thus revised. In the revised logical functions, the effect of r at a given value x_r is conveyed by the values of the regulators leading r to x_r. In other words, we consider the update of the removed component as a *fast process*, which is performed before anything else.

Following this principle, it is impossible to remove an autoregulated component since fixed values of its other regulators may not lead to a unique target value. Thus, the removal of an autoregulated component implies additional decisions, impeding the definition of a systematic procedure. In the following, we will require that autoregulated components should not be removed.

To properly implement an algorithm producing the reduced model, we need further notations to manipulate the logical functions. Given a regulatory graph $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \Gamma, \Theta, \mathcal{K})$ and a node $i \in \mathcal{G}$, we denote:

- $x_i^{\{l\}}$ $(l \in \mathcal{D}_i)$ the Boolean variable with value 1 when $x_i = l$, 0 otherwise.
- x_i^S the Boolean variable that is true if $x_i \in S$, false otherwise. Hence x_i^S is defined by,
$$x_i^S \triangleq \bigvee_{l \in S} x_i^{\{l\}},\ S \subseteq \mathcal{D}_i.$$
Note that $x_i^{\emptyset}$ is always false and $x_i^{\mathcal{D}_i}$ always true.
- For all $v \in \mathcal{D}_i$, the logical function $\mathcal{K}_i^v$ that gives the conditions under which the target value of node i is v. This function is defined as follows:
$$\mathcal{K}_i^v = \bigvee_{n=1,\ldots p} \mathcal{C}_i^n, \tag{1}$$
where $\mathcal{C}_i^n$ are conjunctive clauses $\mathcal{C}_i^n = \bigwedge_{j \in Reg(i)} x_j^{S_{j,i,n}}$, where $S_{j,i,n} \subseteq \mathcal{D}_j$. Each clause $\mathcal{C}_i^n$ defines a situation (*i.e.* sets of combinations of incoming interactions acting upon i) for which the target value of i is v.

In Equation (1), each clause $\mathcal{C}_i^n$ defines a subset of $\mathcal{S}$, $D = \Pi_{j \in \mathcal{G}} S_{j,i,n}$ (with $S_{j,i,n} = \mathcal{D}_j$, $\forall j \notin Reg(i)$), such that for all $x \in D$, $\mathcal{K}_i(x) = v$. Hence, Equation (1) defines a set of cubes in the state space $\mathcal{S}$, where the target value of i is v.

Definition 3. *Given a LRG $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \Gamma, \Theta, \mathcal{K})$, the reduced LRG $\mathcal{R}^r = (\mathcal{G}^r, \mathcal{M}ax^r, \Gamma^r, \Theta^r, \mathcal{K}^r)$ obtained by removing a non-autoregulated component $r \in \mathcal{G}$ is defined as follows:*

- $\mathcal{G}^r = \mathcal{G} \setminus \{r\}$.
- $\mathcal{M}ax^r : \mathcal{G}^r \to \mathbb{N}^*$, *s.t.* $\forall i \in \mathcal{G}^r\ \mathcal{M}ax^r(i) = \mathcal{M}ax_i$.
- *For all $i \in \mathcal{G}^r$, and for all $v \in \mathcal{D}_i$, the logical function $\mathcal{K}_i^{rv}$ is defined as follows. Consider $\mathcal{K}_i^v = \bigvee_{n=1,\ldots p} \mathcal{C}_i^n$, the disjunctive form of $\mathcal{K}_i^v$, as defined previously. For all $n = 1, \ldots p$ (i.e. for each clause $\mathcal{C}_i^n$), let define $\mathcal{F}_i^{rn}$ as:*
$$\mathcal{F}_i^{rn} = \left(\bigvee_{w \in S_{r,i,n}} \mathcal{K}_r^w \right) \wedge \left(\bigwedge_{j \in Reg(i) \setminus \{r\}} x_j^{S_{j,i,n}} \right)$$
Then $\mathcal{K}_i^{rv} = \bigvee_{n=1,\ldots p} \mathcal{F}_i^{rn}$.
- *Γ^r and Θ^r are deduced from $\mathcal{K}^r$; for all $i \in \mathcal{G}^r$, $j \in \mathcal{G}^r$,*
$$m_{i,j}^r = \sum_{v \in [1, \mathcal{M}ax_i]} \mathbb{1}_{i,j,v},$$

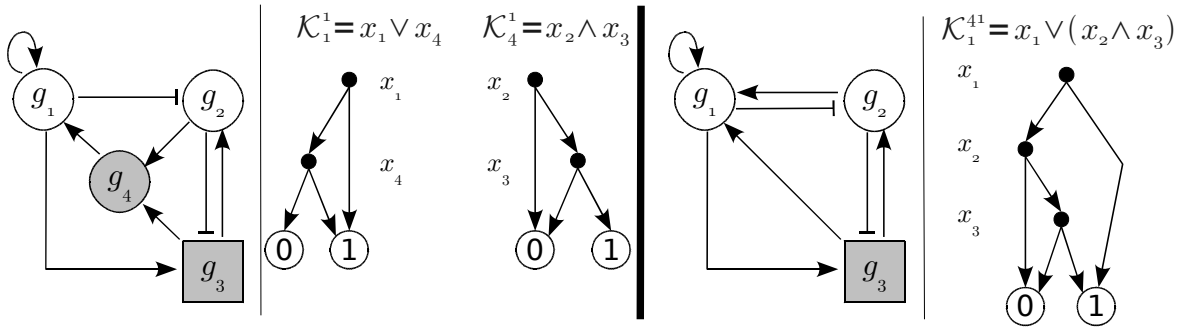


Fig. 2. Reduction in terms of MDDs. Left: the same LRG as in Figure 1, where g_4 (greyed-out) is selected for removal. Logical functions for g_1 and g_4 are shown on the right, along with their MDD representations. Right: the reduced LRG after removal of g_4, along with the resulting logical function for g_1. In the MDDs, internal nodes are labelled with the associated variable (x_i), whereas leaves represent the value of the logical functions. Children of internal nodes are ordered from left to right: the leftmost (resp. rightmost) child is the root of the sub-diagram corresponding to the case $x_i = 0$ (resp. $x_i = \mathcal{M}ax_i$).

where $\mathbb{1}_{i,j,v} = 1$ *if it exists* $x \in \mathcal{S}$ *such that* $x_i = v-1$ *and* $\mathcal{K}_j^r(x) \neq \mathcal{K}_j^r(x+e^i)$. *Then* $(i,j) \in \Gamma^r$ *if* $m_{i,j}^r > 0$ *(and the multiplicity of* (i,j) *in* Γ^r *is given by* $m_{i,j}^r$*). Finally, the ordered set of values* v *such that* $\mathbb{1}_{i,j,v} = 1$ *defines the thresholds* $\theta_{i,j,k}^r$ *(*$k = 1, \ldots, m_{i,j}^r$*).*

The logical function $\mathcal{K}_i^{rv}$ is deduced from the logical function $\mathcal{K}_i^v$ by replacing, in each clause, literals x_r^S by the formulae giving the conditions under which the target value of r is in S (remark that this definition may not give $\mathcal{K}_i^{rv}$ in a proper disjunctive form). Note that if $\mathcal{C}_i^n$ does not depend on r (*i.e.* $r \notin Reg(i)$) then $S_{r,i,n} = \mathcal{D}_r$ and $\mathcal{F}_i^{rn} = \mathcal{C}_i^n$ for all n, therefore $\mathcal{K}_i^{rv} = \mathcal{K}_i^v$.

The set of arcs verifies:

$$\Gamma^r \subseteq \{(i,j) \in \mathcal{G}^r \times \mathcal{G}^r, \text{ s.t. } (i,r),\ (r,j) \in \Gamma \text{ or } (i,j) \in \Gamma\}.$$

In practice, the construction of the new logical function is performed using Reduced Ordered Multivalued Decision Diagrams (ROMDDs or MDDs for short). Decision diagrams are rooted directed acyclic graphs, widely used to represent logical functions (see *e.g.* [7,8]). In these diagrams, internal nodes are labelled with *decision variables* and have one child per value, while leaves represent the values of the function. Decision variables are ordered: each internal node has a rank and the sub-diagrams rooted by the children of a node of rank i do not contain internal nodes of rank $j \leq i$. In [9] we used MDDs to represent the logical functions $\mathcal{K}_i$. In this context, decision variables are the levels of the components of the model. For the sake of simplicity, we consider that the ordering of the MDD variables is the same as that of the LRG components. Given the MDD representation of $\mathcal{K}_i$ and a state x, a unique path from the root of the MDD to one of its leaves is defined. Along this path, the child chosen for each non-terminal node is labelled with the value of the corresponding variable in state

x. The terminal node reached through this path gives the value of $\mathcal{K}_i(x)$. Each clause of $\mathcal{K}_i^v$ corresponds to a path leading to a leaf valued v.

To compute the MDD representing $\mathcal{K}_i^r$, we define the recursive algorithm given in Appendix C and illustrated in Figure 2.

4 Dynamics of the Reduced Model

In this section, the dynamical behaviour of a reduced LRG (as specified in Definition 3) is compared to that of the original LRG. In particular, we show that the reduction preserves existing attractors and does not add any spurious path.

Let $\mathcal{E} = (\mathcal{S}, \mathcal{T})$ be the full state transition graph of $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \Gamma, \Theta, \mathcal{K})$ and $r \in \mathcal{G}$ a node not autoregulated. Let $\mathcal{E}^r = (\mathcal{S}^r, \mathcal{T}^r)$ be the full STG of $\mathcal{R}^r = (\mathcal{G}^r, \mathcal{M}ax^r, \Gamma^r, \Theta^r, \mathcal{K}^r)$, the LRG obtained after the removal of r from $\mathcal{G}$.

Consider the projection $\pi_r : \mathcal{S} \to \mathcal{S}^r$ such that, $\forall i \in \mathcal{G}^r$, $\forall x \in \mathcal{S}, (\pi_r(x))_i = x_i$, and the equivalence relation on $\mathcal{S}$: $\forall x, y \in \mathcal{S}$, $x \sim_r y$ iff $\pi_r(x) = \pi_r(y)$.

We denote $[x]_{\sim r}$ the equivalence class: $[x]_{\sim r} = \{y \in \mathcal{S} \text{ s.t. } y \sim_r x\}$. The class $[x]_{\sim r}$ contains all states of $\mathcal{S}$ that differ only by their r^{th} component, *i.e.* the $(\mathcal{M}ax_r + 1)$ states $\{x^i \in \mathcal{S}, i = 0, \ldots, \mathcal{M}ax_r\}$, such that $x^i \sim_r x$ and $x_r^i = i$. Because r is not autoregulated, $\forall x^i \in [x]_{\sim r}$, $\mathcal{K}_r(x^i) = \mathcal{K}_r(x)$. This implies that:

- $(x^i, x^{i+1}) \in \mathcal{T}$, for all $i < \mathcal{K}_r(x)$,
- $(x^i, x^{i-1}) \in \mathcal{T}$, for all $i > \mathcal{K}_r(x)$,
- $(x^{\mathcal{K}_r(x)}, x^i) \notin \mathcal{T}$, for all i.

Hence, for all $x \in \mathcal{S}$, there exists a path in $\mathcal{S}$ from x to $x^{\mathcal{K}_r(x)}$, which is the representative state of $[x]_{\sim r}$.

Definition 4. *$x \in \mathcal{S}$ is the* representative state *of an equivalence class for $\sim_r$ iff $x_r = \mathcal{K}_r(x)$.*

We can then define the *retrieval* function $s_r : \mathcal{S}^r \to \mathcal{S}$ such that, $\forall z \in \mathcal{S}^r$,

$$(s_r(z))_i = z_i, \text{ for all } i \in \mathcal{G} \setminus \{r\},$$

$$(s_r(z))_r = \mathcal{K}_r(x), \text{ with } x \text{ such that } \pi_r(x) = z.$$

In other words, $s_r(z)$ is the representative state of the equivalence class projected on z (see Figure 3). Relying on this, we can introduce an alternative definition of the logical functions in the reduced LRG: $\forall i \in \mathcal{G}^r$, $\mathcal{K}_i^r : \mathcal{S}^r \mapsto \mathcal{D}_i$ is defined as $\mathcal{K}_i^r(z) = \mathcal{K}_i(s_r(z))$. Note that if $(r, i) \notin \Gamma$ (*i.e.* r is not a regulator of i), $\mathcal{K}_i^r(\pi_r(x)) = \mathcal{K}_i(x)$.

Remark 1. It follows from their definitions that functions π_r and s_r verify:

1. $\pi_r \circ s_r$ is the identity function.
2. For any $x \in \mathcal{S}$, $(s_r \circ \pi_r(x)) \sim_r x$.
3. If $x \in \mathcal{S}$ is a representative state, then, $s_r \circ \pi_r(x) = x$.
4. For any $z \in \mathcal{S}^r$, $\mathcal{K}^r(z) = \pi_r(\mathcal{K}(s_r(z)))$; indeed, $\forall x \in \mathcal{S}$, $\forall i \in \mathcal{G}^r, \mathcal{K}_i^r(\pi_r(x)) = \mathcal{K}_i(s_r \circ \pi_r(x))$.

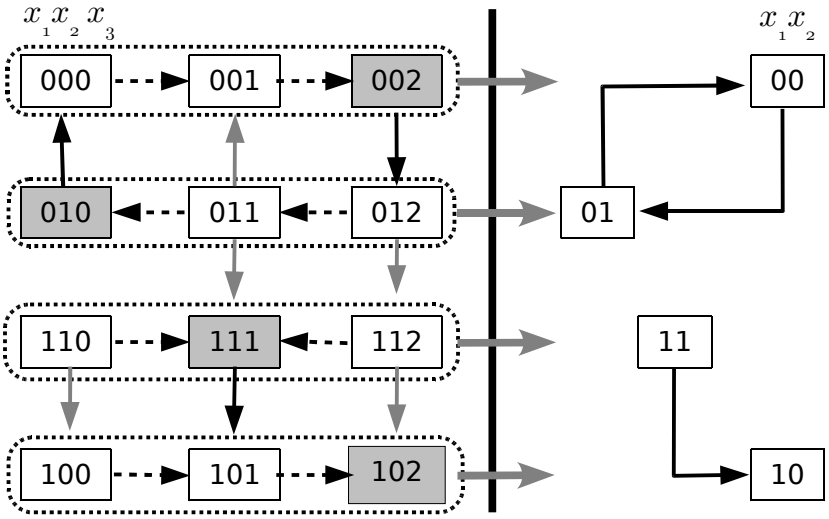


Fig. 3. Dynamical behaviour of the reduced model given in Figure 2, before and after removal of the ternary node g_3. Left: State transition graph (STG), partitioned into four equivalence classes for g_3. Each equivalence class contains 3 states; its representative state is greyed out and internal transitions are dashed. Right: STG of the reduced model, each state corresponding to an equivalence class of the original STG. After the reduction, the stable state 102 is projected on 10 and all transitions are preserved except the one from the second equivalence class to the third one. This results in the isolation of the non-terminal strongly connected component involving the first two equivalence classes of the original STG, hence generating the attractor $(01, 00)$.

The following lemma establishes the relationships between transitions in $\mathcal{E}$ and $\mathcal{E}^r$.

Lemma 1. *1. Let $z, z' \in \mathcal{S}^r$.*

$$(z, z') \in \mathcal{T}^r \implies \exists x \in \mathcal{S} \text{ s.t. } \pi_r(x) = z' \text{ and } (s_r(z), x) \in \mathcal{T}\,.$$

2. Let x, $y \in \mathcal{S}$. If x is a representative state, then

$$(x, y) \in \mathcal{T} \quad \implies \quad (\pi_r(x), \pi_r(y)) \in \mathcal{T}^r\,.$$

Proof. Recall that $\Delta_i(x) \triangleq \frac{\mathcal{K}_i(x) - x_i}{|\mathcal{K}_i(x) - x_i|}$. For $z \in \mathcal{S}^r$ s.t. $z_i \neq \mathcal{K}_i^r(z)$, we similarly denote:

$$\Delta_i^r(z) \triangleq \frac{\mathcal{K}_i^r(z) - z_i}{|\mathcal{K}_i^r(z) - z_i|} = \frac{\mathcal{K}_i(s_r(z)) - (s_r(z))_i}{|\mathcal{K}_i(s_r(z)) - (s_r(z))_i|} = \Delta_i(s_r(z))\,.$$

1. Consider $z, z' \in \mathcal{S}^r$ such that $(z, z') \in \mathcal{T}^r$. Then $\exists i \neq r$ s.t. $\mathcal{K}_i^r(z) \neq z_i$, and $z' = z + \Delta_i^r(z)\, e^i$. By definition, $\mathcal{K}_i^r(z) = \mathcal{K}_i(s_r(z)) \neq (s_r(z))_i = z_i$. This implies that $(s_r(z), x) \in \mathcal{T}$ with $x \in \mathcal{S}$ and $x = s_r(z) + \Delta_i(s_r(z))\, e^i$, and then $\pi_r(x) = z'$.
2. Consider $x, y \in \mathcal{S}$ such that $\mathcal{K}_r(x) = x_r$. The hypothesis $(x, y) \in \mathcal{T}$ implies that $\exists i \in \mathcal{G}, i \neq r$ s.t. $\mathcal{K}_i(x) \neq x_i$, and $y = x + \Delta_i(x)\, e^i$.

We have $\mathcal{K}_i^r(\pi_r(x)) = \mathcal{K}_i(x)$ (since x is a representative state), and $x_i = (\pi_r(x))_i$, since $i \neq r$. So, $\mathcal{K}_i^r(\pi_r(x)) \neq (\pi_r(x))_i$, and then $\exists z \in \mathcal{S}^r$ s.t. $(\pi_r(x), z) \in \mathcal{T}^r$, with

$$z = \pi_r(x) + \Delta_i^r(\pi_r(x))\, e^i = \pi_r(x) + \Delta_i(s_r \circ \pi_r(x))\, e^i = \pi_r(y)\,. \qquad \square$$

The first item of Lemma 1 states that any transition in $\mathcal{T}^r$ corresponds to at least one transition in $\mathcal{T}$. Clearly, the reverse is not true. The second item of the lemma gives a condition under which transitions are preserved from $\mathcal{T}$ to $\mathcal{T}^r$. Of course, it is important to know which transitions are lost through the reduction.

Definition 5. *The reduction preserves a transition* $(x, y) \in \mathcal{T}$ *if* $(\pi_r(x), \pi_r(y)) \in \mathcal{T}^r$, *or* $\pi_r(x) = \pi_r(y)$. *The reduction preserves a path* $(s_1, \ldots, s_n) \in \mathcal{E}$ *if all its transitions are preserved.*

In other words, a path $(s_1, \ldots, s_n)$ in $\mathcal{E}$ is preserved if the reduction preserves the transitions between equivalence classes, in the required order.

The following property characterises the transitions that are not preserved by the reduction.

Property 1. A transition $(x, y) \in \mathcal{T}$ is *not* preserved by the reduction if and only if the three following conditions are satisfied:

1. x is not a representative state,
2. $y \notin [x]_{\sim r}$ ($\Rightarrow\ \exists i \neq r$ s.t. $y_i \neq x_i$),
3. $\Delta_i(x) \neq \Delta_i(s_r \circ \pi_r(x))\,.$

The last condition means that there is no call for updating i in the same direction in state $s_r \circ \pi_r(x)$.

Proof. Consider a transition $(x, y) \in \mathcal{T}$, which satisfies the three conditions. Suppose that (x, y) is preserved by the reduction, then $(\pi_r(x), \pi_r(y)) \in \mathcal{T}^r$ (the case $\pi_r(x) = \pi_r(y)$ is not possible because of the second condition). This means that there exists $j \neq r$ s.t. $(\pi_r(x))_j \neq (\pi_r(y))_j$, and $(\pi_r(x))_k = (\pi_r(y))_k$ for any $k \neq j$. With Condition 2 and by definition of π_r, we deduce that $j = i$. Moreover, we know that:

$$\pi_r(y) = \pi_r(x) + \Delta_i^r(\pi_r(x))\, e^i = \pi_r(x) + \Delta_i(s_r \circ \pi_r(x))\, e^i\,.$$

Finally, $y = x + \Delta_i(x)\, e^i$, and, as $y_i = (\pi_r(y))_i$, we have $\Delta_i(x) = \Delta_i(s_r \circ \pi_r(x))$. This contradicts Condition 3. Hence, (x, y) is not preserved by the reduction.

Conversely, let $(x, y) \in \mathcal{T}$ be a transition not preserved by the reduction.

- Condition 1 is satisfied by the second item of Lemma 1.
- Condition 2 is satisfied because $y \in [x]_{\sim r} \ \Rightarrow \pi_r(x) = \pi_r(y) \ \Rightarrow \ (x, y)$ preserved, hence a contradiction.
- We know that $y = x + \Delta_i(x)\, e^i$. As $\mathcal{K}^r(\pi_r(x)) = \pi_r(\mathcal{K}(s_r \circ \pi_r(x)))$ (cf. Remark 1),

$$\begin{aligned} \mathcal{K}_i^r(\pi_r(x)) &= (\pi_r(\mathcal{K}(s_r \circ \pi_r(x))))_i = \mathcal{K}_i(s_r \circ \pi_r(x)) \\ &\neq x_i = (\pi_r(x))_i\,. \end{aligned}$$

Hence, there exists $z \in \mathcal{S}^r$ s.t. $(\pi_r(x), z) \in \mathcal{T}^r$ with

$$\begin{aligned} z_i &= \pi_r(x) + \Delta_i^r(\pi_r(x))\, e^i \\ &= \pi_r(x) + \Delta_i(s_r \circ \pi_r(x))\, e^i = \pi_r(x) + \Delta_i(x)\, e^i\,. \end{aligned}$$

Consequently, $\pi_r(y) = z$ and (x, y) is preserved, hence a contradiction. □

Given C, a set of states in $\mathcal{S}$, we denote $\pi_r(C) \triangleq \{\pi_r(x),\ x \in C\}$. Given C', a set of states in $\mathcal{S}^r$, we denote $s_r(C') \triangleq \{s_r(z),\, z \in C'\}$. Note that $\pi_r(C)$ may contain less elements than C, and that $s_r(C')$ contains only representative states. The following results relate attractors in $\mathcal{E}$ and $\mathcal{E}^r$. Proofs are provided in Appendix D and E.

Theorem 1. *Consider a LRG $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \Gamma, \Theta, \mathcal{K})$ and $\mathcal{R}^r$ the reduced LRG. Let $\mathcal{E}$ (resp. $\mathcal{E}^r$) be the full STG of $\mathcal{R}$ (resp. of $\mathcal{R}^r$), then:*

1. *Stable states in $\mathcal{E}$ and $\mathcal{E}^r$ verify:*
   - *x stable state in $\mathcal{E} \implies \pi_r(x)$ stable state in $\mathcal{E}^r$. Furthermore no other stable state is projected on $\pi_r(x)$,*
   - *z stable state in $\mathcal{E}^r \implies s_r(z)$ stable state in $\mathcal{E}$.*

   Hence, the number of stable states is conserved by the reduction.
2. *If $(s_1, \dots s_n)$ is a cyclic attractor in $\mathcal{E}$, then $(\pi_r(s_1), \dots \pi_r(s_n))$ is a cyclic attractor in $\mathcal{E}^r$.*
3. *If C is a complex attractor in $\mathcal{E}$, $z \in \pi_r(C)$ and $(z, z') \in \mathcal{T}^r$, then $z' \in \pi_r(C)$. As a consequence, $\pi_r(C)$ contains at least one non-trivial attractor in $\mathcal{E}^r$.*

Theorem 1 characterises the dynamical properties conserved by the reduction. Going further, it is possible to identify the situations leading to the generation of additional non-trivial attractors. A non-trivial attractor in the reduced STG corresponds to a (part of a) strongly connected component of the original STG. This SCC is itself a non-trivial attractor or involves outgoing transitions all in conflict with transitions concerning the removed component. In other words, we can fully characterise the set of states in the original STG giving rise to a non-trivial attractor in the reduced dynamics. Interestingly, this set corresponds to transient oscillatory behaviour from which the system cannot escape provided that updates of the removed component are always faster than other concurrent changes. This is formalised by Theorem 2 in Appendix E.

5 Application: Segment Polarity

We demonstrate the power and flexibility of our reduction method through its application to the segment-polarity network, which plays a key role in the segmentation of the fly embryo. This system has been thoroughly analysed by developmental geneticists and has been already modelled using continuous [10,11,12] and logical approaches [13,14,15]. However, all these studies involved important simplifications of the network, particularly so as a proper modelling of

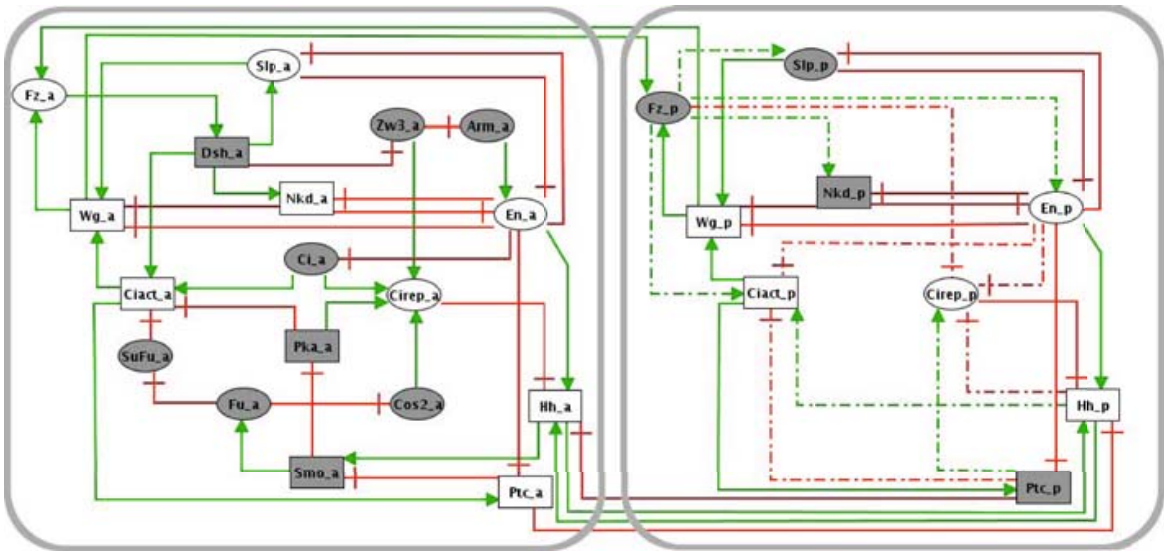


Fig. 4. Logical model of the segment polarity network for two cells, based on [15]. Ellipsoid and rectangular nodes denote Boolean and ternary components, respectively. The two cellular networks have been properly connected to take into account Wg and Hh diffusion, as well as Hh sequestration by Ptc, as in [16]. The anterior cell contains the extended version of the model, where greyed-out components will be removed, leading to the model on the right. Dashed arrows denote indirect interactions resulting from this reduction. Greyed-out components in the posterior cell are candidates for further reduction.

its behaviour requires the chaining of several identical networks to account for inter-cellular interactions through Wingless (Wg) and Hedgehog (Hh) signalling. Describing the most complete model to date, [15] had to discard various components known to play important roles in Wg and Hh signalling to keep dynamical simulations and analyses computationally tractable for up to six cells. Here, we propose a logical model based on their full description of the segment polarity network. The resulting regulatory graph encompasses 18 components and 31 regulatory interactions (left part of Figure 4).

In order to model the intercellular interactions involved in the formation of segment boundaries, we have to connect neighbouring cells (along the anterior-posterior axis) through Wg and Hh signalling. Wg is known to bind its receptor, Frizzled (Fz), only at very short range, amounting here to neighbouring cells. This can be represented by positive arcs linking each Wg node to Fz nodes of neighbouring cells. In contrast, Hh is able to reach more distant cells, but can be sequestered by its receptor Patched (Ptc). Similar interactions have been modelled in [16] in terms of positive arcs between Hh nodes in neighbouring cells (diffusion) and negative arcs from Ptc onto the Hh node of neighbouring cells (sequestering). Figure 4 illustrates the intercellular network obtained after coupling two cells and reducing one of the cellular sub-networks down to nine components.

The reduction method described above can be advantageously applied to ease the identification of all attractors of such intercellular models (Sánchez *et al.*

Table 1. Dynamical characteristics of different reduced models derived from that of Figure 4 (involving 2 x 9 nodes after applying the same reduction to both cells). The number of reachable states decreases drastically with the number of considered nodes. Note that the three stable states remain reachable for all reductions listed, but the last one (removal of Slp).

LRG size	Removed components	Number of reached states	Reached stable states
2x9	–	$> 10^6$	TT, WE, EW
2x7	Fz,Ptc	12476	TT, WE, EW
2x6	Fz,Ptc,Nkd	1625	TT, WE, EW
2x8	Slp	11350	TT, WE

considered six cells). The modeller can select the sets of nodes to discard from the network, depending on biological considerations (*e.g.* different time scales, specific mutations, etc.). In a first step, it is reasonable to conserve transcription factors and components involved in intercellular communications: Wg, Hh and their receptors (Fz and Ptc). However, since the transcription factor Cubitus interruptus is represented by three nodes here (full length immature Ci protein, activator Ci-act and repressor Ci-rep forms), we choose to retain only the two nodes corresponding to active regulatory forms. These choices correspond to the removal of the greyed-out components in the left part of Figure 4.

The reduced model involves half of the nodes of the original one, which amounts to a much higher reduction of the number of possible states, as this grows exponentially with the number of regulatory nodes. The resulting regulatory graph (Figure 4, right) remains easy to grasp as it reasonably unfolds most intra-cellular and inter-cellular regulatory pathways. As we shall see, this logical model can be further reduced to facilitate analyses encompassing more cells.

For proper logical rules (cf. [15] and supplementary material), one can check that the detailed and the reduced two-cells models have exactly three stables states (as predicted by Theorem 1). These multi-cellular stable states combine three types of cellular states: a Wg expressing state (denoted W), an En expressing state (E), and a *trivial* state (T) expressing neither Wg, nor En. The three stable states for the two connected cells correspond to TT, WE and EW cell combinations reported by Sánchez *et al.* All three stable states are reachable from biologically relevant initial conditions (significant amounts of Wg and Slp in the anterior cell, significant amount of En in the posterior cell), provided as an outcome of the activity of the pair-rule system, cf. [17,15]. However, the size of the corresponding state transition graph still impedes detailed dynamical analyses (see Table 1).

As shown in Table 1, the removal of Fz, Ptc and Nkd drastically reduces the number of reached states without changing the reachability of the three stable states from the considered initial state. However, the sole removal of Slp impedes the reachability of the stable state with inverted Wg and En expressing cells. It also suppresses the functionality contexts of all negative circuits [9,15], implying that the state transition graph does not contain any cyclic attractor. Indeed, after further reduction to three nodes per cell (Wg, En and Hh), we were able

to check the absence of non-trivial attractors in the full STG. As the reduction cannot delete existing non-trivial attractors (see Theorem 1), this implies that all attractors of the original model are stable states.

6 Conclusions and Prospects

We have defined a reduction method that can be applied to multi-valued logical models while preserving important dynamical properties. In particular, all attractors of the original dynamics have a counterpart in the dynamics of the reduced model. Furthermore, trajectories in the reduced model can be formally related to trajectories in the original one. This enables to infer the existence of paths in the dynamics of a detailed model whenever it is possible to show (by simulation and graph analysis) that paths exist between the corresponding states in a reduced version of the model. However, the reverse is not true. Indeed, a reduction can lead to the loss of reachability properties. Whenever several asynchronous component updates are possible at a given state, the elimination of one of the updated components amounts to consider it as "faster" than the concurrent ones, leading to the possible exclusion of some transitions in the reduced STG. Such reductions relate to the delineation of specific priority class configurations [18].

One particular feature of the reduction method defined here is that the removal of (functional) autoregulated components is forbidden. This rule is related to previous work on the dynamical roles of the regulatory circuits. Indeed, it has been recently proven in the discrete framework that positive regulatory circuits are necessary to generated multiple attractors, whereas negative circuits are necessary to generate cyclic attractors (cf. [19] and references therein). At least in the discrete framework, these properties depend only on the sign of the regulatory circuit, *i.e.* on the product of the sign of the involved interactions and not on their number. From a qualitative dynamical point of view, it is thus possible to reduce the number of components of a circuit down to a single autoregulated component, while keeping the corresponding property, as long as we conserve the sign of the circuit (along with some functionality constraints).

Our formal presentation of the reduction method mainly focuses on the removal of a single component. However, iterating this process enables the removal of several components. This raises the question of the impact of the order in which reductions are performed. As shown in Appendix B, the removal of a component may be possible only after the prior removal of others. If we aim at removing as many components as possible, the ordering of removals may thus be crucial. Further work is needed to properly define optimal or maximal reductions for the general case. When the removal of a set of components is possible in several orders, we suspect that the dynamics of the resulting model does not depend on the order (work in progress).

The worst case complexity of the algorithm for the reduction of a node r that regulates k targets is in $O(m^d)$, where m is the highest number of levels of the involved components and d is the depth of the MDDs representing the revised logical functions associated to the target nodes. In most cases, $m \leq 3$ and $d \leq 5$.

Applying our reduction method to a detailed model of the segment-polarity network, we were able to show the absence of non-trivial attractors in a state transition graph too large to be stored. As indicated for this application, the reduction method offers a great flexibility to the modeller. Biological arguments (*e.g.* information on relative reaction speeds) can be used to select sets of nodes for consistent model reduction. In the course of the dynamical analysis of complex networks (*e.g.* multicellular networks), further reduction can be performed to identify all attractors and check their reachability from specific initial states.

To ease the maintenance of a detailed model along with its reduced versions, the *GINsim* implementation enables the user to define and record various reductions for the same reference model. In order to handle still larger and more complex networks, such reduction could be combined with algorithmic methods enabling the analysis of large state transition graphs ([20] and references therein), or yet with model checking techniques ([21] and references therein).

Supplementary Materials. *GINsim* can be downloaded from `http://gin.univ-mrs.fr/GINsim`. The Appendix, and the models, are available at the following URL: `http://gin.univ-mrs.fr/GINsim/publications/naldi2009.html`.

Acknowledgments. A.N. has been supported by PhD grant from the French Ministry of Research and Technology. C.C. acknowledges the support provided by the Calouste Gulbenkian Foundation. This work was further supported by research grants from the French National Agency (project ANR-08-SYSC-003), from EU FP7 (APO-SYS Project), and by the Belgian Science Policy Office (IAP BioMaGNet).

References

1. Saez-Rodriguez, J., Simeoni, L., Lindquist, J., Hemenway, R., Bommhardt, U., Arndt, B., Haus, U., Weismantel, R., Gilles, E., Klamt, S., Schraven, B.: A logical model provides insights into t cell receptor signaling. PLoS Comput. Biol. 3(8), e163 (2007)
2. Franke, R., Müller, M., Wundrack, N., Gilles, E.D., Klamt, S., Kähne, T., Naumann, M.: Host-pathogen systems biology: logical modelling of hepatocyte growth factor and helicobacter pylori induced c-met signal transduction. BMC Syst. Biol. 2, 4 (2008)
3. Chaouiya, C., Remy, E., Mossé, B., Thieffry, D.: Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework. LNCIS, vol. 294, pp. 119–126 (2003)
4. Naldi, A., Berenguier, D., Fauré, A., Lopez, F., Thieffry, D., Chaouiya, C.: Logical modelling of regulatory networks with GINsim 2.3. BioSystems (in press)
5. Thomas, R.: Regulatory networks seen as asynchronous automata: A logical description. J. Theor. Biol. 153, 1–23 (1991)
6. Thomas, R., Thieffry, D., Kaufman, M.: Dynamical behaviour of biological regulatory networks–i. biological role of feedback loops and practical use of the concept of the loop-characteristic state. Bull. Math. Biol. 57(2), 247–276 (1995)

7. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Comput. 35, 677–691 (1986)
8. Kam, T., Villa, T., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: Multi-valued decision diagrams: Theory and applications. Int. J. Multi. Logic 4, 9–12 (1998)
9. Naldi, A., Thieffry, D., Chaouiya, C.: Decision diagrams for the representation and analysis of logical models of genetic networks. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 233–247. Springer, Heidelberg (2007)
10. Meinhardt, H.: Hierarchical inductions of cell states: a model for segmentation in drosophila. J. Cell Sci. Suppl. 4, 357–381 (1986)
11. von Dassow, G., Meir, E., Munro, E.M., Odell, G.M.: The segment polarity network is a robust developmental module. Nature 406(6792), 188–192 (2000)
12. Ingolia, N.T.: Topology and robustness in the drosophila segment polarity network. PLoS Biol. 2(6), e123 (2004)
13. Albert, R., Othmer, H.G.: The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. J. Theor. Biol. 223(1), 1–18 (2003)
14. Chaves, M., Albert, R., Sontag, E.D.: Robustness and fragility of boolean models for genetic regulatory networks. J. Theor. Biol. 235(3), 431–449 (2005)
15. Sánchez, L., Chaouiya, C., Thieffry, D.: Segmenting the fly embryo: logical analysis of the role of the segment polarity cross-regulatory module. Int. J. Dev. Biol. 52(8), 1059–1075 (2008)
16. González, A., Chaouiya, C., Thieffry, D.: Logical modelling of the role of the hh pathway in the patterning of the drosophila wing disc. Bioinformatics 24(16), i234–i240 (2008)
17. Sánchez, L., Thieffry, D.: Segmenting the fly embryo: a logical analysis of the pair-rule cross-regulatory module. J. Theor. Biol. 224(4), 517–537 (2003)
18. Fauré, A., Naldi, A., Chaouiya, C., Thieffry, D.: Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. Bioinformatics 22(14), e124–e131 (2006)
19. Remy, E., Ruet, P.: From minimal signed circuits to the dynamics of boolean regulatory networks. Bioinformatics 24(16), i220–i226 (2008)
20. Garg, A., Di Cara, A., Xenarios, I., Mendoza, L., De Micheli, G.: Synchronous versus asynchronous modeling of gene regulatory networks. Bioinformatics 24(17), 1917–1925 (2008)
21. Monteiro, P.T., Ropers, D., Mateescu, R., Freitas, A.T., de Jong, H.: Temporal logic patterns for querying dynamic models of cellular interaction networks. Bioinformatics 24(16), i227–i233 (2008)

A Consistency of the LRG

Regulatory graphs are often manually constructed, based on biological knowledge, and may contain inconsistencies between their structure and their logical functions, for example if an interaction has no effect on its target. Here, we characterise consistent LRG, where the set Γ and the labelling function Θ can be deduced from $\mathcal{K}$.

An interaction (i, j, θ) is said to be *effective* if it appears in the logical function of its target, *i.e.* if it exists a state $x \in \mathcal{S}$ such that: $x_i = \theta - 1$, and $\mathcal{K}_j(x) \neq \mathcal{K}_j(x + e^i)$. Non-effective interactions can be safely removed from the LRG, leading to a LRG structure consistent with the set of its logical functions.

Consider now the case of autoregulated nodes. An effective autoregulation (i, i, θ) is said to be *functional* if the values of the logical function flank the threshold θ of the autoregulation (see [9]), *i.e.* for $x \in \mathcal{D}_i$ with $x_i = \theta - 1$ and $x' = x + e^i$, we have: $\mathcal{K}_i(x) < \theta \leq \mathcal{K}_i(x')$ or $\mathcal{K}_i(x) > \theta \geq \mathcal{K}_i(x')$.

Note that, in the Boolean case, all effective autoregulations are also functional, but effective non-functional autoregulations may appear in multi-valued models. In this case, even if the target values are different ($\mathcal{K}_i(x) \neq \mathcal{K}_i(x')$), x_i tends to the same direction. As a consequence, non-functional autoregulations do not affect the dynamical behaviour and can thus be removed along with non-effective interactions.

Here, we consider consistent LRGs (*i.e.* with all interactions effective and all autoregulations functional).

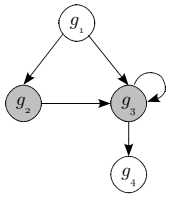
B Removing several components

To reduce a regulatory graph by removing several nodes, one iteratively applies the one-node reduction as defined above. However, the order chosen for the sequence of one-node reductions may be decisive since the removal of one component may have an impact on further reductions. Indeed, a regulatory circuits may be shortened into a single self-regulated component. In contrast, the removal of one component may suppress a self-regulation on one of its targets (see Figure 5).

It goes beyond the scope of this paper to define an order that allows an optimal reduction (with an optimality criteria still to be defined).

C Algorithm

The Algorithm takes as input the MDDs representing $\mathcal{K}_i$ (denoted $\mathcal{K}$ in the algorithm) and $\mathcal{K}_r$. The principle is to browse these two MDDs at the same time until a node of rank r is encountered in $\mathcal{K}$. All its children are then considered until the value of r is determined by reaching a leaf in $\mathcal{K}_r$. The branch of $\mathcal{K}$ corresponding to the value labelling this leaf is then retained and the path from the root of the MDD to the root of this sub-diagram is reconstructed while coming back from the recursive calls.



$\mathcal{K}_1^1 = 0\,(\mathcal{K}_1^0 = 1)$,
$\mathcal{K}_2^1 = x_1^{\{1\}}$,
$\mathcal{K}_3^1 = x_1^{\{1\}} \vee (x_2^{\{1\}} \wedge x_3^{\{1\}})$,
$\mathcal{K}_4^1 = x_3^{\{1\}}$.

Fig. 5. Removing g_2 and g_3 is possible in this order, not in the reverse one. Indeed, g_3 cannot be removed since it is autoregulated. But after the removal of g_2, the autoregulation on g_3 is no longer functional, allowing the removal of g_3.

MDD leaves have a value (denoted $\mathcal{K}$.`value` below), while internal MDD nodes have a rank ($\mathcal{K}$.`rank`) and an array of $\mathcal{M}ax_{\mathcal{K}.rank} + 1$ children ($\mathcal{K}$.`child`, where $\mathcal{K}$.`child[i]` is the i^{th} child, corresponding to $x_{\mathcal{K}.rank} = i$). The function `CREATE(p, child[])` creates an internal node of rank `p` with the children specified in the array `child[]` and takes care of the usual MDD simplifications. MDD representations of logical functions and the outcome of this algorithm are illustrated in Figure 2.

```
// INPUTS:
// - K: MDD associated to the component
// - r: index of the removed component
// - K_r: MDD associated to r
// OUTPUT: modified MDD for the component
REMOVE(K, r, K_r):
  if K is a leaf or K.rank > r:
    return K   //r has no effect
  if K.rank = r:
    if K_r.rank = r:
      ERROR: r is autoregulated
    else: //r found, browse K_r and K's children
      return FIX(K.child[],K_r)

  // recursively build child[] using K, K_r or
  // their children, depending on their rank
  p ← minimum of K_r.rank, K.rank
  for v ← 0 to Max_p:
    if K.rank = K_r.rank:
      child[v] ← REMOVE(K.child[v],r,K_r.child[v])
    else if K.rank > K_r.rank:
      child[v] ← REMOVE(K, r, K_r.child[v])
    else:
      child[v] ← REMOVE(K.child[v],r,K_r)
  return CREATE(p,child[])


//INPUTS:
// - mdd[]: array of the sub-MDDs of the component,
```

```
//            for each value of r
// - K_r :   sub-MDD associated to r
//OUTPUT: new sub-MDD for the component
FIX(mdd[], K_r):
  if K_r is a leaf: // fixed value of r
    return mdd[K_r.value]

  // recursively build the children, with a new
  // child[] filled according to the ranks in mdd[]
  p ← minimum of K_r.rank, K.rank, ∀ K ∈ mdd[]
  for v ← 0 to Max_p:
    for i ← 0 to len(mdd)-1:
      if mdd[i].rank = p:  //consider the v^th child
        t_mdd[i] ← mdd[i].child[v]
      else:  //consider the current MDD
        t_mdd[i] ← mdd[i]
    if p = K_r.rank:
      child[v] ← FIX(t_mdd[], K_r.child[v])
    else:
      child[v] ← FIX(t_mdd[], K_r)
  return CREATE(p, child[])
```

D Proof of Theorem 1

Proof. 1. Let $x \in \mathcal{S}$ be a stable state: $\mathcal{K}(x) = x$ and $\forall y \in [x]_{\sim_r}$, $\mathcal{K}(y) = x$ (by Remark 1, item 2). Then, $\mathcal{K}^r(\pi_r(x)) = \pi_r(\mathcal{K}(s \circ \pi_r(x))) = \pi_r(x)$. Hence, $\pi_r(x)$ is a stable state in $\mathcal{S}^r$. Moreover, $\pi_r(x)$ corresponds to a unique stable state as all states in $[x]_{\sim_r}$ different from x are not stable.

Let $z \in \mathcal{S}^r$ be a stable state. For all $i \in \mathcal{G}^r$, we have $\mathcal{K}^r_i(z) = z_i$. Consequently, for all $i \in \mathcal{G}^r$, $\mathcal{K}_i(s_r(z)) = (s_r(z))_i$, and, by definition of s_r, $\mathcal{K}_r(s_r(z)) = (s_r(z))_r$.

2. A cyclic attractor $(s_1, \dots s_n)$ in $\mathcal{E}$ is an elementary cycle such that $\forall i \in \{1, \dots, n\}$, s_i has a unique successor s_{i+1}. If s_i is a representative state, then $\pi_r(s_i)$ has a unique successor $\pi_r(s_{i+1})$ in $\mathcal{T}^r$. Otherwise, $s_i \sim_r s_{i+1}$ (as the successor is unique). Hence, the path $(\pi_r(s_1), \dots \pi_r(s_n))$ exists in $\mathcal{T}^r$ and each $\pi_r(s_i)$ has a unique successor. Thus $(\pi_r(s_1), \dots \pi_r(s_n))$ is a cyclic attractor in $\mathcal{G}^r$.
3. Given C, a complex attractor in $\mathcal{E}$. For all $z \in \mathcal{S}^r$, if $z \notin \pi_r(C)$ then z is not reachable from any state of $\pi_r(C)$. Suppose that there exist $x \in C$ and $z \in \mathcal{S}^r$ such that $(\pi_r(x), z) \in \mathcal{T}^r$ and $z \notin \pi_r(C)$. Then there exists $y \in \mathcal{S}$ such that $\pi_r(y) = z$ and $(s \circ \pi_r(x), y) \in \mathcal{T}$ (by Lemma 1.1.). Moreover, there exists a path from x to $s \circ \pi_r(x)$ (Remark 1), so $s \circ \pi_r(x) \in C$ (C is a terminal strongly connected component). This implies that $y \in C$, and $z = \pi_r(y) \in \pi_r(C)$, which is a contradiction. As a consequence, all trajectories starting in $\pi_r(C)$, projection of the complex attractor C, are trapped in $\pi_r(C)$. Thus $\pi_r(C)$

contains at least one non trivial attractor, recalling that stable states are conserved and C does not contain any of them.

□

E Theorem 2

Theorem 2. *Given a non-trivial attractor C^r in $\mathcal{E}^r$,*

1. *$s_r(C^r) = \{s_r(z),\, z \in C^r\}$ is part of a strongly connected component C in $\mathcal{E}$.*
2. *Let $C' = \{x \text{ s.t. } \pi_r(x) \in C^r\}$ $(= \cup_{z \in s_r(C^z)}[z]_{\sim r})$, the set of states which projections are in C^r. Suppose $(x, y) \in \mathcal{T}$, such that $x \in C'$ and $y \notin C'$, then (x, y) is not preserved.*
3. *Suppose $(x, y) \in \mathcal{T}$, such that $x \in s_r(C^r)$ and $y \notin C' \cap C$, then (x, y) is not preserved.*

Proof. Consider C^r a non-trivial attractor in $\mathcal{E}^r$, then C^r is a strongly connected component.

1. Let $x, y \in s_r(C^r)$; then $x = s_r(z)$ and $y = s_r(z')$, $z, z' \in C^r$. Let $z^0 = z, z^1, z^2, \ldots, z^l = z'$ a path from z to z'. For all $i \in \{1, \ldots l-1\}$, $(z^i, z^{i+1}) \in \mathcal{T}^r \Rightarrow \exists u \in \mathcal{S}$ such that $(s_r(z^i), u) \in \mathcal{T}$ and $\pi_r(u) = z^{i+1}$. Moreover, we know that there exists a path from u to $s_r \circ \pi_r(u) = s_r(z^{i+1})$. Hence, there is a path from $x = s_r(z)$ to y.
2. Let $(x, y) \in \mathcal{T}$, such that $x \in C'$ and $y \notin C'$. Suppose that (x, y) is preserved, then either $\pi_r(x) = \pi_r(y)$ or $(\pi_r(x), \pi_r(y)) \in \mathcal{T}^r$. Both cases imply $\pi_r(y) \in C^r$, thus $y \in C'$ which is a contradiction.
3. Let $(x, y) \in \mathcal{T}$, such that $x \in s_r(C^r)$ and $y \notin C' \cap C$. Suppose that (x, y) is preserved, then either $\pi_r(x) = \pi_r(y)$ or $(\pi_r(x), \pi_r(y)) \in \mathcal{T}^r$.
   - $\pi_r(x) = \pi_r(y)$ is not possible because it would imply that $y \sim_r x$ and since x is a representative state $(x \in_r (C^r))$, it is stable for r and cannot be the source of a transition leading to a state in its equivalence class.
   - if $(\pi_r(x), \pi_r(y)) \in \mathcal{T}^r$, thus $\pi_r(y) \in C^r$ and $y \in C'$. Furthermore, $s_r \circ \pi_r(y) \in s_r(C^r) \subset C$. A path exists from $x \in C$ to y (since $(x, y) \in \mathcal{T}$) and from y to $s_r \circ \pi_r(y) \in C$ (by item 2 of Remark 1), thus $y \in C$, which is a contradiction.

□

Chapitre 8

Développement de GINsim : un logiciel de modélisation logique

8.1 Présentation de GINsim

GINsim est un logiciel dédié à la modélisation et l'analyse de réseaux de régulation. Développé dans l'équipe, ce logiciel implémente le formalisme logique présenté dans la section 2.2. Ce logiciel utilise un format XML pour la représentation des graphes de régulation : GINML (Gene Interaction Network Markup Language). Avant de débuter ma thèse, j'ai participé à l'élaboration d'un prototype intégrant un éditeur de graphes de régulation utilisant le format GINML, ainsi qu'un moteur de simulation en modes synchrone et asynchrone. La version actuelle de GINsim est basée sur ce prototype et sert de plateforme pour l'implémentation de nouvelles fonctionnalités.

GINsim est écrit en Java[1] et utilise les bibliothèques JGraph[2] et JGraphT[3] pour la définition, la visualisation et l'édition de graphes. Un mécanisme d'extensions (plugins) facilite l'ajout de nouvelles fonctionnalités. La version 2.3 est présentée dans [79], inséré dans la section 8.4. Cette version intègre la plupart des outils présentés ici : les classes de priorités pour la simulation (section 7.1), la recherche d'états stables (section 6.2) et l'analyse des circuits de régulation (section 6.3). Elle contient également des modules d'export vers plusieurs formats (voir section 8.3). La section 8.5 liste les nouveautés implémentées dans la version de développement, dont notamment la méthode de réduction présentée dans la section 7.2.

La fenêtre principale de GINsim permet d'éditer interactivement un graphe de régulation : l'utilisateur peut ajouter, supprimer et paramétriser des composants de régulation et des interactions (voir figure 8.1). Les plugins sont accessibles dans les menus "Actions" et "Fichier/export".

8.2 Architecture

GINsim utilise une architecture modulaire dans laquelle la plupart des fonctionnalités sont fournies par des extensions ("plugins" ou "greffons"). Le coeur du logiciel est constitué de classes génériques pour la représentation de graphes, ainsi que pour la lecture et l'écriture

[1] java.sun.net
[2] www.jgraph.com
[3] www.jgrapht.org

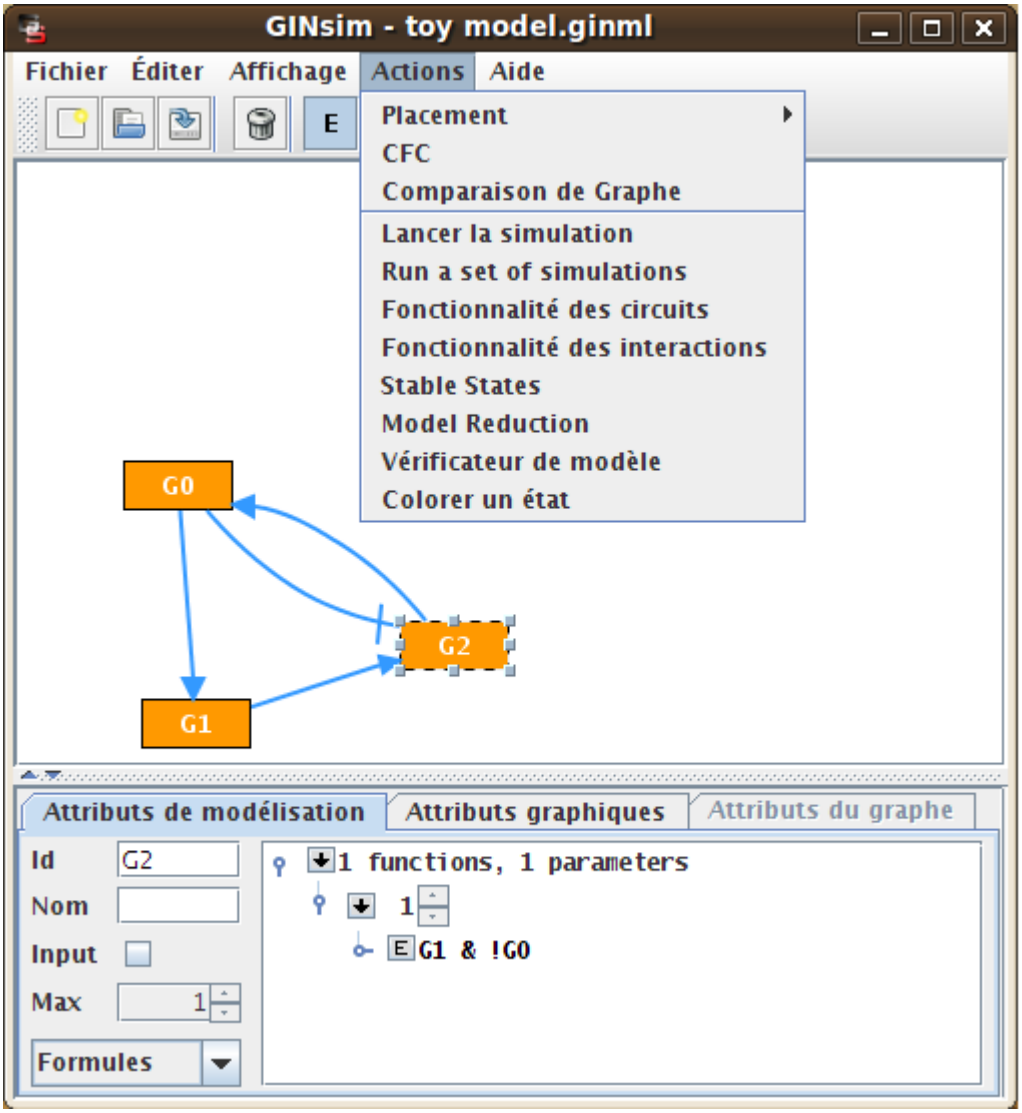


Figure 8.1: La fenêtre principale de GINsim, montrant un graphe de régulation. Les propriétés du composant sélectionné peuvent être éditées dans le panneau du bas (ici on voit le panneau d'édition des fonctions logiques). Le menu "action" permet de lancer des analyses ou des simulations sur ce graphe.

de documents XML. Au lancement, GINsim consulte une liste de plugins à charger. Le chargement d'un plugin consiste à créer un objet de la classe spécifiée et à lancer sa méthode de chargement. Cette méthode enregistre les fonctionnalités du plugin auprès du coeur de GINsim. Les plugins peuvent fournir un large éventail de fonctionnalités :

- Nouveau type de graphe : le coeur de GINsim contient une classe abstraite permettant de stocker et d'afficher une structure de graphe générique. Chaque type de graphe dérive de cette classe et ajoute les objets utilisés comme noeuds et arcs ainsi que les panneaux d'édition des propriétés de la sélection. Nous distinguons les graphes de régulation, les graphes de transitions d'états et les graphes des composantes fortement connexes.
- Actions : le menu action de la fenêtre principale liste les actions qui peuvent s'appliquer au graphe courant. Ces actions peuvent être génériques, comme la recherche de composantes fortement connexes, ou spécifiques à un type de graphe, comme la simulation et la recherche d'états stables pour le graphe de régulation. La sélection d'une de ces entrées de menu déclenche l'appel d'une méthode du plugin l'ayant enregistré.
- Exports : de manière similaire aux actions, un menu export liste les formats vers lesquels le graphe courant peut être exporté (voir section 8.3).
- Les plugins peuvent également associer des données au graphe courant. Par exemple les définitions de mutants, paramètres de simulation ou liste des composants à masquer. Ces données sont identifiées par leur nom et peuvent être utilisées par les autres plugins. Elles peuvent également être sauvegardées avec le graphe, à condition d'utiliser la sauvegarde étendue. Le fichier de sauvegarde est alors un dossier compressé (au format zip) contenant un fichier GINML pour le graphe et un fichier par objet associé. Dans ce cas, le plugin est chargé de la sauvegarde et de l'ouverture du fichier correspondant.

8.3 Lien avec d'autres outils

Afin d'éviter de développer des outils existants par ailleurs, GINsim permet d'exporter les graphes dans les formats utilisés par d'autres outils. Ces exports permettent d'utiliser divers outils de visualisation ou de manipulation de graphes, mais aussi d'autres outils d'analyse ou de modélisation.

De manière générale, une représentation graphique du graphe peut être obtenue aux formats PNG ou SVG. Par exemple la plupart des figures montrant des modèles dans ce document ont été obtenues à partir de l'export SVG, avant d'être transformé en PDF à l'aide d'Inkscape[4]. De même, la structure du graphe peut être exportée vers des outils de visualisation de graphe comme Graphviz[5] ou BioLayout[6]. Les graphes peuvent également être exportés vers le format de l'outil de manipulation de graphes Cytoscape[7].

Un graphe de régulation peut être traduit en réseau de Petri standard ou coloré, selon la méthode présentée dans [19] (inséré en annexe A.3). Seule la traduction en réseau de Petri standard est actuellement implémentée. Dans le réseau de Petri obtenu, chaque composant est représenté par deux places. Les jetons présents dans la première place indiquent le niveau courant du composant associé. Le nombre de jetons dans la seconde place (dite place complémentaire) est égal au niveau maximal du composant moins son niveau courant. Les fonctions logiques sont représentées par des transitions permettant de faire passer un jeton d'une place à l'autre en fonction du nombre de jetons dans les places associées aux

[4] www.inkscape.org
[5] www.graphviz.org
[6] www.biolayout.org
[7] www.cytoscape.org

régulateurs de ces composants. Chacune de ces transitions correspond à un chemin dans le MDD du composant concerné (voir [19], figure 2). Les réseaux ainsi obtenus peuvent être sauvegardés dans plusieurs formats de description de réseaux de Petri : PNML, APNN et PNT.

Les graphes de régulation peuvent également être traduits dans le format du model-checker NuSMV[8]. Il est alors possible de spécifier et de vérifier des propriétés Computation Tree Logic (CTL). L'utilisation de NuSMV a donné des résultats mitigés : de légères variations du modèle font passer le temps de calcul de quelques secondes à plusieurs jours.

Enfin, nous proposons un export vers le format du logiciel de modélisation GNA[9]. Celui-ci utilise des équations différentielles linéaires par morceaux pour la modélisation qualitative de réseaux de régulation génétiques [29]. L'utilisation de ce type de modèle peut être vue comme un raffinement d'un modèle logique.

Nous avons également travaillé sur le support du format d'échange de modèles SBML[10] (Systems Biology Markup Language). Ce format est bien adapté aux modèles de réseaux biochimiques continus mais n'a pas été conçu pour les modèles qualitatifs ou discrets. Nous avons un prototype d'export vers SBML Level 2 (la version actuelle) détournant la sémantique associée à certaines balises de SBML. Nous participons également à la création d'une extension de SBML Level 3 dédiée aux modèles discrets (modèles logiques et réseaux de Petri).

[8] nusmv.irst.itc.it
[9] www-helix.inrialpes.fr/gna
[10] sbml.org

8.4 Article sur GINsim (BioSystem 2009)

BioSystems 97 (2009) 134–139

Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

Logical modelling of regulatory networks with GINsim 2.3

A. Naldi [a], D. Berenguier [a], A. Fauré [a], F. Lopez [a], D. Thieffry [a,b,*], C. Chaouiya [a,c,*]

[a] TAGC - INSERM U928 - Université de la Méditerranée, Campus de Luminy, Case 929, F-13288 Marseille Cedex 9, France
[b] CONTRAINTES project, INRIA Paris-Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay, France
[c] Instituto Gulbenkian de Ciência, Rua da Quinta Grande, P-2780-156 Oeiras, Portugal

ARTICLE INFO

Article history:
Received 9 February 2009
Received in revised form 22 April 2009
Accepted 25 April 2009

Keywords:
Regulatory networks
Logical modelling
Dynamical simulation
Computational systems biology

ABSTRACT

Many important problems in cell biology require the consideration of dense nonlinear interactions between functional modules. The requirement of computer simulation for the understanding of cellular processes is now widely accepted, and a variety of modelling frameworks have been designed to meet this need. Here, we present a novel public release of the Gene Interaction Network simulation suite (*GINsim*), a software designed for the qualitative modelling and analysis of regulatory networks. The main functionalities of *GINsim* are illustrated through the analysis of a logical model for the core network controlling the fission yeast cell cycle. The last public release of *GINsim* (version 2.3), as well as development versions, can be downloaded from the dedicated website (http://gin.univ-mrs.fr/GINsim/), which further includes a model library, along with detailed tutorial and user manual.

1. Introduction

Discrete (logical) formal methods provide a suitable framework to integrate heterogeneous molecular data into rigorous dynamical models, which can be used as a basis for the development of more quantitative models (e.g. using differential or stochastic equations). During the last decade, logical modelling has been applied to various types of biological regulatory systems, including flower morphogenesis (Mendoza et al., 1999), Drosophila embryonic development (González et al., 2008; Sánchez et al., 2008; Chaves et al., 2005), cell cycle control (Fauré et al., 2006; Sahin et al., 2009; Irons, 2009; Abou-Jaoudé et al., 2009), immune responses (Stark et al., 2007 and references therein), in particular T lymphocyte activation and differentiation (Mendoza, 2006; Saez-Rodriguez et al., 2007).

In brief, the logical approach considered here involves the delineation of:

- the crucial regulatory components and their different ranges of activity;
- the main interactions and associated activity ranges;
- the logical rules directing the activity of each network component, depending on incoming interactions;
- the relevant time constraints underlying specific dynamical behaviours.

The dynamics of such models are naturally described by state transition graphs. Further details on this logical formalism can be found in Thomas et al. (1995) and Chaouiya et al. (2003). Although generally considered sound and well adapted to the qualitative modelling of complex biological regulatory networks, this approach is still scarcely used by biologists because of the lack of accessible and efficient computational tools. The development of the Gene Interaction Network simulation suite (*GINsim*) aims at filling this gap by providing a user-friendly logical modelling software, based on the intuitive notion of regulatory graph.

GINsim encompasses three main components:

- a graphical user interface;
- a simulation core (construction of state transition graphs);
- a graph analysis toolbox.

Developed in Java, *GINsim* is based on public graph libraries, *JGraph* and *JGraphT*, and is designed in a modular way to ease extensions and collaborative developments. A previous paper presented an earlier version of the software (González et al., 2006). Since then, *GINsim* has considerably evolved and this software application note aims at describing the most notable characteristics of the new public release, *GINsim 2.3*. For a more detailed presentation, we refer to the tutorial or to the user manual, both available along with the current stable and development versions of *GINsim*.

2. Logical Model Definition

The *GINsim* main window comprises the usual file management, edition and visualisation tools. Clickable icons and menus provide

* Corresponding authors.
E-mail addresses: thieffryg@tagc.univ-mrs.fr (D. Thieffry), chaouiyag@igc.gulbenkian.pt (C. Chaouiya).

0303-2647/$ – see front matter © 2009 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2009.04.008

a friendly access to the functionalities enabling the definition of a regulatory graph (node and arrow drawing, cutting and pasting, etc.). Part of these functionalities is similar to those of most simple drawing tools. For each node of a regulatory graph, the user specifies the range of discrete activity levels (the associated variable takes its values in this range) and the logical parameters (defining the target levels of activity related to sets of incoming interactions). By default, logical parameters and variables are considered Booleans, but can be progressively refined by the modeller as his understanding of the logical formalism and of the properties of the model under development increases. Logical parameters are easily defined by selecting interactions and associating them with discrete values. For each interaction, an interval specifies the range of values for which this interaction is operating (by default, the interval is set to the whole range of activity of the source node, except zero). Graphical settings of nodes and interactions can be modified. *GINsim* also provides means to store textual information supporting a given model, either at the level of the whole regulatory graph, or at the level of each of its elements (nodes or interactions).

Models are stored in a specific XML file format, GINML, but can also be exported into various formats, including PNG (image), SVG, Graphviz and BioLayout (graphs), as well as Cytoscape (xgmml) formats. Thanks to the many plugins available for Cytoscape (e.g. BiNoM, which handles CellDesigner, BioPAX and SBML formats (Zinovyev et al., 2008)), regulatory and state transition graphs defined with *GINsim* can now be converted into virtually any relevant format, thereby enabling the combination of various software tools with complementary functionalities. Finally, this novel public release encompasses the export of logical models towards Petri nets, following the procedure described by Chaouiya et al. (2008).

Fig. 1 shows an example of a regulatory graph corresponding to the core oscillator controlling the fission yeast cell cycle, derived from the Boolean model published by Davidich and Bornholdt (2008). The two Boolean components representing the different levels of activity of Cdc2/Cdc13 have been replaced by a unique ternary component. In addition, the loops originally placed on Start, SK, PP, and Slp1 nodes have been removed, as they do not represent true auto-regulations. Finally, the logical rules governing the behaviours of these components have been reformulated to fit Thomas' framework (Thomas et al., 1995; Chaouiya et al., 2003)(for a complete listing of these rules, see the corresponding entry in the model repository).

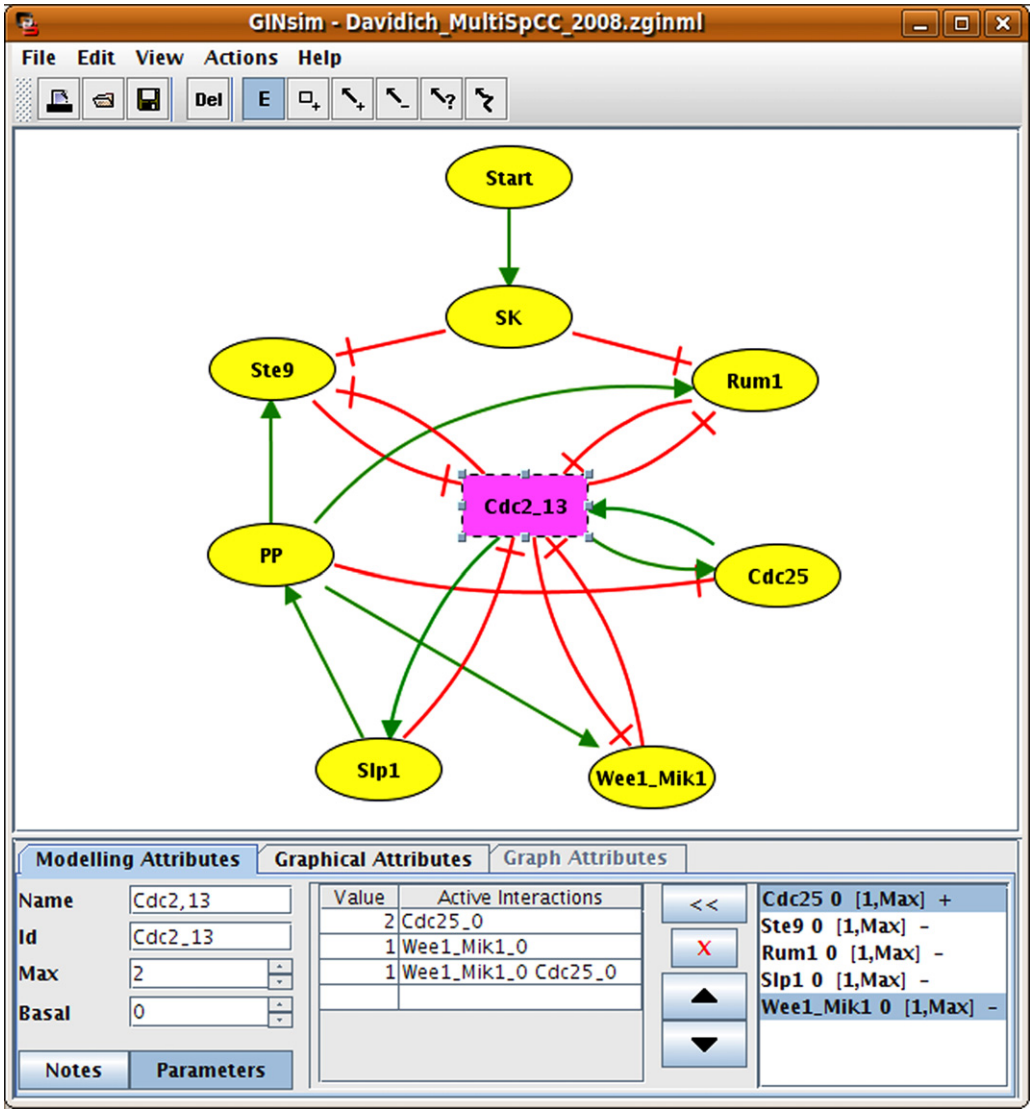


Fig. 1. Main *GINsim* window displaying the regulatory graph for the fission yeast cell-cycle control as derived from Davidich and Bornholdt (2008). The lower panel shows the parameters attached to the selected node (Cdc2_13): maximal level (here 2), basal value (0), five incoming interactions (bottom right), and the logical parameters (bottom center), i.e. target values for combinations of incoming interactions (those which are not listed lead, by default, to value 0).

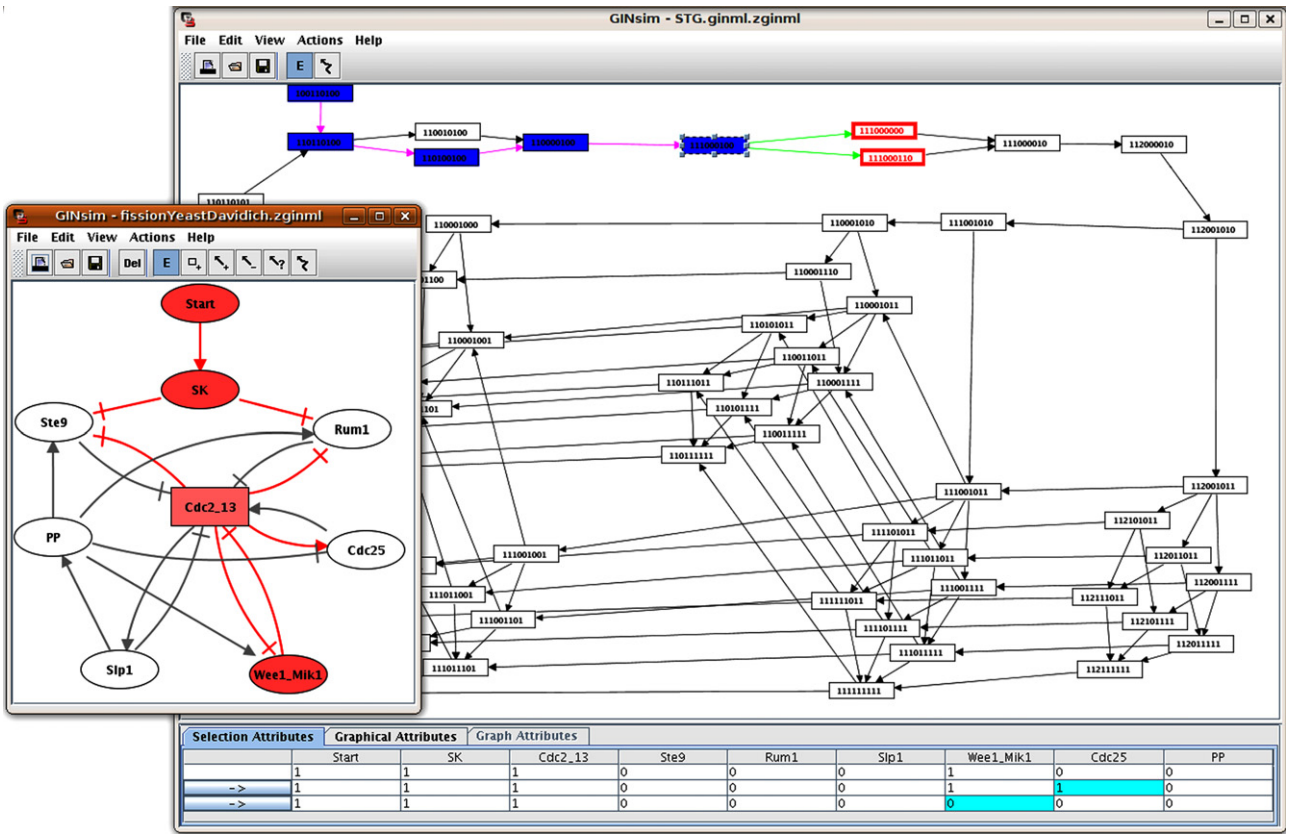


Fig. 2. *GINsim* windows showing a State Transition Graph (STG, *cf.* right panel) resulting from a simulation of the model of Fig. 1 using priority classes (lower priorities for the decreases of Start, SK, Slp1 and PP). A path can be interactively selected in the STG and the active nodes of the model are coloured according to the selected state (left panel).

3. Simulation and Analysis

On the basis of the definition of a regulatory graph and of the associated logical parameters, *GINsim* computes the temporal evolution of the system, which is represented in terms of a *State Transition Graph* (STG). In this graph, each node represents a state (giving the discrete level of each regulatory component), and arrows represent spontaneous transitions (see the sample of simulation results provided in Fig. 2).

Briefly, performing a simulation implies:

- the definition of a single initial state, or set of initial states, through the specification of initial value(s) for each variable;
- the selection of an updating assumption: the classical synchronous or asynchronous modes, or yet mixed modes, relying on the definition of priority classes (Fauré et al., 2006);
- the selection of perturbations (mutant conditions), defined as the blocking of the values of a (set of) variable(s) within a restricted interval (e.g. the definition of a gene knockdown amounts to block its level to zero);
- optionally, the specification of a maximal depth or graph size, or yet of a graph construction strategy (depth or breadth first).

It is possible to define a series of simulation settings (initial conditions, perturbations, updating policies) and store them together with the model. More precisely, *GINsim* creates a compressed archive containing both the model and the simulation settings.

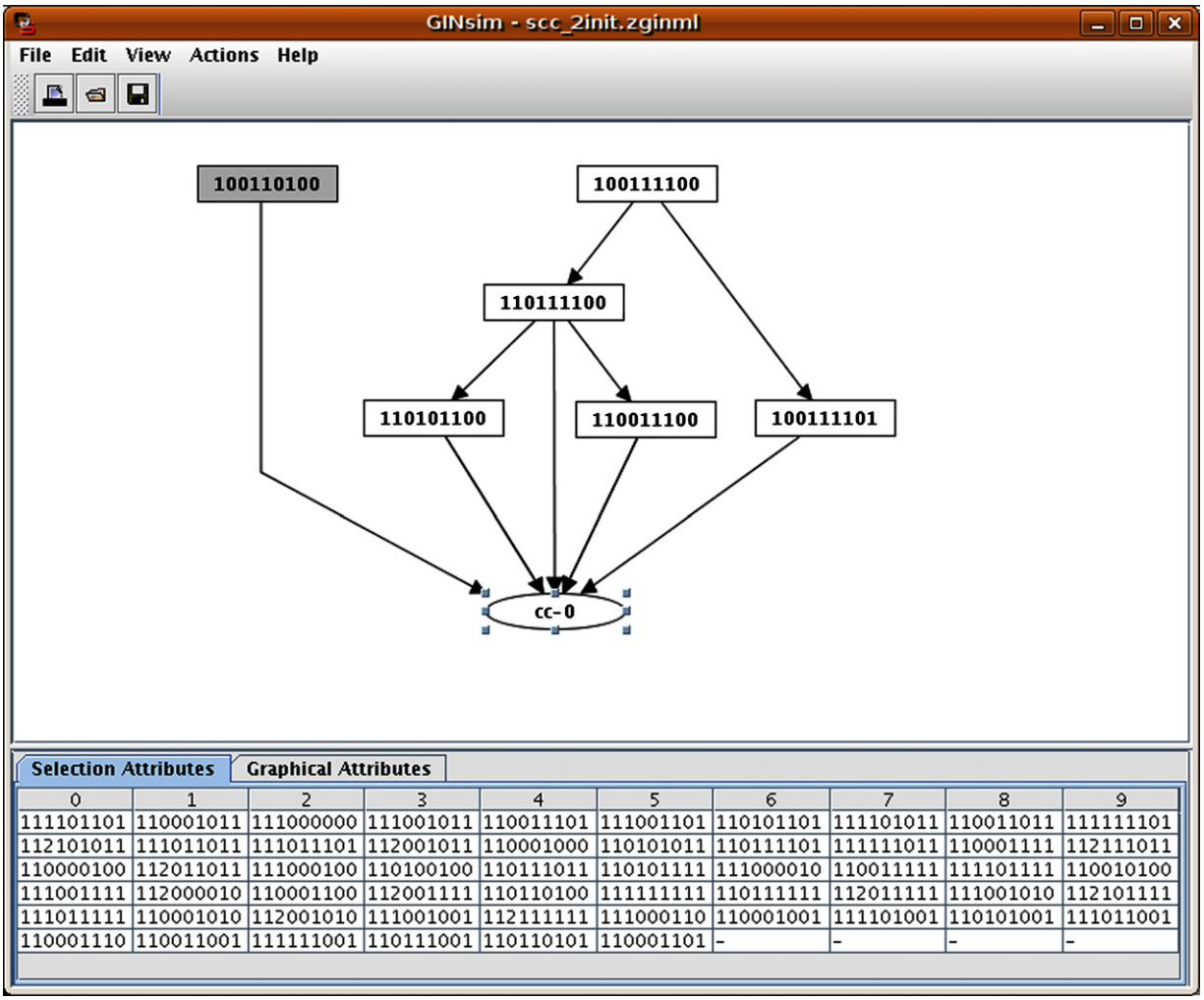
Once a STG has been computed, the user can select a relevant trajectory and export it into a gnuplot file that enables the generation of time plots. Moreover, graph tools are available to analyse and inspect STGs (*cf.* Fig. 3).

Since logical models and simulations rely on graph definitions implemented using standard Java graph libraries, existing graph analysis libraries or original algorithms can be easily integrated in *GINsim*, taking advantage of its modular architecture. In this respect, the current release encompasses tools to:

- identify stable states (terminal nodes in the STG), which can be displayed in a separated panel;
- compute the strongly connected components of regulatory graphs or of STGs, which can then be displayed and further analysed;
- search paths (shortest path between user-defined states);
- visualise state transitions interactively, either within a STG or on the underlying regulatory graph; an automatic colouring of nodes and arrows facilitates stepwise explorations of network dynamics, as well as their recording and further automatic display.

The computation, let-alone the visualisation, of STGs is often intractable, as their sizes grow exponentially with the number of model components. In this respect, *GINsim 2.3* enables an efficient computation of all stable states, without constructing the STG (Naldi et al., 2007).

The application of this method to the model presented in Fig. 1 leads to the identification of a single stable state, corresponding to the G1 state found by Davidich and Bornholdt, with only Ste9, Rum1 and Wee1/Mick1 activated. This means that the other 11 spurious stable states obtained by these authors have been eliminated. Using a synchronous updating, sustained activation of Start leads to SK activation and then to inhibition of Ste9 and Rum1, launching a sequence of state transitions matching that defined by Davidich and Bornholdt, as well as available kinetic data (see Model Documentation for more details about the configuration of the logical



0	1	2	3	4	5	6	7	8	9
111101101	110001011	111000000	111001011	110011101	111001101	110101101	111101011	110011011	111111101
112101011	111011011	111011101	112001011	110001000	110101011	110111101	111111011	110001111	112111011
110000100	112011011	111000100	110100100	110111011	110101111	111000010	110011111	111101111	110010100
111001111	112000010	110001100	112001111	110110100	111111111	110111111	112011111	111001010	112101111
111011111	110001010	112001010	111001001	112111111	111000110	110001001	111101001	110101001	111011001
110001110	110011001	111111001	110111001	110110101	110001101	-	-	-	-

Fig. 3. Graph of the Strongly Connected Components (SCCs) of a STG obtained in the same conditions used for Fig. 2 but considering an additional initial state where Slp1 is also present (the initial state considered in Fig. 2 is greyed out). This graph provides a more compact representation of the complex dynamics by hiding oscillations. In such graphs, attractors (stable states or complex attractors) appear as terminal nodes. Here, the 57 states STG of Fig. 2 has a single complex attractor encompassing 56 states (listed in the lower panel). The additional initial state leads to the same attractor, through some transient states.

simulation). If the Start component is permanently activated, the model gives rise to sustained oscillations.

At this point, the logical approach further enables a systematic analysis of the dynamical roles of the regulatory circuits embedded in the network, depending on the values of input nodes (Thomas et al., 1995). *GINsim 2.3* implements algorithms enabling the decomposition of complex regulatory networks into elementary circuits, as well as the analysis of their functionality contexts (see Naldi et al., 2007 for details on the method).

Turning back to the model presented in Fig. 1, these algorithms lead to the identification of nine regulatory circuits. As shown in Fig. 4, the feedback circuit analysis emphasizes the dynamical roles of four of these circuits (each involving two components):

- a negative circuit: Cdc2_13/Slp1;
- three positive circuits: Cdc2_13/Rum1, Cdc2_13/Ste9, and Cdc2_13/Cdc25.

From a dynamical point of view, the negative circuit enables the generation of oscillations, whereas the positive circuits may contribute to amplify them and to ensure irreversibility of transitions between cell phases.

In *GINsim*, for each circuit, the analysis of its functionality further computes its sign together with the constraints on the values of the components acting on this circuit that enable the corresponding dynamical property. For example, the negative circuit is functional only in the absence of both Ste9 and Rum1. In other words, in the presence of one of these factors, the system cannot exhibit sustained oscillations. This would be the case, for example, in a mutant with constitutive activity of one of these factors.

Finally, *GINsim 2.3* provides straightforward means to simulate various kinds of mutants or perturbations and to check their dynamical effects. For example, Fig. 5 illustrates the definition of a mutant with constitutive Ste9 activity. For initial conditions with both Start and Ste9 blocked at value 1, the STG is devoid of any cycle. The simulation of a mutant with constitutive Rum1 activity leads to a similar result.

4. Discussion and Prospects

GINsim provides an intuitive and versatile environment to define qualitative models for biological regulatory networks and to probe their dynamical properties. In summary, the new release presented here provides improved user interfaces and performances for both model definition and analyses. It implements new exports facilities, a simulation mode using priority classes and the possibility to store simulation parameters as well as mutant specifications

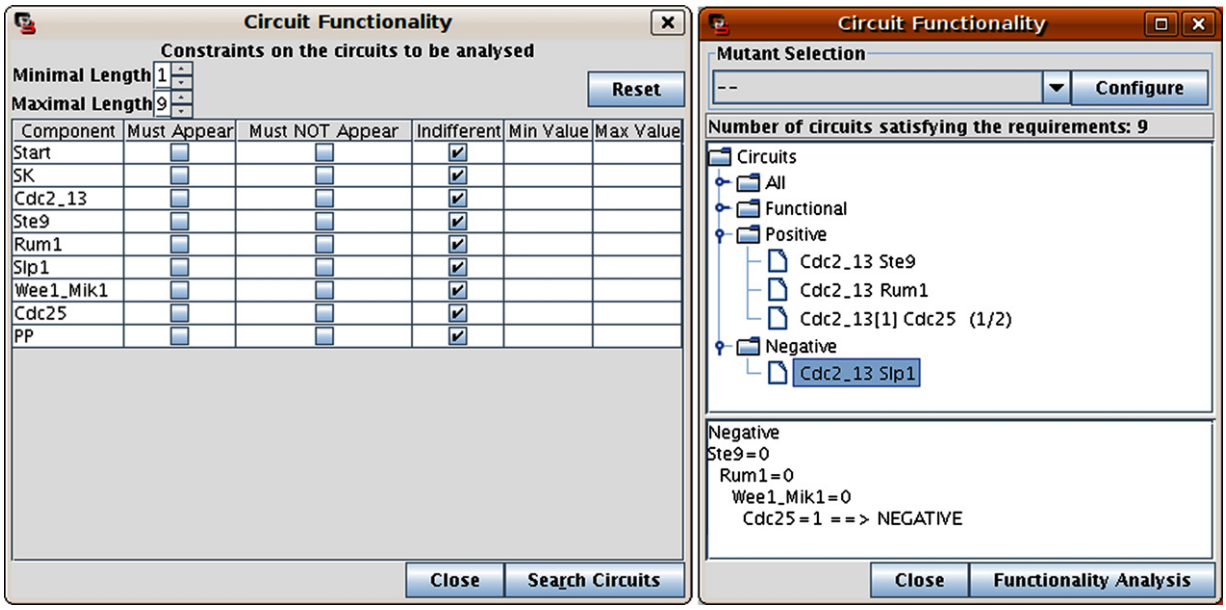


Fig. 4. The circuit functionality tool enables the search of all elementary circuits matching some criteria (length, components, *cf.* dialog box on the left) in a model. The functionality contexts of the corresponding circuits are then computed, enabling the classification of circuits as positive and negative. The application of this tool to the model of Fig. 1 identified 9 circuits, among which 4 are found functional (*cf.* right panel), the functionality context of the selected circuit is displayed at the bottom.

along with a model. Original algorithms have been implemented to identify all potential stable states and to analyse regulatory circuit functionality.

As the use of the logical modelling spreads into the systems biology community, various softwares become available, e.g. DDLab (Wuensche, 1998), CellNetAnalyzer (Klamt et al., 2007), SQUAD (Cara et al., 2007), ChemChains (Helikar et al., 2008), BooleanNet (Albert et al., 2008). In this context, the main assets of *GINsim* are the support of multilevel modelling and of various updating schemes (synchronous and asynchronous modes, as well as mixed or intermediate modes) and, above all, an original and powerful regulatory circuit analysis tool.

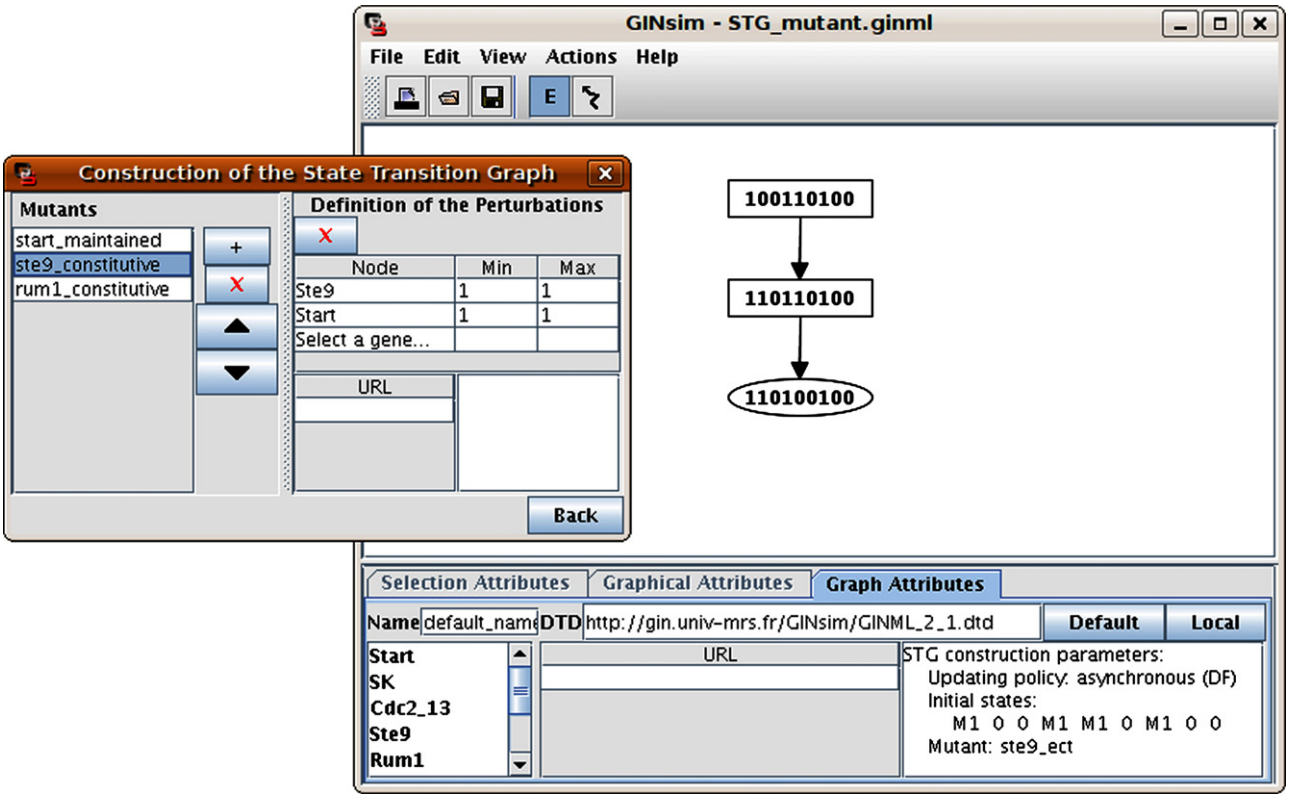


Fig. 5. *GINsim* enables the definition of simple or multiple perturbations (mutants) along with a model. Such perturbations are defined using the dialog box shown on the left. The left part of the dialog box lists the defined perturbations, while the right part enables the edition of the selected one, here a constitutive expression of Ste9. The window on the left shows the STG obtained with this perturbation, all other parameters (updating, initial state) being the same as in Fig. 2.

Unlike DDLab (Wuensche, 1998), GINsim does not handle Random Boolean Networks, but rather parametrised multi-level models, generally analysed under an asynchronous updating. Hence, statistics on the attractors and their bassins of attraction would rely on distinct data in our context. To quantify the trajectories leading to given attractors, we are currently considering the implementation of a compact, hierarchical, representation of state transition graphs, which should greatly ease such analyses.

It is worth noting that *GINsim* is constantly updated as new requirements emerge and novel algorithms are defined. Current developments mainly aim at easing the definition and analysis of larger models, including:

- a parser to support more compact logical rules;
- model composition tools;
- model reduction by "hiding" components;
- automation of graph comparison and merging;
- additional import and export facilities, to enable model exchanges with tools such as Gene Network Analyser (GNA) (de Jong et al., 2003), model checkers (NuMSV), or yet logical and constraints programming environments;
- compact representation of the dynamics, enabling to identify parts of the phase space leading to each attractors;
- scripting facilities allowing to define and launch series of simulations.

The issue of model export is particularly relevant for the modeller willing to apply the logical framework as an initial step to build more quantitative models. However, the reference Systems Biology Modelling Language (SBML) does not yet offer a straightforward format to encode qualitative models. To solve this problem, we are presently engaged in an international collaborative effort aiming at defining an extension dedicated to qualitative models in the context of the development of SBML Level 3.

Acknowledgements

A.N. has been supported by PhD grant from the French Ministry of Research and Technology. This project was further supported by research grants from the EU FP6 (DIAMONDS STREP) and FP7 (APOSYS large scale project) programs, and by the Belgian Science Policy Office (IAP BioMaGNet).

References

Abou-Jaoudé, W., Ouattara, D.A., Kaufman, M., 2009. From structure to dynamics: frequency tuning in the p53-mdm2 network I. Logical approach. J. Theor. Biol. 258, 561–577.

Albert, I., Thakar, J., Li, S., Zhang, R., Albert, R., 2008. Boolean network simulations for life scientists. Source Code Biol. Med. 3, 16.

Cara, A.D., Garg, A., Micheli, G.D., Xenarios, I., Mendoza, L., 2007. Dynamic simulation of regulatory networks using squad. BMC Bioinform. 8, 462.

Chaouiya, C., Remy, E., Mossé, B., Thieffry, D., 2003. Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework. Lect. Notes Control Inform. Sci. (LNCIS) 294, 119–126.

Chaouiya, C., Remy, E., Thieffry, D., 2008. Petri net modelling of biological regulatory networks. J. Discrete Algorithms 6, 165–177.

Chaves, M., Albert, R., Sontag, E.D., 2005. Robustness and fragility of boolean models for genetic regulatory networks. J. Theor. Biol. 235, 431–449.

Davidich, M., Bornholdt, S., 2008. Boolean network model predicts cell cycle sequence of fission yeast. PLoS ONE 3, e1672.

de Jong, H., Geiselmann, J., Hernandez, C., Page, M., 2003. Genetic network analyzer: qualitative simulation of genetic regulatory networks. Bioinformatics 19, 336–344.

Fauré, A., Naldi, A., Chaouiya, C., Thieffry, D., 2006. Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. Bioinformatics 22, 124–131.

González, A., Chaouiya, C., Thieffry, D., 2006. Dynamical analysis of the regulatory network defining the dorsal-ventral boundary of the drosophila wing imaginal disk. Genetics 174, 1625–1634.

González, A., Chaouiya, C., Thieffry, D., 2008. Logical modelling of the role of the hh pathway in the patterning of the drosophila wing disc. Bioinformatics 24, i234–i240.

Helikar, T., Konvalina, J., Heidel, J., Rogers, J.A., 2008. Emergent decision-making in biological signal transduction networks. Proc. Natl. Acad. Sci. U.S.A. 105, 1913–1918.

Irons, D.J., 2009. Logical analysis of the budding yeast cell cycle. J. Theor. Biol. 257 (4), 543–559.

Klamt, S., Saez-Rodriguez, J., Gilles, E.D., 2007. Structural and functional analysis of cellular networks with CellNetAnalyser. BMC Syst. Biol. 1, 2.

Mendoza, L., 2006. A network model for the control of the differentiation process in Th cells. Biosystems 84, 101–114.

Mendoza, L., Thieffry, D., Alvarez-Buylla, E., 1999. Genetic control of flower morphogenesis in arabidopsis thaliana: a logical analysis. Bioinformatics 15, 593–606.

Naldi, A., Thieffry, D., Chaouiya, C., 2007. Decision diagrams for the representation of logical models of regulatory networks. Lecture Notes Comput. Sci. (LNCS) 4695, 233–247.

Saez-Rodriguez, J., Simeoni, L., Lindquist, J.A., Hemenway, R., Bommhardt, U., Arndt, B., Haus, U.-U., Weismantel, R., Gilles, E.D., Klamt, S., Schraven, B., 2007. A logical model provides insights into T cell receptor signaling. PLoS Comput. Biol. 3, e163.

Sahin, O., Fröhlich, H., Löbke, C., Korf, U., Burmester, S., Majety, M., Mattern, J., Schupp, I., Chaouiya, C., Thieffry, D., Poustka, A., Wiemann, S., Beissbarth, T., Arlt, D., 2009. Modeling ERBB receptor-regulated G1/S transition to find novel targets for de novo trastuzumab resistance. BMC Syst. Biol. 3, 1.

Sánchez, L., Chaouiya, C., Thieffry, D., 2008. Segmenting the fly embryo: logical analysis of the role of the segment polarity cross-regulatory module. Int. J. Dev. Biol. 52, 1059–1075.

Stark, J., Chan, C., George, A.J.T., 2007. Oscillations in the immune system. Immunol. Rev. 216, 213–231.

Thomas, R., Thieffry, D., Kaufman, M., 1995. Dynamical behaviour of biological regulatory networks-i. biological role of feedback loops and practical use of the concept of the loop-characteristic state. Bull. Math. Biol. 57, 247–276.

Wuensche, A., 1998. Genomic regulation modeled as a network with basins of attraction. Pac. Symp. Biocomput., 89–102.

Zinovyev, A., Viara, E., Calzone, L., Barillot, E., 2008. Binom: a Cytoscape plugin for manipulating and analyzing biological networks. Bioinformatics 24, 876–877.

8.5 Bientôt dans GINsim

La méthode de réduction présentée dans la section 7.2 est implémentée dans la branche de développement de GINsim et doit faire partie de la prochaine version. Une ou plusieurs listes de composants à masquer peuvent être sauvegardées avec le modèle. Ceci permet d'appliquer facilement la même réduction à plusieurs versions du modèle. La méthode de réduction d'un unique composant est appliquée itérativement sur tous les composants listés. Les interactions indirectes dans le modèle réduit (passant par les composants masqués) sont inférées à partir des fonctions logiques.

D'autres nouvelles fonctionnalités visent à faciliter la définition de modèles.

La définition des interactions ne nécessite plus qu'un seuil au lieu de l'intervalle utilisé dans les versions précédentes. Le seuil de l'interaction suivante sert de borne supérieure implicite.

Nous avons également introduit une interface de définition de fonctions logiques (visible dans la figure 8.1). Cette interface coexiste avec celle de définition des paramètres logiques. L'utilisation de fonctions logiques est particulièrement utile dans le cas de composants ayant de nombreux régulateurs. De tels composants peuvent avoir plusieurs milliers de paramètres logiques. C'est le cas, par exemple, dans le modèle du cycle cellulaire de la levure bourgeonnante développé par Fauré et al. [35] (voir Annexe A.4).

Toujours dans l'optique de simplifier la définition de modèles, les composants peuvent désormais être marqués comme entrées. Ils ont alors une fonction logique implicite empêchant leur niveau de changer, ce qui évite de leur assigner une auto-régulation artificielle. Il est alors possible de définir des combinaisons d'entrées indépendamment des états initiaux internes, ce qui facilite la définition d'états initiaux combinant états internes et conditions environnementales. Ces combinaisons d'entrées sont par exemple utilisées pour la définition des différents environnements auxquels sont soumises les cellules T dans le chapitre 9.

GINsim s'assure qu'une interaction ne peut intervenir dans la fonction logique que si elle est présente dans le graphe mais n'empêche pas de définir une interaction dans le graphe sans lui donner de rôle réel dans les fonctions logiques. Il est donc possible de définir un modèle incohérent car certaines interactions ne jouent aucun rôle. Ceci est courant pendant la construction d'un modèle : les interactions sont ajoutées avant les fonctions logiques. En cas d'incohérence, lors des simulations et analyses du graphe de régulation, seules les fonctions logiques sont prises en compte. Autrement dit, le graphe sert uniquement d'aide visuelle pendant la conception du modèle. Afin de diagnostiquer ces situations, les fonctions logiques peuvent être confrontées aux interactions. Un nouvel outil permet ainsi de mettre en évidence les interactions ne jouant aucun rôle, ainsi que les incohérences de signe.

Il est également possible de documenter les modèles en associant des annotations à leurs composants et interactions. Ces annotations peuvent désormais contenir des liens vers des entrées de bases de données (par exemple "pubmed : <PMID>"). Il est facile d'ajouter de nouvelles bases de données dans le code en plus de celles déjà disponibles (PubMed, HUGO, GeneBank). Les annotations peuvent aussi correspondre à des références bibliographiques ("ref : <id>"). L'utilisation de références bibliographiques nécessite d'associer un fichier de bibliographie (au format BibTeX) au modèle, le lien peut alors ouvrir, selon leur disponibilité, le PDF associé ou une page web (PubMed ou DOI).

Enfin, certaines parties internes de GINsim ont été remaniées afin de fonctionner sans interface graphique. Ceci permet en particulier l'écriture de scripts à l'aide de Jython. Jython est une implémentation en Java du langage Python permettant l'utilisation de code Java. De tels scripts sont utilisés pour lancer des séries de simulations ou d'analyses sur un ou plusieurs modèles et la création de comptes rendus (en HTML, ODT ou LaTeX). Quelques exemples de scripts sont présentés dans l'annexe B.2. Ces scripts ont par exemple servi à implémenter les "simulations en cascade" appliquées au modèle présenté dans le chapitre 9.

Chapitre 9

Modèle étendu de différenciation Th

Le modèle présenté ici se base sur deux modèles existants traitant de la différenciation Th1/Th2 et de l'activation via le TCR (voir [70, 95] et chapitre 4). Nous avons intégré des composants et interactions supplémentaires tirés de la littérature (voir table récapitulative dans l'article et documentation du modèle en annexe) afin de prendre en compte les lignées cellulaires Th17 et Treg, récemment mises en évidence.

La construction de ce modèle a nécessité de nombreuses étapes de raffinement. Il faut tout d'abord identifier les composants. Certains d'entre eux étaient déjà présents dans les modèles de départ. Nous avons également choisi de montrer explicitement certains composants qui y étaient masqués (en particulier les sous-chaînes des récepteurs de cytokine). Parmi les nouveaux composants, foxp3, IL17 et RORγt sont bien connus pour leur rôle dans les lignées cellulaires Th17 et Treg. La plus grande partie du modèle porte sur les cytokines et leurs voies de signalisations. Nous avons également ajouté un composant représentant la prolifération cellulaire, déclenchée par IL-2, vraisemblablement nécessaire pour rendre les promoteurs des gènes codants pour les cytokines accessibles. Le modèle comporte 13 composants d'entrée représentant l'environnement auquel est soumise la cellule (cytokines et cellule présentatrice d'antigène).

L'analyse des circuits de ce modèle permet d'identifier 5 circuits positifs fonctionnels impliquant les principaux régulateurs des lignées cellulaires Th1, Th2, Th17 et Treg. Sur la base de ces circuits, on peut prédire 48 "signatures stables" potentielles concernant les composants impliqués dans ces circuits. Il reste alors à déterminer les conditions environnementales avec lesquelles ces signatures sont compatibles. Dans un premier temps, l'analyse des états stables élimine 5 de ces signatures. Nous avons ensuite réalisé une étude d'atteignabilité de ces états stables. Pour cela, nous avons effectué des simulations sur une version réduite du modèle pour une sélection de 8 conditions environnementales. Le modèle réduit comporte 34 composants, dont toujours 13 entrées, fixées pour une simulation donnée. En partant de l'état Th0 et en effectuant une simulation pour chaque environnement, le modèle prédit plusieurs états stables, correspondant aux lignées cellulaires attendues. Nous avons ensuite réutilisé ces états stables comme états initiaux pour une nouvelle série de simulations, et ce jusqu'à ce qu'aucun nouvel état stable ne soit atteint. Cette analyse montre une certaine plasticité dans les signatures cellulaires, en fonction des conditions environnementales.

9.1 Article présentant le modèle (en préparation)

Diversity and plasticity of Th cell types predicted from regulatory network modelling

Aurélien Naldi[1], Jorge Carneiro[2], Claudine Chaouiya[1,2], and Denis Thieffry[1,3]
[1] TAGC-INSERM U928, Marseille, France
[2] Instituto Gulbenkian de Ciencia, Oeiras, Portugal
[3] CONTRAINTES Project, INRIA Paris-Rocquencourt, France

2nd November 2009

1 Introduction

Alternative cell differentiation pathways are believed to arise from the concerted action of signalling pathways and transcriptional regulatory networks. However, the prediction of mammalian cell differentiation from the knowledge of the presence of specific signals and transcriptional factors is currently a daunting challenge. In this respect, the vertebrate hematopoietic system, with its many branching differentiation pathways and cell types, is a compelling case study. In particular, numerous publications describe molecular and genetic interactions involved in the control the late stages of helper CD4 T (Th) cell differentiation, and yet our current knowledge on cell lineage branching is clearly incomplete, as demonstrated by recent reports on novel Th cell types [16, 13, 41, 23]. The existence of some of these cell types was revealed following the identification of novel transcription factors [16] or cytokines [13, 41] However, several Th cell phenotypes recently described likely result from reassorted expression of already known genes [7, 23].

Beyond the expression of diverse and cell-specific antigen receptor genes, the appreciation of the heterogeneity of late Th cell lineages emerged from the characterisation of Th1 and Th2 cell types by Mosmann and Coffman [37]. Both cell types arise following the sustained activation of uncommitted Th0 cell precursors and can be characterised by the expression of mutually exclusive sets of cytokines: Th1 cells produce IFN-γ, whereas Th2 cells express IL-4, IL-5 and IL-6. These cytokine profiles have a critical influence on the selection of a specific immune response, driving pro-inflammatory or allergic responses, and promoting alternative antibody classes. The cellular dichotomy was mechanistically explained by the mutual inhibition between two master transcription factors encoded by T-bet and GATA-3 genes at the single cell level, as well as by cross-regulatory mechanisms at the cell population level. Indeed, the cytokines produced by each Th subtype drives the differentiation of precursors into the same pathway while inhibiting alternative pathways.

Additional T-helper subtypes have been recently identified. Dependent on the transcription factor Foxp3, regulatory T cells are capable of preventing (auto)immunity by inhibiting the activation and proliferation of other cells [53]. Furthermore, proinflammatory Th17 cells expressing IL-17 and dependent on RORγt have been characterised. Current evidence indicates that late Th cell differentiation pathways are more complex and likely comprise further, non-canonical cell types [19, 7, 52, 59, 25], whose mechanistic underpinnings and functional roles remain to be established.

The issue is not only how heterogeneous are these cell types, but, perhaps more importantly, how is the heterogeneity of Th cells sustained and controlled? How is antigen-specific memory mapped to the predominance of an appropriate Th cell branch? How plastic or resilient are the differentiated Th cells? Effective immunity to many fungi and bacteria requires that the T cell response is dominated by proinflammatory effector Th1 or Th17 cells. Allergic reactions, whether beneficial or deleterious, are strictly dependent on Th2 cells. Avoiding spontaneous autoimmunity or controlling the collateral damage of effective immune responses to infection involves a fine balance between regulatory T cells and other Th cells. Genetic defects or accidental failures affecting this delicate balance can lead to irreversible immunopathology.

During their lifetime, naive and memory Th cells face a changing environment in time and space. The lymphoid tissues are heteregeneous and provide variable local cytokine contexts to circulating Th cells. How robust are the Th subtypes with respect to these heterogeneous environments? Will a Th cell that differentiated into a Th1 phenotype in a lymph node always remain in this state or can it switch to another cell type (be reprogrammed) if it faces a different environment? Evidence for substantial plasticity has been recently reported. For example, stimulation of Th2 cells in vitro in the presence of TGF-β generates a non-canonical cell type expressing IL-10 and IL-9 in the absence of Foxp3 [7, 52]. Furthermore, Foxp3+ regulatory T cells loose

the expression of this key transcription factor in the absence of effector T cells *in vivo* and can then drive inflammatory responses [9].

How many of such conversions should be expected? Are some Th cells irreversibly committed and others more plastic? It is difficult to address these questions directly due to the current impossibility to follow a single cell as it circulates in the body, during the rather long time scale of cell differentiation. Instead, studies are usually made on cell populations and therefore measure the predominance of one or another cell type. However, these questions can also be addressed in terms of stability and robustness of differentiation states of the molecular network underpinning the cellular phenotypes, using mathematical modelling.

Mathematical modelling has been used recurrently to formalise hypothetic regulatory schemes in immunology. Early phenomenological models represented the development of Th1 vs Th2 responses from naive Th0 cells using ordinary differential equations [11, 3]. In these models, the subcellular molecular network controlling the state of the cell was implicit behind the transitions between cell types. These models accounted for the role of cell interactions in driving population commitment and sustained polarised responses. In general they featured mutual inhibitions among cell populations, enabling multistability. Alternative population stationary states were interpreted as polarised cell responses. However, such cell population models are unable to predict novel cell types or to question their plasticity because cell properties are hardwired in the model structure.

More recently, models of the cellular networks driving Th cell differentiation and polarisation have been formulated using logical [34] or ordinary differential equations [14, 51]. These models assume a straightforward relationship between cell types and mutually exclusive auto-activated master transcription factors, which precludes any plasticity or reprogramming potential.

In this paper, we propose an integrated, comprehensive model of the regulatory network and signalling pathways controlling Th cell differentiation. Our main aim is to gain insight into the potential heterogeneity and plasticity of late Th cell lineages. As the majority of available data are qualitative, we rely on a qualitative, logical formalism to perform extensive dynamical analyses. To cope with the size and complexity of the resulting network, we use an original model reduction approach recently described elsewhere [39].

To assess the effects of heterogeneous environments on Th cell differentiation, we have performed a systematic, extensive series of simulations, considering various prototypic environments. As we shall see, stable states corresponding to cannonical Th1, Th2, Th17 and Treg subtypes are readily identified, but they are found to coexist with other cell types, notably Treg cells expressing Th1, Th2 and also Th17 'specific' genes in an environment-dependent fashion. In the process, our logical analysis highlights cell types and their relationship to canonical Th subtypes.

2 Methods

2.1 Logical modelling formalism

The precise roles of the different molecular species involved in the regulation of T cell differentiation are sparsely known. Even in the cases where *in vivo* direct regulatory interactions have been documented, little or no quantitative information is available on the relative strengths or rates of these processes.

The extended logical formalism [48, 5] is a discrete modelling framework well adapted to biological systems where most available information is qualitative. In this framework, a regulatory network is modelled in terms of a directed *regulatory graph*, where nodes represent regulatory components (protein, complex, gene,...), whereas arcs represent interactions between these components (*i.e* transcriptional activation or inhibition, phosphorylation, etc.). In addition, each regulatory component is associated with a logical variable denoting its qualitative concentration or *level of activity*. In many cases, Boolean variables capture the most relevant situations (*i.e.*, a Boolean variable takes the value 1 if the component is present or active, 0 otherwise). Whenever needed, *i.e* when a component has distinct functional levels, multi-valued variables are introduced. It is worth noting that components may represent phenomenological features as well as specific molecular species (this is the case for the proliferation component in the Figure 2).

Logical rules are further assigned to each regulatory component to specify its *target* activity level according to the levels of its regulatory components. Whereas some authors consider standard logical functions for all components, we do not impose such a restriction. For example, in the regulatory graph displayed in Figure 2, STAT1 and IL21 both receive three activatory interactions but their logical rules differ: the rule assigned to STAT1 stipulates that STAT1=1 **if** IFNbR=1 **or** IFNgR=1 **or** IL27R=1 (otherwise STAT1=0), whereas the rule assigned to IL21 stipulates that IL21=1 **if** NFAT=1 **and** proliferation=1 **and** STAT3=1.

Given a regulatory graph, the trajectories of the system can be simulated, starting from a (set of) initial state(s) and recursively computing the successors states. This results in the computation of a new graph, called *state transition graph*, which describes the dynamical behaviour of the system. In this graph, a node denotes a *state* of the system (*i.e.*, a tuple giving the levels of the regulatory component), whereas an arc from one state to another denotes a possible *state transition*.

Given a state, if the activity level of a component differs from the target level given by its logical rule, there is a transition towards a state accounting for the update of the corresponding variable. Note that, we assume that a state has as many successors as updated components (fully asynchronous dynamics), potentially leading to alternative behaviours.

Considering a state transition graph, it is particularly interesting to identify the *attractors*, which correspond to potential asymptotical behaviours, being either *stable states* (states without any successor) or *cyclical components* (denoting oscillatory behaviours).

2.2 Model reduction

The asynchronous dynamical analysis of increasing regulatory graphs can be very challenging due to the exponential growth of the state transition graphs. A solution consists in reducing the model by removing (intermediate) components. For example, in the case of the comprehensive regulatory graph displayed in Figure 2, cytokine receptors can be readily considered for reduction (see Section 3.3). To ease such reductions, we have recently proposed an algorithm to automatically compute the logical rules for a user-defined reduced model [39].

Starting with a detailed model, the computation of a reduced model is performed by iteratively removing regulatory components (note that auto-regulated components are not entitled for removal). Removing a regulatory component G implies a revised wiring where all targets of G are directly regulated by the regulators of G. The logical rules of the targets of G are modified accordingly to conserve the (indirect) effects of G's regulators.

This reduction method insures the preservation of a number of dynamical properties of the original model. In particular, stable states and more complex attractors are conserved. However, additional cyclic attractors may arise from the isolation of transient cycles of the original system. An attractor which is reachable in the reduced model is also reachable in the full model, but the reverse is not always true, as the reduction may lead to the loss of some trajectories (see [39] for further details).

2.3 Feedback circuit analysis

One asset of the logical framework is the possibility to analyse the dynamical roles of the *regulatory circuits* (circular chains of interactions) imbedded in the regulatory graph. The dynamical roles of isolated regulatory circuits differ depending on their signs. A circuit is positive if it encompasses an even number of inhibitions, whereas a negative circuit encompasses an odd number of inhibitions. In the eighties, R. Thomas conjectured that a positive circuit is required to generate multiple attractors, whereas a negative circuit is necessary for sustained oscillations [47]. In the meantime, these rules have inspired several theorems in different mathematical frameworks (see [46] for a brief survey). In this work, we focus on positive circuits, which enable the existence of alternative differentiated cell lineages, each corresponding to an attractor.

While the presence of a positive circuit is necessary for the existence of multiple stable states, this condition is not sufficient. Indeed, regulatory circuits are embedded in large networks, and external regulators might prevent the circuit from generating the expected dynamical behaviour. Hence the definition of functionality contexts defining regions of the state space enabling the corresponding dynamical properties. More precisely, functionality contexts are defined in terms of constraints on the regulators of circuit members (see [40] for further details). In practice, in large regulatory graphs, a tiny fraction of the regulatory circuits are usually found functional (*i.e.* have a non-empty functionality context). Such circuit functionality analysis can thus pinpoint crucial regulatory structures (with a predominance of short circuits) that are responsible for salient dynamical features.

2.4 GINsim, a software dedicated to logical dynamical modelling

To ease the definition and analysis of logical models, our team is involved in the development of the *GINsim* software suite [38]. GINsim provides a user interface to set up logical models as well as functionalities dedicated to the construction of states transition graphs, under diverse updating schemes. GINsim further implements an efficient algorithm for the determination of all stable states of a logical model. It also implements the reduction method and the regulatory circuit analysis briefly mentioned above.

We used GINsim to build a comprehensive logical model by integrating a large set of documented molecular actors and interactions. The resulting model will be made available in the model repository linked to the GINsim website. As GINsim allows the user to document the model associating comments (free text, links and references) to each component or interaction, extensive documentation has been integrated in the model file.

2.5 Model simulations

In the case of the model presented here, we have to deal with many input nodes that collectively handle local environment conditions. Varying the values of these inputs, one can simulate all possible fates of a given cell,

represented in terms of the attractors reachable in these diverse local environments. In turn, one can take each of these attractors, stable for a given local environment, and run new simulations changing the local environment (*i.e.* the values of the input nodes).

Our simulations aim at determining how naive cells can differentiate into specialised cells, depending on local environments, and how these differentiated cells respond to environmental changes. In this respect, we have defined a set of prototypic local environments (*i.e.* in terms of specific combinations of values for APC and external cytokines ; see Section 3.2). For each of these environments, we performed a simulation starting from a state corresponding to naive Th cells. All the resulting attractors consist in stable states (*i.e.* attractors reduced to a unique state, see Section 3.3 for a justification), which each corresponds to a cell lineage. These stable states are in turn taken as initial states for a new round of simulations, considering different local environments. We have chained such simulations until no other stable states could be found, thereby allowing us to asses the behaviour of each cell lineage depending on the local environment (see results summarised in Tables 4 and 5).

3 Results

3.1 Building blocks of the T cell regulatory network

The logical formalism enables a modular approach for the construction of large regulatory network models, where parts of the network are defined and studied separately, before merging them to generate a comprehensive model. The Th cell network was constructed based on several "modules", displayed in Figure 1 and described below.

TCR signalling

The first building block is the TCR signalling module that controls all aspects of Th cell's life cycle including activation and differentiation. A comprehensive Boolean model of the TCR signalling cascade has been recently published [44]. In brief, T cells receives coordinated inputs from the antigen presenting cell via the TCR itself and via CD2. For sake of simplicity, we represent these coordinated inputs by a single variable denoted APC and assume that these signals converge to activate AP1/NFAT as well as NFkB transcriptional activity (see Figure 1, left). This simplified pathway capture the essential dynamical features of the more complete model network analysed in [44].

Cytokines signalling

Cytokines play an important role in the control of T cell differentiation. Cytokine signaling proceeds via the JAK-STAT pathway. Cytokine binding to its receptor chains leads to the phosphorylation of specific JAK and STAT, the latter being translocated into the nucleus were it activates the transcription of target genes. A generic logical model for cytokine signalling is shown in Figure 1 (middle). This module has been replicated and adapted for a list of cytokines, taking into account the relevant receptor chains, as well as JAK and STAT components (see Table 1). ILR represents the activated state of the cytokine receptor: it is active when its subunits are crosslinked to the corresponding cytokine, which may have an autocrine or paracrine origin. This can be represented by the logical function $ILR1$ **and** $ILR2$ **and** (IL **or** IL_e). Note that several subchains and STATs are shared between different cytokine modules. Here, we consider that a STAT is active when at least one of its activating cytokine receptor is active.

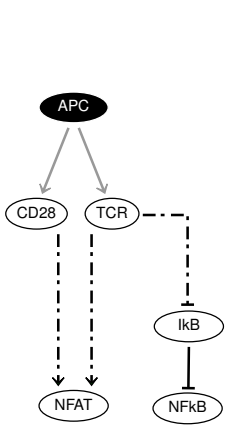
To enable the analysis of the model in terms of stable states, we provisionally ignore the arrest of stimulation and the SOCS-dependant negative feedbacks.

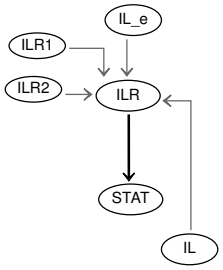
IL2 and cell cycle

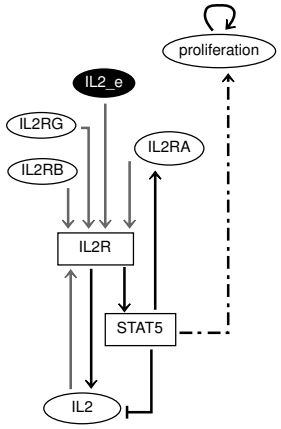
As IL2 plays a particular role in the control of Th cell differentiation and proliferation, it deserves a more detailed description. The IL2 module shown in Figure 1 (right) extends the general cytokine module by considering explicit positive and negative feedbacks onto IL2, as well as the effect of STAT5 on cell proliferation [36]. Briefly, IL2Rα increases the affinity of the IL2 receptor, enabling a stronger response, required to activate cell proliferation (for more details on the regulation of IL2 and of its receptor, see [24]). Note that, in our full model, IL2 and IL2Rα are further regulated by AP-1/NFAT and Foxp3, not shown here.

3.2 Regulatory network controlling Th cell activation and differentiation

All the modules must then be integrated to build a comprehensive model. The TCR signalling module functions as an input, as it is not regulated by any other component. Several cytokine receptors share subchains and targets. For example, the common gamma chain (CGC) is widely used and STAT5 is activated by all cytokine receptors using it (cf. Table 1). Furthermore, the cytokine modules are connected through a number of cross-regulations. Figure 2 displays the whole regulatory graph assembled for this study, while the supplementary







NFAT=1 if CD28=1 ∧ TCR=1 **NFAT=0 otherwise**	**ILR=1 if** (ILR1=1 ∧ ILR2=1) ∧ (IL_e=1 ∨ IL=1) **ILR=0 otherwise**	**IL2R=2 if** (IL2RA=1 ∧ IL2RB=1 ∧ CGC=1) ∧ (IL2_e=1 ∨ IL2=1) **IL2R=1 if** (IL2RA=0 ∧ IL2RB=1 ∧ CGC=1) ∧ (IL2_e=1 ∨ IL2=1) **IL2R=0 otherwise**

Figure 1: Model building blocks. **Left:** (simplified) TCR signalling pathway. **Middle:** generic cytokine module. IL_ext represents the cytokine present in the environment; IL represents the autocrine production of the same cytokine; ILR1 and ILR2 represent two different receptor sub-chains; ILR represents the activated receptor, which in turn activates STAT. **Right:** IL2 regulation and its effect on cell proliferation. The bottom row gives the logical functions used for one of the components. "∧" and "∨" stand for AND and OR logical operators, respectively.

Cytokine	Chains	Target	Producers	Targets
IFNG	IFNGR1, IFNGR2	STAT1	Th1	Th1
IL4	IL4RA, CGC	STAT6	Th2	Th2
IL6	IL6RA, GP130	STAT3		Th17
IL10	IL10RA,IL10RB	STAT3		Th2
IL12	IL12RB1, IL12RB2	STAT4		Th1
IL15	IL2RB, CGC, IL15RA	STAT5		
IL21	CGC, GP130	STAT3		Th17
IL23	IL12RB1, GP130	STAT3		Th17
IL27	GP130, IL27RA	STAT1, STAT3		

Table 1: List of the cytokines considered in our model, each corresponding to an instance of the generic module shown in Figure 1 (middle). For each cytokine, the corresponding receptor sub-chains and downstream targets are specified, along with the producing and targeted Th subtypes. CGC stand for Common Gamma Chain. The IL-15 receptor has three subchains (versus two in the generic module), all of which are required for proper signalling.

material provides the logical rules associated with each component, along with biological justifications and bibliographical references.

3.3 Model reduction

What cell types and differentiation pathways can be predicted from the logical model just described? The different cell types correspond to attractors in the state transition graph, *i.e.* regions of the state transition graph from which the system cannot escape. Among these attractors, stable states can be readily determined [40]. Other attractors may consist in (intertwined) terminal cycles. Their identification requires a thorough analysis of the state transition graph.

As our regulatory network encompasses too many components to enable a direct analysis of the full state transition graph, we decided to apply the reduction method described in Section 2.2). Regarding the selection of the components to "hide", we face a compromise between computational performance and biological readability. By hiding cytokines receptors and their subchains from the model, along with the intermediate components RUNX3, IRF1, SMAD3, IkB, NFkB, CD28 and TCR, we obtain a regulatory graph containing 34 components (see Figure 3). Although the resulting logical model is still too large to compute the whole state transition graph (encompassing over 25.10^9 states), it is now amenable to simulations starting from selected initial states. Indeed, in this regulatory graph, 13 out of 34 components represent environmental conditions (inputs), which are fixed for each simulation. With 21 internal components, the system has "only" about 4.5 millions of possible states, most of which are not reachable when starting from a specific initial state.

A stronger reduction leads to a graph conserving the 13 inputs, but only 7 internal nodes: Tbet, GATA3, RORgt, Foxp3, STAT3, IFNg and proliferation. Using this compact model, we could compute the full state transition graphs for relevant input combinations. This led us to identify 28 stable signatures and to verify the absence of cyclic attractor. Since our reduction preserve attractors [39], we can conclude that all attractors of the original model are indeed stable states. Furthermore, whenever a stable signature can be reached in the compact model from specific initial condition, we can conclude that it is also the case for the full model.

3.3.1 Regulatory circuits and cell fate decisions

Regulatory circuits are known to play a role in the emergence of essential dynamical properties (cf. Section 2.3). The regulatory graph presented in Figure 2 encompasses 565 circuits in total. We expect only a small fraction of them to be functional, with a predominance of small positive circuits. We are mostly interested in the short positive circuits involving the four master regulators: Tbet, GATA3, Foxp3, and RORgt. Indeed, each of these circuits may function as a switch to maintain the differentiation of a specific Th subtype.

Considering that each functional positive circuit can lead to two attractors, the four auto-activations can possibly generate 16 (*i.e.*, 2^4) attractors (combining the two possible states of the four related components). However, the cross-inhibitory circuit (Tbet-GATA3) ensures that Tbet and GATA3 expressions are exclusive, leading to the twelve possible combinations shown in the Table 3. The first five patterns correspond to the expected canonical Th0, Th1, Th2, Th17 and Treg subtypes, while the last seven patterns involve the expression of several master regulators. Each of these stable states may be compatible or not with specific input configurations. Moreover, additional positive circuits may be functional and thereby introduce further expression variability. For example, the circuit involving STAT3 can lead to stable patterns differing by the value of STAT3 and the downstream cytokines.

To analyse how the cell fates depend on initial and environmental conditions, we have iterated several round of simulations, using the prototypic environments described in Table 2 (cf. Section 2.5). In the first round of simulations, we used an initial state corresponding to the Th0 subtype (all components inactive). As a result, we obtained the stable states listed in Table 4.

3.3.2 Plasticity of Th cell types

To check the stability of the identified cell types in changing environments, we performed a series of simulations using each of the stable states listed in Table 4 as initial configuration with each of the prototypic environments listed in Table 2. The results are summarised in Table 5 and Figure 4.

In the absence of any input from the environment, resting Th0, Th1, and Th2 are the only three attractors. These three cell types can thus be considered as reference states, predominant outside lymphoid tissues, where APC and cytokines are scarce.

The other stable states can be grouped with one of these reference states, depending on the expression of characteristic markers:

- the states Th0, Th0_a, Th17_n, Treg_a, and Treg_r can be grouped with Th0 cells;
- the states Th1, Th1_a, Th1_n, Th1_ra, Th17_1, Treg_1, and Treg_r1 can be grouped with Th1 lineage

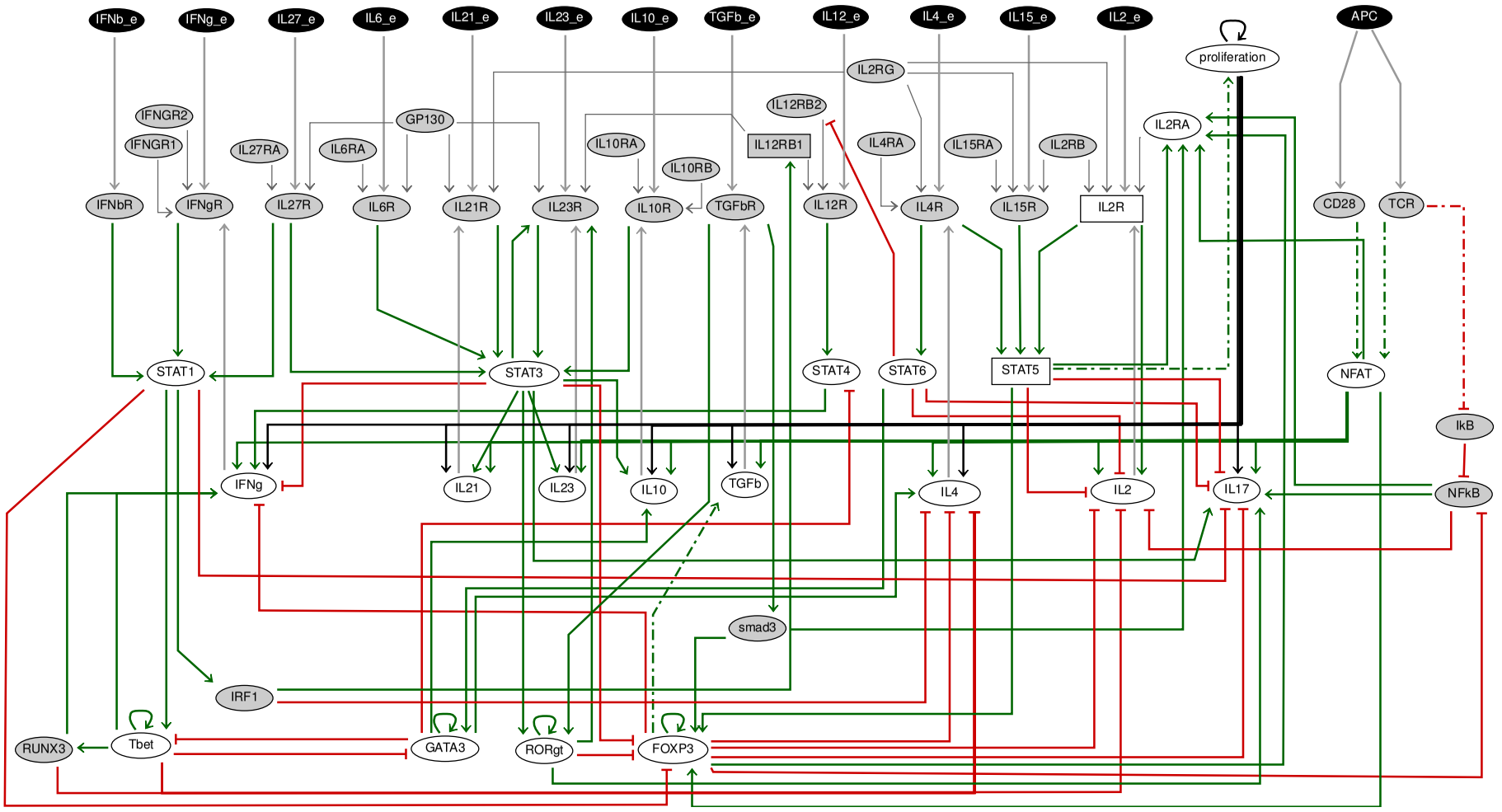


Figure 2: Th differentiation regulatory graph, encompassing 64 components. The 13 input components are colored in black. Ellipses denote Boolean components while rectangles denote ternary components. Green arrows denote activations, whereas red blunt ones denote inhibitions. Dashed arcs denote indirect (or poorly documented) interactions. The greyed-out components have been hidden to generate the regulatory graph displayed in Figure **??**

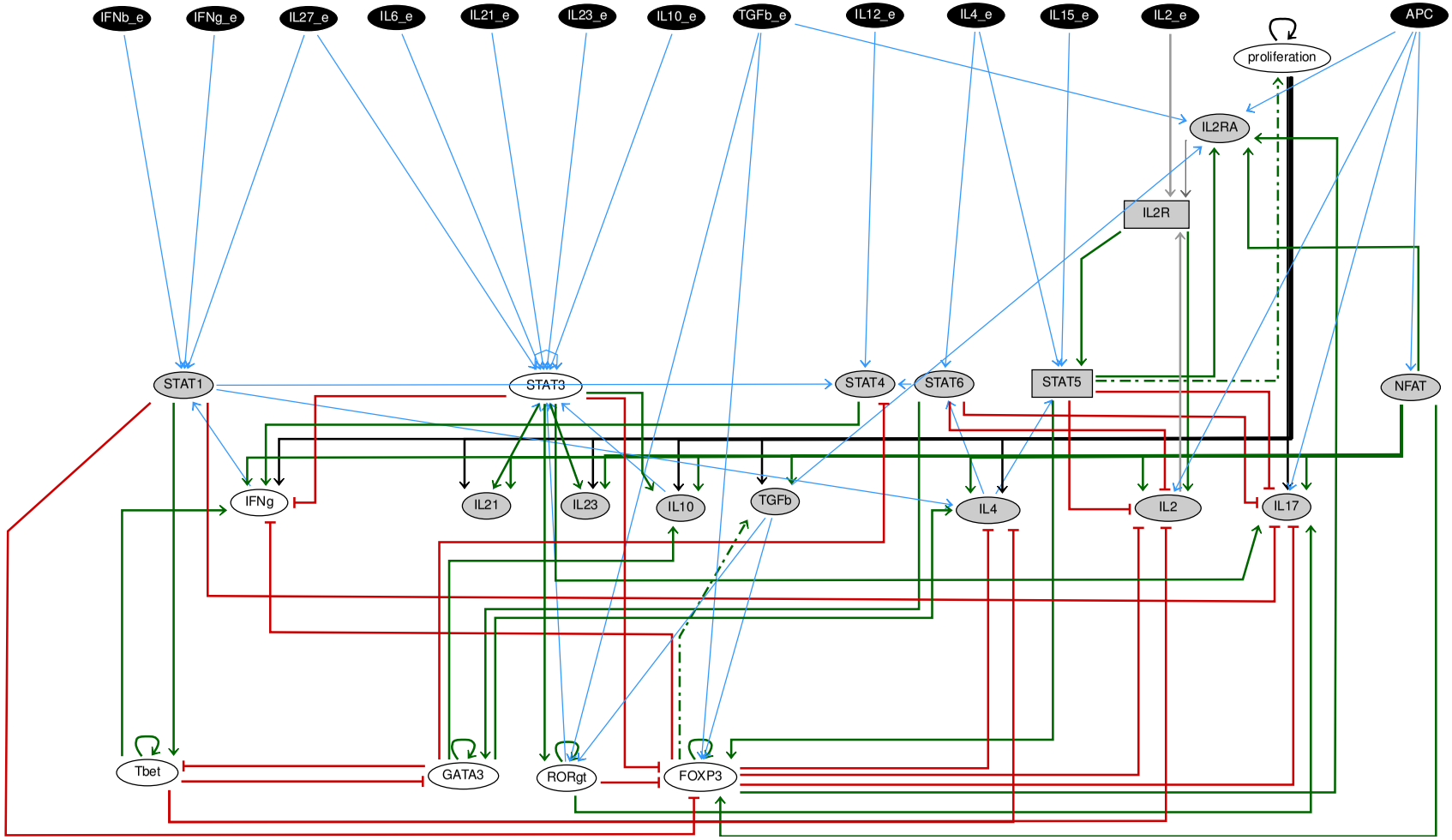


Figure 3: Reduced Th regulatory graph, encompassing 34 components. This graph has been obtained by applying the reduction method described in Section 2.2 to the full model shown in Figure 2. Indirect interactions resulting from the reduction are displayed using dotted lines. Greyed-out components can be further reduced to generate a more compact model.

code	description	Inputs: APC	IL2_e	IL4_e	IL6_e	IL10_e	IL12_e	TGFb_e	IFNg_e
	no stimulation								
	APC only	■							
	pro Th1	■	■						■
	pro Th1 bis	■					■		
	pro Th2	■		■	■				
	pro Th17	■			■			■	
	pro regulatory	■	■					■	
	pro Tr1	■				■			

Table 2: Environmental conditions used for the simulations. Each row correspond to one prototypic environment, in terms of combinations of APC and seven different cytokine inputs. Input presences are denoted by greyed cells. The visual code of the first column is used in Table 5.

Cell lineage	Master regulators: Tbet	GATA3	RORgt	Foxp3	Other circuits: None	Prolif	STAT3	Both
Th0	0	0	0	0	32	48	112	240
Th1	1	0	0	0	372	512	2880	3968
Th17	0	0	1	0	16	16	240	496
Th2	0	1	0	0	312	256	2400	3968
Treg	0	0	0	1	8	24	28	–
Treg_1	1	0	0	1	96	224	720	–
Treg_2	0	1	0	1	96	–	720	–
Treg_r	0	0	1	1	4	24	60	192
Treg_r1	1	0	1	1	48	224	1488	3584
Treg_r2	0	1	1	1	48	–	1488	3584
Th1_r	1	0	1	0	180	256	5952	8064
Th2_r	0	1	1	0	152	128	4960	8064

Table 3: Definition of alternative Th subtypes based on master regulator expression. Each of the four master genes considered (Tbet, GATA3, RORgt and Foxp3) auto-regulates itself positively. The first four rows correspond to the canonical Th lineages expressing no (Th0) or a single master regulator (Th1, Th17, Th2, Treg). The remaining lineages correspond to mixed expression patterns. We considered that Foxp3 has a "dominant" effect over the other regulator, while RORgt has a relatively minor effect. Suffixes (r for RORgt, 1 for Th1, 2 for Th2) are used to denote the co-expression of additional regulators. The last four columns denote the four possible states of the other positive circuits (proliferation and STAT3-related). The circuit analysis predicts 48 stable core signatures (4 for each of the 12 groups). Among these 48 signatures, only 28 are compatible with at least one of the input combinations considered here (grey cells). The values in the cells indicate how many input combinations are compatible with this stable state. Five signatures are not compatible with any input combination (cells with dashes). With 13 input nodes, we obtain 393216 (48.2^{13}) possible combinations, among which 56284 are found consistent and thus constitute stable states for the whole system, reflecting the variety of possible environmental inputs.

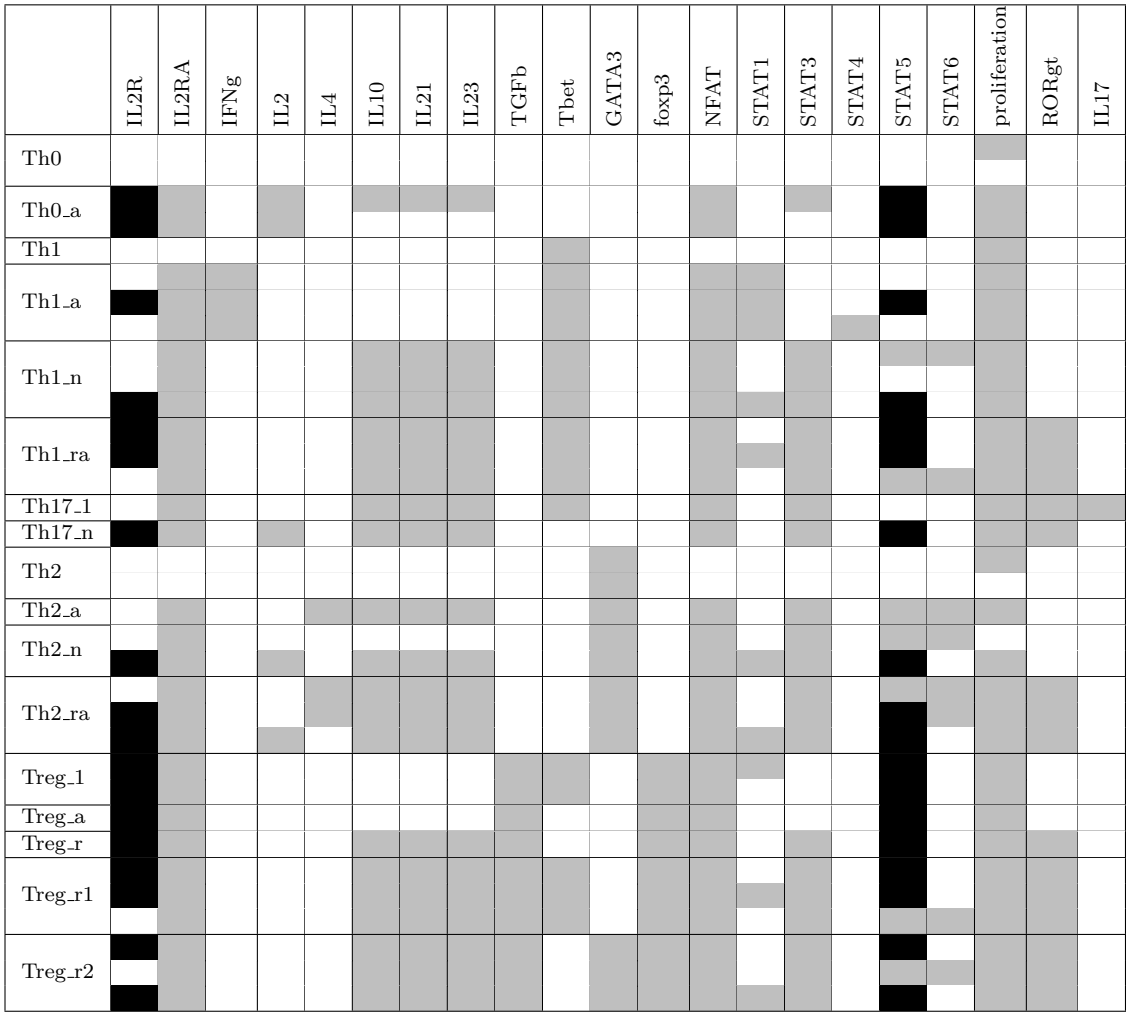


Table 4: Stable states found during the simulations. Each row corresponds to a state. A grey cell in a row denotes that the corresponding component is active in the corresponding stable state. Black cells denote higher activity levels (in the case of multi-level components). Note that the values of the input nodes are omitted here. A stable state for a given input combination may not be stable for another one. The link between these stable states and selected environmental conditions (described in Table 2) are given in Table 5. This table encompasses stable corresponding to naive, Th1 (expressing Tbet),Th2 (expressing GATA3), Th17 (expressing RORgt) and Treg (expressing Foxp3) subtypes. The states are named following the same rules as in Table 3: the name indicates the prominent phenotype while the suffix denote the co-expression of additional regulators ("r" for RORgt, "1" for Th1, and "2" for Th2). Two additional suffixes were added to further distinguish the activity of the cells: "a" denotes activated cells (*i.e.* producing lineage-specific cytokines), and "n" denotes anergic cells (*i.e.* expressing NFAT but no lineage-specific cytokine). Note that when the expression of the master regulators is identical, two or three stable states may be associated with a single cell type (see, *e.g.*, Treg_r2, at the end of the table).

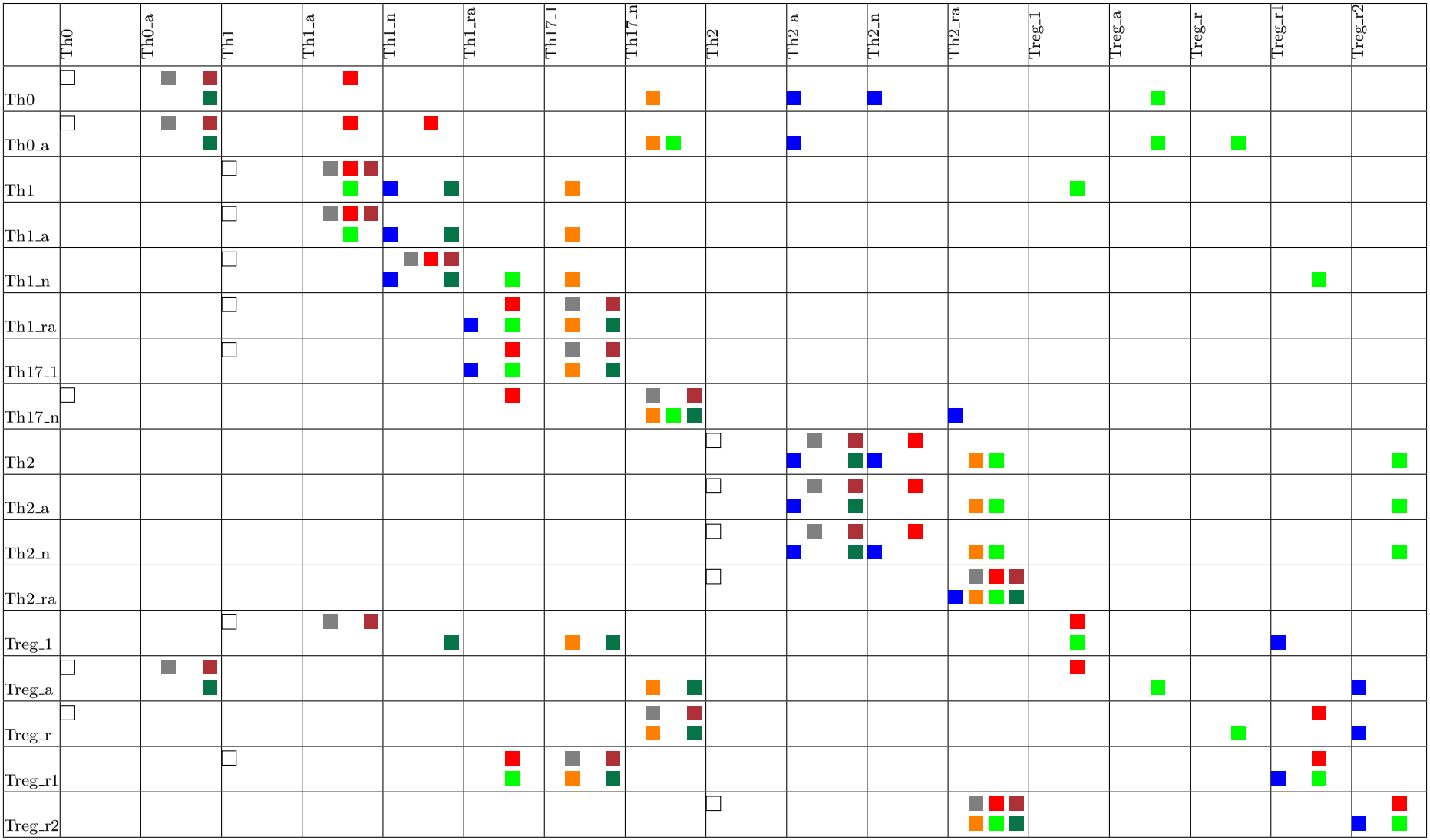


Table 5: Stability of Th subtypes and induced transitions. This table summarises several simulation rounds, displaying the stable signatures (column entries) reached depending on the initial signatures (row entries) and the environmental conditions (coloured tiles in the cells). The environmental conditions corresponding to the different tiles are described in Table 2.

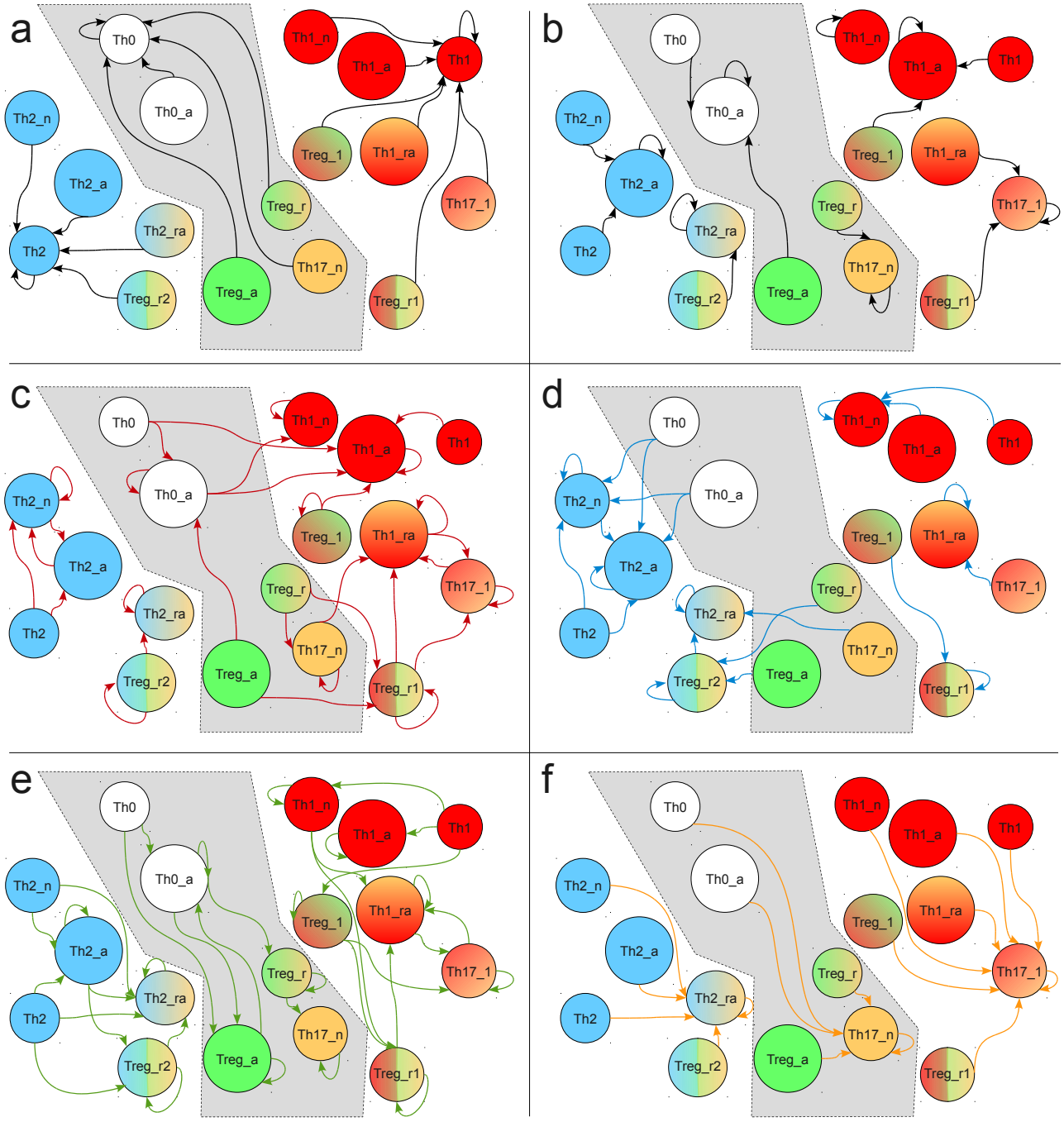


Figure 4: Graphical representation of the plasticity of cell lineages depending on the environment. The cell lineages observed in silico are grouped in three main *constellations* (Th0, Th1 and Th2, delimited by different backgrounds). Each sub-figure represents environmental conditions from Table 2: **(a)** no stimulation, **(b)** APC only, **(c)** Th1 or Th1 bis, **(d)** Th2, **(e)** regulatory or Tr1, and **(f)** Th17. Arrows between cell lineages denote possible switches occurring in each specific environment.

- finally, Th2, Th2_a, Th2_n, Th2_ra, and Treg_r2 can be grouped with Th2 lineage.

However, these *constellations* of states are not disconnected. Obviously, any cell turning into 'precursor' Th0 or Th0_a subtypes can readily switch to states belonging to either of the two other constellations depending on further stimulation. According to our simulations, using proper sequences of environmental inputs, several differentiated Th subtypes could likely be *reprogrammed* into other subtypes.

By and large, Th1 cells will tend to remain within the Th1 state constellation, which includes activated Th1_a, resting Th1, and anergic Th1_n. For example, stimulation by APC will activate a resting Th1 into an activated Th1_a cells, inclusively in a pro-Treg environment. In either a pro-Th2 or proTr1 environment, Th1 cell will become anergic, while they will tend to express RORgt with or without IL-17 in a pro-Th17 environment. Anergic Th1 cells, as well as non-canonical Th1 and Th17 mixed cell types, will remain so provided that APC stimulus is sustained, but most likely turn to resting Th1 cells in the absence of sustained TCR signalling. In a pro-Treg environment, resting Th1 cells can up-regulate Foxp3 and further express both T-bet and Foxp3. This chimeric cell type can loose Foxp3 under several conditions: in absence of TCR stimulus and of other stimuli, they will revert to the Th1 resting state; in the presence of APC alone (without pro-Treg stimuli) or together with IL-12 (pro-Th1bis), they turn into the canonical Th1 activated state (Th1_a); whereas in the presence of IL-12 they turn into an anergic Th1 state.

Th2 differentiated cells expressing GATA-3 are even more robust than Th1 cells and will always remain within the Th2 constellation (labeled in blue in Figure 4). In pro-Th1 environments, all Th2-like cells converge towards the anergic Th2_a type, devoid of cytokine production. According to our model, in pro-Th17 or pro-Treg environments, Th2 cells may turn on RORgt and Foxp3, respectively. While Foxp3 expression will be lost in the absence of IL-2, the expression of RORgt is more robust and remains in all the environments providing TCR stimulus.

A subset of Th17 cells expressing RORgt overlaps with all other lineages. This results from the positive auto-regulation of RORgt, which is barely affected by other transcription factors. Thus, an environment rich in TGFβ and IL-6 can lead to the activation of RORgt in virtually any cell type (Figure 4). This expression will be stable until the cell switches to a resting state in the absence of TCR signals from the APC.

Treg cells can be maintained only in the presence of TCR-stimulus delivered by APC and of IL-2 produced by other cells. However, in strong polarizing environments, Tregs may differentiate into mixed cell types associated with the Th1 or Th2 cell constellations. For example, Foxp3+ T cells expressing RORgt and t-bet (Treg_r1) can be obtained by moving Treg_1 cells into a pro-Th2 environment, or when several Foxp3+ cells dwell in a pro-Th1 environment. These cells may loose Foxp3 in several environmental conditions, from pro-regulatory to pro-inflammatory environments, coming closer to the Th1 subtype. However, they can be maintained in other experimental conditions, in particular in sustained pro-Th2 environments. Foxp3+, RORgt+, and GATA-3+ cells can arise from either Th17_n or several Foxp3+ precursors by sustained pro-Th2 environment. Finally, Th0 cells can potentially turn into Th2 via regulatory or Th17 intermediates.

4 Discussion

We have reconstructed the molecular network controlling the activation and differentiation of Th cells and asked how many stable states to expect, considering that these cells face a changing local environment during their life span. We conclude that while canonical states such Th1, Th2, Th17 and Treg are identified, transient, environment-dependent non-canonical cell types are also expected. These non-canonical states are often chimeric or hybrid, displaying partial gene expression profiles characteristic of two or more canonical cell types.

Several aspects of the network topology are currently based on unverified assumptions. Components and cross-regulatory links were extracted from the literature and, in some cases, from previous logical models [34, 35]. It is likely that components or links were missed. Furthermore, we face another challenge with the definition of the logical functions driving the behaviour of the components, particularly those involving complex regulatory mechanisms. Unfortunately, there is no rigorous, objective procedures yet to infer logical functions from available data.

This cautionary note notwithstanding, our current model allows to recapitulate the differentiation of naive cells into 4 main lineages and generates some unexpected predictions. A surprising result is the plasticity and lability of the regulatory T cells considering the fundamental role of these cells in avoiding autoimmunity. According to our model analysis, the only way to sustain a regulatory T cell phenotype is the specific condition of sustained TCR signalling and IL-2 rich environment. The absence of TCR stimulation allows the cell to loose Foxp3 and revert to a Th0 phenotype. From this state, the cell can regain Foxp3+ but also differentiate into other cell types. Our model further predicts that any polarizing environment will promote the differentiation into a Th1 or Th2 related phenotype. Turning Tregs into effector cells is consistent with some recent observations [9] but not with others [26]. In our model, the lability of regulatory T cells is clearly dependent on the assumption that Foxp3 requires STAT5 and AP1/NFAT transcriptional activities. This might turn out to be an oversimplification indicating that we overlooked some mechanisms preventing regulatory cells to become effector

cells and vice-versa. Some data suggests that DNA methylation is needed for sustainable foxp3 expression [12]. miRNA have probably a role as well (a knockout of a gene known for its role in miRNA generation interferes with the repressive activity of Tregs [60]). However, the integration of these results into a model would require the clarification of the underlying mechanisms. Indeed, blocking all miRNA probably likely leads to pleiotropic effects, and the loss of Treg activity could be just a side effect.

The lability of regulatory T cells could be eliminated by making strengthening Foxp3 auto-activation, to render Foxp3 expression autonomous (similarly to the case of RORgt). Alternatively, higher order regulatory mechanisms stemming from intercellular interactions and population dynamics could be determinant. The modelling of interacting populations dynamics is beyond the scope of the present study, but differential equation models [29] suggest that stable regulatory T cells pools could be maintained despite a continuous interconversion of cells. Evidence that persistence of regulatory T cell pools and the maintenance of their phenotype depend on higher level population effects is provided by results of loss or gain of phenotype depending on the remaining populations. This may therefore indicate that the Foxp3 module considered in our molecular network might not be over-simplified.

It is worth discussing wether the predicted *chimeric* cell types are realistic. If we consider the production of the main lineage-specific cytokines (IFNg, IL4, IL17 and TGFb), these cell types are not really surprising and could have been overlooked in standard essays. Indeed, some of these cell types are anergic, while others express only one kind of cytokine. Recent experiments highlighting such chimeric cells as discussed below.

Many of these chimeric cell types express RORgt, which is supposed to be Th17 specific. It may mean that it is not the right component for the job (and that the real one is still unknown) or that cross-inhibitory mechanisms between RORgt and other master regulators remain to be uncovered. Published evidences indicate that Foxp3 and RORgt may inhibit each other [12, 58]. RORgt expression has been observed in Treg. When expressed along with Foxp3, it is unable to activate IL17. Thus, RORgt is needed for the differentiation of the Th17 lineage but is not sufficient to maintain it. Furthermore, co-expression of IL17 and IFNg has been observed in memory cells [31]. Thus, the chimeric stable states involving IL17 expression along with Th1 or Th2 markers could be realistic.

We have identified several environment-dependent regulatory Th cell types that express genes characteristic of Th1, Th2 or Th17 cells. In an environment dominated by other effector Th cells types, Foxp3+ regulatory T cells would thus up-regulate a subset of genes specific of these cell types. The upregulation of only a subset of genes not suppressed by Foxp3 could then allow these chimeric cells to mingle within effector T cells, profiting from local growth or survival factors, as suggested by population dynamical modelling [4]. This hypothesis is supported by the recent observation that the capacity of regulatory T cells to suppress Th1 and Th2 responses is lost if key transcription factors are selectively knocked out in the Foxp3 expressing cells [25, 57].

The list of cytokines is increasing and most likely will still grow more. It is not unlikely that these cytokines are predominantly expressed by specific, yet undiscovered, cell types. How much will the information already obtained in our network be retained as the network grows? This is in fact a question about the modularity of the network and the robustness of the modules. We extended a Th1/Th2 differentiation model to take into account two additional lineages. If additional lineages are still to be discovered, proper model extensions should enable to take them into account.

Our results indicate that the pool of CD4 T cells is highly heterogeneous and that the structure of the gene network promotes this heterogeneity. Diversity of antigen receptors and their crossreactivity are the most salient feature of the adaptive immune system of the vertebrates, allowing the host to recognize and potentially react to numerous antigens. The heterogeneity and plasticity of Th cell types emphasised here adds to this open-ended capacity to deal with wide possible contingencies. Furthermore, such plasticity does not fit well with the classical depiction of T cell differentiation potential in terms of a branching tree. Instead, our computational study points to a reticulate network of alternative, environment-dependent, differentiation and reprogramming events.

Acknowledgments.

A.N. has benefited from a PhD grant from the French Ministry of Research and Technology. This work has been supported by research grants from the French National Agency (projects ANR-06-BYOS-0006 and ANR-08-SYSC-003), and from the Belgian Science Policy Office (IAP BioMaGNet). C.C. and J.C have been further supported by the Calouste Gulbenkian Foundation.

5 Appendix: description of all components considered in the Th model

Component(s)	Qualification	Behaviour	Reference(s)
IFNG_e, IL2_e, IL4_e, IL6_e, IL10_e, IL12_e, IL15_e, IL17_e, IL21_e, IL23_e, IL27_e, TGFb_e	external cytokines	input of the model representing the external environment. We do not consider the arrest of the activation.	
APC	denotes the presence of an Antigen-Presenting Cell	input of the TCR module. We do not consider the arrest of the activation.	
CGC, IFNGR1, IFNGR2, IL4RA, IL6RA, GP130, IL10RA, IL10RB, IL2RB, IL15RA, IL27RA	subchains of the cytokine receptors	considered as always present	
IL12RB2	subchain of IL12R	inhibited by STAT6 (present otherwise)	[45, 34]
IL12RB1	subchain of IL12R and IL23R	always present with a higher level (required for IL12 signalling) in presence of IRF1	[22]
IL2RA	high affinity subchain of the IL2 receptor	activated by NFAT, NFkB, STAT5, smad3 and foxp3	[24, 33]
IFNGR, IL4R, IL6R, IL10R, IL15R, IL17R, IL21R, IL23R, IL27R, TGFbR	cytokine receptors, composed of subchains as described in Table 1	active when their subchains and the cytokine (external or from the same cell) are present	
IL12R	IL12 receptor	as other receptors but requires a higher level of IL12RB1	[22]
IL23R	IL23 receptor	as other receptors but also requires RORγt and STAT3	[20]
IL2R	IL2 receptor, composed of 3 subchains (CGC, IL2RA and IL2RB)	CGC and IL2RB are mandatory but IL2RA is only needed for higher levels of IL2R	[24]
TCR, CD28	T Cell Receptor and its co-receptor	activated by APC	
IkB		inhibited by the TCR pathway	[44]
NFkB		inhibited by IkB and foxp3	[1]
IRF1		activated by STAT1	[22]
STAT1		activated by IFNbR, IFNgR and IL27R	[34, 21, 55]
STAT3		activated by IL6R, IL10R, IL21R, IL23R, and IL27R	[35, 55]
STAT4		activated by IL12R and inhibited by GATA3	[34]
STAT5		activated by IL2R, IL4R, and IL15R. High levels of IL2R are required for high levels of STAT5	[24]
STAT6		activated by IL4R	[34]
proliferation	the cell started proliferating	triggered by high levels of STAT5, its arrest is not considered here. We assume that cell proliferation is required for the production of all cytokines but IL2.	[42, 2]
NFAT		activated by TCR and CD28. We assume it is required for the production of all cytokines.	[15, 18, 24]
Tbet	the master switch for the Th1 sub type	activated by itself and STAT1 and inhibited by GATA3	[34]
RUNX3		activated by Tbet	[8]
GATA3	the master switch for the Th2 sub type	activated by itself and STAT6 and inhibited by Tbet	[34]
Foxp3	specific to Treg cells	activated by NFAT, TGFb (through smad3), and IL2 (through STAT5) and inhibited by IL6 (through STAT3). Binding sites for Tbet, STAT1, GATA and RORgt have been observed in its promoter region. Tbet and GATA3 are likely inhibitors but no evidence further supports it yet.	[49, 56, 12, 33, 30, 54, 50]
RORgt	required for the production of IL17	self-maintained and activated by STAT3 and TGFβR Is it an intermediate in STAT3 activation by TGFbR?	[31, 59, 58, 20]
TGFb		produced by the Treg, we assume that it is activated by foxp3	
smad3		activated by TGFb	[33, 30]
IFNg		activated by NFAT, proliferation, Tbet/RUNX3 and STAT4/IRAK. Activation by NFAT inhibited by foxp3. Inhibited by STAT3	[34, 8, 1]
IL2		activated by IL2R and NFAT. STAT5 and STAT6 cooperate to inhibit IL2 production. foxp3 cooperates with NFAT to inhibit IL2. Tbet cooperates with relA (NFkB subunit) to inhibit IL2	[24, 54, 43, 17]
IL4		activated by GATA3, NFAT and proliferation. Tbet and RUNX3 inhibit it cooperatively, foxp3 blocks its activation by NFAT. STAT1 inhibits it through IRF1	[34, 8, 1, 27, 10]
IL10		activated by NFAT, proliferation, GATA3 IL6 and TGFbR (probably through STAT3)	[35, 32]
IL17		STAT3 and RORgt activate IL17 cooperatively. It is and inhibited by IL2 (through STAT5) and Foxp3 (which blocks RORgt). STAT1 and STAT6 are likely inhibitors of IL17.	[59, 55, 58, 20, 28, 6]
IL21		activated by STAT3	[58]
IL23		activated by STAT3	[58, 20]

References

[1] E. Bettelli, M. Dastrange, and M. Oukka. Foxp3 interacts with nuclear factor of activated T cells and NF-kappa B to repress cytokine gene expression and effector functions of T helper cells. *Proc. Natl. Acad. Sci. U.S.A.*, 102(14):5138–43, 2005.

[2] J. J. Bird, D. R. Brown, A. C. Mullen, N. H. Moskowitz, M. A. Mahowald, J. R. Sider, T. F. Gajewski, C. R. Wang, and S. L. Reiner. Helper T cell differentiation is controlled by the cell cycle. *Immunity*, 9(2):229–37, 1998.

[3] J. Carneiro, J. Stewart, A. Coutinho, and G. Coutinho. The ontogeny of class-regulation of CD4+ T lymphocyte populations. *Int. Immunol.*, 7(8):1265–77, 1995.

[4] J. Carneiro, K. Leon, I. Caramalho, C. van den Dool, R. Gardner, V. Oliveira, M.-L. Bergman, N. Sepúlveda, T. Paixão, J. Faro, and J. Demengeot. When three is not a crowd: a Crossregulation model of the dynamics and repertoire selection of regulatory CD4+ T cells. *Immunol. Rev.*, 216:48–68, 2007.

[5] C. Chaouiya, E. Remy, B. Mossé, and D. Thieffry. Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework. *Lect Notes Comp Inf Sci*, 294:119–126, 2003.

[6] Z. Chen, A. Laurence, Y. Kanno, M. Pacher-Zavisin, B. Zhu, C. Tato, A. Yoshimura, L. Hennighausen, and J. O'Shea. Selective regulatory function of Socs3 in the formation of IL-17-secreting T cells. *Proc. Natl. Acad. Sci. U.S.A.*, 103(21):8137–42, 2006.

[7] V. Dardalhon, A. Awasthi, H. Kwon, G. Galileos, W. Gao, R. A. Sobel, M. Mitsdoerffer, T. B. Strom, W. Elyaman, I.-C. Ho, S. Khoury, M. Oukka, and V. K. Kuchroo. IL-4 inhibits TGF-beta-induced Foxp3+ T cells and, together with TGF-beta, generates IL-9+ IL-10+ Foxp3(-) effector T cells. *Nat. Immunol.*, 9(12):1347–55, 2008.

[8] I. Djuretic, D. Levanon, V. Negreanu, Y. Groner, A. Rao, and K. Ansel. Transcription factors T-bet and Runx3 cooperate to activate Ifng and silence Il4 in T helper type 1 cells. *Nat. Immunol.*, 8(2):145–53, 2007.

[9] J. H. Duarte, S. Zelenay, M.-L. Bergman, A. C. Martins, and J. Demengeot. Natural Treg cells spontaneously differentiate into pathogenic helper cells in lymphopenic conditions. *Eur. J. Immunol.*, 39(4):948–55, 2009.

[10] B. Elser, M. Lohoff, S. Kock, M. Giaisi, S. Kirchhoff, P. H. Krammer, and M. Li-Weber. IFN-gamma represses IL-4 expression via IRF-1 and IRF-2. *Immunity*, 17(6):703–12, 2002.

[11] M. A. Fishman and A. S. Perelson. Th1/Th2 cross regulation. *J. Theor. Biol.*, 170(1):25–56, 1994.

[12] S. Floess, J. Freyer, C. Siewert, U. Baron, S. Olek, J. Polansky, K. Schlawe, H.-D. Chang, T. Bopp, E. Schmitt, S. Klein-Hessling, E. Serfling, A. Hamann, and J. Huehn. Epigenetic control of the foxp3 locus in regulatory T cells. *PLoS Biol.*, 5(2): e38, 2007.

[13] L. E. Harrington, R. D. Hatton, P. R. Mangan, H. Turner, T. L. Murphy, K. M. Murphy, and C. T. Weaver. Interleukin 17-producing CD4+ effector T cells develop via a lineage distinct from the T helper type 1 and 2 lineages. *Nat. Immunol.*, 6 (11):1123–32, 2005.

[14] T. Höfer, H. Nathansen, M. Löhning, A. Radbruch, and R. Heinrich. GATA-3 transcriptional imprinting in Th2 lymphocytes: a mathematical model. *Proc. Natl. Acad. Sci. U.S.A.*, 99(14):9364–8, 2002.

[15] P. G. Hogan, L. Chen, J. Nardone, and A. Rao. Transcriptional regulation by calcium, calcineurin, and NFAT. *Genes Dev.*, 17(18):2205–32, 2003.

[16] S. Hori, T. Nomura, and S. Sakaguchi. Control of regulatory T cell development by the transcription factor Foxp3. *Science*, 299(5609):1057–61, 2003.

[17] E. S. Hwang, J.-H. Hong, and L. H. Glimcher. IL-2 production in developing Th1 cells is regulated by heterodimerization of RelA and T-bet and requires T-bet serine residue 508. *J. Exp. Med.*, 202(9):1289–300, 2005.

[18] S.-H. Im, A. Hueber, S. Monticelli, K.-H. Kang, and A. Rao. Chromatin-level regulation of the IL10 gene in T cells. *J. Biol. Chem.*, 279(45):46818–25, 2004.

[19] I. I. Ivanov, B. S. McKenzie, L. Zhou, C. E. Tadokoro, A. Lepelley, J. J. Lafaille, D. J. Cua, and D. R. Littman. The orphan nuclear receptor RORgammat directs the differentiation program of proinflammatory IL-17+ T helper cells. *Cell*, 126(6): 1121–33, 2006.

[20] I. I. Ivanov, L. Zhou, and D. R. Littman. Transcriptional regulation of Th17 cell differentiation. *Semin. Immunol.*, 19(6): 409–17, 2007.

[21] S. Kamiya, T. Owaki, N. Morishima, F. Fukai, J. Mizuguchi, and T. Yoshimoto. An indispensable role for STAT1 in IL-27-induced T-bet expression but not proliferation of naive CD4+ T cells. *J. Immunol.*, 173(6):3871–7, 2004.

[22] S.-i. Kano, K. Sato, Y. Morishita, S. Vollstedt, S. Kim, K. Bishop, K. Honda, M. Kubo, and T. Taniguchi. The contribution of transcription factor IRF1 to the interferon-gamma-interleukin 12 signaling axis and TH1 versus TH-17 differentiation of CD4+ T cells. *Nat. Immunol.*, 9(1):34–41, 2008.

[23] G. Kassiotis and A. O'Garra. Establishing the follicular helper identity. *Immunity*, 31(3):450–2, 2009.

[24] H. Kim, J. Imbert, and W. Leonard. Both integrated and differential regulation of components of the IL-2/IL-2 receptor system. *Cytokine Growth Factor Rev.*, 17(5):349–66, 2006.

[25] M. A. Koch, G. Tucker-Heard, N. R. Perdue, J. R. Killebrew, K. B. Urdahl, and D. J. Campbell. The transcription factor T-bet controls regulatory T cell homeostasis and function during type 1 inflammation. *Nat. Immunol.*, 10(6):595–602, 2009.

[26] N. Komatsu, M. E. Mariotti-Ferrandiz, Y. Wang, B. Malissen, H. Waldmann, and S. Hori. Heterogeneity of natural Foxp3+ T cells: a committed regulatory T-cell lineage and an uncommitted minor population retaining plasticity. *Proc. Natl. Acad. Sci. U.S.A.*, 106(6):1903–8, 2009.

[27] H.-K. Kwon, J.-S. So, C.-G. Lee, A. Sahoo, H.-J. Yi, J.-N. Park, S.-y. Lim, K.-C. Hwang, C.-D. Jun, J.-S. Chun, and S.-H. Im. Foxp3 induces IL-4 gene silencing by affecting nuclear translocation of NFkappaB and chromatin structure. *Mol. Immunol.*, 45(11):3205–12, 2008.

[28] A. Laurence, C. Tato, T. Davidson, Y. Kanno, Z. Chen, Z. Yao, R. Blank, F. Meylan, R. Siegel, L. Hennighausen, E. Shevach, and J. O'shea. Interleukin-2 signaling via STAT5 constrains T helper 17 cell generation. *Immunity*, 26(3):371–81, 2007.

[29] K. León, R. Peréz, A. Lage, and J. Carneiro. Modelling T-cell-mediated suppression dependent on interactions in multicellular conjugates. *J. Theor. Biol.*, 207(2):231–54, 2000.

[30] Y. Liu, P. Zhang, J. Li, A. Kulkarni, S. Perruche, and W. Chen. A critical function for TGF-beta signaling in the development of natural CD4(+)CD25(+)Foxp3(+) regulatory T cells. *Nat. Immunol.*, 2008.

[31] N. Manel, D. Unutmaz, and D. R. Littman. The differentiation of human T(H)-17 cells requires transforming growth factor-beta and induction of the nuclear receptor RORgammat. *Nat. Immunol.*, 9(6):641–9, 2008.

[32] M. McGeachy, K. Bak-Jensen, Y. Chen, C. Tato, W. Blumenschein, T. McClanahan, and D. Cua. TGF-beta and IL-6 drive the production of IL-17 and IL-10 by T cells and restrain T(H)-17 cell-mediated pathology. *Nat. Immunol.*, 8(12):1390–7, 2007.

[33] S. C. McKarns and R. H. Schwartz. Distinct effects of TGF-beta 1 on CD4+ and CD8+ T cell survival, division, and IL-2 production: a role for T cell intrinsic Smad3. *J. Immunol.*, 174(4):2071–83, 2005.

[34] L. Mendoza. A network model for the control of the differentiation process in Th cells. *BioSystems*, 84(2):101–14, 2006.

[35] L. Mendoza and I. Xenarios. A method for the generation of standardized qualitative dynamical systems of regulatory networks. *Theoretical biology & medical modelling*, 3:13, 2006.

[36] R. Moriggl, D. J. Topham, S. Teglund, V. Sexl, C. McKay, D. Wang, A. Hoffmeyer, J. van Deursen, M. Y. Sangster, K. D. Bunting, G. C. Grosveld, and J. N. Ihle. Stat5 is required for IL-2-induced cell cycle progression of peripheral T cells. *Immunity*, 10(2):249–59, 1999.

[37] T. R. Mosmann and R. L. Coffman. TH1 and TH2 cells: different patterns of lymphokine secretion lead to different functional properties. *Annu. Rev. Immunol.*, 7:145–73, 1989.

[38] A. Naldi, D. Berenguier, A. Fauré, F. Lopez, D. Thieffry, and C. Chaouiya. Logical modelling of regulatory networks with GINsim 2.3. *BioSystems*, 2009.

[39] A. Naldi, E. Remy, D. Thieffry, and C. Chaouiya. A reduction method of logical regulatory graphs preserving essential dynamical properties. *In Proc. CMSB 2009, Lect. Notes Comp. Sc.*, 5688:266–280, 2009.

[40] A. Naldi, D. Thieffry, and C. Chaouiya. Decision Diagrams for the Representation and Analysis of Logical Models of Genetic Networks. *In Proc. CMSB 2009, Lect. Notes Comp. Sc.*, 4695/2007:233–247, 2007.

[41] H. Park, Z. Li, X. O. Yang, S. H. Chang, R. Nurieva, Y.-H. Wang, Y. Wang, L. Hood, Z. Zhu, Q. Tian, and C. Dong. A distinct lineage of CD4 T cells regulates tissue inflammation by producing interleukin 17. *Nat. Immunol.*, 6(11):1133–41, 2005.

[42] A. Richter, M. Löhning, and A. Radbruch. Instruction for cytokine expression in T helper lymphocytes in relation to proliferation and cell cycle progression. *J. Exp. Med.*, 190(10):1439–50, 1999.

[43] A. Y. Rudensky, M. Gavin, and Y. Zheng. FOXP3 and NFAT: partners in tolerance. *Cell*, 126(2):253–6, 2006.

[44] J. Saez-Rodriguez, L. Simeoni, J. Lindquist, R. Hemenway, U. Bommhardt, B. Arndt, U. Haus, R. Weismantel, E. Gilles, S. Klamt, and B. Schraven. A logical model provides insights into T cell receptor signaling. *PLoS Comput. Biol.*, 3(8):e163, 2007.

[45] S. J. Szabo, A. S. Dighe, U. Gubler, and K. M. Murphy. Regulation of the interleukin (IL)-12R beta 2 subunit expression in developing T helper 1 (Th1) and Th2 cells. *J. Exp. Med.*, 185(5):817–24, 1997.

[46] D. Thieffry. Dynamical roles of biological regulatory circuits. *Brief. Bioinformatics*, 8(4):220–5, 2007.

[47] R. Thomas. On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations. *Springer series in Synergetics*, 9:180–193, 1981.

[48] R. Thomas, D. Thieffry, and M. Kaufman. Dynamical behaviour of biological regulatory networks–I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bull. Math. Biol.*, 57(2):247–76, 1995.

[49] Y. Tone, K. Furuuchi, and al. Smad3 and NFAT cooperate to induce Foxp3 expression through its enhancer. 2008.

[50] D. Q. Tran, H. Ramsey, and E. M. Shevach. Induction of FOXP3 expression in naive human CD4+FOXP3 T cells by T-cell receptor stimulation is transforming growth factor-beta dependent but does not confer a regulatory phenotype. *Blood*, 110(8): 2983–90, 2007.

[51] H.-J. van den Ham and R. J. de Boer. From the two-dimensional Th1 and Th2 phenotypes to high-dimensional models for gene regulation. *Int. Immunol.*, 20(10):1269–77, 2008.

[52] M. Veldhoen, C. Uyttenhove, J. van Snick, H. Helmby, A. Westendorf, J. Buer, B. Martin, C. Wilhelm, and B. Stockinger. Transforming growth factor-beta 'reprograms' the differentiation of T helper 2 cells and promotes an interleukin 9-producing subset. *Nat. Immunol.*, 9(12):1341–6, 2008.

[53] D. A. A. Vignali, L. W. Collison, and C. J. Workman. How regulatory T cells work. *Nat. Rev. Immunol.*, 8(7):523–32, 2008.

[54] A. V. Villarino, C. M. Tato, J. S. Stumhofer, Z. Yao, Y. K. Cui, L. Hennighausen, J. J. O'Shea, and C. A. Hunter. Helper T cell IL-2 production is limited by negative feedback and STAT-dependent cytokine signals. *J. Exp. Med.*, 204(1):65–71, 2007.

[55] C. Weaver, R. Hatton, P. Mangan, and L. Harrington. IL-17 family cytokines and the expanding diversity of effector T cell lineages. *Annu. Rev. Immunol.*, 25:821–52, 2007.

[56] Z. Yao, Y. Kanno, M. Kerenyi, G. Stephens, L. Durant, W. T. Watford, A. Laurence, G. W. Robinson, E. M. Shevach, R. Moriggl, L. Hennighausen, C. Wu, and J. J. O'Shea. Nonredundant roles for Stat5a/b in directly regulating Foxp3. *Blood*, 109(10):4368–75, 2007.

[57] Y. Zheng, A. Chaudhry, A. Kas, P. deRoos, J. M. Kim, T.-T. Chu, L. Corcoran, P. Treuting, U. Klein, and A. Y. Rudensky. Regulatory T-cell suppressor program co-opts transcription factor IRF4 to control T(H)2 responses. *Nature*, 458(7236):351–6, 2009.

[58] L. Zhou, I. Ivanov, R. Spolski, R. Min, K. Shenderov, T. Egawa, D. Levy, W. Leonard, and D. Littman. IL-6 programs T(H)-17 cell differentiation by promoting sequential engagement of the IL-21 and IL-23 pathways. *Nat. Immunol.*, 8(9):967–74, 2007.

[59] L. Zhou, J. E. Lopes, M. M. W. Chong, I. I. Ivanov, R. Min, G. D. Victora, Y. Shen, J. Du, Y. P. Rubtsov, A. Y. Rudensky, S. F. Ziegler, and D. R. Littman. TGF-beta-induced Foxp3 inhibits T(H)17 cell differentiation by antagonizing RORgammat function. *Nature*, 453(7192):236–40, 2008.

[60] X. Zhou, L. T. Jeker, B. T. Fife, S. Zhu, M. S. Anderson, M. T. McManus, and J. A. Bluestone. Selective miRNA disruption in T reg cells leads to uncontrolled autoimmunity. *J. Exp. Med.*, 205(9):1983–91, 2008.

Troisième partie

Discussion

Chapitre 10

Conclusions et perspectives

Ce travail comporte trois principaux volets : la mise au point de méthodes d'analyse pour le formalisme logique, l'implémentation de ces méthodes dans le logiciel de modélisation GINsim, et leur application à la modélisation de la différenciation des lymphocytes T auxiliaires. Plus précisément, il a débuté par l'implémentation de GINsim, un logiciel permettant la définition et la simulation de graphes de régulation. Des idées issues de la collaboration avec des modélisateurs sont rapidement venues enrichir cet outil : définition de classes de priorités, détermination des états stables et analyse de la fonctionnalité des circuits. Je suis ensuite devenu utilisateur de GINsim pour mon travail sur la différenciation des lymphocytes T auxiliaires, ce qui a inspiré la mise au point d'une méthode de réduction de modèles. Ce chapitre présente les principales conclusions et perspectives sur ces trois aspects : méthodologie, implémentation et application.

10.1 Manipulation de grands réseaux de régulation

Le fil conducteur de mon travail méthodologique est de faciliter la construction et l'analyse de grands graphes de régulation. Ce travail porte, d'une part, sur la réduction des graphes de régulation et de leur dynamique et, d'autre part, sur l'extraction d'informations sur la dynamique à partir des fonctions logiques. Nous avons commencé par introduire une méthode de mise à jour permettant de restreindre la région de l'espace des états générée lors d'une simulation. Cette méthode permet de combiner comportements synchrones et asynchrones, tout en prenant en compte des informations qualitatives sur les vitesses relatives des processus impliqués. Nous avons ensuite utilisé des diagrammes de décision pour représenter et manipuler les fonctions logiques décrivant le comportement dynamique des composants d'un modèle. En particulier, cela nous a permis d'exprimer, pour chaque composant, une contrainte de stabilité sous forme de diagramme de décision. La combinaison de ces contraintes donne la liste des états stables du modèle, sans énumération exhaustive.

10.2 Analyse des circuits de régulation

Basée sur la représentation des fonctions logiques en terme de diagrammes de décision, la méthode d'analyse des circuits présentée dans la section 6.3 permet l'identification des circuits de régulation pouvant jouer un rôle important dans la détermination des attracteurs. En effet, notre méthode détermine la région de l'espace des états dans laquelle un circuit

est susceptible de générer des oscillations ou encore de créer une séparatrice entre bassins d'attractions. Nous souhaitons affiner ces résultats afin de tirer des informations plus précises sur le nombre et le type des attracteurs.

Tout d'abord, il reste à affiner la gestion des court-circuits. On parle de court-circuit lorsqu'un des membres du circuit régule un membre du circuit autre que son successeur. Cela signifie qu'il existe un circuit plus court impliquant uniquement certains membres de ce circuit. Le composant responsable du court-circuit est alors à la fois un membre et un régulateur du circuit. Dans le cas où les seuils de ses interactions sortantes vers les autres membres du circuits sont différents, le court-circuit n'est pas gênant. Dans le cas où les seuils sont confondus, nous avons choisi d'appliquer une contrainte forte, spécifiant que le circuit doit être fonctionnel de part et d'autre des seuils de chacune de ses interactions. Cette contrainte élimine en général un grand nombre de circuits ne jouant aucun rôle significatif mais il est possible que dans certains cas, encore à préciser, elle élimine également des circuits importants.

D'autre part, le contexte de fonctionnalité obtenu pour un circuit implique uniquement ses régulateurs directs, ce qui rend leur interprétation difficile. Il serait intéressant de remonter vers leurs propres régulateurs afin de donner une information plus précise. Dans le cas (idéal mais peu courant en pratique) de modèles fortement hiérarchiques, il devrait être possible d'exprimer les contextes de fonctionnalité directement en fonction des entrées du modèle. Dans le cas général, il reste à préciser jusqu'où remonter pour rendre le contexte aussi informatif que possible.

Plus généralement, nous souhaitons utiliser les informations sur les circuits pour déterminer le nombre et la nature des attracteurs d'un modèle. L'interprétation des contextes de fonctionnalité nous permet souvent d'obtenir une vue d'ensemble, ou de construire une liste d'attracteurs possibles. Pour améliorer les prédictions, il est nécessaire de définir et d'identifier les séparatrices dans l'espace des états. L'extension des contextes de fonctionnalité vers les régulateurs de plus haut niveau est un premier pas dans cette direction, mais l'analyse des circuits pris indépendamment est insuffisante. Il est nécessaire de croiser les résultats obtenus afin de voir quels circuits collaborent ou entrent en conflit dans la définition de ces séparatrices. À partir de ces contextes de fonctionnalité, il devrait être possible de déterminer une partition du graphe de transitions d'états telle que chaque partition contienne au maximum un attracteur dont on connaisse le type (état stable ou attracteur complexe) et la signature (composants stables ou oscillants). En combinant cette information avec la recherche d'états stables, on peut déterminer les zones susceptibles de contenir des attracteurs cycliques.

La construction du graphe des composantes fortement connexes (CFC) peut aider à hiérarchiser les circuits. Si l'on peut déterminer le comportement de chaque CFC en fonction de ses entrées, on peut alors déterminer celui du graphe dans son ensemble. En pratique, les modèles que nous étudions contiennent généralement une CFC principale regroupant la plupart des circuits. À l'intérieur d'une CFC, les circuits courts jouent souvent un rôle plus important que les autres. En effet, ils ont en moyenne moins de régulateurs et par conséquent un contexte de fonctionnalité moins contraint.

10.3 Réduction de modèle

La détermination des états stables et l'analyse des circuits sont utiles pour évaluer le nombre et la nature des attracteurs du modèle dont les conditions d'atteignabilité doivent ensuite être établies. Nous avons proposé une méthode de réduction visant notamment à faciliter les analyses d'atteignabilité. Cette méthode, présentée dans la section 7.2 permet de supprimer itérativement des composants dans le graphe de régulation. Afin de masquer un composant tout en conservant l'essentiel des propriétés dynamiques, l'effet de ses régulateurs est reporté sur ses cibles.

10.3.1 Impact de l'ordre

Lorsque l'on supprime plusieurs composants, il est important de savoir si l'ordre choisi a un impact sur le modèle réduit. Voici quelques raisons de penser que lorsque plusieurs ordres sont possibles, la combinaison de réductions est commutative, c'est-à-dire que la réduction d'une série de composants dans plusieurs ordres donne le même résultat.

Afin de définir le comportement dynamique du graphe de régulation obtenu après une série de réductions, on souhaite étendre les notions de projection et de représentant de classe définies dans [80]. Il est facile de vérifier que l'état réduit obtenu après une série de projections ne dépend pas de l'ordre dans lequel ces projections sont effectuées. La projection π et la relation d'équivalence que nous avons introduites peuvent donc s'étendre à un ensemble de composants. Notre réduction repose sur l'existence d'un unique état représentant dans chaque classe d'équivalence, à partir duquel sont définies les transitions sortant de l'état réduit. De la même manière, on peut définir un représentant pour une classe d'équivalence étendue, comme un état qui soit représentant de classe pour chacun des composants que l'on souhaite supprimer. En effet, un tel état sera sélectionné comme représentant de classe pour chacune des réductions successives, quel que soit l'ordre choisi. Si cet état n'existe pas ou n'est pas unique, alors la suppression de cet ensemble de composants n'est pas possible.

Dans certains cas, la suppression d'un composant ne peut se faire qu'après la suppression d'un autre composant (voir [80] ainsi que la remarque à ce sujet dans la section suivante). Ce type de blocage peut être contourné lors de suppressions multiples. Lorsque la suppression d'un composant est impossible, on le déplace à la fin de la liste et on passe au suivant. Les composants restant à supprimer seront donc considérés avant que l'on ne revienne à celui-ci. La réduction de l'ensemble de composants sélectionnés est impossible lorsque la liste ne contient plus que des composants bloquants.

Le comportement dynamique obtenu après réduction est comparable à celui obtenu en utilisant des classes de priorités. Il est donc possible de définir des classes de priorités correspondant à une série de réductions. Si la réduction est indépendante de l'ordre, alors l'ensemble des définitions de classes de priorités correspondant à des réductions dans des ordres différents des mêmes composants sont "équivalentes". En effet, le détail des comportements dynamiques obtenus avec ces classes de priorités peut varier, mais les attracteurs atteignables sont comparables. Étant donné un ensemble de composants que l'on peut supprimer, toutes les définitions de classes de priorités telles que ces composants sont dans une ou plusieurs classes prioritaires sur la classe asynchrone contenant tous les autres composants donnent des comportements dynamique équivalents du point de vue des composants non-prioritaires.

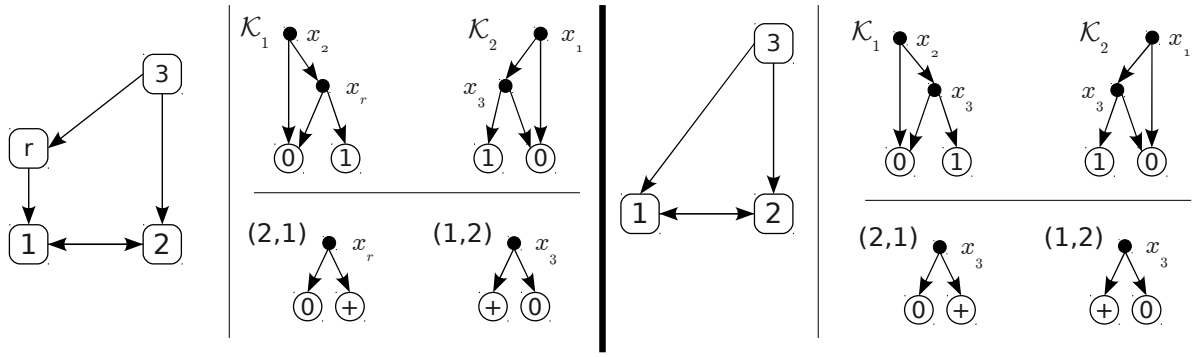


Figure 10.1: Réduction et suppression de circuits. Dans le graphe de régulation donné à gauche, le circuit (g_1, g_2) est fonctionnel en présence de r et en absence de g_3. Dans le graphe obtenu après suppression de r (à droite), l'effet de r sur le circuit est reporté sur g_3. L'intersection des contextes de fonctionnalité des deux interactions est alors vide.

10.3.2 Impact sur les circuits

Il est également intéressant d'étudier l'impact de la méthode de réduction sur les circuits et leurs contextes de fonctionnalité. En effet, l'un des objectifs de cette méthode est de conserver les circuits fonctionnels. Comme le calcul des contextes de fonctionnalité ne prend en compte que les membres du circuit et leurs régulateurs directs, la réduction peut affecter le contexte de fonctionnalité d'un circuit de deux manières :

- Suppression d'un des membres du circuit. Le circuit est alors "raccourci", une seule interaction remplace les deux interactions entourant le composant supprimé. Le contexte de fonctionnalité des autres interactions est inchangé (sauf en cas de court-circuit, voir paragraphe suivant) et le contexte de la nouvelle interaction est identique à l'intersection des contextes des deux interactions d'origine (dans le cas Booléen, il faut encore s'en assurer dans le cas multi-valué). Dans ce cas, le contexte de fonctionnalité du circuit reste inchangé.
- Suppression d'un régulateur du circuit (qui peut en être membre dans le cas de court-circuits). Le contexte prend alors en compte des régulateurs plus lointains du circuit. On peut noter que ceci correspond justement à l'une des améliorations que l'on souhaite apporter à la définition des contextes de fonctionnalité. La réduction peut alors relaxer ou restreindre, voire supprimer complètement, le contexte de fonctionnalité d'un circuit (voir figure 10.1). En effet, les contraintes concernant le composant supprimé sont remplacées par des contraintes sur ses régulateurs.

La perte du contexte de fonctionnalité d'un circuit lors de la suppression d'un de ses régulateurs est à l'origine des cas dans lesquels un composant ne peut être supprimé que si un autre l'a été auparavant. En effet, un circuit fonctionnel peut se réduire jusqu'à ne comporter qu'un seul composant. Celui-ci est alors impossible à supprimer car porteur d'une auto-régulation fonctionnelle. Si la suppression d'un de ses régulateurs rend l'auto-régulation non-fonctionnelle, la suppression de cet élément devient possible. Il est important de noter que cette propriété est liée au problème évoqué plus haut concernant le calcul des contextes de fonctionnalité : seuls les régulateurs directs sont considérés. Des contextes approfondis permettraient de détecter plus tôt que le circuit en question n'est fonctionnel que dans des zones "instables" de l'espace des états.

10.3.3 Projections alternatives

Dans notre méthode de réduction, nous définissons la dynamique réduite comme une projection de la dynamique d'origine. Si la projection des états est unique, il est envisageable de choisir les transitions projetées de plusieurs manières. Celle que nous avons choisie permet de formaliser et d'automatiser les réductions manuelles appliquées précédemment. Cette projection supprime certains chemins dans le graphe de transitions d'états. La dynamique obtenue est donc une sous-approximation de la dynamique d'origine. Dans ce type d'approximation, un état atteignable dans la version réduite l'est également dans la version d'origine, mais l'inverse n'est pas toujours vrai.

On peut également définir des projections qui sur-approximent la dynamique. Un état atteignable dans la version d'origine le reste alors dans la version projetée. Une sur-approximation est obtenue si l'on conserve une transition entre x' et y' dès qu'il existe une transition entre au moins un x et un y tels que $\pi(x) = x'$ et $\pi(y) = y'$. Cette projection ne peut pas être représentée dans le formalisme logique que nous utilisons car elle nécessite qu'un composant puisse avoir plusieurs niveaux cibles dans un état donné.

10.3.4 Réductions conservatives

La sélection des composants à supprimer lors d'une réduction peut se baser sur plusieurs critères. Si l'objectif est de conserver autant que possible le comportement dynamique, il est intéressant de vérifier que la suppression d'un composant ne modifie pas les propriétés d'atteignabilité. Si le représentant de chaque classe d'équivalence a exactement les mêmes transitions sortantes que les autres états de la classe, alors on est certain que la dynamique est inchangée. Cette propriété n'est vérifiée que si le composant supprimé n'a aucune cible. Il est donc possible de supprimer itérativement les composants de sortie d'un modèle sans aucune modification de son comportement dynamique.

Les propriétés d'atteignabilité sont vraisemblablement conservées dans d'autres cas, en particulier lors de la suppression d'intermédiaires dans de longues chaînes. En effet, le délai entre le changement de niveau de ses régulateurs et celui du composant supprimé peut se reporter sur ses cibles sans que cela ne perturbe la dynamique globale.

Il serait intéressant de définir formellement les conditions sous lesquelles une réduction conserve les propriétés d'atteignabilité des attracteurs, de manière à garantir qu'une réduction donnée est conservative, voire proposer directement de telles réductions.

10.4 Dynamique compacte

Les méthodes de réduction présentées ici visent à réduire la taille de la région de l'espace des états que l'on doit étudier. Il est également possible de réduire la représentation de cet espace, en particulier à l'aide de schémas représentant des ensembles d'états [5, 122]. On peut ainsi représenter un ensemble d'états de manière compacte, mais l'on perd les informations sur les transitions entre ces états. Afin de conserver une vue d'ensemble du comportement dynamique, il est possible d'utiliser cette méthode pour représenter le graphe des composantes fortement connexes (CFC) du graphe de transitions d'états. L'ensemble d'états composant chaque CFC est alors représenté par un diagramme de décision, et les

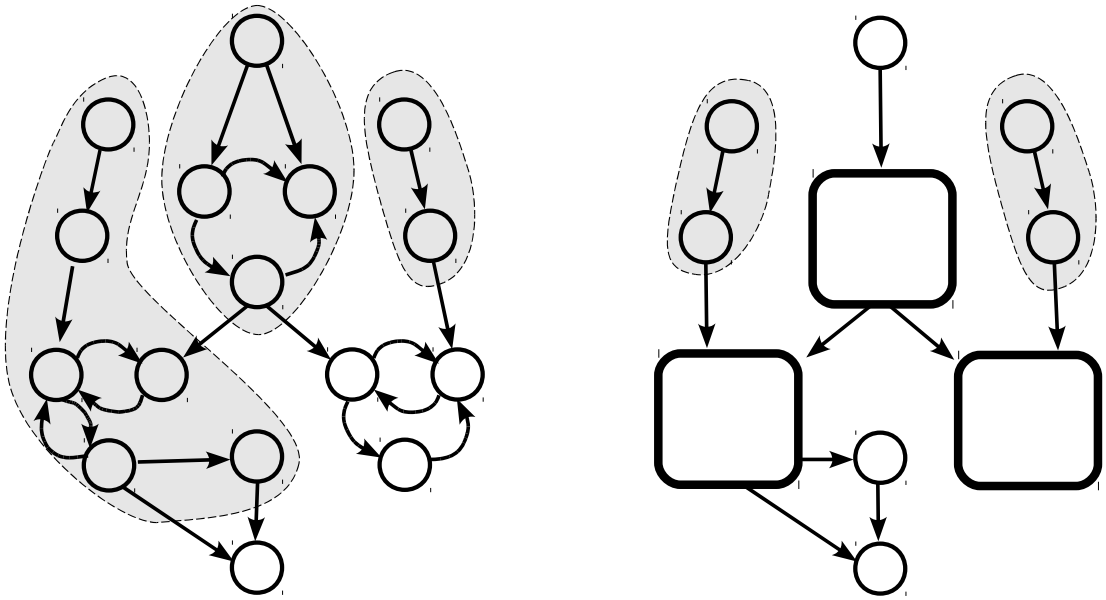


Figure 10.2: Représentation compacte de la dynamique. A gauche, un graphe de transitions d'états ayant deux attracteurs : un état stable et un attracteur cyclique. Les autres états sont regroupés en bassins d'attraction (mis en évidence sur fonds colorés), spécifiques à l'un de ces attracteurs ou pouvant mener aux deux. A droite le graphe des composantes fortement connexes correspondant. On peut construire une représentation plus compacte en regroupant également les états transients menant aux mêmes ensembles d'états (mis en évidence dans le graphe des CFC), voire les bassins d'attraction dans leur ensemble.

transitions entre états de CFC différentes deviennent des transitions entre les CFC correspondantes. GINsim propose déjà ce type de représentation (sans la compaction de la liste des états au sein de chaque CFC), mais le graphe des CFC est déterminé sur la base du graphe de transitions d'états. Cette représentation nécessite donc des calculs et de l'espace mémoire conséquents. Pour l'appliquer à de grands graphes, il serait préférable de calculer directement le graphe des CFC pendant la simulation, plutôt que de construire le graphe de transitions d'états.

Duncan Berenguier travaille actuellement sur une méthode de simulation regroupant à la volée les états en CFC. Nous souhaitons intégrer d'autres critères de regroupement permettant par exemple de compacter les chemins menant aux mêmes ensembles d'états. Selon le degré de détail que l'on souhaite conserver, ces regroupements peuvent être étendus aux CFC transientes. Deux groupes A et B (qui peuvent être des états isolés, des ensembles de chemins acycliques ou des CFC) peuvent être fusionnés si B fait partie des successeurs de A et que tous les autres successeurs de A sont également des successeurs de B (voir figure 10.2).

10.5 Compositions

Afin de faciliter la définition de modèles complexes, nous travaillons sur la définition d'un modèle composé à partir d'un ensemble de sous-modèles. Plusieurs modèles logiques évoqués dans la section 2.2.4 impliquent plusieurs cellules. Dans ces réseaux, chaque cellule contient le même réseau de régulation et certains composants d'une cellule agissent sur ceux de cellules voisines. Ces réseaux multi-cellulaires ont été construits manuellement, en répliquant le réseau intra-cellulaire et en ajoutant des interactions entre cellules voisines.

Les fonctions logiques des cibles de ces interactions inter-cellulaires doivent ensuite être adaptées pour prendre en compte ces nouvelles interactions. Le modèle résultant est ensuite traité comme un modèle logique normal. Si l'on souhaite ensuite modifier le modèle intra-cellulaire, il faut soit recommencer cette opération, soit appliquer le même changement dans chaque cellule. Une méthode de définition compositionelle faciliterait la création et la manipulation de tels modèles, mais aussi la mise au point d'outils d'analyse tirant partie de la réutilisation du même réseau.

Le cas le plus favorable est celui dans lequel chaque modèle a des entrées et sorties bien identifiées, qui servent de lien entre les différents modèles élémentaires. Il est alors possible de définir un "meta-modèle" : un nouveau graphe dans lequel chaque noeud représente un modèle élémentaire et les arcs représentent des liens entre ces modèles. Les composants d'entrée d'un modèle élémentaire intègrent alors les effets des composants de sortie de ceux situés en amont. Cette approche s'applique naturellement à la définition de modèles multi-cellulaires, que chaque cellule soit de même type ou non. Elle peut également s'appliquer dans le cas de modèles intégrant plusieurs voies de signalisation différentes, chacune étant modélisée séparément. Claudine Chaouiya collabore avec Hanna Klaudel et Franck Pommereau sur la définition de tels meta-modèles en utilisant les réseaux de Petri de haut niveau. Gregory Batt travaille sur une approche similaire, basée sur du model-checking compositionnel, au sein de l'équipe Contraintes[1].

Le problème se complique lorsque les modèles élémentaires partagent des composants autres que les entrées/sorties. C'est notamment le cas dans le modèle du cycle cellulaire de la levure construit à partir de trois sous-modèles par Adrien Fauré (voir [35], reproduit en annexe A.4). La topologie du modèle composé est facile à déterminer, mais les seuils et les fonctions logiques des composants partagés sont plus délicats à établir.

Ce type de composition peut se ramener au cas précédemment décrit en éclatant chaque composant partagé en un composant de sortie et un composant d'entrée. Un modèle de liaison, contenant tous les composants partagés, est utilisé pour définir les modalités d'intégration (voir figure 10.3). Ces trois modèles peuvent être induits automatiquement à partir des deux modèles à coupler, il reste à définir les fonctions d'intégration dans le modèle de liaison. On peut alors définir un modèle composé dans lequel chaque composant partagé est éclaté en plusieurs composants (les entrées et sorties de chaque modèle élémentaire ainsi que les composants d'intégration). L'application de la méthode de réduction permet ensuite de supprimer les composants artificiels et d'obtenir une version lisible et compacte du modèle composé.

10.6 Vers des modèles plus quantitatifs

Le formalisme logique que nous utilisons décrit qualitativement le comportement d'un système. Cette approche ne permet pas de répondre à toutes les questions que l'on peut se poser sur le système étudié. Une fois le comportement qualitatif bien défini, il peut être nécessaire d'affiner le modèle pour prendre en compte des aspects quantitatifs. Il est bien entendu possible de redéfinir le modèle dans un autre formalisme mais il serait préférable de se servir du modèle logique existant comme base pour le nouveau modèle. Afin de réutiliser des outils existants, nous souhaitons traduire les modèles logiques dans des formats supportés par d'autres outils de modélisation. Plusieurs types de formalismes peuvent être

[1] INRIA, Rocquencourt

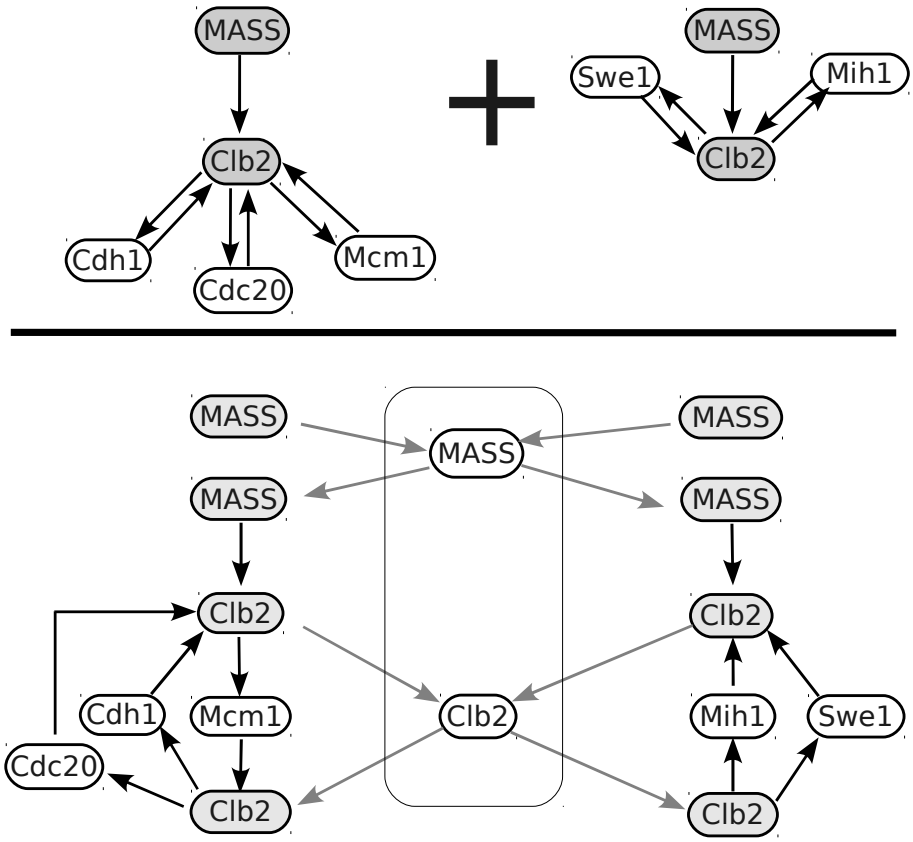


Figure 10.3: Construction d'un modèle composé. En haut : deux sous-modèles partageant deux composants (grisés). En bas : le modèle éclaté utilisé pour la composition. Chaque composant partagé des modèles d'origine est alors représenté par 5 composants. Les règles de compositions sont définies par les fonctions logiques des composants du modèle de liaison (au milieu). Une réduction des composants grisés donne le modèle composé final.

utilisés pour affiner un modèle logique. Le choix du formalisme dépend des questions auxquelles on souhaite répondre.

Dans certains cas, on souhaite affiner la dynamique par la prise en compte de délais. Il est possible d'ajouter des délais sur les transitions d'un réseau de Petri mais aussi d'utiliser une extension du formalisme logique, comme proposé par Siebert et Bockmayr [107].

Il peut également être intéressant d'ajouter des probabilités aux différentes transitions, en particulier afin de déterminer non pas seulement les différents attracteurs atteignables mais la probabilité d'atteindre chacun d'entre eux. Ce type d'information serait très utile pour l'étude d'une population cellulaire (voir section 10.8.2).

L'export existant vers des formats de réseaux de Petri peut également servir de base pour raffiner un modèle. En effet, ce formalisme permet, à l'aide d'extensions, d'utiliser plusieurs types d'approches (stochastique, différentielle, hybride). La structure du réseau est alors conservée mais sa dynamique est largement redéfinie.

Un export vers le format du logiciel de modélisation GNA permet l'utilisation de simulations numériques basées sur un système d'équations linéaires par morceaux. GNA est également un outil de modélisation qualitative, utilisant un formalisme proche du formalisme logique mais permettant d'étudier le comportement des états sur seuils.

Enfin, plusieurs méthodes de traduction de réseaux logiques en systèmes d'équations différentielles ordinaires, avec des termes de forme prédéfinie, ont été récemment proposées [30, 123].

10.7 GINsim

Une grande partie de mon travail a porté sur le logiciel de modélisation GINsim. L'objectif de ce projet est de faciliter le développement et l'analyse de modèles logiques, d'une part en proposant une interface conviviale et, d'autre part, en facilitant l'intégration de nouvelles méthodes et l'interopérabilité avec d'autres outils. Des fonctionnalités sont régulièrement ajoutées à GINsim, pour répondre aux besoins soulevés par les modèles sur lesquels nous travaillons. GINsim est à la fois un projet de recherche en lui-même et un outil pour d'autres membres de l'équipe ainsi que pour d'autres groupes avec lesquels nous collaborons (en particulier : Institut Curie[2], DKFZ[3], CIB[4]).

Bien entendu nous souhaitons continuer à améliorer l'interface utilisateur ainsi que les performances. Afin de compléter l'outil de simulation, nous souhaitons également prendre en charge les méthodes de mise à jour séquentielle et bloc-séquentielle. Ceci ne pose *a priori* pas de problème technique : le moteur de simulation actuel permet l'ajout de nouvelles méthodes de mise à jour.

Sur le plan de l'ergonomie, le principal problème est l'absence d'annuler-refaire. Cette fonction peut être mise en place selon deux approches différentes. La première consiste à copier la totalité du document édité à chaque modification. Cette copie peut se faire en mémoire ou sur le disque (sauvegardes automatiques). Elle a l'avantage d'être relativement simple à mettre en place au sein d'un logiciel existant mais occupe une grande place mémoire pour les documents de taille importante et risque de ralentir fortement l'édition. La seconde

[2] Laurence Calzone, Paris
[3] Özgür Sahin, German Cancer Research Center, Heidelberg
[4] Lucas Sánchez, Centro de Investigaciones Biológicas, Madrid.

approche consiste à définir, pour chaque opération modifiant le document, une opération inverse. Cette approche est bien plus complexe à mettre en place car elle nécessite de recenser chaque modification possible. Si l'une d'elles n'est pas prise en compte il existe un risque de créer une version incohérente du document. D'autre part, certaines actions effectuées dans GINsim déclenchent d'autres changements (par exemple la suppression d'une interaction déclenche la mise à jour des fonctions logiques de sa cible). Cette seconde approche est bien meilleure mais plus difficile à intégrer.

De nombreuses fonctionnalités dépendent d'interactions avec l'utilisateur, ce qui a compliqué l'ajout d'un mode automatique (scripts jython). Une refonte de l'architecture de GINsim permettrait de mieux séparer les fonctionnalités de l'interface graphique. Les scripts auraient ainsi accès à l'ensemble des fonctionnalités.

Avant d'encourager plus largement l'écriture de scripts pour GINsim, il est également nécessaire de stabiliser et de documenter au moins une partie de l'API. Une API stable nous permettrait aussi d'encourager le développement de plugins externes, ce qui faciliterait le partage de ressources avec les équipes travaillant sur des problématiques similaires.

Enfin, le format GINML, utilisé par GINsim, béneficierait de quelques adaptations, en particulier pour intégrer l'utilisation de fonctions logiques et permettre des annotations plus riches.

L'implémentation actuelle des MDD étant assez naïve, un travail d'optimisation pourrait améliorer les performances. Des gains de performance peuvent également être obtenus en améliorant l'ordre des combinaisons de diagrammes. Enfin, lorsqu'on souhaite combiner de nombreux diagrammes, il est possible de travailler directement sur la liste de diagrammes au lieu de les prendre en compte l'un après l'autre. Un prototype en Python utilisant une autre implémentation naïve donne des résultats très encourageants. Une approche alternative consiste à adapter les algorithmes pour n'utiliser que des contraintes Booléennes afin d'utiliser une implémentation existante (en particulier JavaBDD et ses passerelles vers plusieurs implémentations natives). Cette seconde approche est limitée aux opérations classiques sur les diagrammes, ce qui la rend plus complexe à mettre en place et risque de limiter les gains de performance.

Au sein de notre laboratoire, plusieurs chercheurs travaillent sur l'analyse de données de génomique fonctionnelle (essentiellement des données de transcriptome et d'interactome) et sur la construction d'une base de données de signatures transcriptionnelles, mettant en évidence des gènes souvent co-exprimés. Ces données sont accessibles à travers l'outil Transcriptome Browser [67]. L'intégration de cet outil avec GINsim permettra d'identifier de bons candidats pour l'extension d'un modèle.

On pourrait envisager la prise en compte d'incertitudes dans un modèle, en particulier en associant, dans certaines conditions, plusieurs valeurs cibles à une fonction logique. La prise en compte de ces incertitudes semble facile dans les simulations asynchrones, qui sont déjà non-déterministes. Les autres modes de simulation et les méthodes d'analyse nécessitent d'être adaptés. En attendant, un modèle avec incertitudes peut servir de base pour la génération d'un ensemble de modèles "classiques". Des modèles logiques partiellement paramétrisés ont déjà été utilisés pour des techniques d'inférence ou de sélection de modèles satisfaisant un ensemble de contraintes [9, 23].

10.8 Modèle de la différenciation des Th

Partant d'un modèle existant de la différenciation Th1/Th2, j'ai intégré des données de la littérature pour recouvrir les lignées Th17 et Treg, récemment mises en évidence. Comportant 65 composants, ce modèle s'est révélé difficile à analyser par des méthodes classiques.

10.8.1 Extension du modèle

Bien que nous ayons construit un modèle relativement détaillé, il est impossible de prendre en compte chaque détail du système réel. Nous avons donc choisi de simplifier certaines parties du modèle. Ces simplifications peuvent empêcher l'étude de mécanismes impliquant une perturbation d'un composant masqué dans le modèle. Par exemple, la plupart des patients atteint par la maladie de Vasquez (et d'autres maladies myéloprolifératives) ont une mutation du gène codant pour JAK2 et un dérèglement de l'équilibre entre les différentes populations Th. Les Janus Kinases (JAK) sont impliquées dans l'activation des STAT par les récepteurs de cytokine. L'étude de l'impact de cette mutation sur le comportement du modèle nécessite la prise en compte explicite des JAK.

D'autres simplifications ont été apportées au modèle pour permettre son analyse en terme d'états stables. En particulier, l'arrêt des stimulations (APC, cytokines) et certains rétrocontrôles négatifs (par les protéines SOCS [125]) ne sont pas pris en compte explicitement. Le fait de réutiliser les états stables comme états initiaux pour d'autres environnements rend compte de certains changements d'environnements, mais ceux-ci peuvent se produire avant que le système n'atteigne un état stable. L'étude d'un modèle plus réaliste de ce point de vue nécessite vraisemblablement la prise en compte de délais pour éliminer les trajectoires artefactuelles.

La mise en évidence des Th17 et des Treg est récente et les mécanismes contrôlant leur différenciation sont imparfaitement connus. D'autres acteurs seront vraisemblablement mis en évidence dans les années à venir. De nouveaux acteurs potentiels peuvent être sélectionnés par l'analyse de données de génomique fonctionnelle, comme cela a déjà été fait pour la différenciation Th1/Th2 [68]. L'intégration de Transcriptome Browser et de GINsim (voir section précedente) peut faciliter l'interrogation de ces données. Malheureusement, peu d'expériences portant sur des populations lymphocytaires purifiées sont disponibles pour le moment, mais des travaux sont en cours dans cette direction [48].

La prise en compte de l'effet des cellules régulatrices sur les autres lymphocytes nécessite l'étude des voies de signalisation de cytokines supplémentaires, en particulier IL-35, mais aussi IL-10, et TGFb, présentes dans le modèle, mais dont le rôle inhibiteur n'est pas pris en compte. De plus en plus de données sont disponibles sur le rôle de ces cytokines ainsi que sur les autres effets suppresseurs des cellules régulatrices, mais de nombreuses questions subsistent sur leur mode d'action [119].

Certaines études indiquent un rôle possible pour les petits ARN, en particulier dans le fonctionnement des cellules régulatrices [127]. De façon générale, de nombreuses études indiquent que ces ARN jouent un rôle majeur dans de nombreux processus de régulation génique. L'intégration de ces acteurs au sein d'un modèle semble cependant encore prématuré, leur rôle précis étant encore inconnu. En effet, si l'étude mentionnée montre que la suppression de ces mécanismes dans les cellules régulatrices leur fait perdre leur fonction suppressive, on ne connaît pas le niveau auquel ils agissent. Ils peuvent être nécessaires

pour des mécanismes cellulaires généraux et la perte de la fonction suppressive ne serait alors qu'un effet de bord. De façon générale, une meilleure prise en compte des "nouveaux" mécanismes de régulation (ARN, modifications de l'ADN, conformation de la chromatine) serait utile dans de nombreux modèles. Dans notre modèle, nous avons notamment pris en compte, de façon très simplifiée, l'accessibilité de certains promoteurs. Une connaissance approfondie des mécanismes sous-jacents permettrait de mieux les prendre en compte au sein de modèles dynamiques.

10.8.2 Extension de l'analyse

L'analyse de notre modèle a permis d'identifier tous ses états stables ainsi que les principaux circuits positifs impliqués dans la génération de ces états stables alternatifs. Nous avons ensuite défini une version réduite du modèle afin d'étudier l'atteignabilité de ces états stables. L'environnement cellulaire (APC et cytokines présentes) déclenche et oriente cette différenciation. Afin d'étudier son impact, nous avons défini un ensemble de conditions environnementales. En partant d'un état initial correspondant à une cellule Th0, nous avons pu atteindre, en fonction de l'environnement, des états stables correspondants aux quatre lignées cellulaires attendues. Chaque état stable atteint sert à son tour d'état initial pour chaque environnement, jusqu'à ce qu'aucun nouvel état stable ne soit trouvé. Ces simulations successives ont mis en évidence une diversité dans les signatures cellulaires au delà des cinq lignées considérées *a priori*.

Il est malheureusement difficile d'obtenir des données expérimentales fiables sur le comportement des différentes lignées cellulaires. En effet, les études *in vitro* sont peu représentatives et les données *in vivo* portent sur des populations hétérogènes. Ces populations cellulaires sont généralement triées par cytométrie de flux (FACS) pour connaître leur composition. Il est difficile d'interpréter les changements de l'équilibre entre les différentes sous-populations qui peuvent être dues à de multiples causes (prolifération, mort cellulaire, différenciation). De plus, les filtres permettant d'étudier une sous-population précise sont encore imparfaits et les sous-populations identifiées sont encore hétérogènes. La forte variabilité et la plasticité observées ici peuvent paraître surprenantes face aux lignées cellulaires considérées comme bien établies. Des résultats récents, encore controversés, présentent cependant une vue plus mitigée, en particulier pour les cellules régulatrices, dont il pourrait exister plusieurs variantes répondant aux mêmes stimuli que certaines cellules effectrices [60, 126]. D'autres résultats semblent indiquer que ces mêmes cellules régulatrices peuvent perdre leur activité suppressive, voire devenir des cellules effectrices [32]. Nos prédictions peuvent donc refléter une réelle variabilité de comportement, ou le manque de connaissance des mécanismes empêchant cette diversité *in vivo*.

Nos résultats se concentrent sur le devenir d'une cellule en fonction de l'environnement. Or les différents acteurs du système immunitaire sécrètent des cytokines dans cet environnement et modifient ainsi l'équilibre entre les populations des différents types de lymphocytes présents. Afin d'étudier la réponse immunitaire dans son ensemble, il est donc important de prendre en compte les différentes populations cellulaires et leur influence sur l'environnement. La prise en compte de la prolifération et de la mort cellulaire programmée (apoptose) est essentielle. Notre modèle contient déjà un composant représentant le déclenchement de la prolifération cellulaire mais ne considère pas son arrêt. Ce composant sert essentiellement de feu vert pour la synthèse de cytokines via un effet (supposé) sur l'accessibilité de leur promoteurs. L'arrêt de la prolifération n'annule pas forcément cet effet, il faut donc probablement découpler ces deux paramètres. L'apoptose n'est pas encore

prise en compte dans notre modèle, bien que plusieurs études montrent que celle-ci peut jouer un rôle important dans la sélection de lymphocytes non auto-réactifs, ainsi que dans l'arrêt de la réponse immunitaire [27, 83].

Il est tentant d'utiliser une approche compositionnelle (voir section 10.5) pour générer un meta-modèle couvrant une population des différents types cellulaires. Au vu de la taille de notre modèle, cette approche n'est probablement pas applicable sans le développement de méthodes d'analyse dédiées. De plus, un modèle logique n'est peut-être pas le plus adapté pour prendre en compte une grande population cellulaire hétérogène.

Nous pouvons envisager une approche hybride dans laquelle le système est décrit de façon quantitative (concentrations des différentes cytokines et nombre de cellules de chaque type), mais le comportement des cellules dépend d'un modèle comme celui que nous proposons ici. Les entrées de ce modèle dépendent de l'environnement. Comme il n'est pas réaliste de considérer que chaque cellule a le même environnement, les niveaux de leurs composants d'entrées peuvent être déterminés de manière stochastique en fonction de l'environnement. Dans ce cadre, l'ajout de probabilités ou d'informations temporelles sur les différentes transitions permettrait d'affiner les résultats et d'obtenir de meilleures prédictions quantitatives sur l'équilibre entre les différentes populations.

Un modèle complet nécessite également l'intégration d'autres types cellulaires, comme les cellules dendritiques et les macrophages (APC). Pour les prendre en compte, il est sans doute plus réaliste, pour le moment, d'utiliser des modèles différents que d'intégrer le comportement de nombreux types cellulaires au sein d'un même modèle.

Bibliography

[1] W. Abou-Jaoudé, D. A. Ouattara, et M. Kaufman. From structure to dynamics: frequency tuning in the p53-Mdm2 network I. Logical approach. *J. Theor. Biol.*, 258(4):561–77, 2009.

[2] H. Alla et R. David. Continuous and hybrid Petri nets. *Journal of Circuits, Systems, and Computers*, 8: 159–188, 1998.

[3] M. Antoniotti, A. Policriti, N. Ugel, et B. Mishra. Model building and model checking for biochemical processes. *Cell Biochem. Biophys.*, 38(3):271–86, 2003.

[4] J. Aracena, E. Goles, A. Moreira, et L. Salinas. On the robustness of update schedules in Boolean networks. *BioSystems*, 97(1):1–8, 2009.

[5] R. J. Bagley et L. Glass. Counting and classifying attractors in high dimensional dynamical systems. *J. Theor. Biol.*, 183(3):269–84, 1996.

[6] G. Balbo. Introduction to Generalized Stochastic Petri Nets. *In Proc. SFM 2007, Lect. Notes Comput. Sci.*, 4486:83–131, 2007.

[7] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page, et D. Schneider. Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in Escherichia coli. *Bioinformatics*, 21 Suppl 1:i19–28, 2005.

[8] C. Bergmann, J. L. van Hemmen, et L. A. Segel. How instruction and feedback can select the appropriate T helper response. *Bull. Math. Biol.*, 64(3):425–46, 2002.

[9] G. Bernot, J. Comet, A. Richard, et J. Guespin. Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *J. Theor. Biol.*, 229(3):339–47, 2004.

[10] B. Bollig et I. Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Trans. Comput.*, 45: 993–1002, 1996.

[11] B. Bollig, M. Lobbing, et I. Wegener. On the effect of local changes in the variable ordering of ordered decision diagrams. *Inform. Process. Lett.*, 59:233–239, 1996.

[12] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Comput.*, 35:677–91, 1986.

[13] T. Bultan. BDD vs. Constraint-Based Model Checking: An Experimental Evaluation for Asynchronous Concurrent Systems. *In Proc. TACAS 2000, Lect. Notes Comput. Sci.*, 1785:441–455, 2000.

[14] J. Carneiro, K. Leon, I. Caramalho, C. van den Dool, R. Gardner, V. Oliveira, M.-L. Bergman, N. Sepúlveda, T. Paixão, J. Faro, et J. Demengeot. When three is not a crowd: a Crossregulation model of the dynamics and repertoire selection of regulatory CD4+ T cells. *Immunol. Rev.*, 216:48–68, 2007.

[15] F. Castellino et R. Germain. Cooperation between CD4+ and CD8+ T cells: when, where, and how. *Annu. Rev. Immunol.*, 24:519–40, 2006.

[16] N. Chabrier et F. Fages. Symbolic Model Checking of Biochemical Networks. *In Proc. CMSB 2003, Lect. Notes Comput. Sci.*, 2602:149–162, 2003.

[17] D. L. Chao, M. P. Davenport, S. Forrest, et A. S. Perelson. The effects of thymic selection on the range of T cell cross-reactivity. *Eur. J. Immunol.*, 35(12):3452–9, 2005.

[18] C. Chaouiya, E. Remy, B. Mossé, et D. Thieffry. Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework. *Lect Notes Comp Inf Sci*, 294:119–126, 2003.

[19] C. Chaouiya, A. Naldi, E. Remy, et D. Thieffry. Petri net representation of multi-valued logical regulatory graphs. *Natural Computing*, Special Issue: Petri Nets and Biosystems, in press.

[20] C. Chaouiya. Petri net modelling of biological networks. *Brief. Bioinformatics*, 8(4):210–9, 2007.

[21] C. Chaouiya, E. Remy, et D. Thieffry. Qualitative Petri Net Modelling of Genetic Networks. *In Proc. TCSB 2006, Lect. Notes Comput. Sci.*, 4220:95–112, 2006.

[22] E. M. Clarke, O. Grumberg, et D. A. Peled. *Model Checking*. The MIT Press, 1999.

[23] F. Corblin, S. Tripodi, E. Fanchon, D. Ropers, et L. Trilling. A declarative constraint-based method for analyzing discrete genetic regulatory networks. *BioSystems*, 2009.

[24] T. Cormen, C. Leiserson, et R. Rivest. *Introduction à l'algorithmique*. Dunod, 1994.

[25] M. Davidich et S. Bornholdt. The transition from differential equations to Boolean networks: a case study in simplifying a regulatory network model. *J. Theor. Biol.*, 255(3):269–77, 2008.

[26] M. I. Davidich et S. Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. *PLoS ONE*, 3(2):e1672, 2008.

[27] I. M. de Alborán, M. S. Robles, A. Bras, E. Baena, et C. Martínez-A. Cell death during lymphocyte development and activation. *Semin. Immunol.*, 15(3):125–33, 2003.

[28] H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *J. Comput. Biol.*, 9 (1):67–103, 2002.

[29] H. De Jong, J.-L. Gouzé, C. Hernandez, M. Page, T. Sari, et J. Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bull. Math. Biol.*, 66(2):301–40, 2004.

[30] A. Di Cara, A. Garg, G. De Micheli, I. Xenarios, et L. Mendoza. Dynamic simulation of regulatory networks using SQUAD. *BMC Bioinformatics*, 8:462, 2007.

[31] R. Drechsler et N. Gockel. Minimization of BDDs by evolutionary algorithms. In *International Workshop on Logic Synthesis (IWLS'97)*, 1997.

[32] J. H. Duarte, S. Zelenay, M.-L. Bergman, A. C. Martins, et J. Demengeot. Natural Treg cells spontaneously differentiate into pathogenic helper cells in lymphopenic conditions. *Eur. J. Immunol.*, 39(4):948–55, 2009.

[33] C. Espinosa-Soto, P. Padilla-Longoria, et E. R. Alvarez-Buylla. A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles. *Plant Cell*, 16(11):2923–39, 2004.

[34] A. Fauré, A. Naldi, C. Chaouiya, et D. Thieffry. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–31, 2006.

[35] A. Fauré, A. Naldi, F. Lopez, C. Chaouiya, A. Ciliberto, et D. Thieffry. Modular logical modelling of the budding yeast cell cycle. *Mol. BioSyst*, 2009.

[36] J. Fisher et T. A. Henzinger. Executable cell biology. *Nat. Biotechnol.*, 25(11):1239–49, 2007.

[37] D. Flanagan. *Java In A Nutshell*. O'Reilly, 5 edition, 2005.

[38] A. Garg, L. Mendoza, I. Xenarios, et G. Demicheli. Modeling of multiple valued gene regulatory networks. *Conference proceedings : Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society*, pages 1398–404, 2007.

[39] H. Garreta. *C : langage, bibliothèque, applications*. Dunod, 1992.

[40] C. Georgescu, W. J. R. Longabaugh, D. D. Scripture-Adams, E.-S. David-Fung, M. A. Yui, M. A. Zarnegar, H. Bolouri, et E. V. Rothenberg. A gene regulatory network armature for T lymphocyte specification. *Proc. Natl. Acad. Sci. U.S.A.*, 105(51):20100–5, 2008.

[41] R. N. Germain et I. Stefanová. The dynamics of T cell receptor signaling: complex orchestration and the key roles of tempo and cooperation. *Annu. Rev. Immunol.*, 17:467–522, 1999.

[42] A. Ghysen et R. Thomas. The formation of sense organs in Drosophila: a logical approach. *Bioessays*, 25(8): 802–7, 2003.

[43] B. Goldstein, J. R. Faeder, et W. S. Hlavacek. Mathematical and computational models of immune-receptor signalling. *Nat. Rev. Immunol.*, 4(6):445–56, 2004.

[44] A. González, C. Chaouiya, et D. Thieffry. Dynamical analysis of the regulatory network defining the dorsal-ventral boundary of the Drosophila wing imaginal disc. *Genetics*, 174(3):1625–34, 2006.

[45] A. González, C. Chaouiya, et D. Thieffry. Logical modelling of the role of the Hh pathway in the patterning of the Drosophila wing disc. *bioinformatics*, 24(16):i234–40, 2008.

[46] S. Hardy et P. N. Robillard. Modeling and simulation of molecular biology systems using petri nets: modeling goals of various approaches. *Journal of bioinformatics and computational biology*, 2(4):595–613, 2004.

[47] L. E. Harrington, R. D. Hatton, P. R. Mangan, H. Turner, T. L. Murphy, K. M. Murphy, et C. T. Weaver. Interleukin 17-producing CD4+ effector T cells develop via a lineage distinct from the T helper type 1 and 2 lineages. *Nat. Immunol.*, 6(11):1123–32, 2005.

[48] T. S. P. Heng, M. W. Painter, The Immunological Genome Project Consortium, K. Elpek, V. Lukacs-Kornek, N. Mauermann, S. J. Turley, D. Koller, F. S. Kim, A. J. Wagers, N. Asinovski, S. Davis, M. Fassett, M. Feuerer, D. H. D. Gray, S. Haxhinasto, J. A. Hill, G. Hyatt, C. Laplace, K. Leatherbee, D. Mathis, C. Benoist, R. Jianu, D. H. Laidlaw, J. A. Best, J. Knell, A. W. Goldrath, J. Jarjoura, J. C. Sun, Y. Zhu, L. L. Lanier, A. Ergun, Z. Li, J. J. Collins, S. A. Shinton, R. R. Hardy, R. Friedline, K. Sylvia, et J. Kang. The Immunological Genome Project: networks of gene expression in immune cells. *Nat. Immunol.*, 9(10):1091–4, 2008.

[49] T. Höfer, H. Nathansen, M. Löhning, A. Radbruch, et R. Heinrich. GATA-3 transcriptional imprinting in Th2 lymphocytes: a mathematical model. *Proc. Natl. Acad. Sci. U.S.A.*, 99(14):9364–8, 2002.

[50] S. Huang et D. E. Ingber. Shape-dependent control of cell growth, differentiation, and apoptosis: switching between attractors in cell regulatory networks. *Exp. Cell Res.*, 261(1):91–103, 2000.

[51] D. Irons. Logical analysis of the budding yeast cell cycle. *J. Theor. Biol.*, 2009.

[52] G. Kalmanovich et R. Mehr. Models for antigen receptor gene rearrangement. III. Heavy and light chain allelic exclusion. *J. Immunol.*, 170(1):182–93, 2003.

[53] T. Kam, T. Villa, R. K. Brayton, et A. L. Sangiovanni-Vincentelli. Multi-valued decision diagrams: Theory and applications. *Int. J. Multiple-Valued Logic*, 4:9–12, 1998.

[54] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.*, 22 (3):437–67, 1969.

[55] M. Kaufman et R. Thomas. Model analysis of the bases of multistationarity in the humoral immune response. *J. Theor. Biol.*, 129(2):141–62, 1987.

[56] M. Kaufman, J. Urbain, et R. Thomas. Towards a logical analysis of the immune response. *J. Theor. Biol.*, 114 (4):527–61, 1985.

[57] M. Kaufman, F. Andris, et O. Leo. A logical analysis of T cell activation and anergy. *Proc. Natl. Acad. Sci. U.S.A.*, 96(7):3894–9, 1999.

[58] H. Kitano. A graphical notation for biochemical networks. *Biosilico*, 1(5):169–176, 2003.

[59] S. Klamt, J. Saez-Rodriguez, J. Lindquist, L. Simeoni, et E. Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, 7:56, 2006.

[60] M. A. Koch, G. Tucker-Heard, N. R. Perdue, J. R. Killebrew, K. B. Urdahl, et D. J. Campbell. The transcription factor T-bet controls regulatory T cell homeostasis and function during type 1 inflammation. *Nat. Immunol.*, 10(6):595–602, 2009.

[61] K. W. Kohn. Molecular interaction map of the mammalian cell cycle control and DNA repair systems. *Mol. Biol. Cell*, 10(8):2703–34, 1999.

[62] L. Kristensen, S. Christensen, et K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2:98–132, 1998.

[63] K.-H. Lee, A. R. Dinner, C. Tu, G. Campi, S. Raychaudhuri, R. Varma, T. N. Sims, W. R. Burack, H. Wu, J. Wang, O. Kanagawa, M. Markiewicz, P. M. Allen, M. L. Dustin, A. K. Chakraborty, et A. S. Shaw. The immunological synapse balances T cell receptor signaling and degradation. *Science*, 302(5648):1218–22, 2003.

[64] K. León, A. Lage, et J. Carneiro. Tolerance and immunity in a mathematical model of T-cell mediated suppression. *J. Theor. Biol.*, 225(1):107–26, 2003.

[65] F. Li, T. Long, Y. Lu, Q. Ouyang, et C. Tang. The yeast cell-cycle network is robustly designed. *Proc. Natl. Acad. Sci. U.S.A.*, 101(14):4781–6, 2004.

[66] S. Liang, S. Fuhrman, et R. Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 18–29, 1998.

[67] F. Lopez, J. Textoris, A. Bergon, G. Didier, E. Remy, S. Granjeaud, J. Imbert, C. Nguyen, et D. Puthier. TranscriptomeBrowser: a powerful and flexible toolbox to explore productively the transcriptional landscape of the Gene Expression Omnibus database. *PLoS ONE*, 3(12):e4001, 2008.

[68] R. Lund, H. Ahlfors, E. Kainonen, A.-M. Lahesmaa, C. Dixon, et R. Lahesmaa. Identification of genes involved in the initiation of human Th1 or Th2 cell commitment. *Eur. J. Immunol.*, 35(11):3307–19, 2005.

[69] L. Mariani, M. Löhning, A. Radbruch, et T. Höfer. Transcriptional control networks of cell differentiation: insights from helper T lymphocytes. *Prog. Biophys. Mol. Biol.*, 86(1):45–76, 2004.

[70] L. Mendoza. A network model for the control of the differentiation process in Th cells. *BioSystems*, 84(2): 101–14, 2006.

[71] L. Mendoza et E. R. Alvarez-Buylla. Dynamics of the genetic regulatory network for Arabidopsis thaliana flower morphogenesis. *J. Theor. Biol.*, 193(2):307–19, 1998.

[72] L. Mendoza et I. Xenarios. A method for the generation of standardized qualitative dynamical systems of regulatory networks. *Theoretical biology & medical modelling*, 3:13, 2006.

[73] L. Mendoza, D. Thieffry, et E. Alvarez-Buylla. Genetic control of flower morphogenesis in Arabidopsis thaliana: a logical analysis. *Bioinformatics*, 15(7-8):593–606, 1999.

[74] C. G. Moles, P. Mendes, et J. R. Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Res.*, 13(11):2467–74, 2003.

[75] I. Mura et A. Csikász-Nagy. Stochastic Petri Net extension of a yeast cell cycle model. *J. Theor. Biol.*, 254(4): 850–60, 2008.

[76] E. Muraille, D. Thieffry, O. Leo, et M. Kaufman. Toxicity and neuroendocrine regulation of the immune response: a model analysis. *J. Theor. Biol.*, 183(3):285–305, 1996.

[77] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proc. IEEE*, (77):541–580, 1989.

[78] M. Nagasaki, A. Doi, H. Matsuno, et S. Miyano. A versatile petri net based architecture for modeling and simulation of complex biological processes. *Genome informatics. International Conference on Genome Informatics*, 15(1):180–97, 2004.

[79] A. Naldi, D. Berenguier, A. Fauré, F. Lopez, D. Thieffry, et C. Chaouiya. Logical modelling of regulatory networks with GINsim 2.3. *BioSystems*, 2009.

[80] A. Naldi, E. Remy, D. Thieffry, et C. Chaouiya. A reduction method of logical regulatory graphs preserving essential dynamical properties. *In Proc. CMSB 2009, Lect. Notes Comput. Sci.*, 5688:266–280, 2009.

[81] A. Naldi, D. Thieffry, et C. Chaouiya. Decision Diagrams for the Representation and Analysis of Logical Models of Genetic Networks. *In Proc. CMSB 2009, Lect. Notes Comput. Sci.*, 4695/2007:233–247, 2007.

[82] A. Naldi, J. Carneiro, C. Chaouiya, et D. Thieffry. Diversity and plasticity of Th cell types predicted from regulatory network modelling. in prep.

[83] P. Pandiyan, L. Zheng, S. Ishihara, J. Reed, et M. J. Lenardo. CD4+CD25+Foxp3+ regulatory T cells induce cytokine deprivation-mediated apoptosis of effector CD4+ T cells. *Nat. Immunol.*, 8(12):1353–62, 2007.

[84] T. J. Perkins, M. Hallett, et L. Glass. Inferring models of gene expression dynamics. *J. Theor. Biol.*, 230(3): 289–99, 2004.

[85] V. N. Reddy, M. N. Liebman, et M. L. Mavrovouniotis. Qualitative analysis of biochemical reaction systems. *Comput. Biol. Med.*, 26(1):9–24, 1996.

[86] E. Remy, B. Mossé, C. Chaouiya, et D. Thieffry. A description of dynamical graphs associated to elementary regulatory circuits. *Bioinformatics*, 19 Suppl 2:ii172–8, 2003.

[87] E. Remy, P. Ruet, L. Mendoza, D. Thieffry, et C. Chaouiya. From Logical Regulatory Graphs to Standard Petri Nets: Dynamical Roles and Functionality of Feedback Circuits. *Transactions on Computation Systems Biology (TCSB) VII*, 4230:55–72, 2006.

[88] E. Remy, P. Ruet, et D. Thieffry. Graphic requirement for multistability and attractive cycles in a Boolean dynamical framework. *Advances in Applied Mathematics*, 41(3):335–350, 2008.

[89] E. Remy et P. Ruet. From minimal signed circuits to the dynamics of Boolean regulatory networks. *Bioinformatics*, 24(16):i220–6, 2008.

[90] A. Richard. Positive circuits and maximal number of fixed points in discrete dynamical systems. *Discrete Applied Mathematics*, 157:3281–3288, 2009.

[91] A. Richard et J. Comet. Necessary conditions for multistationarity in discrete dynamical systems. *Discrete Applied Mathematics*, 155(18):2403–2413, 2007.

[92] K. H. Rosen. *Discrete mathematics and its applications*. Random House, 1988. ISBN 0394367685.

[93] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *ICCAD '93: Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design*, pages 42–47, Los Alamitos, CA, USA, 1993. IEEE Computer Society Press. ISBN 0-8186-4490-7.

[94] A. Sackmann, M. Heiner, et I. Koch. Application of Petri net based analysis techniques to signal transduction pathways. *BMC Bioinformatics*, 7:482, 2006.

[95] J. Saez-Rodriguez, L. Simeoni, J. Lindquist, R. Hemenway, U. Bommhardt, B. Arndt, U. Haus, R. Weismantel, E. Gilles, S. Klamt, et B. Schraven. A logical model provides insights into T cell receptor signaling. *PLoS Comput. Biol.*, 3(8):e163, 2007.

[96] S. Sakaguchi, N. Sakaguchi, M. Asano, M. Itoh, et M. Toda. Immunologic self-tolerance maintained by activated T cells expressing IL-2 receptor alpha-chains (CD25). Breakdown of a single mechanism of self-tolerance causes various autoimmune diseases. *J. Immunol.*, 155(3):1151–64, 1995.

[97] R. Samaga, J. Saez-Rodriguez, L. G. Alexopoulos, P. K. Sorger, et S. Klamt. The logic of EGFR/ErbB signaling: theoretical properties and analysis of high-throughput data. *PLoS Comput. Biol.*, 5(8):e1000438, 2009.

[98] L. Sánchez et D. Thieffry. A logical analysis of the Drosophila gap-gene system. *J. Theor. Biol.*, 211(2): 115–41, 2001.

[99] L. Sánchez et D. Thieffry. Segmenting the fly embryo: a logical analysis of the pair-rule cross-regulatory module. *J. Theor. Biol.*, 224(4):517–37, 2003.

[100] L. Sanchez, C. Chaouiya, et D. Thieffry. Segmenting the fly embryo: logical analysis of the role of the Segment Polarity cross-regulatory module. *Int. J. Dev. Biol.*, 52(8):1059–75, 2008.

[101] A. Scherer, A. Noest, et R. J. de Boer. Activation-threshold tuning in an affinity model for the T-cell repertoire. *Proc. Biol. Sci.*, 271(1539):609–16, 2004.

[102] T. Schlitt et A. Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8 Suppl 6:S9, 2007.

[103] M. Schwarick et M. Heiner. CSL model checking of biochemical networks with Interval Decision Diagrams. *In Proc. CMSB 2009, Lect. Notes Comput. Sci.*, 5688:296–312, 2009.

[104] N. Sepúlveda, L. Boucontet, P. Pereira, et J. Carneiro. Stochastic modeling of T cell receptor gamma gene rearrangement. *J. Theor. Biol.*, 234(2):153–65, 2005.

[105] H. Siebert. Local Structure and Behavior of Boolean Bioregulatory Networks. *In Proc AB 2008, Lect. Notes Comput. Sci.*, 5147:185–199, 2008.

[106] H. Siebert. Deriving behavior of Boolean bioregulatory networks from subnetwork dynamics. *Math. Comp. Sci.*, 2(3):421–442, 2009.

[107] H. Siebert et A. Bockmayr. Incorporating Time Delays into the Logical Analysis of Gene Regulatory Networks. *In Proc. CMSB 2006, Lect. Notes Comput. Sci.*, 4210:169–183, 2006.

[108] H. Siebert et A. Bockmayr. Context Sensitivity in Logical Modelling with Time Delays. *In Proc. CMSB 2007, Lect. Notes Comput. Sci.*, 4695:64–79, 2007.

[109] H. Siebert et A. Bockmayr. Relating Attractors and Singular Steady States in the Logical Analysis of Bioregulatory Networks. *In Proc. AB 2007, Lect. Notes Comput. Sci.*, 4545:36–50, 2007.

[110] D. Sieling. On the existence of polynomial time approximation schemes for OBDD minimization. *Lecture Notes in Computer Science*, 1373:205 –215, 1998.

[111] E. Simão, E. Remy, D. Thieffry, et C. Chaouiya. Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in E.coli. *Bioinformatics*, 21 Suppl 2:ii190–6, 2005.

[112] C. Soulé. Graphic Requirements for Multistationarity. *Complexus*, 1:123–133, 2003.

[113] D. Thieffry et R. Thomas. Dynamical behaviour of biological regulatory networks–II. Immunity control in bacteriophage lambda. *Bull. Math. Biol.*, 57(2):277–97, 1995.

[114] D. Thieffry. Dynamical roles of biological regulatory circuits. *Brief. Bioinformatics*, 8(4):220–5, 2007.

[115] R. Thomas. On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations. *Springer series in Synergetics*, 9:180–193, 1981.

[116] R. Thomas. Boolean formalization of genetic control circuits. *J. Theor. Biol.*, 42(3):563–85, 1973.

[117] R. Thomas. Regulatory networks seen as asynchronous automata: A logical description. *J Theor Biol*, 153: 1–23, 1991.

[118] R. Thomas et R. D'Ari. *Biological Feedback*. CRC Press, Boca Raton, Forida, 1990.

[119] D. A. A. Vignali, L. W. Collison, et C. J. Workman. How regulatory T cells work. *Nat. Rev. Immunol.*, 8(7): 523–32, 2008.

[120] C. T. Weaver, L. E. Harrington, P. R. Mangan, M. Gavrieli, et K. M. Murphy. Th17: an effector CD4 T cell lineage with regulatory T cell ties. *Immunity*, 24(6):677–88, 2006.

[121] I. Wegener. BDDs - Design, Analysis, Complexity, and Applications. In *Discrete Applied Mathematics 138*, pages 229–251, 2004.

[122] K. Willadsen et J. Wiles. Robustness and state-space structure of Boolean gene regulatory models. *J. Theor. Biol.*, 249(4):749–65, 2007.

[123] D. Wittmann, J. Krumsiek, J. Saez-Rodriguez, D. Lauffenburger, S. Klamt, et F. Theis. Transforming Boolean models to continuous models: methodology and application to T-cell receptor signaling. *BMC systems biology*, 3(1):98, 2009.

[124] A. Yates, R. Callard, et J. Stark. Combining cytokine signalling with T-bet and GATA-3 regulation in Th1 and Th2 differentiation: a model for cellular decision-making. *J. Theor. Biol.*, 231(2):181–96, 2004.

[125] C.-R. Yu, R. M. Mahdi, S. Ebong, B. P. Vistica, J. Chen, Y. Guo, I. Gery, et C. E. Egwuagu. Cell proliferation and STAT6 pathways are negatively regulated in T cells by STAT1 and suppressors of cytokine signaling. *J. Immunol.*, 173(2):737–46, 2004.

[126] Y. Zheng, A. Chaudhry, A. Kas, P. deRoos, J. M. Kim, T.-T. Chu, L. Corcoran, P. Treuting, U. Klein, et A. Y. Rudensky. Regulatory T-cell suppressor program co-opts transcription factor IRF4 to control T(H)2 responses. *Nature*, 458(7236):351–6, 2009.

[127] X. Zhou, L. T. Jeker, B. T. Fife, S. Zhu, M. S. Anderson, M. T. McManus, et J. A. Bluestone. Selective miRNA disruption in T reg cells leads to uncontrolled autoimmunity. *J. Exp. Med.*, 205(9):1983–91, 2008.

Annexes

Annexe A

Autres publications

J'aimerais présenter ici d'autres travaux auxquels j'ai participé avant ou en marge de mon travail principal de thèse.

A.1 GINsim 2.2

Cette publication introduit la version 2.2 de GINsim, sur laquelle j'ai travaillé avant de démarrer ma thèse. Il introduit également un modèle du contrôle de la formation du disque imaginal d'aile durant le développement de la Drosophile. J'ai essentiellement travaillé sur l'implémentation de GINsim.

Available online at www.sciencedirect.com
SCIENCE DIRECT
BioSystems 84 (2006) 91–100
Bio Systems
www.elsevier.com/locate/biosystems

GINsim: A software suite for the qualitative modelling, simulation and analysis of regulatory networks

A. Gonzalez Gonzalez[a,b], A. Naldi[a], L. Sánchez[b], D. Thieffry[a], C. Chaouiya[a,*]

[a] *Institut de Biologie du Développement de Marseille (IBDM), CNRS/INSERM/Université de la Méditerranée, Campus de Luminy, Case 907, F-13288 Marseille Cedex 9, France*
[b] *Centro de Investigaciónes Biológicas, Calle Ramiro de Maeztu, 9, E-28040 Madrid, España, Spain*

Received 21 April 2005; received in revised form 13 September 2005; accepted 4 October 2005

Abstract

This paper presents GINsim,[1] a Java software suite devoted to the qualitative modelling, analysis and simulation of genetic regulatory networks. Formally, our approach leans on discrete mathematical and graph-theoretical concepts. GINsim encompasses an intuitive graph editor, enabling the definition and the parameterisation of a regulatory graph, as well as a simulation engine to compute the corresponding qualitative dynamical behaviour. Our computational approach is illustrated by a preliminary model analysis of the inter-cellular regulatory network activating Notch at the dorsal–ventral boundary in the wing imaginal disc of *Drosophila*. We focus on the cross-regulations between five genes (within and between two cells), which implements the dorsal–ventral border in the developing imaginal disc. Our simulations qualitatively reproduce the wild-type developmental pathway, as well as the outcome of various types of experimental perturbations, such as loss-of-function mutations or ectopically induced gene expression.

Keywords: Regulatory networks; Logical modelling; Dynamical simulation; Wing imaginal disc; Boundary formation

1. Introduction

Most biological processes are controlled by regulatory networks, which involve various kinds of molecular interactions, including protein-mediated transcriptional regulations, polypeptide receptor–ligand associations, protein modifications by specific enzymes, etc. Decades of genetic and molecular analyses, more recently complemented by high-throughput functional genomic experiments, have progressively uncovered many of the numerous interactions controlling several crucial biological processes (including cell cycle and various developmental pathways). The complexity of the networks delineated often defies intuitive reasoning, consequently calling for the development of proper computational tools. Different mathematical approaches have been proposed to model such genetic networks and to simulate their dynamical behaviour, ranging from quantitative formalisms (e.g. sets of differential or stochastic equations) to crude Boolean models (see de Jong, 2003, for an extensive overview of these different methods).

In many instances, in particular in the case of the study of animal development, precise quantitative data are lacking. For this reason, we lean on a qualitative (but yet rigorous) and flexible formalism, initially proposed by Thomas (1991). Although this formalism has proven its usefulness for the modelling of various genetic regulatory networks (see, for example, Sánchez and Thieffry, 2001, 2003), a user-friendly software to edit, simulate

* Corresponding author.
E-mail address: chaouiya@ibdm.univ-mrs.fr (C. Chaouiya)
[1] GINsim is available at http://gin.univ-mrs.fr/GINsim/.

0303-2647/$ – see front matter © 2005 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2005.10.003

and analyse the resulting dynamics was not available yet. We are developing GINsim precisely to fulfill this gap.

To illustrate the applicability of GINsim, we present a preliminary model of the regulatory network involved in the definition of the dorsal–ventral (DV) boundary in the developing wing imaginal disc of the fly *Drosophila melanogaster* during the second and third larval instars. This simple model encompasses two cells flanking the DV boundary and five genes/proteins in each cell. Using GINsim, we have performed a series of simulations corresponding to the wild-type case, as well as to several mutant situations.

2. Logical modelling of regulatory networks

Our approach for the modelling and analysis of regulatory networks roots in the logical formalism previously developed by Thomas and colleagues (Thomas, 1991; Thomas et al., 1995). It is based on the definitions of: (i) *logical regulatory graphs*, describing regulatory interactions between genes and (or via) their products, and (ii) *state transition graphs*, which represent the qualitative dynamical behaviour associated with a given regulatory graph, for given initial states. Hereafter, we provide a brief description of these two types of graphs (a detailed definition of the formalism can be found in Chaouiya et al., 2003).

A *regulatory graph* is a labelled directed graph where nodes represent genes (or, more generaly, regulatory components) and arcs (directed edges) represent interactions between genes. Let $\mathcal{G} = \{g_1, \ldots, g_n\}$ be the set of genes (or nodes of the regulatory graph). Then, for each $g_i \in \mathcal{G}$ we define its *maximum expression level* $\max_i$ ($\max_i \in \mathbb{N}$). Its current expression level is denoted x_i ($x_i \in \{0, 1, \ldots, \max_i\}$). An expression or activity *state* of the system is thus given by a n-tuple of the expression levels of the n genes.

Each arc (interaction) is defined by its source, its target and an integer interval defining the condition under which the interaction is *operating*. An interaction is operating when the current level of expression of its source gene belongs to the attached interval. Note that this interval must be included in [1, max] (max being the maximum level of the source).

At this point, we have defined the regulatory structure, encompassing the nodes, their interactions, together with the definition of the corresponding levels. Now, we have to define the rules governing the dynamics of the network. It is the purpose of the logical functions defined below. Indeed, for each gene, the corresponding logical function allows the qualitative specification of the effects of any combination of incoming interactions.

For each gene g_i, we consider the set $\mathcal{I}(i)$, called *input of* g_i, defined by the set of arcs ending in g_i. The application K_i, called *logical function*, associates a *parameter* $K_i(X)$ to each subset X of $\mathcal{I}(i)$.

For each g_i, given a state $(x_1, \ldots, x_n)$, one can determine the subset $X \subseteq \mathcal{I}(i)$ of the incoming interactions which are effectively operating upon g_i (those for which the levels of expression of their sources belong to the related interval). Then, the value of the logical parameter $K_i(X)$ defines the level towards which g_i tends when X is the set of operating incoming interactions. The function K_i takes its values in $\{0, \ldots, \max_i\}$.

Summarising, a regulatory graph is defined by:

- a set of nodes $\mathcal{G} = \{g_1, \ldots, g_n\}$ with the maximum expression level $\max_i \in \mathbb{N}$ of each g_i;
- a set of labelled arcs $\mathcal{I}$ defined by the union of the sets of arcs targeting the genes g_i of $\mathcal{G}$: $\mathcal{I} = \bigcup_{i=1,\ldots,n} \mathcal{I}(i)$;
- a set of logical functions $\mathcal{K} = \{K_i : \mathcal{P}(\mathcal{I}(i)) \rightarrow \{0, \ldots \max_i\}, i = 1, \ldots, n\}$ ($\mathcal{P}(\mathcal{I}(i))$ being the power set of $\mathcal{I}(i)$).

An interaction is called an *activation* (respectively, *repression* or inhibition) when its effect on the targeted gene tends to be positive (respectively, negative). Note, however, that effective activatory or inhibitory effects generally depend on the presence or on the absence of cofactors. Indeed, one gene can be the target of several interactions. Interaction signs (activation versus inhibition) are implicit in our definition of regulatory graphs.

The (discrete) dynamics of the system can then be represented by *state transition graphs*, where vertices represent states of the system (i.e. n-tuples giving the expression levels of the n genes), and arcs represent *transitions* between these states.

Given a state of the system $(x_1, \ldots x_n)$ (x_i being the current level of g_i), one can determine the set X_i of interactions which are operating upon g_i. It is then straightforward to determine whether g_i may change its level value: it suffices to consider the relevant parameter $K_i(X_i)$. If $x_i < K_i(X_i)$, there is a call to increase the level of g_i, whereas if $x_i > K_i(X_i)$, there is a call to decrease the level of g_i (otherwise there is no call for updating g_i).

In most applications, state transition graphs are generated either on the basis of a fully *synchronous assumption* (all updating calls are then executed simultaneously) (Wuensche, 1998) or on the basis of a fully *asynchronous* approach (Thomas et al., 1995) (all updating calls are then considered independently). Under both synchronous and asynchronous assumptions, we only consider elementary transitions (level increase or decrease by exactly 1).

3. GINsim: a software for the analysis and simulation of Gene Interaction Networks

GINsim supports the definition, the simulation and the analysis of regulatory graphs based on the logical formalism introduced in the previous section. We have developed a series of Java classes, encompassing four main modules: a user interface, a parser, a core simulator and a graph analysis toolbox. The general organisation of GINsim is presented in Fig. 1, and details are given hereafter. A plug-in based architecture allows the implementation of new modules in GINsim, in particular to complete the graph analysis toolbox. GINsim is available at http://gin.univ-mrs.fr/GINsim. This web site further proposes a tutorial and a collection of models of genetic regulatory modules, several involved in the development of the fruit fly.

3.1. Specification of a regulatory graph model

The main window of GINsim, the *graph editor*, allows the specification and visualisation of regulatory graphs (a typical screenshot is provided in Fig. 2). Similar to a simple drawing software, it has been developed using *JGraph*, an open source Swing component for the visualisation of graphs. The internal graph structures have been implemented using the free Java graph library *JGraphT*, which includes mathematical graph-theory objects and algorithms. Besides the graphical specification of the regulatory graph (nodes and interactions), a dedicated panel further allows the definition of the maximal level of each node (default value is 1) and of the logical parameters (default value is zero). To define a logical parameter associated to a given node, the user simply selects the corresponding group of incoming interactions and specifies a non-zero value.

Once defined, the corresponding logical model can be stored in a file, using the GINML format (Gene Interaction Networks Markup Language, see description at http://gin.univ-mrs.fr/GINsim). GINML is dedicated to the storage of regulatory graphs, as well as of state transition graphs. It has been defined as an extension of GXL, a standard XML format for graphs (GXL's home page: http://www.gupro.de/GXL). Logical regulatory graphs and state transition graphs are stored in GINML format, but GINsim also allows exports to the Scalable Vector Graphics (SVG) standard and to the format of Graphviz, an open source graph visualisation tool (see http://www.graphviz.org/).

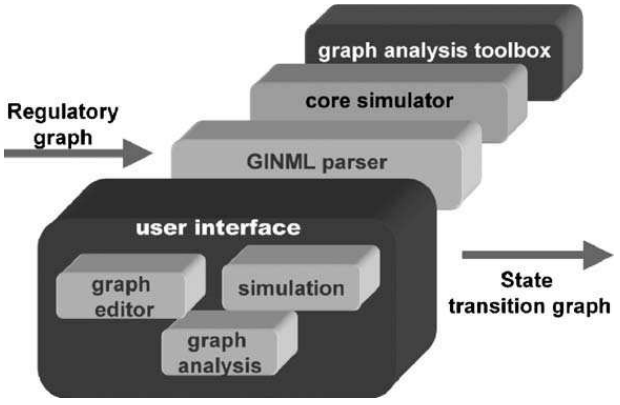


Fig. 1. Schematic illustration of GINsim architecture.

3.2. Simulation and analysis

Given a logical regulatory graph, the user can trigger a simulation, that is the construction of a state transition graph. A dedicated interface allows the specification of the initial state(s), the updating policy (asynchronous versus synchronous), as well as other options described hereafter.

In the case of realistic applications, the state transition graphs can be very large. Indeed, the simulation of a Boolean regulatory graph with n genes can lead to a transition graph with up to 2^n states. We thus face the classical problem of the exponential growth of the size of the state space. To overcome this problem, we have first developed simple means to drive the construction of the state transition graph and to focus on relevant parts of the complete dynamics. The user can choose a depth first or a breadth first search algorithm to explore the state transition graph corresponding to a given parameterised regulatory graph. Next, a maximal search depth and/or a limit on the number of nodes considered can be specified. As the order of the genes on the description of the states has a clear impact on the generation of the state transition graph (it corresponds to the order of the search, beginning with the first gene called to update its level value), the modification of the initial order (that of the original definition of the regulatory graph) is facilitated. Furthermore, the user can prevent the updating of the level of specific genes, thereby avoiding the exploration of specific regions of the state transition graph. This later option also facilitates the simulation of mutations such as loss-of-function and ectopic expression of given genes. Finally, an absolute priority order can be specified upon the genes, to systematically determine a unique path within all possible trajectories.

For state transition graphs of limited size (less than 500 nodes), GINsim allows their visualisation by means of the same graph editor as for the regulatory graphs (a typical screenshot is provided in Fig. 3).

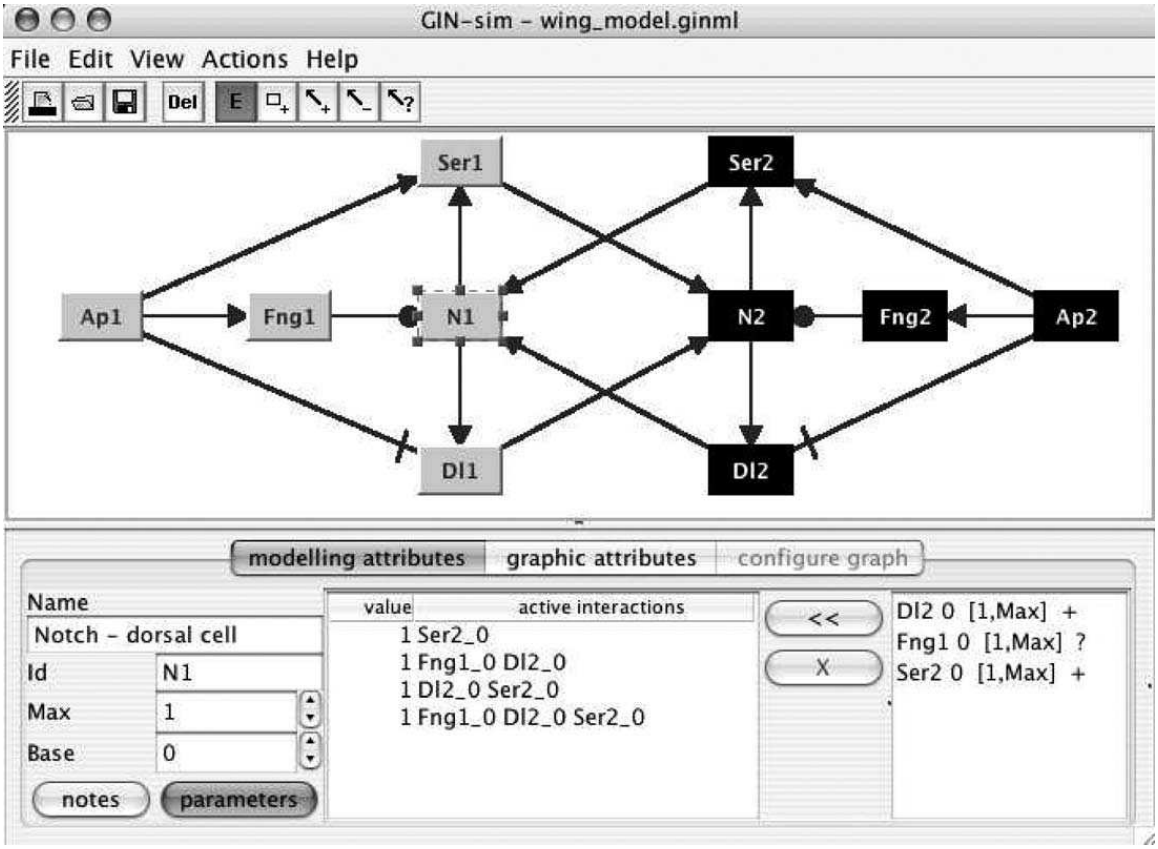


Fig. 2. Key regulatory components in the dorsal cell (grey) and ventral cell (black). Positive interactions are shown with normal arrows, negative interactions with blunt arrows, and ambivalent interactions (Fng on N) with bullet arrows. *Symbols*: Ap, Apterous; Fng, Fringe; Ser, Serrate; Dl, Delta; N, Notch. The number (1 or 2) following each symbol indicates whether it corresponds to the dorsal (1, left) or the ventral (2, right) cell. The frame at the bottom shows the list of the non-zero logical parameters for N1 (selected node). As N1 is the target of three interactions (coming from Fng1, Ser2 and Dl2), eight parameters have to be defined. Four of them take the value 1, corresponding to the following situations: (a) Ser2 alone (and therefore Dl2 and Fng1 are both absent), (b) Fng1 together with Dl2, (c) Dl2 together with Ser2 and (d) all three proteins together activate N1 (see also Table 2). Note that GINsim requires only the definition of the non-zero parameters.

The *graph analysis toolbox* (partially implemented at the moment) embodies two main parts:

- In the case of regulatory graphs, the core of the analysis consists in determining the regulatory circuits and their functionality (i.e. the constraints on the parameters which ensure the functionality of a given regulatory circuit).
- In the case of state transition graphs, the most frequent questions deal with the identification of the stationary states or other attractors, of the corresponding basins of attraction, as well as with the delineation of transition paths from given states to specific attractors. The current version of GINsim provides means to determine the strongly connected components (simple or intertwined cycles) of the state transition graph, as well as the paths going through specific states. Moreover, it might be practical to view the evolution of some genes along a path in a more conventional way, that is plotting the value of these genes as a function of time. This functionality is now available in GINsim, through an export to a GnuPlot file,[2] provided that the user has first determined a specific path of interest.

3.3. Prospects

We have already stressed the problem of the exponential growth of the size of the state transition graphs. To address this question, we are considering different strategies to allow a progressive exploration of the dynamics of the system. Amid these strategies, the specification of priorities among genes reduces the size of the state transition graph by avoiding some paths. This option is partially implemented in GINsim. Priorities can be viewed

[2] GnuPlot is a well-known free function and data plotting programme.

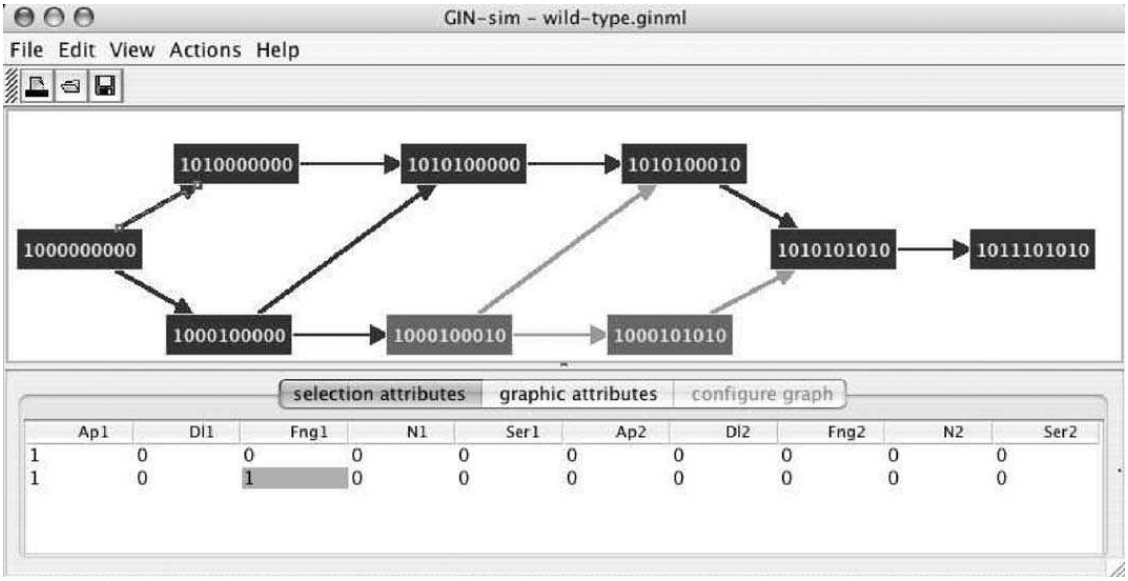


Fig. 3. The state transition graph corresponding to the situation where *ap* is expressed in dorsal cells. States are labelled by a vector giving the levels of the different regulatory products in the following order: Ap1, Dl1, Fng1, N1, Ser1, Ap2, Dl2, Fng2, N2, Ser2 (dorsal cell followed by ventral cell genes). The state transition graph is displayed using the same graph visualisation interface as for the regulatory graph (see Fig. 2). The lower frame shows the detail of a selected transition (arrow at the top left), giving the levels of the regulatory nodes, highlighting the regulatory component changing through this transition (Fng in the dorsal cell).

as a complementary information on the transition delays. Within this proposal, we are currently developing a procedure to specify groups of priorities, and thus allowing a combined synchronous/asynchronous simulation.

We are also currently implementing classical graph theory algorithms to delineate further properties of regulatory and state transition graphs. We are also developing analytical approaches to identify key features of state transition graphs without generating them explicitly (a first attempt in this direction is described in Remy et al., 2003). It is well-known that regulatory circuits play a prevailing role for the dynamical properties of regulatory networks (they are at the origin of multistationarity and homeostasis). Therefore, the next release of GINsim will provide means to analyse the functionality of the regulatory circuits. Moreover, it is possible to locate all stable states for large regulatory networks (involving several hundreds of genes) using constraint programming (as initially shown by Devloo et al., 2003). We are also considering the use of a model-checking approach to verify specific properties of the dynamics.

4. Application: modelling of a gene network involved in the early development of the fly wing

4.1. Description of the biological system

In order to illustrate the use of our computational approach, we have selected an application dealing with the early steps of pattern formation during the fly wing development. The *Drosophila* adult wing develops from a larval tissue called "wing imaginal disc", initially made of a group of about 20 embryonic epithelial cells. At the late first larval instar, this group of cells starts proliferating and reaches a final number of about 75,000 cells at the end of the larval period, organised in a two-dimensional cellular layer. Using appropriate genetic markers, the wing imaginal disc reveals a precise fate map of the adult wing.

Proper patterning and growth of the wing disc is controlled by anterior–posterior and dorsal–ventral boundaries through the production of morphogenetic signals. The protein Wingless is the morphogenetic signal at the DV boundary. In the cells straddling the DV boundary, the expression of the gene *wingless* is cell-autonomously mediated by the activation of the receptor Notch (N). This expression is regulated by a complex regulatory network. The analysis of published wild-type and perturbed gene expression patterns allows the delineation of epistatic relationships between the genes involved in this process (reviewed in Irvine and Vogt, 1997). The resulting network involves genes acting up to the late third larval instar. A full model analysis goes beyond the scope of this paper in which we only consider the earliest steps of this process.

The primary signal triggering N activation is the expression of the gene *apterous* (*ap*) in the presumptive dorsal cells (Diaz-Benjumea and Cohen, 1993). The

ap gene activates the genes *Serrate* (*Ser*) (Couso et al., 1995) and *fringe* (*fng*) (Irvine and Wieschaus, 1994), and represses the expression of the gene *Delta* (*Dl*) (Milan and Cohen, 2000). The products Serrate (Ser) and Delta (Dl) are ligands of the receptor N. Ser from the dorsal cells activates N in the adjacent ventral cells (Couso et al., 1995; Panin et al., 1997). This results in an increase of *Dl* transcription in these cells (de Celis and Bray, 1997). Dl from the ventral cells activates N in the adjacent dorsal cells (Doherty et al., 1996; Panin et al., 1997). Thus, active N at the boundary becomes stabilised through the induction of its two ligands in adjacent cells.

The key to understand the activation of N at the boundary lies in the different response of dorsal and ventral cells to Dl and Ser, respectively, as a result of the differential expression of *fng*. In dorsal cells, Fringe (Fng) prevents N from being activated by Ser in dorsal neighbouring cells, but allows N to be activated non-autonomously by Dl from ventral cells (Panin et al., 1997). On the other hand, in ventral cells, Dl cannot activate N in the absence of Fng and N becomes only activated non-autonomously by dorsal Ser (Panin et al., 1997). The asymmetry of Ap signal, which is only activated in dorsal cells, is mediated by this dual function of Fng.

4.2. Logical modelling and wild-type simulation

To model the process of early N activation, it suffices to consider two cells, each located on one side of the DV boundary. The regulatory network in Fig. 2 encompasses the most relevant factors involved at these early stages. Note that the network includes both intra- and inter-cellular interactions. From a molecular point of view, some of these interactions correspond to protein–protein interactions (e.g. from Dl to N or from Ser to N), others to transcriptional regulation and others to post-translational modifications (from Fng to N). As this model is mainly based on genetic experiments, the molecular mechanisms underlying these interactions are not always known.

The main experimental evidences supporting the interactions shown in Fig. 2 are given in Table 1, whereas the logical parameters are defined in Table 2. Here, a Boolean model is sufficient to cover the main biological aspects of the system. The dynamical analysis of this network was performed using GINsim (see the GINsim web site for the GINML file of the parameterised model).

In the absence of dorsal Ap, the system has a unique steady state, corresponding to a situation where there is neither significant expression of *Dl*, *fng*, *Ser*, nor activation of N (Diaz-Benjumea and Cohen, 1993). However, once the gene *ap* is expressed in the dorsal cell, the system is progressively led to a state where *fng* and *Ser* are expressed in the dorsal cell, *Dl* in the ventral cell and the receptor N appears in both cell types (see Fig. 3). The unique final state, as well as the transient states occurring in our simulation, are coherent with the published data.

Table 1
Main experimental evidences supporting the regulatory interactions of Fig. 2

Interactions	Main experimental observations and references
$ap \rightarrow fng$	Ap protein is found in the same pattern as *fng* and *Ser* expression
$ap \rightarrow Ser$	In loss-of-function *ap* mutants, *Ser* and *fng* are not expressed (Couso et al., 1995; Irvine and Wieschaus, 1994)
$ap \dashv Dl$	Ap inactivation is required for late *Dl* expression in dorsal cells (Milan and Cohen, 2000)
$N \rightarrow Dl$	Ectopic expression of actived N induces Dl and Ser, (Panin et al., 1997)
$N \rightarrow Ser$	
$fng \multimap N$	Fng prevents activation of N by Ser from dorsal neighbouring cells and allows N activation by Dl from ventral neighbouring cells (Panin et al., 1997)
$Ser1 \rightarrow N2$	Dorsal Ser activates the ventral N receptor because of the absence of *fng* (Panin et al., 1997)
$Dl2 \rightarrow N1$	Ventral Dl is a ligand that activates the dorsal N receptor in the presence of *fng* (Panin et al., 1997)

In the left column, the normal arrows ($\rightarrow$) represent positive interactions, the blunt arrow ($\dashv$) represents a negative interaction and the bullet arrow ($\multimap$) represents an ambivalent interaction. Note that the last two rows correspond to the intercellular interactions. In this case, the number following the symbol of the component precises whether it is in the dorsal (1) or in the ventral (2) cell. The main supporting experimental evidences and the corresponding bibliographical references are provided in the right column.

4.3. Mutant simulations

A means for testing the qualitative robustness and consistency of our model is to check the effect of different types of perturbations on its dynamics. To simulate a mutant clone, we force the gene value in the corresponding cell to a fixed level as described in Sánchez and Thieffry (2003). In brief, a loss-of-function mutation of a given gene implies that this gene produces a non-functional product, which amounts to assign the value zero to the corresponding logical variable (regulatory product concentration). The ectopic expression of a gene implies that this gene is expressed in an unregulated

Table 2
Logical parameters defining the effect of genes/proteins in and between the dorsal and the ventral boundary cells

Gene	Dorsal cell	Ventral cell
apterous	$K_{Ap1}(\emptyset) = 1$	$K_{Ap2}(\emptyset) = 0$
Delta	$K_{Dl1}(\emptyset) = 0$, $K_{Dl1}(\{Ap1\}) = 0$, $K_{Dl1}(\{N1\}) = 1$, $K_{Dl1}(\{Ap1, N1\}) = 0$	$K_{Dl2}(\emptyset) = 0$, $K_{Dl2}(\{Ap2\}) = 0$, $K_{Dl2}(\{N2\}) = 1$, $K_{Dl2}(\{Ap2, N2\}) = 0$
fringe	$K_{Fng1}(\emptyset) = 0$, $K_{Fng1}(\{Ap1\}) = 1$	$K_{Fng2}(\emptyset) = 0$, $K_{Fng2}(\{Ap2\}) = 1$
Notch	$K_{N1}(\emptyset) = 0$, $K_{N1}(\{Dl2\}) = 0$, $K_{N1}(\{Fng1\}) = 0$, $K_{N1}(\{Ser2\}) = 1$, $K_{N1}(\{Dl2, Fng1\}) = 1$, $K_{N1}(\{Dl2, Ser2\}) = 1$, $K_{N1}(\{Ser2, Fng1\}) = 0$, $K_{N1}(\{Dl2, Fng1, Ser2\}) = 1$	$K_{N2}(\emptyset) = 0$, $K_{N2}(\{Dl1\}) = 0$, $K_{N2}(\{Fng2\}) = 0$, $K_{N2}(\{Ser1\}) = 1$, $K_{N2}(\{Dl1, Fng2\}) = 1$, $K_{N2}(\{Dl1, Ser1\}) = 1$, $K_{N2}(\{Ser1, Fng2\}) = 0$, $K_{N2}(\{Dl1, Fng2, Ser1\}) = 1$
Serrate	$K_{Ser1}(\emptyset) = 0$, $K_{Ser1}(\{Ap1\}) = 1$, $K_{Ser1}(\{N1\}) = 0$, $K_{Ser1}(\{Ap1, N1\}) = 1$	$K_{Ser2}(\emptyset) = 0$, $K_{Ser2}(\{Ap2\}) = 1$, $K_{Ser2}(\{N2\}) = 0$, $K_{Ser2}(\{Ap2, N2\}) = 1$

Interactions are denoted by the name of their source node. The Boolean values of the parameters define the effect of a given combination of active genes/proteins (set in parentheses) on a given gene/protein (subindex of the parameter). All subsets of possible combinations of incoming interactions are listed (the parameter value for an empy set of incoming interactions is the basal functional level). Note that the values of the parameters are strictly identical for the dorsal and ventral cells (numbers 1 and 2), except for *ap* whose basal expression differs in dorsal and ventral cells. The distinct values of the parameters for N1 in the presence of Ser2 alone vs. Ser2 together with Fng1, and Dl2 alone vs. Dl2 together with Fng1, indicate that N is activated exclusively in the cells flanking the DV boundary.

manner beyond its normal spatial–temporal expression domain. Accordingly, to simulate an ectopic expression, the corresponding variable and function are forced to take a higher value, which in our case is value 1. This is easily done with GINsim when triggering a simulation: it suffices to assign ranges of values (fixed values in the Boolean case) to selected genes. Then, along a simulation, once the level of such a selected gene falls in the assigned interval, its future values are forced to remain in this interval (see Fig. 4).

To illustrate the potential of GINsim in mutant analyses, we have simulated a set of perturbations corresponding to loss-of-function mutations in either dorsal or ventral cellular clones, or to ectopic gene expression in similarly defined clones. The effect of these perturbations has been monitored at the level of N expression in

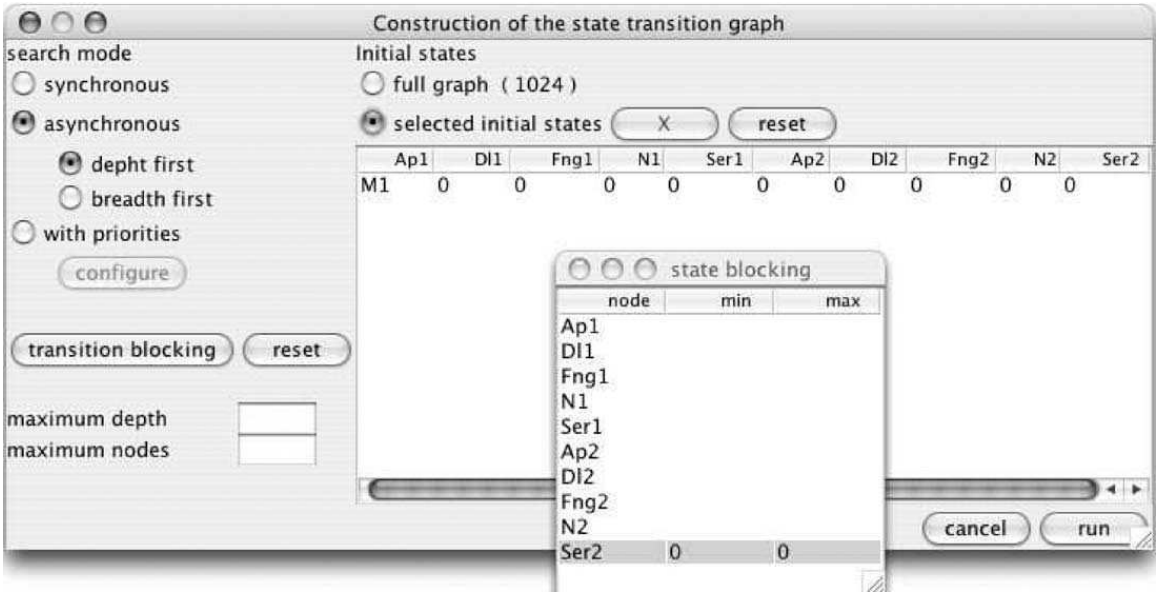


Fig. 4. The GINsim interface for the specification of the simulation. The floating frame shows how to constraint the level of a regulatory product (gene Ser2 is set to 0, a situation corresponding to a ventral Ser loss of function clone).

Table 3
Simulation results fo the wild-type and mutant cases

Clone genotype and position	Steady-state vector (dorsal cell–ventral cell [Ap1, Dl1, Fng1, **N1**, Ser1–Ap2, Dl2, Fng2, **N2**, Ser2])	Boundary phenotype
Wild-type	[1 0 1 **1** 1–0 1 0 **1** 0]	
Dorsal Dl^- clone at the boundary	[1 0 1 **1** 1–0 1 0 **1** 0]	Wild-type (Doherty et al., 1996)
Ventral Dl^- clone at the boundary	[1 0 1 **0** 1–0 0 0 **1** 0]	Perturbed (Doherty et al., 1996)
Dorsal fng^- clone at the boundary	[1 0 0 **0** 1–0 1 0 **1** 0]	Perturbed (Irvine and Wieschaus, 1994)
Ventral fng^- clone at the boundary	[1 0 1 **1** 1–0 1 0 **1** 0]	Wild-type (Irvine and Wieschaus, 1994)
fng^+ clone crossing the boundary	[1 0 1 **0** 1–0 0 1 **0** 0]	Perturbed (Kim et al., 1995)
Dorsal Ser^- clone at the boundary	[1 0 1 **0** 0–0 0 0 **0** 0]	Perturbed (Couso et al., 1995)
Ventral Ser^- clone at the boundary	[1 0 1 **1** 1–0 1 0 **1** 0]	Wild-type (Couso et al., 1995)

The first column describes the mutant genotypes. The second column shows the steady state reached in the simulation. Notch values (in bold) in the second column serve as a reference to compare simulated mutant phenotypes with that of the wild-type. Perturbations in mutant simulations (second column) agree with experimental phenotypes (third column).

the cells adjacent to the DV boundary. Table 3 describes the simulation results for seven different situations.

In the case of a *Dl* loss-of-function clone in the ventral cell (fixing the value of Dl to 0), we obtain a zero value for N in the dorsal cell, while the simulation of a *Dl* loss-of-function clone in the dorsal cell does not lead to any visible perturbation (second and third rows in Table 3). This result is consistent with the observation that *Dl* loss-of-function clones perturb the N signal at the boundary, when the clones are on the ventral side of the boundary (Doherty et al., 1996).

In contrast, the simulations of *fng* and *Ser* loss-of-function clones do cause a mutant phenotype only when they are on the dorsal side of the DV boundary (for *fng*, fourth and fifth rows of Table 3, for *Ser*, seventh and eighth rows). This again agrees with the literature (Irvine and Wieschaus, 1994; Couso et al., 1995). Finally, simulating an ectopic expression of *fng* in mutant clones crossing the boundary (fixing the value of Fng1 and Fng2 to 1) leads to a stable state with no activated N in both cells (sixth row of Table 3). This last result agrees with the observed interruption of the N boundary (Kim et al., 1995).

5. Discussion and prospects

In this paper, we have presented GINsim which is dedicated to the qualitative modelling of genetic and molecular regulatory networks. Formally, our approach leans on the multilevel, asynchronous logical method initially developed by Thomas (1991). This approach has been successfully applied to various types of biological regulatory networks, including the dynamical modelling of the regulatory networks involved in the setting of the segmental gene expression pattern, early during the development of the fly (Sánchez and Thieffry, 2001, 2003).

GINsim includes an intuitive graph editor for the definition of a regulatory graph, as well as of its parameterisation. A core logical simulator and several graph analysis and layout algorithms have been developed as plugins. Referring to a generic graph representation, we have taken advantage of the existence of free standard Java libraries, allowing the future integration of new algorithms implemented by an active community. Moreover, we have defined GINML, an XML format to store regulatory graph models, with the intent to facilitate the exchange of models between different formalisms (in particular between the logical and the Petri net approaches).

We are now turning to the implementation of specific graph analysis packages, in particular to locate dynamical attractors and to analyse specific transition pathways. In parallel, we are developing complementary modules taking advantage of the analytical power of the logical formalism, in particular regarding the delineation of the dynamical roles of specific regulatory structures (feedback circuits).

To illustrate our computational approach, we have further provided a simple (Boolean) application dealing with the early steps of wing formation during the larval development of the fly. Although this application may appear relatively simple (as meant for the sake of illustrative purpose), it already implements various important characteristics of developmental systems, including the combination of intra- and inter-cellular interactions, the occurrence of intertwined feedbacks, as well as the occurrence of positional information in terms of asymmetric inputs (e.g. Apterous activity).

Focusing on two cells implementing the dorsal–ventral border in the developing imaginal disc, our model allows the qualitative reproduction of the wild-type developmental pathway, in terms of a well-defined se-

quence of logical states. As shown by the simulation, this pathway leads to a specific state matching the known wild-type expression pattern.

Using appropriated parameter changes or blocking the value of one or another variable, GINsim allows the simulation of abnormal situations corresponding to experimental induction of cellular clones lacking or over-expressing a specific factor. We have limited ourselves here to the simulation of a series of known situations, in order to demonstrate that the model leads to logical states which are consistent with the results found in the literature. However, more interestingly, we can also use our model to address new situations and perform original in silico experiments. If proper experimental validation is ultimately necessary, the possibility to quickly screen various types of new mutants in silico could help the biologist to delineate the most promising or striking experiments, i.e. those potentially leading to unexpected results or allowing the distinction between concurrent modelling hypotheses.

We are presently working on the extension of this model to include all relevant genes (about seven in each cell), and to encompass explicitly all relevant cell layers (at least four qualitatively different layers). As cells rapidly increase in number during the wing disc development, to simulate the growing tissue we need to take advantage of the repetition of identical units (cells) involved in symmetrical interactions. We are now working on the implementation of graph rewriting rules to explicitly simulate cell duplications. In parallel, we are developing a series of extensions of GINsim to locate and characterise various types of dynamical features, including attractors, their basins of attraction or yet separator states (corresponding to the separatrices in ordinary differential systems). In this respect, we are building new plugins, either in Java or in Prolog.

Finally, we have defined systematic rules to obtain Petri net models from logical regulatory graphs (see Simão et al., 2005 and references therein). The use of Petri nets should allow us to take advantage of their well founded mathematical background and of the analysis tools which have already been developed (see Murata, 1989 for a general introduction to Petri nets).

Acknowledgements

We thank M. van Caneghem for his participation in the software development. The IBDM team has been financially supported by the CNRS, the French Research Ministry (*ACI IMPbio* programme) and by the European Commission (FP6 STREP DIAMONDS, contract LSHG-CT-2004-512143). L. Sánchez is supported by the grant BMC2002-02858 from C.I.C.Y.T., Ministerio de Educación y Ciencia. Collaboration between the two groups has been enhanced by a cooperation grant in the context of a CSIC–CNRS international cooperation programme.

References

Chaouiya, C., Remy, E., Mossé, B., Thieffry, D., 2003. Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework. Lectures Notes Control Inf. Sci. 294, 119–126.

Couso, J.P., Knust, E., Martinez-Arias, A., 1995. *Serrate* and *wingless* cooperate to induce vestigial gene expression and wing formation in *Drosophila*. Curr. Biol. 5, 1437–1448.

de Celis, J.F., Bray, S., 1997. Feed-back mechanisms affecting Notch activation at the dorsoventral boundary in the *Drosophila* wing. Development 124, 3241–3251.

de Jong, H., 2003. Modeling and simulation of genetic regulatory systems: a literature review. J. Comput. Biol. 9, 67–103.

Devloo, V., Hansen, P., Labbe, M., 2003. Identification of all steady states in large networks by logical analysis. Bull. Math. Biol. 65, 1025–1051.

Diaz-Benjumea, F.J., Cohen, S.M., 1993. Interaction between dorsal and ventral cells in the imaginal disc directs wing development in *Drosophila*. Cell 75, 741–752.

Doherty, D., Feger, G., Younger-Shepherd, S., Jan, L.Y., Jan, Y.N., 1996. Delta is a ventral to dorsal signal complementary to Serrate, another Notch ligand, in *Drosophila* wing formation. Genes Dev. 10, 421–434.

Irvine, K.D., Vogt, T.F., 1997. Dorsal–ventral signaling in limb development. Curr. Opin. Cell Biol. 9, 867–876.

Irvine, K.D., Wieschaus, E., 1994. Fringe, a boundary-specific signaling molecule, mediates interactions between dorsal and ventral cells during *Drosophila* wing development. Cell 79, 595–606.

Kim, J., Irvine, K.D., Carroll, S.B., 1995. Cell recognition, signal induction, and symmetrical gene activation at the dorsal–ventral boundary of the developing *Drosophila* wing. Cell 82, 795–802.

Milan, M., Cohen, S.M., 2000. Temporal regulation of apterous activity during development of the *Drosophila* wing. Development 127, 3069–3078.

Murata, T., 1989. Petri nets: properties, analysis and applications. Proc. IEEE 77, 541–580.

Panin, V.M., Papayannopoulos, V., Wilson, R., Irvine, K.D., 1997. Fringe modulates Notch–ligand interactions. Nature 387, 908–912.

Remy, E., Mossé, B., Chaouiya, C., Thieffry, D., 2003. A description of dynamical graphs associated to elementary regulatory circuits. Bioinformatics 19 (Suppl. 2), 172–178.

Sánchez, L., Thieffry, D., 2001. A logical analysis of the *Drosophila* gap–gene system. J. Theor. Biol. 211, 115–141.

Sánchez, L., Thieffry, D., 2003. Segmenting the fly embryo: a logical analysis of the pair-rule cross-regulatory module. J. Theor. Biol. 224, 517–537.

Simão, E., Remy, E., Thieffry, D. Chaouiya, C. 2005. Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in *E. Coli*. Bioinformatics 21, ii190–ii196.

Thomas, R., 1991. Regulatory networks seen as asynchronous automata: a logical description. J. Theor. Biol. 153, 1–23.

Thomas, R., Thieffry, D., Kaufman, M., 1995. Dynamical behaviour of biological regulatory networks, I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. Bull. Math. Biol. 57, 247–276.

Wuensche, A., 1998. Genomic regulation modeled as a network with basins of attraction. In: Proc. Pac. Symp. Biocomput., pp. 89–102.

A.2 Classes de priorités et cycle cellulaire mammifère

Cette publication présente un modèle du cycle cellulaire mammifère. Ce modèle a inspiré la mise en place d'une méthode de simulation utilisant des classes de priorités (voir section 7.1). Là encore ma contribution est principalement méthodologique.

BIOINFORMATICS

Vol. 22 no. 14 2006, pages e124–e131
doi:10.1093/bioinformatics/btl210

Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle

Adrien Fauré, Aurélien Naldi, Claudine Chaouiya and Denis Thieffry*

Institut de Biologie du Développement de Marseille-Luminy, Campus scientifique de Luminy, CNRS case 907, 13288 Marseille, France

ABSTRACT

Motivation: To understand the behaviour of complex biological regulatory networks, a proper integration of molecular data into a full-fledge formal dynamical model is ultimately required. As most available data on regulatory interactions are qualitative, logical modelling offers an interesting framework to delineate the main dynamical properties of the underlying networks.

Results: Transposing a generic model of the core network controlling the mammalian cell cycle into the logical framework, we compare different strategies to explore its dynamical properties. In particular, we assess the respective advantages and limits of synchronous versus asynchronous updating assumptions to delineate the asymptotical behaviour of regulatory networks. Furthermore, we propose several intermediate strategies to optimize the computation of asymptotical properties depending on available knowledge.

Availability: The mammalian cell cycle model is available in a dedicated XML format (GINML) on our website, along with our logical simulation software GINsim (http://gin.univ-mrs.fr/GINsim). Higher resolution state transitions graphs are also found on this web site (Model Repository page).

Contact: thieffry@ibdm.univ-mrs.fr

1 INTRODUCTION

A proper understanding of the structure and temporal behaviour of biological regulatory networks requires the integration of regulatory data into a formal dynamical model (for a review, see de Jong, 2002). Although this issue has been recurrently addressed by applying standard mathematical approaches (*e.g.*, differential or stochastic equations) borrowed from physical sciences, it is notably complicated by the diversity and sophistication of regulatory mechanisms, as well as by the chronic lack of reliable quantitative information.

This situation has motivated the development of intrinsically qualitative approaches leaning on Boolean algebra or generalisation thereof (Glass & Kauffman, 1973; Thomas, 1991).

In this paper, we lean on previous work refining, extending and implementing the logical approach initially formulated by R. Thomas *et al*. (Thomas, 1991; Thomas *et al*., 1995). The corresponding framework is summarised in the following section (see Chaouiya *et al*., 2003, for more detail). This framework is then used to derive a logical version of the differential model for the control of the mammalian cell cycle recently published by Novak and Tyson (2004). The corresponding regulatory network is described in the last chapter of the introduction, together with citations of the most relevant experimental articles (for a didactic introduction to cell cycle modelling, see Fuß *et al*., 2005).

In their landmark model analysis, Novak and Tyson have heavily relied on numerical integration techniques (temporal simulations, phase space analyses, and bifurcation diagrams) to delineate the main dynamical properties of the complex regulatory system under study. Their results are valid for specific sets of parameter values and function shapes, which are difficult to establish quantitatively. Furthermore, such parametric analyses can only handle a few parameters at once.

In contrast, although much more qualitative, the logical framework enables a more systematic and extensive characterisation of all the behaviours compatible with a given regulatory graph. Furthermore, this framework offers enumerative or analytical means to identify relevant asymptotical behaviours (stable states, state transition cycles, etc.). Finally, extending a logical model to encompass additional regulatory modules is relatively easy.

However, one difficulty with the logical approach lies in the implicit treatment of time. In this respect, different approaches have been proposed, either considering all transitions under a synchronicity assumption, or considering them under a fully asynchronous assumption, *i.e.*, selecting a single transition at each dynamical step. The first assumption is simple but leads to well known artefacts, whereas the results obtained under the second assumption are more difficult to evaluate. In this paper, we explore the use of different strategies enabling a honourable compromise between these two extreme approaches.

1.1 Logical modelling of regulatory networks

The specification of a logical model involves three main steps: (i) the building of a *regulatory graph*; (ii) the definition of the *logical parameters* of the system; (iii) the specification of the *updating assumption*(s).

Cross-regulations between regulatory components are formalized in terms of an oriented graph. In this *regulatory graph*, the vertices represent the different regulatory components (activity of a gene, concentration of a regulatory product, or level of activity of a protein), whereas the edges represent regulatory interactions between these components (including self-regulations). The level or activity

*To whom correspondence should be addressed.

of a regulatory component i is given by an integer, taking its values in the interval [0, Max_i], where Max_i is the maximal value considered for this element (in the simplest, Boolean case, Max_i is set to 1). Each edge is labelled with an interval of integers defining the set of values for which the source of the interaction influences its target. Naturally, this interval must be compatible with the values allowed for the source of the interaction. Furthermore, for sake of simplification, the maximal value of the interval is usually set to the maximal value of the source of the interaction (notion of *threshold*). Note that this definition allows the specification of multiple interactions between two components, provided that each interaction involves a specific *threshold* (alternatively, disjoint contiguous intervals can be used).

Finally, an edge can also be optionally labelled with a positive or negative sign, which then specifies that the effect of the source on the target is monotonous, either potentially activating or inhibiting the target, respectively. The specification of interaction signs only affects the graphical representation and must be translated into proper parameter values to obtain coherent regulatory effects.

The next step consists in defining the combinatory effects of the regulatory inputs on the expression or activity of a given component of the regulatory graph. The set of inputs is already specified at the level of the regulatory graph. However, the effect of each regulator usually depends on the presence of the co-regulators. For the sake of conciseness, we consider only the combinations of interactions allowing a significant (non zero, from a logical point of view) expression or activity of the regulated component. The corresponding *logical parameters* are each univocally identified by the set of interactions acting on the regulated genes and take their values in [1,Max_{target}] (see the next section for a concrete illustration).

The dynamical behaviour of a logical regulatory model is represented in terms of an oriented graph, where each vertex represents a specific *logical state* of the system (*i.e.*, a vector giving the discrete levels of expression/activity of all the components), whereas the edges represent (possible) *transitions* between these states.

Together with the regulatory graph, the logical parameters define the rules directing the dynamics of a network, *i.e.*, the potential occurrence of specific edges in the *state transition graph*. Indeed, at a given state, a specific logical parameter can be associated with each component. If the value of this parameter is smaller or greater than that of the concentration/activity level of the corresponding component, this level will tend to decrease or increase, respectively. Otherwise (when the parameter value and the corresponding component level are equal), the component will tend to keep its current value.

At this stage, different assumptions might be considered. According to the simplest one, at a given state, all increase or decrease calls are realized simultaneously (*synchronous updating*), changing the component levels by one unit at a time (see *e.g.*, Kauffman, 1993). Easy to implement and computationally efficient, this approach leads to well known dynamical artefacts (in particular spurious cycles). At the other extreme of the spectrum, the transition calls can be asynchronously updated, *i.e.*, one single transition will be selected at a time. This assumption requires additional rules to sort out concurring transitions (*e.g.*, the specification of time delays or of priorities). These additional rules are tricky to define, as they may perfectly be context sensitive, *i.e.*, finely depend on the levels of various regulatory components (although these combinations might correspond to identical parameter values). For this reason, all possible transitions are often generated, and an *asynchronous transition graph* is built where all single possible transitions are considered, although all resulting dynamical pathways cannot be followed for a single set of transition rules.

Whatever the updating assumption, of particular interest is the asymptotical behaviour of the system, *e.g.*, the terminal vertices (*stable states*, with no updating calls) or the attractive cycles found in the state transition graph. Note that such *attractors* (in particular the stable states and simple terminal cycles) can easily be located in the context of the synchronous updating assumption. As we shall see, the synchronous assumption can often (but not always) be considered as a shortcut for the computation of the asynchronous dynamics. This point will be further assessed below through the analysis of a logical model of the core network controlling the mammalian cell cycle.

To ease the definition of a regulatory graph and of the associated logical parameters, as well as the construction of the (a)synchronous state transition graphs, we have developed a logical modelling/analysis/simulation software called *GINsim* (Gonzalez *et al.*, 2006). A new release of *GINsim* now implements the possibility to play with the different updating assumptions and to define different *priority classes*.

Let consider a regulatory graph with n nodes $\{g_1, g_2, \ldots g_n\}$. A logical state is a vector $S=(s_1, s_2, \ldots s_n)$ where s_i is the current level of the ith regulatory product ($s_i \in \{0, \ldots Max_i\}$). Given such a state, it is possible to determine the evolution of g_i, for all $i = 1, \ldots n$. Indeed, given any regulatory component g_i, the interactions which are operating on g_i in the state S can be identified, and the relevant logical parameter (*i.e.*, corresponding to the right combination of incoming interactions) gives the value k_i to which g_i should tend. If $s_i > k_i$ (the current level is greater than the parameter value), there is a call for decreasing the level of g_i, (a decrease call on g_i is denoted $g_i\downarrow$); if $s_i < k_i$, there is a call for increasing the level of g_i (denoted $g_i\uparrow$); otherwise (if $s_i = k_i$), there is no updating call for this component. A stable state is thus a state without updating call.

The synchronicity assumption amounts to apply all concurrent transitions simultaneously, all states having thus at most one successor; under the asynchronous assumption, concurrent transitions are applied separately, and a state with q updating calls has then exactly q successors.

Here, we introduce a new functionality of *GINsim* consisting in the definition of p priority classes $C_1, C_2, \ldots C_p$, *with* $p \leq n$, which gather regulatory products depending on their qualitative production and/or degradation delays:

(i) each class C_i is associated with a rank $r(C_i)$ ($1 \leq r(C_i) \leq$ p, 1 being the highest rank), as well as with an updating policy (synchronous or asynchronous);

(ii) several classes may have the same rank; and concurrent transitions on genes of different classes with identical rank are triggered asynchronously;

(iii) at any state S, among all concurrent transitions, only those on genes of the classes with the highest rank are triggered;

(iv) concurrent transitions inside a class are triggered accordingly to the updating policy associated to that class;

(v) finally, increasing and decreasing transitions of each gene can be distinguished and associated to classes with different ranks.

1.2 Regulation of the mammalian cell cycle

The cell cycle involves a succession of molecular events leading to the reproduction of the genome of a cell (Synthesis or S phase) and its division into two daughter cells (Mitosis, or M phase). The M phase itself encompasses different sub-phases (prophase, metaphase, anaphase, telophase) characterised by specific chromosomal and nuclear changes (respectively: condensation of the chromatin, alignment of the chromosomes, separation of the sister chromatids, and formation of the two daughter nuclei). The S and M phases are preceded by two gap phases, called G1 and G2 respectively (for a review, see, for example, Tessema *et al*., 2004). These events are very well known and can easily be monitored with an optical microscope.

During the late 1970s and early 1980s, yeast geneticists have identified the cell-division-cycle (cdc) genes, encoding for new classes of molecules including the cyclins (so called because of their cyclic pattern of activation), and their cyclin dependent kinases (cdk) partners. Since then, our knowledge of the molecular network that controls cell cycle events has tremendously progressed, but the number of components and interactions known to be involved has so much increased that proper formal modelling becomes necessary to understand the behaviour of such a complex system.

Our model analysis is rooted in the seminal work of Novak and Tyson, who have recently derived and analyzed a set of 18 ordinary differential equations (ODE) to model the control of the restriction point of the mammalian cell cycle (Novak and Tyson, 2004). Based on this differential model and using numerical integration techniques, the authors were able to qualitatively reproduce the main known dynamical features of the wild-type biological system, as well as the consequences of several types of perturbations. This state-of-the-art model study nevertheless appears difficult to extend, although there is clearly many more regulators, variants and interactions to consider (see, *e.g*., Kohn's map at http://discover.nci.nih.gov/kohnk/fig6a.html).

In this respect, the logical formalism offers an appropriate framework to qualitatively explore the dynamical properties of relatively complex regulatory graphs. However, up to now, it has been mostly applied to transcriptional regulatory networks, and its application to the numerous and various protein interactions at the core of the cell cycle network was thus a challenge.

As a starting point, we have used Novak and Tyson's diagram and model to build a logical regulatory graph (see Figure 1). In the process, we were led to derive a proper logical representation for each type of regulatory interaction. In what follows, we summarize the main experimental data and assumptions underlying our regulatory graph. In the context of this paper, we further focus on a specific Boolean version of this model.

Mammalian cell division is tightly controlled, for it must be coordinated with the overall growth of the organism, as well as answer specific needs, such as wound healing for example. This coordination is achieved through extra-cellular positive and negative signals whose balance decides whether a cell will divide or remain in a resting state (quiescence or G0 phase), which can be reached and left by the cell during the G1 phase. The positive signals or growth factors ultimately elicit the activation of Cyclin D in the cell. In our model, CycD thus represents the input, and its activity is considered constant. Note that cdk4 and cdk6, the partners of Cyclin D, are not explicitly represented in our model, for their activity essentially depends on the presence or absence of their cyclin. In other words, CycD stands here for the whole cdk4/6-Cyclin D complex. The same approach has been adopted for the other cyclin/cdk pairs.

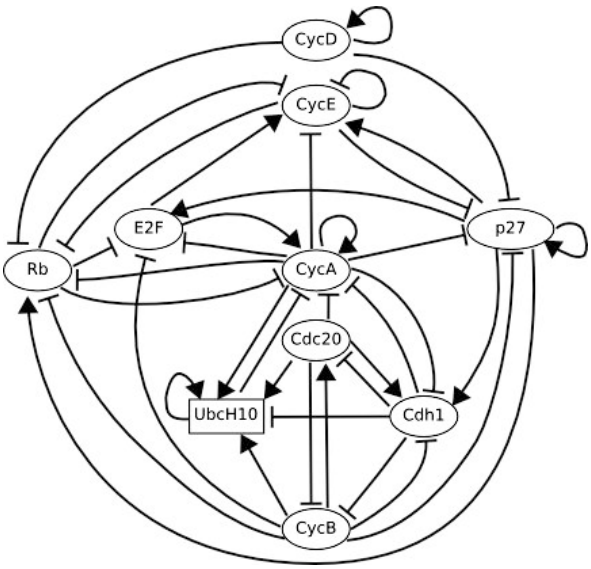


Fig. 1. Logical regulatory graph for the mammalian cell cycle network. Each node represents the activity of a key regulatory element, whereas the edges represent cross-regulations. Blunt arrows stand for inhibitory effects, normal arrows for activations.

In our model, CycD is necessary for the inactivation of the retinoblastoma protein Rb, and for the sequestration of p27/Kip1 (p27 in the sequel). This protein is a cdk inhibitor that sequesters cdk2/Cyclin E (CycE) and cdk2/Cyclin A (CycA), preventing them from phosphorylating their targets (reviewed in Coqueret, 2003). It is usually considered that Cyclin D remains active when in complex with p27, though the issue is still debated (Olashaw *et al*., 2004). For the sake of simplicity, we consider that CycD directly inhibits p27.

In contrast, the complexes formed by p27 and CycE or CycA are represented in a subtler way, though this formation remains implicit: when both p27 and CycE or CycA are active, the complex forms, and the activity of the cyclin is blocked. To model the fact that the cyclins remain present though sequestered when linked to p27, we consider that p27 opposes their activities on their targets, instead of directly inhibiting them. In our model, this is embodied by arrows from p27 onto the targets of CycE and CycA, with a sign opposite to that corresponding to the effect of the cyclins on their targets in the absence of p27.

The other target of CycD, Rb, is a key tumour suppressor, which is found mutated in a large variety of cancers. Rb is inactivated by phosphorylation, and CycD is involved in the first step of this process (reviewed in Tamrakar *et al*., 2000). However, in this simplified Boolean model, we consider that Rb inactivation by CycD is total.

Rb forms a complex with members of the E2F family of transcription factors, turning them from transcriptional activators to repressors, in part through recruitment of chromatin remodelling

complexes. For this reason, we model the action of Rb by direct inhibitions of E2F targets (which include E2F itself).

E2F is a wide family of dimeric transcription factors, formed by a member of the E2F family, and a member of the DP family. It is usually divided into activators E2Fs (E2F1, E2F2, E2F3a) and repressors E2Fs (E2F3b, E2F4, E2F5), plus the recently discovered E2F6, E2F7 and E2F8, whose structure, regulation and mode of action are slightly different from those of the *regular* E2Fs (Dimova and Dyson, 2005). In our present model, E2F represents the activator members (together with their DP partners), the other E2Fs being implicit.

At the G1/S transition, E2F activates the transcription of Cyclin E, which in turns causes the inactivation of Rb. CycE also phosphorylates p27, eliciting its destruction. Phosphorylated Rb dissociates from E2F, allowing more Cyclin E to be transcribed, further increasing the phosphorylation of Rb and the destruction of p27, in a positive feedback loop.

Cyclin A is another target of E2F, which is activated slightly after Cyclin E, when Rb is more completely inactivated (Zhang *et al*., 2000). The action of CycA contributes to maintain Rb and p27 inhibition, inactivates E2F and CycE and most importantly, inactivates the Anaphase Promoting Complex (APC).

The APC is an important E3 ubiquitin ligase that is activated in a cyclic fashion (reviewed in Harper *et al*., 2002). The APC complex is represented by its two activators, Cdh1 and Cdc20. Around the G2-to-M-phase transition, CycA inactivates Cdh1, which switches the APC OFF, allowing Cyclin B to appear. Cyclin B in turn activates Cdc20, sowing the seeds of its own destruction, since CycB is a target of Cdc20. Cdc20 is responsible for the metaphase-to-anaphase transition: it activates separase through the destruction of its inhibitor securin; this activation elicits the cleavage of the cohesin complexes that maintain the cohesion between the sister chromatids, thus leading to their separation. Cdc20 also participates in degrading CycA, and indirectly activates Cdh1. Cdh1 completes CycA and CycB inactivation, and inactivates Cdc20. In absence of its inhibitors, E2F can be reactivated and a new cycle begins.

How Cyclin A can rise a level high enough to inactivate its own inhibitor has long remained a paradox. Rape and Kirshner (2004) solved it by highlighting the role of the E2 ubiquitin conjugating enzyme UbcH10). They found that UbcH10 is necessary for Cdh1 dependent degradation of Cyclin A, but not of the other APC substrates; once all of its substrates have been degraded, UbcH10 can ubiquitinate itself, preventing the APC from degrading Cyclin A, which can thus reappear. These findings make the activation of Cyclin A in S phase coherent with the observation that Cdh1 is still active at this point of the cycle (Huang *et al*., 2001). At the present stage, the explicit inclusion of UbcH10 constitutes the most remarkable extension of Novak & Tyson's model. It further allows us to incorporate an important additional interaction, the inactivation of CycA by Cdh1 (within the APC complex).

2 RESULTS

2.1 Regulatory graph and its parameterization

The Figure 1 displays the regulatory graph integrating all the data briefly reviewed in the introductory section.

On the basis of this graph and using additional information from the literature, it is possible to derive a set of rules enabling the activation of each of the regulatory component encompassed by this graph. Presented in Table 1, these rules are sufficient to derive all the non-zero logical parameters enabling the recovery of the main known features of the wild-type cell cycle.

In its present Boolean version (*i.e*., $Max_i = 1$ for all regulatory components), our model is still simple enough to allow an exhaustive dynamical analysis with the logical simulation software *GINsim*. The complete state transition graph contains 1024 vertices (*i.e*., Boolean states). To study the dynamical trajectories corresponding to the asymptotical behaviour of the system, we still need to specify an updating assumption. As we shall see, this specification further determines the pathway(s) followed by the system, in particular with respect to the cyclic attractor.

2.2 Synchronous *versus* asynchronous updating

Starting with the simplest, synchronous assumption, we obtain two attractors. The first one is a stable state with only Rb, p27 and Cdh1 active, in the absence of CycD; this state is reached from all the other states lacking CycD activity (*i.e*., in the lack of growth factors; this state thus corresponds to the phase G0 or cell quiescence).

In contrast, in the presence of CycD, all trajectories lead to a unique dynamical cycle, made of a sequence of seven successive states (Figure 2, bottom left). From a qualitative point of view, the order of activity switching (off or on) matches the available data, as well as the time plots published by Novak and Tyson (2004).

Looking more carefully at this synchronous cycle, one can note that only two arrows correspond to single transitions, namely the activations of CycA and Cdc20, whereas three arrows correspond to double transitions, and two arrows to triple ones. In such situations, the synchronous approach impedes any further refined analysis of these transitions.

One may also consider a fully asynchronous assumption and generate all the trajectories compatible with the regulatory graph and the logical rules. Naturally, the stable state is conserved and can still be reached from all states lacking CycD activity.

Similarly, in the presence of CycD activity, the system has a unique attractor, but this now includes many intertwined cycles (see Figure 2, top; see also the web supplementary material for higher resolution graphs). Composed by 112 states, this attractor is a terminal strongly connected component in the sense of graph theory. In addition, the part of the state transition graph with CycD active encompasses several dozens of additional (non terminal) strongly connected components, each involving a small number of states (typically four) and potentially representing transient oscillations of few components on the way to the canonical cell cycle.

Interestingly, the seven states forming the synchronous cycle are also found in the terminal strongly connected component found in the asynchronous transition graph (see grey shaded states in Figure 2, top), together with the corresponding single transitions. However, the synchronous transitions may now correspond to multiple asynchronous paths.

2.3 Mixed a/synchronous updating

Between these two extreme updating assumptions, it is possible to define middle terms. One option is to consider the possibility that the realisation of some transitions requires several updating steps. Chaves *et al*. (2005) have recently explored this option to improve their Boolean model analysis of the segment polarity network

Table 1. Logical rules underlying the definition of the logical parameters associated with the regulatory graph of Figure 1

Product	Logical rules leading to an activity of the product	Justification/References
CycD	*CycD*	CycD is an input, considered as constant.
Rb	$(\overline{CycD} \wedge \overline{CycE} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{CycD} \wedge \overline{CycB})$	Rb is expressed in the absence of the cyclins, which inhibit it by phosphorylation (Novak and Tyson, 2004; Taya, 1997); it can be expressed in the presence of CycE or CycA if their inhibitory activity is blocked by p27 (Coqueret, 2003).
E2F	$(\overline{Rb} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{Rb} \wedge \overline{CycB})$	E2F is active in the absence of Rb, that blocks E2F self-transcriptional activation (Helin, 1998), and in the absence of CycA and CycB, that inhibit E2F (Novak and Tyson, 2004); CycA may be present, if its inhibitory activity is blocked by p27 (Coqueret, 2003).
CycE	$(E2F \wedge \overline{Rb})$	CycE activity requires the presence of E2f and the absence of Rb (Helin, 1998).
CycA	$(E2F \wedge \overline{Rb} \wedge \overline{Cdc20} \wedge \overline{(Cdh1 \wedge Ubc)}) \vee (CycA \wedge \overline{Rb} \wedge \overline{Cdc20} \wedge \overline{(Cdh1 \wedge Ubc)})$	The transcription of CycA is activated by E2F in the absence of Rb, which blocks this activation (Helin, 1998), in the absence of Cdc20, as well as of the pair formed by Cdh1 and UbcH10, which both lead to the degradation of CycA (Harper *et al*., 2002; Rape and Kirschner, 2004); CycA is stable in the absence of its inhibitors Rb, Cdc20, and of the pair Cdh1 and UbcH10.
p27	$(\overline{CycD} \wedge \overline{CycE} \wedge \overline{CycA} \wedge \overline{CycB}) \vee (p27 \wedge \overline{(\boldsymbol{CycE} \wedge \boldsymbol{CycA})} \wedge \overline{\boldsymbol{CycB}} \wedge \overline{\boldsymbol{CycD}})$	p27 is active in the absence of the cyclins; when p27 is already present, it blocks the action of CycE or CycA (but not both of them) by sequestration (Coqueret, 2003).
Cdc20	*CycB*	CycB indirectly activates Cdc20 (Harper *et al*., 2002).
Cdh1	$(\overline{CycA1} \wedge \overline{CycB}) \vee (Cdc20) \vee (p27 \wedge \overline{CycB})$	The activity of Cdh1 requires the absence of CycB and CycA, which inhibit it by phosphorylation (Harper *et al*., 2002); Cdc20 further activates Cdh1. (Novak and Tyson, 2004); p27 allows the presence of CycA, by blocking its activity.
UbcH10	$(\overline{Cdh1}) \vee (Cdh1 \wedge Ubc \wedge (Cdc20 \vee CycA \vee CycB))$	UbcH10 is active in the absence of Cdh1; this UbcH10 activity can be maintained in the presence of Cdh1 when at least one of its other targets is present (CycA, Cdc20, or CycB) (Rape and Kirschner, 2004).
CycB	$(\overline{Cdc20} \wedge \overline{Cdh1})$	CycB is active in the absence of both Cdc20 and Cdh1, which target CycB for destruction (Harper *et al*., 2002).

The names of the components of the regulatory graph of Figure 1 are listed in the first column. For each one, the second column gives the logical rules specifying its behaviour. More precisely, we have described only the situations where the component is activated (value of the corresponding Boolean variable set to 1), all other situations leading to an inactivation. This description is based on the classical logical formulation, where ''$\wedge$'' stands for ''AND'', ''$\vee$'' stands for (inclusive) ''OR'', and the negation is written by a bar over the term. As an example, considering the case of CycE, there are eight non-zero parameters attached to CycE, specifying the different combinations of incoming interactions which lead to an activation of CycE (cf. the GINML file on the *GINsim* website). These can be summarised by the logical formula ''E2F active and Rb not active, whatever the state of the other components''. Finally the last column provides some justifications for the logical rules, together with references.

involved in the segmentation of the trunk of Drosophila embryos. Here, we propose an alternative approach enabling the combination of synchronous and asynchronous assumptions depending on the regulatory element or on the nature of transition considered. Indeed, depending on available knowledge or on the biological questions addressed, it may be necessary to go into fine grain dynamical analysis for only a subset of regulatory components. To deal with these issues, the last version of *GINsim* enables the user to group components into different classes, and to assign a *priority level* to each of these classes. In case of concurrent transition calls, *GINsim* first updates the gene(s) belonging to the class with the highest *ranking*. For each regulatory component class, the user can further specify the desired updating assumption, which then determines the treatment of concurrent transition calls inside that class. When several classes have the same ranking, concurrent transitions are treated under an asynchronous assumption (no priority).

To illustrate this approach, we have first built two priority classes, which arguably group faster *versus* slower biochemical processes. In the highest ranked transition priority class, we have included the degradations of E2F, CycE, CycA, Cdc20, UbcH10, CycB, as well as all transitions (in both directions) for CycD, Rb, p27 (Kip1) and Cdh1. The remaining transitions corresponding to synthesis rates (of E2F, CycE, CycA, Cdc20, UbcH10, and CycB) are grouped in a lower priority class. Using these two priority classes, both considered under the asynchronous assumption, we still obtain a single terminal strongly connected component (not shown) involving 34 states (to compare with the seven states obtained with the standard synchronous treatment, *versus* the 112 states in the fully asynchronous case without priority).

The analysis of this component reveals that some pathways are clearly unrealistic, as they skip the activation of some crucial cyclins, for example. To eliminate these spurious pathways, one can further refine the priority classes, taking into account additional information. Here, we can exploit the fact that several transitions are controlled by similar regulatory mechanisms and group them into synchronous classes. This leads to the definition of the four transition classes displayed in Table 2.

For this last prioritisation, we obtain a smaller terminal strongly connected component involving 18 states, which combine single and multiple transitions. This mixed graph is thus much simpler that the fully asynchronous transition graph. This graph enables a finer description of the sequence of events characteristic of the normal cell cycle than in the fully synchronous case. However, the data presently available do not allow a clear distinction between the different alternative pathways.

2.4 Mutant simulations

Beyond a faithful reproduction of the wild-type behaviour, a good cell cycle model should enable the simulation of various types of

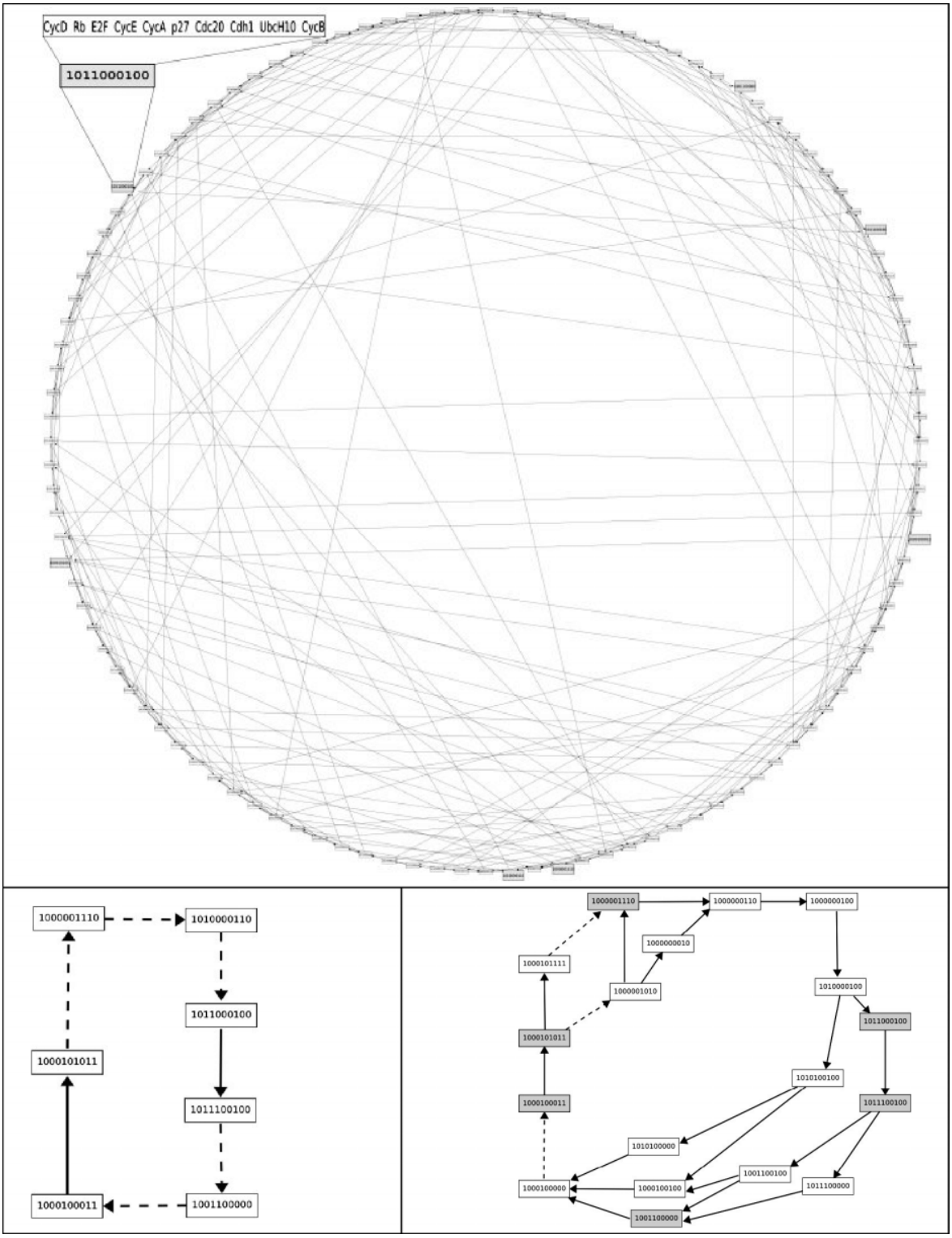


Fig. 2. Simulations of the wild-type cell cycle based on the Boolean model defined in Figure 1 and Table 1. Each vertex (node) represents one state, with the regulatory components ordered as mentioned in the top panel. The three state transition graphs correspond to the comprehensive asynchronous (top), the synchronous (bottom left), and a mixed (bottom right) assumptions. Note the difference of complexity between the asynchronous and synchronous graphs. In the bottom panels, solid arrows stand for single transitions, and dotted arrows for multiple transitions. The seven states involved in the synchronous cycle are grey-shaded in the asynchronous and mixed state transition graphs. For larger resolution pictures, see *GINsim* website.

Table 2. Priority transition classes used to obtain the strongly connected component shown in the bottom right panel of Figure 2

Rank	Type	Transitions
1	Asynchronous	CycD, Rb, p27, Cdh1, E2F↓, CycE↓
1	Synchronous	CycA↓, Cdc20↓, Ubc↓, CycB↓
2	Asynchronous	E2F↑, CycE↑, CycA↑, Cdc20↑
2	Synchronous	Ubc↑, CycB↑

The symbols ↓ and ↑ specify the (decreasing or increasing) direction of the considered transitions (by default, both directions are considered, *e.g.*, for CycD).

perturbations, in particular, the addition of drugs interfering with cell growth or cyclin activities, or the presence of loss-of-function or gain-of-function mutations in some of the core regulatory genes.

In this respect, *GINsim* provides a simple interface to constraint selected regulatory components within specific value intervals. Depending on the initial state(s), once a regulatory component has reached the corresponding value interval in the course of the simulation, all transitions leading outside of this interval are automatically discarded. This function greatly eases the simulation of loss-of-function and gain-of-function mutants (a table compiling our mutant analyses is maintained on the GINsim web page). As we have represented molecule families by single components, the comparison between *in silico* and experimental results are not always straightforward. However, up to now, all our simulation results are consistent with available experimental data (loss *versus* preservation of cell cycle depending on the mutant considered), but a few exceptions like the case of the p27 loss-of-function, for which our model predicts a stable state in the absence of CycD, whereas published data support the existence of oscillations in this situation. This discrepancy is likely due to the crude representation of CycE and Rb activity levels in terms of Boolean variables and could be solved by using ternary variables for these elements.

3 CONCLUSIONS AND PROSPECTS

In this paper, we have assessed the power of the logical approach, already in its simplest Boolean form, for the modelling of a complex protein interaction network. We have further presented extensions of our software *GINsim* to enable detailed studies of the asymptotical behaviour of complex systems, with synchronous, asynchronous or mixed treatment of concurrent transitions.

As shown through the analysis of a model for the mammalian cell cycle, a relatively simple logical model captures most qualitative dynamical features of the wild-type network, as well as of documented mutants. Strikingly, even simplistic synchronous simulations give rise to (only) two attractors consistent with available data, as well as with the simulations of Novak and Tyson (2004). On the one hand, we obtain a stable state in the absence of CycD, which matches our knowledge of quiescent cellular states when growth factors are lacking. On the other hand, in the presence of CycD, all trajectories converge towards a unique complex dynamical cycle. This is a favourable situation for the synchronous assumption, as no spurious cycle is generated.

However, the synchronous dynamics obtained does not allow the temporal separation of multiple regulatory activity changes. In contrast, asynchronous updating does allow finer temporal analyses, but the resulting state transition graph is very complex and encompasses many incompatible or unrealistic pathways. Leaning on specific *GINsim* functions, we have thus considered systematic ways to combine synchronous and asynchronous transitions, taking advantage of existing information on kinetics or regulatory mechanisms. This application thus illustrates the flexibility of the combination of different updating assumptions.

The logical formalism used should further enable the identification of the regulatory circuits playing the most crucial dynamical roles (Thomas *et al.*, 1995). Our present model comprises 132 different circuits, involving from one to nine regulatory elements. A preliminary analysis suggests that only a dozen of these circuits are functional in some region of the variable space, most of the time only in the absence of CycD. The precise role of these different circuits has still to be clarified.

Modelling the molecular regulatory network controlling mammalian cell cycle is clearly a challenging and long-term enterprise. Focusing on the core network controlling the mammalian cell cycle, our present Boolean model corresponds to a relatively high level abstraction of our knowledge of the cellular system, which involves many variants for several of the molecular species considered (E2F, RB...). In this respect, the generation of extensive functional genomics data sets should prove of great help to delineate the specific expression and interaction patterns of these variants (for a pioneering attempt to exploit various kinds of functional genomic data sets to dynamically characterise the molecular network controlling the cell cycle in yeast, see the recent article by de Lichtenberg *et al.*, 2005). On the basis of our generic, abstract model, several extensions or refinements can now be considered, including the use of multilevel variables wherever biological justifications can be advanced, further specifications and enrichments of this model in reference to specific cell types, or yet the inclusion of additional control modules.

A substantial increase in the sophistication of the logical models considered will lead to combinatorial problems, *e.g.*, to identify all attractors or to analyze the trajectories leading to these attractors. To prepare the ground to deal with such combinatorial problems, we are exploring different approaches. First, we use constraint programming to delineate attractors from simple or composed logical models without computing the whole state transition graph (Devloo *et al.*, 2003). Next, we have developed and implemented a set of translations rules enabling the export of parameterised regulatory graphs into standard or coloured Petri nets, thereby enabling the use of the various dynamical analysis tools developed by this lively community (Chaouiya *et al.*, in press). Finally, we are presently evaluating the application of temporal logic formalisms (*e.g.*, Computational Tree Logics) to assess the existence of specific dynamical pathways, or to encompass specific temporal information (Bernot *et al.*, 2004; Batt *et al.*, 2005).

Ultimately, more quantitative models are needed to explore fine grain aspects of the control of the cell cycle, *e.g.*, modulations of the cycle period or of its amplitude. In this respect, Petri nets constitute an interesting framework to refine discrete models, leaning on existing hybrid or stochastic extensions. Alternatively, one may use sets of differential or stochastic equations, but even in this case, a preparatory logical analysis should prove useful when dealing with large and complex regulatory networks.

ACKNOWLEDGEMENTS

We wish to thank A. Ciliberto and K. Helin, as well as E. Remy, P. Ruet and B. Mossé, for insightful discussions on biological, and mathematical aspects of this work. We acknowledge financial support from the European Commission (contract LSHG-CT-2004-512143), the French Research Ministry (ACI IMPbio), the CNRS, and the INRIA (ARC MOCA).

REFERENCES

Batt,G., Ropers,D., de Jong,H., Geiselmann,J., Mateescu,R., Page,M., Schneider,D. (2005) Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in *Escherichia coli*. *Bioinformatics*, **21**, i19–i28.

Bernot,G., Comet,J.P., Richard,A., Guespin,J. (2004) Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *J. Theor. Biol*., **229**, 339–347.

Chaouiya,C., Remy,E., Mossé,B., Thieffry,D. (2003) Qualitative analysis of regulatory graphs: A computational tool based on a discrete formal framework. *Lect. Notes Control Inf. Sci*., **294**, 119–126.

Chaouiya,C., Remy,E., Thieffry,D. (in press) Petri net modelling of biological regulatory networks. *J. Discrete Algorithms.*

Chaves,M., Albert,R., Sontag,E.D. (2005) Robustness and fragility of Boolean models for genetic regulatory networks. *J. Theor. Biol*., **235**, 431–449.

Coqueret,O. (2003) New roles for p21 and p27 cell-cycle inhibitors: a function for each cell compartment? *Trends Cell Biol*., **13**, 65–70.

de Jong,H. (2002) Modelling and simulation of genetic regulatory systems: A literature review. *J. Comput. Biol*., **9**, 67–103.

de Lichtenberg,U., Jensen,L.J., Brunak,S., Bork,P. (2005) Dynamic complex formation during the yeast cell cycle. *Science*, **307**, 724–727.

Devloo,V., Hansen,P., Labbe,M. (2003) Identification of all steady states in large networks by logical analysis. *Bull. Math. Biol*., **65**, 1025–1051.

Dimova,D.K., Dyson,N.J. (2005) The E2F transcriptional network: old acquaintances with new faces. *Oncogene*, **24**, 2810–2826.

Fuß,H., Dubitzky,W., Downes,C.S., Kurth,M.J. (2005) Mathematical models of cell cycle regulation. *Brief. Bioinformatics*, **6**, 163–177.

Glass,L., Kauffman,S.A. (1973) The logical analysis of continuous non-linear biochemical control networks. *J. Theor. Biol*., **39**, 103–129.

Gonzalez,A.G., Naldi A.,Sánchez,L., Thieffry,D., Chaouiya,C. (2006) GINsim: a software suite for the qualitative modelling, simulation and analysis of regulatory networks. *Biosystems*, **84**, 91–100.

Harper,J.W., Burton,J.L., Solomon,M.J. (2002) The anaphase-promoting complex: it's not just for mitosis anymore. *Genes Dev*., **16**, 2179–2206.

Helin,K. (1998). Regulation of cell proliferation by the E2F transcription factors. *Curr. Opin. Genet. Dev*., **8**, 28–35.

Huang,J.N., Park,I., Ellingson,E., Littlepage,L.E., Pellman,D. (2001) Activity of the APCCdh1 form of the anaphase-promoting complex persists until S phase and prevents the premature expression of Cdc20p. *J. Cell Biol*., **154**, 85–94.

Kauffman,S.A. (1993) *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press.

Novak,B., Tyson,J.J. (2004) A model for restriction point control of the mammalian cell cycle. *J. Theor. Biol*., **230**, 563–579.

Olashaw,N., Bagui,T.K., Pledger,W.J. (2004) Cell Cycle Control A Complex Issue. *Cell Cycle*, **3**, 263–264.

Rape,M., Kirshner,W.W. (2004) Autonomous regulation of the anaphase-promoting complex couples mitosis to S-phase entry. *Nature*, **432**, 588–595.

Tamrakar,S., Rubin,E., Ludlow,J.W. (2000) Role of pRb dephosphorylation in cell cycle regulation. *Front. Biosci*., **5**, 121–137.

Taya,Y. (1997) RB kinases and RB-binding proteins: new points of view. *Trends Biochem. Sci*., **22**, 14–17.

Tessema,M., Lehmann,U., Kreipe,H. (2004) Cell cycle and no end. *Virchows Arch*., **444**, 313–323.

Thomas,R. (1991) Regulatory networks seen as asynchronous automata: a logical description. *J. Theor. Biol*., **153**, 1–23.

Thomas,R., Thieffry,D., Kaufman,M. (1995) Dynamical behaviour of biological regulatory networks—I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bull. Math. Biol*., **57**,247–276.

Zhang,H.S., Gavin,M., Dahiya,A., Postigo,A.A., Ma,D., Luo,R.X., Harbour,J.W., Dean,D.C. (2000) Exit from G1 and S phase of the cell cycle is regulated by repressor complexes containing HDAC-Rb-hSWI/SNF and Rb-hSWI/SNF. *Cell*, **101**, 79–89.

A.3 Traduction de modèles logiques en réseaux de Petri

Cette publication donne une vue d'ensemble de la traduction des modèles logiques en réseaux de Petri (voir section 8.3). Le principe de cette traduction a été mis au point avant mon arrivée dans l'équipe, j'ai participé à son adaptation aux diagrammes de décision et implémenté la méthode résultante dans GINsim.

L'article inclus ici est en cours de révision.

Petri net representation of multi-valued logical regulatory graphs

C. Chaouiya[1,2], A. Naldi[1,3], E. Remy[4], D. Thieffry[1,3,5]

[1] INSERM U928 - TAGC, Marseille, France
[2] Instituto Gulbenkian de Ciência, Oeiras, Portugal
[3] Université de la Méditerranée, Marseille, France
[4] Institut de Mathématiques de Luminy, Marseille, France
[5] CONTRAINTES Project, INRIA-Rocquencourt, Le Chesnay, France

October 23, 2009

Abstract

Relying on a convenient logical representation of regulatory networks, we propose a generic method to qualitatively model regulatory interactions in the standard elementary and coloured Petri net frameworks.

Logical functions governing the behaviours of the components of logical regulatory graphs are efficiently represented by Multivalued Decision Diagrams, which are also at the basis of the translation of logical models in terms of Petri nets. We further delineate a simple strategy to sort trajectories through the introduction of priority classes (in the logical framework) or priority functions (in the Petri net framework).

We also focus on qualitative behaviours such as multistationarity or sustained oscillations, identified as specific structures in state transition graphs (for logical models) or in marking graphs (in Petri nets). Regulatory circuits are known to be at the origin of such properties. In this respect, we present a method that allows to determine the functionality contexts of regulatory circuits, *i.e.* constraints on external regulator states enabling the corresponding dynamical properties.

Finally, this approach is illustrated through an application to the modelling of a regulatory network controlling T lymphocyte activation and differentiation.

Keywords: Gene regulation, biological networks, regulatory circuits, logical modelling, Petri nets, signal transduction, cell differentiation.

1 Introduction

Most essential cellular processes are controlled by regulatory networks involving diverse regulatory interactions, *e.g.* transcriptional regulations of target genes, protein modifications, diffusion and sequestring of signalling molecules. Due to the growing complexity of these networks, proper understanding of their dynamical properties requires the development of adequate formal representations, as well as efficient modelling and analysis tools. Approaches generally used to model such regulatory networks include graph theory, Boolean networks, differential equations, or yet stochastic equations (see reviews in [8, 34]). It is worth noting that current data on the molecular processes governing regulatory interactions remains largely qualitative. Indeed, precise information

on kinetic parameters or concentration levels are often lacking. This is particularly true for the regulation of gene expression, as the role of a regulatory product is often just characterised as activating or inhibiting a target gene in a given context. The semantics associated with biological interactions varies: while in a chemical reaction, the reactants are consumed, the expression levels of a transcriptional regulator seldomly change during the regulatory process. One successful approach to qualitatively model such regulatory networks is the *logical method* initially developed by René Thomas and collaborators [42, 43, 44, 3] (see also the related approach in [19]). Indeed this method was productively used to model a diversity of regulatory interactions, beyond transcriptional genetic regulation, as demonstrated in [10, 11, 14, 33]. However, we face a classical combinatorial explosion when we analyse large networks. This is one of the motivations that drove us to propose a systematic translation of logical regulatory models into Petri nets (PNs), to take advantage of the corresponding simulation and analysis tools. Another interesting prospect of the PN representation of regulatory networks lies in the coupling of metabolic pathways with regulatory processes acting upon these pathways (see [37] for a first step in this direction). Finally, PNs open the way to quantitative extensions (*e.g.* adding stochastic rates on the transitions [15, 39, 24], or considering hybrid models [25, 9]).

So far, PNs have been mainly employed to model and analyse metabolic networks, since PNs are well adapted to the representation of the consumption/production semantics related to chemical reactions. The representation of a regulatory relation by means of standard PNs is less straighforward. Section 2 introduces the logical approach for the modelling of regulatory networks. This allows us to define the representation of logical regulatory graphs by means of standard or coloured PNs (Section 3). One strategy to ease the analysis of large regulatory networks consists in cutting irrelevant trajectories thanks to the consideration of priority classes [10]. This drives us to discuss the introduction of such priorities in our models of regulatory networks in Section 4. Another interesting point of the logical framework is the establishment of the relationships between the occurrence of regulatory circuits and specific dynamical properties (see [41] and references therein). In Section 5, we recall how regulatory structures relate to multi-stationarity and oscillatory behaviours. The determination of functionality contexts is explained and its transposition in the context of PN representation of logical regulatory graphs is discussed. All these concepts are illustrated by a biological application dealing with a simplified model of the control of T lymphocyte activation and differentiation. The paper ends with a discussion of future prospects on the use of Petri nets for the modelling and analysis of complex biological regulatory networks.

2 Multi-valued logical modelling of regulatory networks

Regulation refers to the molecular mechanisms responsible for changes in concentration or activity of a functional product. Such mechanisms range from transcriptional regulation to protein modifications [2]. Regulation functions (or response functions) are usually represented by sigmoid or step functions [42]. It is then assumed that the regulatory effect becomes noticeable when the level of the regulator reaches a given threshold. Hence it makes sense to consider regulatory networks as discrete event systems. In this context, we rely on the logical modelling of regulatory networks introduced by René Thomas and collaborators [43, 44, 3]. In this framework, a discrete variable is associated to each regulatory component to represent its qualitative levels of expression (for a gene) or of activity (for a protein), and logical rules define the behaviours of the regulatory components as functions of the levels of their regulators.

Definition 1 A logical regulatory graph (LRG) *is a directed labelled multigraph*[1] $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \mathcal{E}, \Theta, \mathcal{K})$ *where,*

- $\mathcal{G} = \{g_1, \ldots, g_n\}$ *is the set of nodes, representing* genes *(or, more generally,* regulatory components*).*
- $\mathcal{M}ax : \mathcal{G} \to \mathbb{N}^*$ *associates a maximum level*[2] $\mathcal{M}ax(g_i) = Max_i$ *to node* g_i*. In the sequel,* x_i *denotes the* current level *of* g_i *(*$x_i \in \{0, \ldots, Max_i\}$*), a state of the system is thus given as a vector* $x = (x_i)_{i=1,\ldots,n}$*.*
- $\mathcal{E}$ *is a finite multiset of ordered pairs of elements of* $\mathcal{G}$ *representing* regulatory interactions*. If* $Max_i > 1$*,* g_i *may have different effects onto a component* g_j*, depending on level* x_i*. Hence, the arc connecting* g_i *to* g_j *may be a multi-arc encompassing different interactions. The multiplicity of the arc* (g_i, g_j) *(i.e. the number of its constitutive interactions), is denoted* $m_{i,j}$ *(*$1 \leq m_{i,j} \leq Max_i$*). Loops (even multi-loops) are allowed: an arc* (g_i, g_i) *denotes a self-regulation of* g_i*.*
- Θ *is a labelling function, which associates a* threshold *to each element of* $\mathcal{E}$*. More precisely,* $\theta_{i,j,k}$ *is associated to the* k^{th} *interaction between* g_i *and* g_j *(denoted* $(g_i, g_j, k), k \in \{1, \ldots, m_{i,j}\}$*), with* $1 \leq \theta_{i,j,1} < \cdots < \theta_{i,j,m_{i,j}} \leq Max_i$*. This interaction is* active*, when* x_i*, the level of its source* g_i *lays between the threshold of this interaction and that of the next interaction:* $\theta_{i,j,k} \leq x_i < \theta_{i,j,k+1}$ *(by convention,* $\theta_{i,j,m_{i,j}+1} = Max_i + 1$*). In other words, the interaction* (g_i, g_j, k) *is active in all states* x *for which* $\theta_{i,j,k} \leq x_i < \theta_{i,j,k+1}$*.*

  For each $g_j \in \mathcal{G}$*,* $Reg(j)$ *denotes the set of its regulators:* $g_i \in Reg(j)$ *if and only if* $(g_i, g_j) \in \mathcal{E}$*.*
- $\mathcal{K} = (\mathcal{K}_1, \ldots, \mathcal{K}_n)$ *defines the* logical rules *attached to the nodes specifying their behaviours: each* $\mathcal{K}_i$ *is a multi-valued logical function that gives the target value of* g_i *(the value to which* g_i *should tend), depending on the interactions acting on* g_i *at any state:*

$$\mathcal{K}_i : \left(\prod_{g_j \in Reg(i)} \{0, \ldots, m_{j,i}\} \right) \to \{0, \ldots, Max_i\}\,.$$

Later on, when adequate, g_i will be denoted by i and $(i, j, \theta_{i,j,k})$ will stand for interaction (g_i, g_j, k). Also for convenience, since we work in a discrete framework, we will use the notation $[a, b]$ to represent the integer interval $\{a, \ldots, b\}$ $(a, b \in \mathbb{N},\ a \leq b)$.

Note that the biologists often consider two types of interactions: activations have a positive effect on their targets, whereas repressions have a negative effect on their targets. However the actual effect of an interaction may depend on the presence of co-factors; its sign may even change depending on the context. In any case, the interactions and their signs can be derived from the logical functions.

We represent the behaviour of a LRG by a *state transition graph*, where nodes correspond to states (*i.e.* vectors encompassing the levels of the regulatory components), whereas arcs denote transitions between states. This graph is computed using the functions which indicate the transitions leading from the current state to its potential successor states (see Definitions 3 and 4). Here, we consider an asynchronous updating, where each transition corresponds to a change of a unique variable (see [3] for further details). This choice leads to non-deterministic behaviours.

Definition 2 *A state x of the LRG $\mathcal{R}$ is a n-tuple $(x_1, \ldots, x_n)$ of the levels of the regulatory components. The set of all potential states (or state space) is denoted by $\mathcal{S} = \prod_{i=1}^{n} [0, Max_i]$.*

From Definition 1 it follows that a state $x \in \mathcal{S}$ fully determines, for each g_i, the set of active incoming interactions (hence there is no memory of past states). The next definition specifies the behaviour of each regulatory node as a function defined on the state space $\mathcal{S}$.

Definition 3 *Given a LRG $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \mathcal{E}, \Theta, \mathcal{K})$, the dynamics of $g_i \in \mathcal{G}$ is defined by:*

$$\begin{aligned} \mathcal{F}_i^{\mathcal{K}} : \quad \mathcal{S} &\longrightarrow [0, Max_i] \\ x &\mapsto \mathcal{K}_i \left(\left(\sum_{k=1}^{m_{j,i}} k \, \mathbf{1}_{[\theta_{j,i,k}, \theta_{j,i,k+1}[}(x_j) \right)_{j \in Reg(i)} \right), \end{aligned}$$

where $\mathbf{1}$ *denotes the indicator function.*

In the rest of the paper, to simplify the notations and because of the unique correspondance between $\mathcal{K}_i$ and $\mathcal{F}_i^{\mathcal{K}}$, we will refer to the dynamics as $\mathcal{K}_i$.

Remark 1 *For all $g_i \in \mathcal{G}$, the dynamics $\mathcal{K}_i$ only depends on the values of x_j for $j \in Reg(i)$. In other words, $\mathcal{K}_i(x) = \mathcal{K}_i(x')$, for all $x, x' \in \mathcal{S}$ such that $\forall j \in Reg(i)$, $x_j = x'_j$.*

Definition 4 *Given a LRG $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \mathcal{E}, \Theta, \mathcal{K})$ and a set of initial states $Init \subseteq \mathcal{S}$, the (asynchronous) state transition graph $(\mathcal{S}_{Init}, \mathcal{T})$ is the (finite) directed graph defined as follows:*

- $\forall x \in Init$, $x \in \mathcal{S}_{Init}$,
- $\forall x \in \mathcal{S}_{Init}$, $\exists i | \, \mathcal{K}_i(x) \neq x_i \Rightarrow x' \in \mathcal{S}_{Init}$ *s.t.* $(x, x') \in \mathcal{T}$ *and:*

$$\begin{cases} x'_j = x_j & \forall j \neq i, \\ x'_i = x_i + 1 & \text{if } \mathcal{K}_i(x) > x_i, \\ x'_i = x_i - 1 & \text{if } \mathcal{K}_i(x) < x_i. \end{cases} \tag{1}$$

In other words, given a state x, the level of each g_i tends toward the *target value* given by $\mathcal{K}_i(x)$. If this is greater (*resp.* lower) than x_i (the current value of g_i), there is a call to increase (*resp.* decrease) by one the value of g_i, hence a transition towards a new state having the corresponding updated value for x_i.

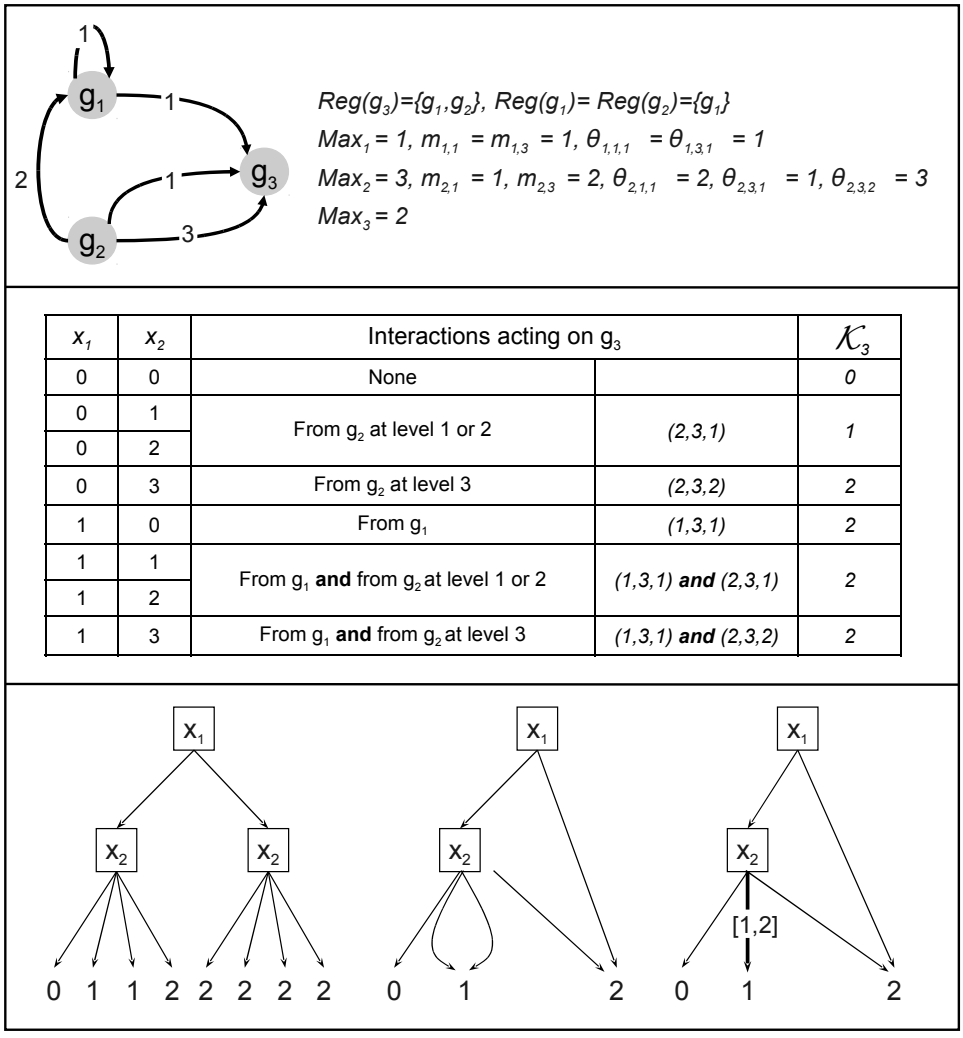
It is worth noting that one might consider $Init = \mathcal{S}$ (as defined in Definition 2) and then obtain the *whole* state transition graph, encompassing all potential states and possible transitions defined by $\mathcal{K}$.

Terminal strongly connected components in the state transition graphs denote regions of the state space where the system is eventually trapped, which we call *attractors*. These attractors might be stable states (states with no successor, *e.g.* corresponding to stable patterns of expressions in a gene regulatory network), or encompass states involved in elementary cycles or in intertwined

cycles (these components indicate stable oscillations of the system, or yet homeostasis that ensures the maintenance of a certain equilibrium).

Throughout the paper, *state transition graphs* will refer to the dynamics of LRGs as defined above, whereas *marking graphs* will be used in the context of Petri net models.

For several years, we have been developing GINsim [13, 27], a software dedicated to the definition and analysis of logical models. Performance considerations led us to propose the use of Reduced



x_1	x_2	Interactions acting on g_3		$\mathcal{K}_3$
0	0	None		*0*
0	1	From g_2 at level 1 or 2	*(2,3,1)*	*1*
0	2			
0	3	From g_2 at level 3	*(2,3,2)*	*2*
1	0	From g_1	*(1,3,1)*	*2*
1	1	From g_1 **and** from g_2 at level 1 or 2	*(1,3,1)* ***and*** *(2,3,1)*	*2*
1	2			
1	3	From g_1 **and** from g_2 at level 3	*(1,3,1)* ***and*** *(2,3,2)*	*2*

Figure 1: Example of a *logical regulatory graph*. The **top panel** displays interactions between three nodes along with the specification of the thresholds and maximal levels. The **middle panel** shows the function $\mathcal{K}_3$, which defines the target value of g_3 depending on the levels of its regulators g_1 and g_2 (equivalently, depending on the combinations of active interactions). Definitions of $\mathcal{K}_1$ and $\mathcal{K}_2$ are omitted here. The **bottom panel** displays the decision tree representing $\mathcal{K}_3$, the corresponding reduced decision diagram (MDD), and, on the right, an illustration of the merging of consecutive edges linked to the same child.

Ordered Multi-valued Decision Diagrams (ROMDDs, denoted MDDs from now on) to internally represent logical functions [26]. In [12], decision diagrams were used to represent state transition graphs and analyse the dynamical properties of Boolean models. In our context, the MDD representation further facilitates the translation of logical models into Petri nets. In particular, it is used by GINsim modules exporting logical models into several PN formats.

To apply efficient algorithms for the analysis of our models, the function $\mathcal{K}_i$, which takes its values in $[0, Max_i]$, is represented in terms of a MDD with the levels x_j of the regulators $g_j \in Reg(i)$ as decision variables [26, 18] (see Figure 1, bottom panel). It is possible to further compact this MDD by merging consecutive edges leading to the same child as explained hereafter.

Following the classical MDD representations, given $g_i \in \mathcal{G}$, for each decision variable x_j that appears in the diagram of $\mathcal{K}_i$, there are $Max_j + 1$ outgoing edges, implicitly labelled with the corresponding value in $[0, Max_j]$. Edges labelled with consecutive values pointing towards the same child can be merged into a unique edge, which is then labelled with the integer interval of these consecutive values. Remaining edges are labelled by intervals containing a unique value for the decision variable. In the resulting diagram, each decision path Φ (from the root node to a leaf labelled $v_\Phi \in [0, Max_i]$) corresponds to a set of assignments of the regulators $g_j \in Reg(i)$ for which the value of $\mathcal{K}_i$ is v_Φ:

- If path Φ encompasses an edge going out the decision variable x_j, the set of assignments of x_j equals the label $[\phi_j, \phi'_j] \subsetneq [0, Max_j]$ (called the Φ assignment interval for x_j) of the edge going out the decision variable x_j.
- If, along the path Φ, a decision variable x_j does not appear (due to the simplification of the MDD), it means that $\mathcal{K}_i(x) = v_\Phi$ does not depend on x_j.

The use of MDD generally leads to a simplified expression of $\mathcal{K}_i$, but the resulting diagram and its complexity may vary depending on the ordering of the decision variables (see *e.g.* [18] and Figure 5 for an illustration).

Figure 1 illustrates a logical graph, the function $\mathcal{K}_3$ of the node g_3, as well as the decision tree and resulting decision diagrams representing $\mathcal{K}_3$. Note that, hereafter, we sometimes depict MDDs as non-completely reduced decision diagrams to facilitate their interpretation.

Remark 2 *Given a function $\mathcal{F} : \mathbb{N}^n \to \mathbb{N}^n$, one can recover a LRG with n regulatory components, the maximal values, the logical rules, the interactions and their thresholds. The recovered interactions will all be* functional *interactions, i.e. interactions with an effect on the levels of their targets observed through the dynamics $\mathcal{F}$ (see Section 5 for the notion of functionality).*

3 Petri net representation of logical models

Petri nets consumption/production semantics is well adapted to the representation of chemical reactions. However, it is less obvious to use the standard PN framework to represent regulatory mechanisms, where (1) there is no modification of the activity level of a regulator acting on its target, (2) the absence of a regulator may have an effect on its target. One could use inhibitory arcs to take into account this last situation, but we aim at proposing a representation using standard elementary Petri nets to take advantage of their powerful analysis framework. Because the level of each component in a LRG is bounded, we can use complementary places instead of inhibitory arcs. In a first step, a rewriting of LRGs into standard PNs is briefly presented. Next, we define a coloured PN version of this translation.

3.1 Standard Petri net representation

The definition below allows the explicit construction of a Petri net from a LRG, based on the MDD representation of the dynamics $\mathcal{K}_i$, where edges are labelled by integer intervals. The resulting Petri net has a behaviour equivalent to that of the original logical model (Property 1). Further details, basic properties and applications of this PN representation of LRGs are provided in [4] for the Boolean case, and in [5] for the multi-valued case. Note that the examples presented throughout this section are not meant to be realistic, but rather to illustrate the rules governing the rewriting and facilitate the understanding.

Definition 5 *Given a LRG $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \mathcal{E}, \Theta, \mathcal{K})$, we define the corresponding Multi-valued Regulatory Petri Net (MRPN) as follows:*

- *For each $g_i \in \mathcal{G}$, there are two places g_i and $\widetilde{g_i}$ that satisfy, for all markings M:*

$$M(g_i) + M(\widetilde{g_i}) = Max_i, \tag{2}$$

  meaning that $\widetilde{g_i}$ is the complementary place of g_i.

- *For $g_i \in \mathcal{G}$, for each path Φ from the root to a leaf of the MDD representing $\mathcal{K}_i$, at most two transitions are defined, one accounting for the increasing tendency (denoted $t^+_{i,\Phi}$), the second accounting for the decreasing tendency (denoted $t^-_{i,\Phi}$). This simplifies when the leaf is associated with an extreme value (see below). For the considered component g_i, a path Φ defines assignment intervals of the levels of g_j in $Reg(i)$: $x_j \in [\phi_j, \phi'_j]$, where $\phi_j, \phi'_j \in [0, Max_j]$ and $\phi_j \leq \phi'_j$.*

- *Transitions $t^+_{i,\Phi}$ and $t^-_{i,\Phi}$ are connected to:*

  - *place g_j, $j \in Reg(i)$, with a test arc weighted ϕ_j,*
  - *place $\widetilde{g_j}$, $j \in Reg(i)$, with a test arc weighted $Max_j - \phi'_j$.*

  When $\phi_j = \phi'_j$, from Equation 2, it suffices to consider only one of these test arcs. If $[\phi_j, \phi'_j] = [0, Max_j]$, the decision variable should be omitted in the reduced MDD, because all possible values lead to the same result (hence places g_j and $\widetilde{g_j}$ are not connected to $t^+_{i,\Phi}$ nor to $t^-_{i,\Phi}$).

  Transition $t^+_{i,\Phi}$ is further connected to:

  - *place g_i, with an outgoing arc (increasing the level of g_i),*
  - *place $\widetilde{g_i}$, with an incoming arc weighted $Max_i - v_\Phi + 1$ (ensuring that the current level of g_i is less than the target value v_Φ) and an outgoing arc weighted $Max_i - v_\Phi$ (accounting for the decreasing by one of the current marking of this complementary place).*

  Symmetrically, transition $t^-_{i,\Phi}$ is further connected to:

  - *place $\widetilde{g_i}$, with an outgoing arc (decreasing the level of g_i),*
  - *place g_i, with an incoming arc weighted $v_\Phi + 1$ (ensuring that the current level of g_i is more than the target value v_Φ) and an outgoing arc weighted v_Φ (accounting for the decreasing by one of the current marking).*

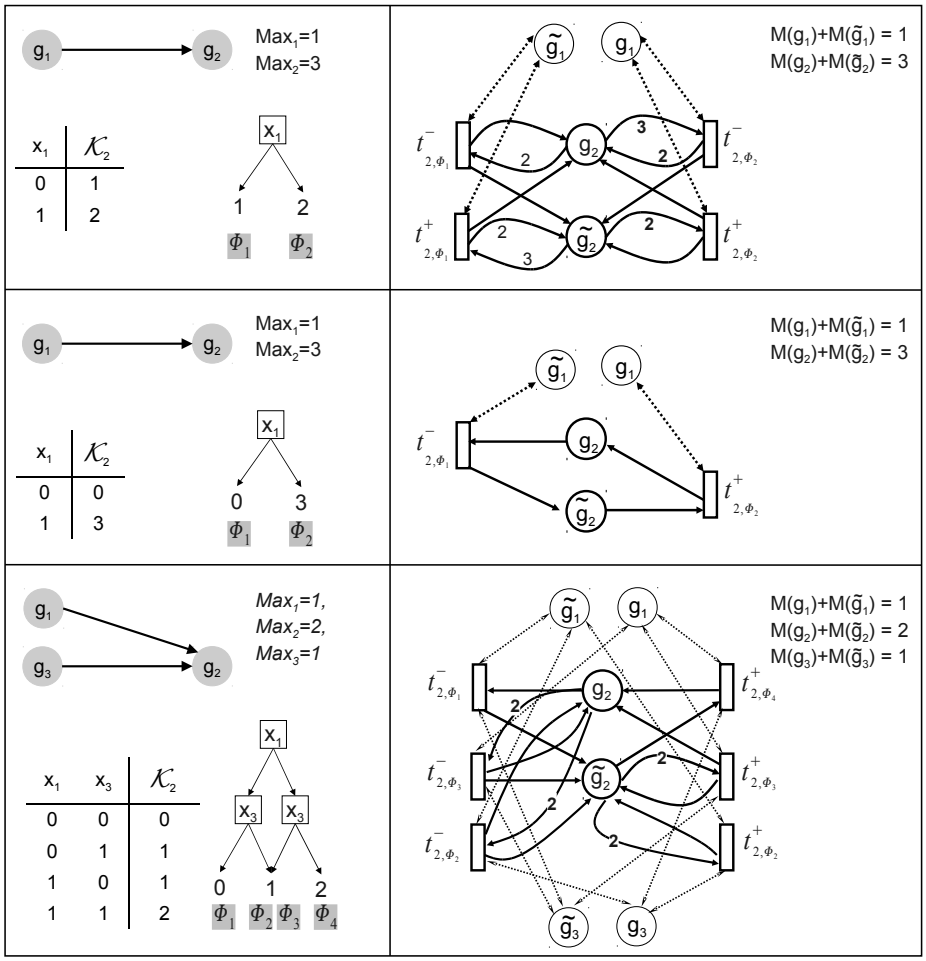


Figure 2: LRG translations into standard PNs. The **top panel** displays a component g_1 (with 2 levels) that regulates component g_2 (4 levels); $\mathcal{K}_2$ is given in the form of a table and the corresponding MDD. When g_1 is present, g_2 tends to 2, while the *basal value* of g_2 is 1 (*i.e.* the value of g_2 in the absence of its regulator g_1). On the right side, the corresponding MRPN is displayed. For each component, two *complementary* places are defined. For each path in the decision diagram of $\mathcal{K}_2$, two transitions are defined: *e.g.*, $\mathcal{K}_2(1) = 2$ is represented by t^+_{2,Φ_2} and t^-_{2,Φ_2}. If g_1 is marked (g_1 at level 1) and g_2 not marked (g_2 at level 0, 3 tokens in $\widetilde{g_2}$), then t^+_{2,Φ_2} is enabled and its firing *increases* the marking of g_2; if g_1 is marked and g_2 has 3 tokens (its highest level), then t^-_{2,Φ_2} is enabled and its firing *decreases* the marking of g_2. In the **middle panel** the same toy example is considered but with $\mathcal{K}_2$ taking extreme values. On the right side, the corresponding MRPN is simpler with only one transition for each situation (t^+_{2,Φ_1} and t^-_{2,Φ_2} are of no use here). The **bottom panel** gives a third example, with a resulting MRPN encompassing one transition for each path leading to extreme values (Φ_1 and Φ_4), and two transitions for each path leading to the intermediate value 1 (Φ_2 and Φ_3).

From the definition above, it follows that, for all $g_i \in \mathcal{G}$ and Φ a path in the decision diagram associated to $\mathcal{K}_i$, when $v_\Phi = 0$ or $v_\Phi = Max_i$ (the value of the $\mathcal{K}_i$ for this assignment of the regulators is *extreme*), only one transition is relevant. Indeed, if $v_\Phi = 0$, transition $t^+_{i,\Phi}$ can be omitted as, by construction, there will never be Max_i+1 tokens in place $\widetilde{g}_i$. Similarly, if $v_\Phi = Max_i$, transition $t^-_{i,\Phi}$ can be omitted as there will never be Max_i+1 tokens in place g_i. Figure 2 illustrates the standard PN representation of LRGs.

Remark 3 *Regarding self-regulations, in Petri nets, all transitions are meant to represent events that effectively change the state of the modelled system. If g_i is a self-regulator, then it appears as a decision variable in the diagram of $\mathcal{K}_i$. If Φ is a path leading to a value v_Φ and if g_i is assigned to value v_Φ (i.e. $\phi_i = v_\Phi$), then no transition will be generated for Φ. Figure 3 illustrates this situation.*

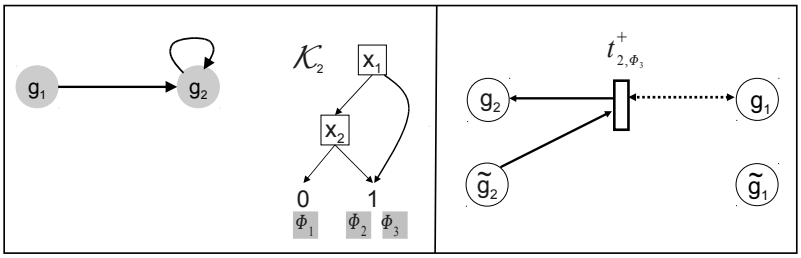


Figure 3: Representation of a self-regulation in Petri nets. In the example shown, there is no need to define transitions for g_2 self-maintainance, *i.e.* transitions related to the paths: $\Phi_1 = (g_1 = 0,\, g_2 = 0)$ and $\Phi_2 = (g_1 = 0,\, g_2 = 1)$.

Property 1 *In the state transition graph $(\mathcal{S}, \mathcal{T})$ of a LRG $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \mathcal{E}, \Theta, \mathcal{K})$, there exists a transition between two states x and x' iff there exists an enabled transition t in the associated MRPN defined as in Definition 5 such that $M[t\rangle M'$ (t is enabled by the marking M and its firing leads to the new marking M') with, for all $k = 1, \ldots n$:*

$$\begin{aligned} M(g_k) &= x_k \qquad & M(\widetilde{g_k}) &= Max_k - x_k\,, \\ M'(g_k) &= x'_k \qquad & M'(\widetilde{g_k}) &= Max_k - x'_k\,. \end{aligned}$$

Proof: Let consider $x, x' \in \mathcal{S}$ such that $(x, x') \in \mathcal{T}$, and let M be the marking of the associated MRPN such that $M(g_i) = x_i, \quad M(\widetilde{g_i}) = Max_i - x_i$, for all $g_i \in \mathcal{G}$. We show first that this marking enables a transition that, when fired, leads to a new marking M' with $M'(g_i) = x'_i, \quad M'(\widetilde{g_i}) = Max_i - x'_i$, for all $g_i \in \mathcal{G}$.

From Definition 4, there exists an i such that $x_i = x'_i \pm 1$ and $\mathcal{K}_i(x) \neq x_i$. The state x determines a unique decision path $\Phi(x)$ in the MDD representing $\mathcal{K}_i$. Recall that if the decision variable x_j has no interval assignment $[\phi_j(x), \phi'_j(x)]$ in the path $\Phi(x)$, places g_j and $\widetilde{g_j}$ are not connected to the transitions $t^+_{i,\Phi(x)}$ and $t^-_{i,\Phi(x)}$. Let $v_{\Phi(x)} = \mathcal{K}_i(x) \in [0, Max_i]$. Then,

1. If $0 \leq v_{\Phi(x)} < x_i$, then $M(g_i) \in [v_{\Phi(x)} + 1, Max_i]$. Moreover, for all $g_j \in Reg(i)$ such that the decision variable x_j is assigned in $\Phi(x)$, we have:

$$\begin{array}{rcl} x_j = M(g_j) & \in & [\phi_j(x), \phi'_j(x)], \\ Max_j - x_j = M(\widetilde{g_j}) & \in & [Max_j - \phi_j(x), Max_j - \phi'_j(x)]. \end{array}$$

   Hence $t^-_{i,\Phi(x)}$ is enabled (if $v_{\Phi(x)} \neq 0$, $\Phi(x)$ also defines a transition $t^+_{i,\Phi(x)}$ that is not enabled in M because $M(\widetilde{g_i}) \in [0, v_{\Phi(x)}]$).

2. If $x_i < v_{\Phi(x)} \leq Max_i$, then $M(\widetilde{g_i})(= Max_i - x_i) \in [Max_i - v_{\Phi(x)} + 1, Max_i]$. Moreover, for all $g_j \in Reg(i)$ such that the decision variable x_j is assigned in $\Phi(x)$, we have:

$$\begin{array}{rcl} x_j = M(g_j) & \in & [\phi_j(x), \phi'_j(x)], \\ Max_j - x_j = M(\widetilde{g_j}) & \in & [Max_j - \phi_j(x), Max_j - \phi'_j(x)]. \end{array}$$

   Hence $t^+_{i,\Phi(x)}$ is enabled (if $v_{\Phi(x)} \neq Max_i$, $\Phi(x)$ also defines a transition $t^-_{i,\Phi(x)}$ that is not enabled in M because $M(g_i) \in [0, v_{\Phi(x)}]$).

3. When $t^-_{i,\Phi(x)}$ is fired ($v_{\Phi(x)} < x_i$),

$$M'(g_i) = M(g_i) - 1 = x_i - 1 = x'_i, \qquad M'(\widetilde{g_i}) = M(\widetilde{g_i}) + 1 = Max_i - x_i + 1 = Max_i - x'_i.$$

4. When $t^+_{i,\Phi(x)}$ is fired ($x_i < v_{\Phi(x)}$),

$$M'(g_i) = M(g_i) + 1 = x_i + 1 = x'_i, \qquad M'(\widetilde{g_i}) = M(\widetilde{g_i}) - 1 = Max_i - x_i - 1 = Max_i - x'_i.$$

Reciprocally, let consider a marking M enabling a transition t, and M' the marking such that $M[t > M'$, then there exists i such that:

$$\begin{array}{rcl} & M'(g_i) = M(g_i) + 1 \quad \text{and} \quad M'(\widetilde{g_i}) = M(\widetilde{g_i}) - 1, \\ \textit{or} & M'(g_i) = M(g_i) - 1 \quad \text{and} \quad M'(\widetilde{g_i}) = M(\widetilde{g_i}) + 1. \end{array}$$

Transition t being enabled by M, for all $g_j \in Reg(i)$, the marking of place g_j verifies $M(g_j) \in [\omega_{t,j}, Max_j - \omega'_{t,j}]$, where $\omega_{t,j}$ is the weight of the test arc connecting t to g_j (there is no arc if $\omega_{t,j} = 0$), and $\omega'_{t,j}$ is the weight of the test arc connecting t to $\widetilde{g_j}$ (there is no arc if $\omega'_{t,j} = 0$). Recall that $M(g_j)$ defines a level x_j of the component $g_j \in Reg(i)$, hence M defines a state assignment, in particular it gives the levels of the regulators of g_i. Therefore, from Definition 5, M allows us to recover a path Φ in the MDD of $\mathcal{K}_i$. The assignment of any x_j along Φ verifies: $\phi_j = \omega_{t,g_j}$ and $\phi'_j = \omega'_{t,g_j}$. Moreover, we have (because t is enabled in M):

$$\begin{array}{rcl} & t = t^+_{i,\Phi} \quad \text{and} \quad M(g_i) = x_i \leq v_\Phi - 1, & (\text{arc from } \widetilde{g_i} \text{ to } t \text{ weighted } Max_i - v_\Phi + 1), \\ \text{or} & t = t^-_{i,\Phi} \quad \text{and} \quad M(g_i) = x_i \geq v_\Phi + 1, & (\text{arc from } g_i \text{ to } t \text{ weighted } v_\Phi + 1). \end{array}$$

Therefore, $v_\Phi \neq x_i$ and there exists a transition in $(\mathcal{S}, \mathcal{T})$ from state x (defined by M) to state x' such that $\forall j \in \mathcal{G}, x'_j = x_j$, and $x'_i = x_i + 1$ if $t = t^+_{i,\Phi}$, $x'_i = x_i - 1$ if $t = t^-_{i,\Phi}$. ■

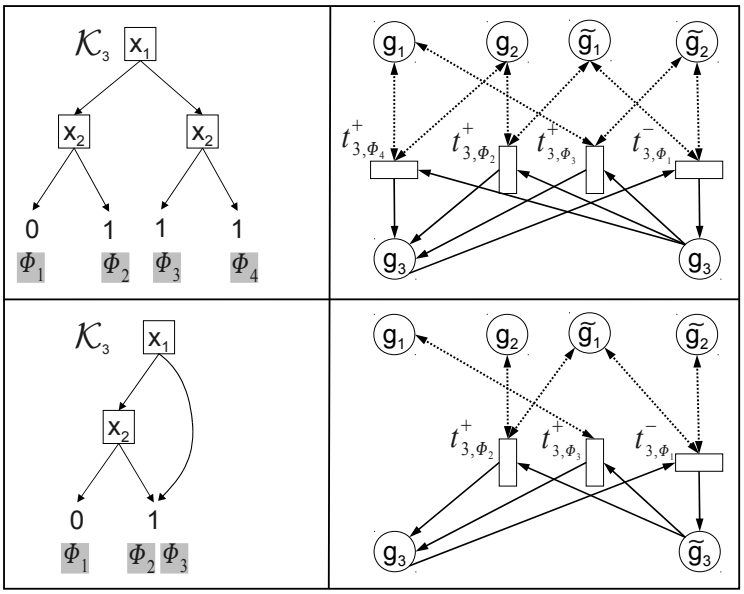


Figure 4: Translations of decision tree *versus* decision diagram representations of a logical function into standard Petri nets. Here, we consider a node g_3 with two regulators g_1 and g_2. Top: the decision tree representing $\mathcal{K}_3$ is given with the corresponding MRPN, which encompasses one transition for each path of the tree (hence four transitions). Bottom: the decision diagram representing $\mathcal{K}_3$, leading to three relevant paths, and thus to three transitions in the derived MRPN.

The MDD representation of $\mathcal{K}$ leads to more compact Petri nets compared to those obtained from decision trees (as in [5]; see Figure 4 for an illustration). Different orderings of the variables in the MDD may generate different reductions. However, although the number of transitions may vary, it can be proved that the resulting dynamics (the marking graphs) are isomorph. This leads to the following property.

Property 2 *Given a LRG $\mathcal{R}$, two different orderings of the regulatory nodes can lead to different MRPNs, which have the same dynamical behaviour (i.e. their marking graphs are isomorph for a given initial state x).*

The proof easily follows from Property 1. Figure 5 displays the two MRPNs obtained from the same LRG, considering different orderings of the variables.

Two invariant properties easily follow from Definition 5. The MRPN corresponding to a LRG is covered by n P-invariants (n being the number of regulatory components). Moreover, T-invariants are always defined as pairs of transitions related to pairs of complementary places. These properties can be used as a consistency check of the MRPN. This leads to the question of a possible reverse transformation, which is not addressed here. Note that several LRGs might be recovered from a given MRPN (*i.e.* a PN satisfying structural constraints such as the P and T-invariants properties). This has been already emphasized for the Boolean case in [4]. Indeed, if we consider a pair of complementary places, we have no way to determine which one corresponds to the current level of the regulatory product (see Figure 6).

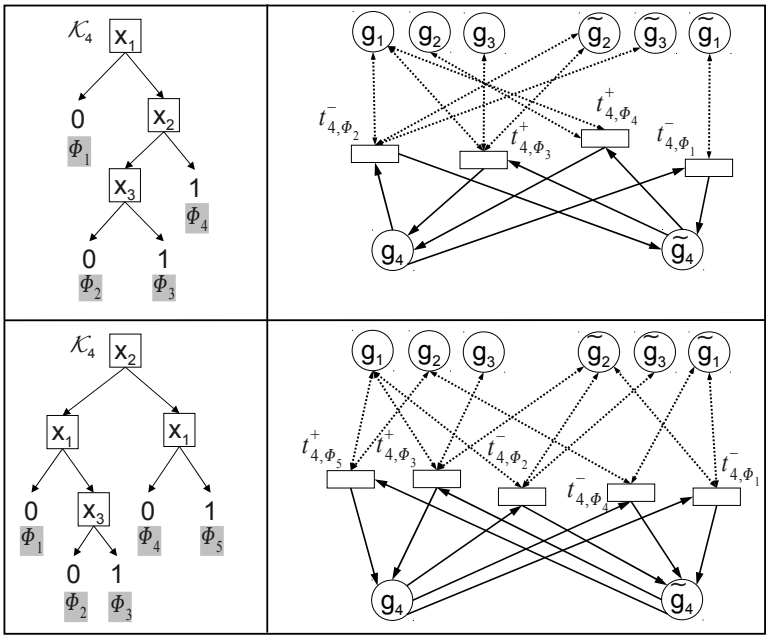


Figure 5: Effect of decision variable ordering on the structure of the resulting MRPN. Here, we consider a node g_4 with three regulators g_1, g_2 and g_3. On the left, two decision trees are displayed, both representing $\mathcal{K}_4$. On the right, the resulting MRPNs are given. The number of paths in the decision tree determines the number of transitions in the corresponding MRPN. Here, there is a unique transition for each path since the values labelling the leaves are 0 or $Max_4 = 1$.

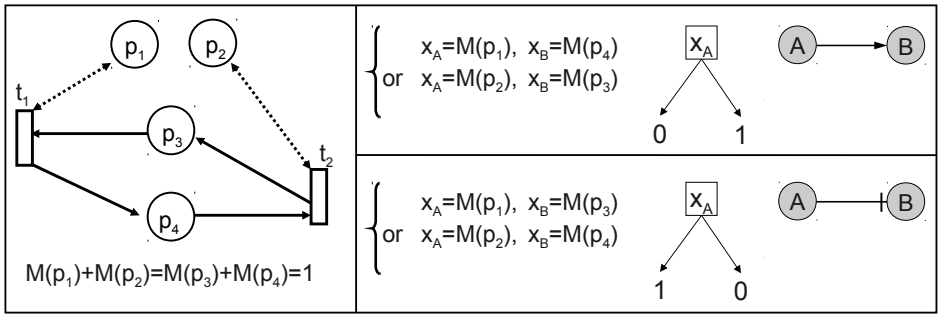


Figure 6: Two different LRGs induced from a single MRPN-like PN. On the left, part of a MRPN is given, with two pairs of complementary places (p_1, p_2) and (p_3, p_4), defining two regulatory nodes A and B. On the right, the four possible choices for the places representing the current levels of A and B, and the corresponding decision trees and associated regulatory structures (activation *versus* inhibition). For example (top case), if x_A is given by the marking of p_1 and x_B by the marking of p_4, then the network structure indicates that when $x_A = 1$, if $x_B = 0$ (*i.e.* $M(p_3) = 1$) transition t_1 may fire and increase the level of B. In other words, this choice denotes a situation where A is an activator of B.

This systematic rewriting has been implemented in our software GINsim [27]. In [6], the rewriting rules are demonstrated for a multi-valued logical model of the genetic switch controlling the

lysis-lysogeny decision in the bacteriophage lambda. This and other PN models can be downloaded from the GINsim web site, as text files in the INA format [17].

3.2 Coloured Petri net representation

MRPNs might seem complex and the regulatory structures underlying the model are not easy to visualize. In this section, we present a coloured version of MRPNs, called Coloured Regulatory Petri Nets (CRPNs), where there is one place for each regulatory component g_i, and one transition governing the evolution of the marking $M(g_i)$ (for further details, see [5]).

Definition 6 *A LRG $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \mathcal{E}, \Theta, \mathcal{K})$ can be represented as a Coloured Regulatory Petri Net (CRPN), with, for all $g_i \in \mathcal{G}$:*

- *one place g_i containing a unique token, which value is the current level x_i of g_i ($x_i \in [0, Max_i]$);*
- *one transition t_i connected :*
  - *to each place g_j such that $g_j \in Reg(i)$, by tests arcs labelled with the arc expression x_j;*
  - *to place g_i, by an incoming arc labelled with the arc expression x_i, and by an outgoing arc labelled with the arc expression[3]: $x_i + sign(\mathcal{K}_i(x) - x_i)$;*
- *a guard associated to transition t_i, to ensure that t_i is enabled only if g_i is called to change its current value, hence this guard is defined as $x_i \neq \mathcal{K}_i(x)$.*

Figure 7 illustrates the CRPN representing the logical regulation of a gene.

A property similar to Property 1 can be stated, equating the LRG state transition graph to the related CRPN marking graph.

Finally, contrary to MRPNs, a unique LRG can be recovered from a given CRPN.

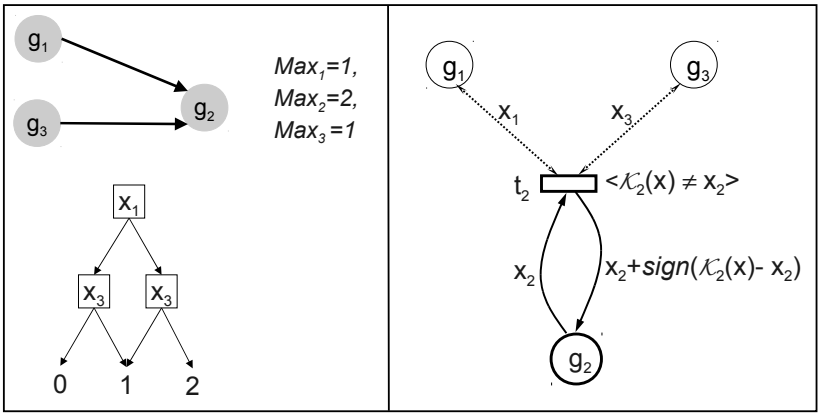


Figure 7: Coloured Petri net representation of a LRG. In the simple case depicted here, the unique transition t_2 that governs the behaviour of g_2 is connected to the regulators of g_2 by test arcs, which read the current values of g_1 and g_3. A guard associated to t_2 ensures that the current level of g_2 (x_2) is different from its target value $\mathcal{K}_2(x)$. When t_2 is enabled, it increases or decreases the value x_2 by one (according to the comparison of x_2 and $\mathcal{K}_2(x)$). See Figure 2, bottom panel, for a "flat" (MRPN) version of this model.

4 Introducing priorities

For specific initial conditions, the asynchronous dynamics of realistic regulatory graphs often leads to huge numbers of states. In the logical framework, as introduced in Section 2, we consider asynchronous updatings for the definition of state transition graphs, which match the corresponding MRPN marking graphs. Another updating policy frequently considered consists in applying all updating calls at once [19]. In [40], the authors propose a PN rewriting of Boolean regulatory networks with such a synchronous updating. Although generating simpler, deterministic state transition graphs, the synchronous updating often generates spurious behaviours [43]. In contrast, since the asynchronous updating makes no assumption on the delays related to the increase or decrease orders, the generated dynamics is highly non-deterministic and realistic trajectories are hidden by numerous unrealistic ones. The inclusion of qualitative delays in logical models has been considered by several authors (*e.g.* [43, 1, 35, 36]). Siebert and Bockmayr pointed out in [36] that delays for synthesis or decay of a regulatory product might be context sensitive. Sound delays are generally difficult to obtain from experimental data. Moreover, other questions arise in case of preemption: when a process (synthesis or decay) is interrupted for a while, which delay value should be considered when the process resumes?

To circumvent these difficult questions, we chose a simple strategy consisting in sorting trajectories through the introduction of priority classes [10]. This possibility has been introduced in GINsim by allowing the user to group components into different classes, and to assign a priority level to each of these classes. In case of concurrent transition calls, GINsim only updates the component(s) belonging to the class with the highest ranking. For each regulatory component class, the user can further specify the desired updating assumption (synchronous or asynchronous), which then determines the treatment of concurrent transition calls inside that class. When several classes have the same ranking, concurrent transitions are treated under an asynchronous assumption (no priority). Moreover, similarly to the definitions of transitions t^+ and t^- in Definition 5, one can distinguish between increasing and decreasing tendencies when updating node levels. In biological terms, increasing (respectively decreasing) tendencies generally correspond to synthesis (respectively degradation) processes. GINsim further enables the distinction between these two types of updatings.

Introduction of priorities in MRPNs is straighforward, if we only consider asynchronous classes. Taking into account asynchronous classes in an MRPN simply consists in defining a *priority function* that assigns a positive integer to each transition defining its priority level [22]. As priorities restrict the enabling of transitions, it is clear that the marking graph of the prioritised model is a subgraph of the marking graph of the original model. Hence, we must be aware that the introduction of priorities in a model may affect the reachability of attractors, and may even result to additional or modified attractors. Figure 8 illustrates how prioritisation may affect the configuration of attractors.

The use of priority classes eases the analysis of large regulatory networks (see, *e.g.*, [33]). Priorities can be set on the basis of biological knowledge. Hence, the paths lost in the marking graphs are arguably not realistic. It could be interesting to define subtler classes, for example depending not only on the sign of the update, but also on the combination of interactions leading to this update. In this case, it would be necessary to impede the automatic simplification of the MRPN (as in Figure 4), because this amounts in grouping several situations, which could then belong to distinct priority classes.

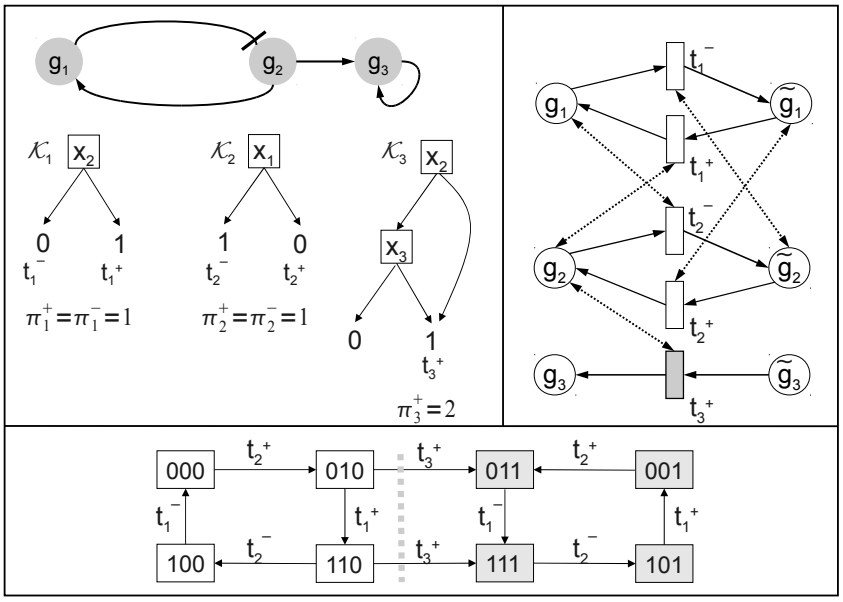


Figure 8: Effect of prioritisation on the dynamics. The **top left panel** shows a LRG with the specification of two priority classes: all updating calls for g_1 and g_2 have a higher priority ($\pi_1 = \pi_2 = 1$), g_3 has a lower priority ($\pi_3 = 2$). The **top right panel** displays the corresponding MRPN, and transition t_3^+ that has a lower priority is greyed out. The marking graph is given in the **bottom panel**, generated from the initial state $(0, 0, 0)$. If priorities are implemented, the dynamics is restricted to the left-hand cycle, otherwise it will end up in the rightmost terminal cycle. The transition from the first cycle to the terminal cycle is only possible through t_3^+, which is always in conflict with another transition with a higher priority. However, the right hand cycle is still an attractor for the prioritised model, if one considers an initial marking in this cycle.

5 Feedback circuits and functionality context analysis

For complex regulatory networks, R. Thomas enunciated several rules binding the dynamical behaviour to the presence of specific types of regulatory circuits. More precisely, he has conjectured that a necessary condition for multistationarity is the presence of a positive circuit (*i.e.*, containing an even number of inhibitions), whereas a necessary condition for homeostasis and/or sustained, stable oscillations is the presence of a negative circuit (with an odd number of inhibitions), cf. [42, 44, 41] and references therein. These rules inspired several theorems refering to different frameworks (see [38, 29, 31] and references therein).

However, the presence of a regulatory circuit is not sufficient to enable the corresponding dynamical property. The simple example presented in Figure 9 illustrates this point. When g_3 is present ($x_3 = 1$), the negative circuit involving g_1 and g_2 induces a cycle in the state transition graph; when g_3 is absent ($x_3 = 0$), the cyclic behaviour disappears: the circuit is no longer *functional*. A circuit is said to be *functional* if it does generate homeostasis in the case of a negative circuit, or multistationarity in the case of a positive circuit. In [26], we associate a functionality context to each circuit of a regulatory graph. When a circuit is embedded in a regulatory graph, its (external) inputs may annihilate the expected property (as illustrates in Figure 9). The func-

tionality context of a circuit defines constraints on the levels of external regulators of the circuit enabling the circuit to be functional.

In what follows, we propose a formal definition for regulatory circuits and their functionality contexts along with a method to determine these contexts.

Definition 7 *A circuit* $\mathcal{C} = \{(i_l, i_{l+1}, \theta_{l,l+1}),\ l = 1, \ldots, k\}$ *of a regulatory graph* $\mathcal{R} = (\mathcal{G}, \mathcal{M}ax, \mathcal{E}, \Theta, \mathcal{K})$ *is a subgraph such that* $\{i_1, \ldots, i_k\} \in \mathcal{G}$ *and* $(i_l, i_{l+1}, \theta_{l,l+1}) \in \mathcal{E}$ *for* $l = 1, \ldots k$ *with the convention* $k + 1 = 1$.

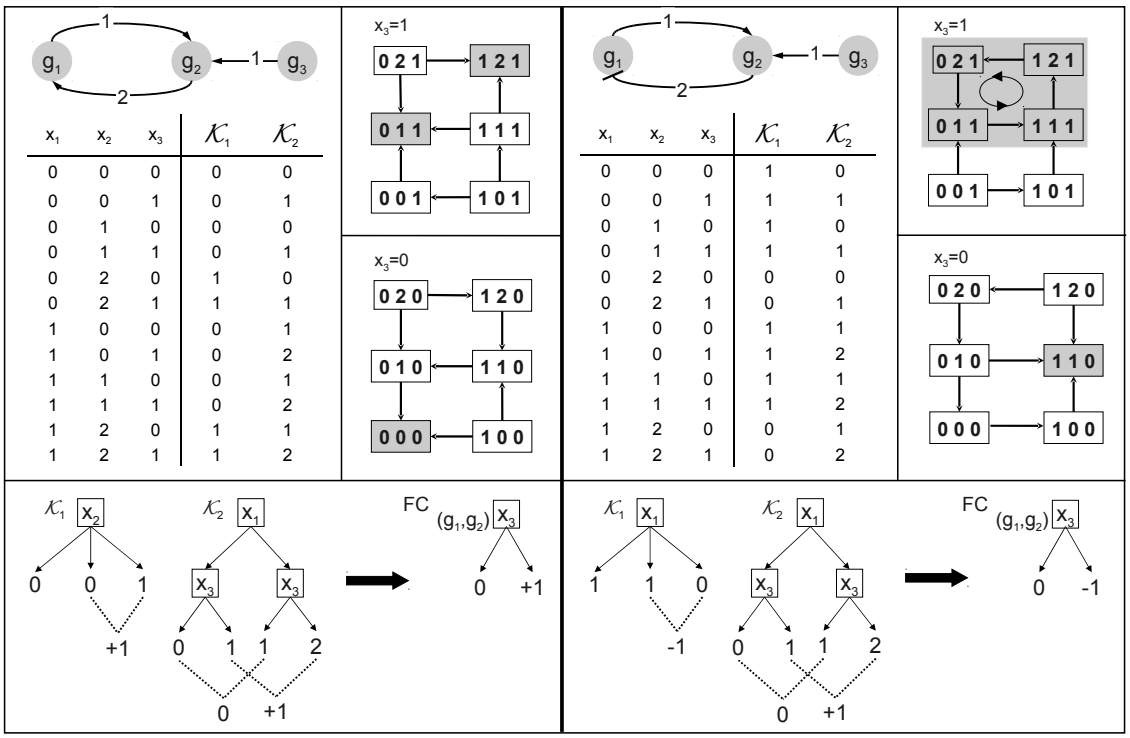


Figure 9: Functionality context of regulatory circuits. An example of positive circuit is displayed on the left, along with a negative circuit on the right. For each case, the upper-left panel displays the circuit and the table defining $\mathcal{K}_1$ and $\mathcal{K}_2$. The state transition graph in presence (*resp.* absence) of g_3 is shown in the upper-right (*resp.* middle-right) panel. The bottom panels show the decision trees representing the functions $\mathcal{K}_i$ and their use to determine the functionality context of the circuit. For example, the functionality context of the interaction $(1, 2, 1)$ within the circuit is determined from the decision tree of $\mathcal{K}_2$ by comparing the values of leaves reached from $x_1 = 0$ (absence of g_1) with those reached from $x_1 = 1$ (illustrated by the dotted lines below the trees). In both circuits, the interaction $(2, 1, 2)$ is always functional. The interaction $(1, 2, 1)$ is functional within the circuit only in the presence of g_3. Indeed, $\mathcal{K}_2(0, 0) = 0$ and $\mathcal{K}_2(1, 0) = 1$: even if g_1 has a positive effect on g_2, in the absence of g_3, this effect is not sufficient to make g_2 cross the threshold of the next interaction. The context of the whole circuit is then obtained by combining those of the individual interactions. The state transition graphs are consistent with these results: for the positive (*resp.* negative) circuit, multistability (*resp.* oscillations) appears only for $x_3 = 1$. Attractors (stable states and attracting cycles) are emphasized in grey.

The following definition formalises the notion of functionality of an interaction within a regulatory circuit.

Definition 8 *An interaction $(i_l, i_{l+1}, \theta_{l,l+1})$ of a circuit $\mathcal{C}$ is functional if and only if there exists $x \in \mathcal{S}$ with $x_{i_l} = \theta_{l,l+1} - 1$ and $x' \in \mathcal{S}$ with $x'_{i_l} = \theta_{l,l+1}$ and $x'_k = x_k$ for all $k \neq i_l$, such that:*

$$\begin{array}{rrll} & \mathcal{K}_{i_{l+1}}(x) & < \theta_{l+1,l+2} & \leq \mathcal{K}_{i_{l+1}}(x'), \\ \text{or} & \mathcal{K}_{i_{l+1}}(x') & < \theta_{l+1,l+2} & \leq \mathcal{K}_{i_{l+1}}(x). \end{array}$$

Definition 8 establishes that the interaction $(i_l, i_{l+1}, \theta_{l,l+1})$ is functional provided its activity affects the activity of the following interaction of the circuit (going out i_{l+1}). This depends on the values of $\mathcal{K}_{l+1}$, considering values $\theta_{i_l,i_{l+1}} - 1$ and $\theta_{i_l,i_{l+1}}$ for i_l and all possible values of other regulators of i_{l+1}.

We can then define the sign of an interaction: when the increase of the source across its threshold drives an increase (*resp.* decrease) of the target across the threshold of the following interaction, the interaction is functional and positive (value $+1$) (*resp.* negative, value -1), otherwise the interaction is not functional (value 0).

From Definition 8, one can determine the set of assignments for the regulators of i_{l+1} (except for i_l) for which the interaction $(i_l, i_{l+1}, \theta_{l,l+1})$ is functional (its sign is not 0). This gives the context of functionality of the interaction.

Definition 9 *The functionality context of a circuit $\mathcal{C}$ is given by the intersection of the functionality contexts of its interactions.*

Note that if the functionality context is empty, then the circuit is not functional.

In [26], we define a logical function that yields the sign of the interaction targeting i_{l+1}. This function is again represented as an MDD and is obtained from the logical function $\mathcal{K}_{l+1}$. The functionality context of a circuit is defined as the intersection of the contexts of its constitutive interactions. Hence, combining the functionality conditions of all the interactions of a circuit in a logical conjunction, we obtain the functionality context of the whole circuit.

Summarising, the computational method given in [26] allows the determination of the functionality contexts (and signs) of a circuit (notice that, depending on the values of its regulators, the sign of a regulatory circuit might change). The procedure involves two steps:

- the determination of the sign of each interaction, using decision diagrams,
- the computation of the product of these values by combining these MDDs.

In the resulting decision diagram, the paths leading to non-zero leaves define the functionality context of the circuit.

In [30], we defined how to determine, in Boolean Regulatory PNs (BRPNs), the functionality contexts of regulatory circuits. The method relies on the comparison of the effects of relevant pairs of transitions along the circuit. This comparision is performed by analysing the matrices *Pre* and *Post* of the net (see Section 6 for an illustration). The proposed procedure is based on the analysis of large matrices and, more important, is only valid for Boolean models. Its extension to the multi-valued case is not straighforward. However, this seminal work inspired the principles driving the computational method mentioned above, which is based on MDDs manipulations. Hence, currently

we analyse the functionality contexts of regulatory circuits in the logical framework, using the MDD representation of the logical functions. The algorithm has been implemented into GINsim and efficiently determines the circuit functionality contexts for large LRGs (including multi-valued cases). In principle, this algorithm could be adapted to the analysis of regulatory circuits in CRPNs as a unique transition is associated to each regulatory component carrying its logical function.

6 Biological illustration: a simplified qualitative model for T cell activation and differentiation

6.1 Logical model

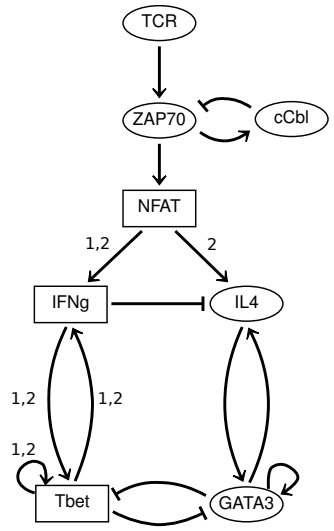
T lymphocytes play a central role in the regulation of the adaptive immune response. Early differentiation steps in the thymus lead to a pool of antigen-naive T helper cells (denoted Th0), displaying different antigen-specific T Cell Receptor (TCR). Once in the periphery, when a Th0 cell encounters a matching antigen (displayed by an antigen presenting cell), the activation of the TCR receptor triggers a cascade of events ultimately leading to the activation of the cell and cell differentiation. Th0 cells can then differentiate into Th1 or Th2 subtypes, which enhance different (cellular *versus* humoral) immune responses. To illustrate the definitions and methods introduced above, we introduce a model (see Figure 10) integrating two recently published models. The first of these models encompasses 45 components transducing the signals received by the TCR and two co-receptors on the cell membrane down to transcription factors in the nucleus [20]. Involving 17 components, the second model focuses on the control of the differentiation of Th0 cells into Th1 *versus* Th2 subtypes, characterised by the activation of Tbet and the secretion of IFNγ, versus the activation of GATA3 and the secretion of IL4, respectively [23]. These two models share only two components (TCR and NFAT) and can thus be easily coupled using the logical formalism. The coupled model was then reduced (in part automatically with a novel GINsim prototype, in part manually). We retained only eight interacting components (out of 60), selected to preserve the main dynamical properties of the original models:

- the transient oscillations of ZAP70 and cCbl upon T cell activation;
- the coexistence of four stable states accounting for different T cell populations.

For proper parameterisation, this logical model has four stable states, corresponding to the following cell types:

- Th0: all components are inactive;
- Th1: Tbet and IFNγ are active at medium level (1);
- Th1*: Tbet and IFNγ are active at their highest level (2);
- Th2: GATA3 and IL4 are active.

Starting from the Th0 state, a transient activation of the TCR leads to a transient activation of NFAT. Under constant TCR activation, the level of NFAT oscillates with that of ZAP70, which is involved in a negative circuit with cCbl. The activations of IFNγ and IL4 are coupled to that of NFAT. These two cytokines in turn control the activation of Tbet and GATA3. Self-regulations and



Node	*Value*	*Function*
TCR	–	–
$ZAP70$	1	$TCR \wedge \neg cCbl$
$cCbl$	1	$ZAP70$
$NFAT$	2	$ZAP70$
IFNg	1	$(NFAT_1 \wedge \neg Tbet) \vee (Tbet_1 \wedge \neg NFAT)$
	2	$(NFAT_1 \wedge Tbet_1) \vee NFAT_2 \vee Tbet_2$
Tbet	1	$\neg GATA3 \wedge (Tbet_1 \vee IFNg_1)$ $\wedge \neg (Tbet_2 \vee IFNg_2)$
	2	$\neg GATA3 \wedge (Tbet_2 \vee IFNg_2)$
$IL4$	1	$NFAT_2 \vee GATA3$
$GATA3$	1	$(GATA3 \vee IL4) \wedge \neg Tbet$

Figure 10: Reduced logical model for T cell activation and differentiation. On the left, the regulatory graph is displayed. Blunt arrows denote inhibitions while regular ones denote activations. Circled components are represented by Boolean variables, while rectangular boxes emphasise ternary components. Interaction thresholds are specified on the drawing if different from 1 ($1, 2$ denote two thresholds for multi-arcs). The table on the right gives the logical rules for each component of the model. For the sake of readability, the logical functions are displayed as logical statements, using the classical connectors ($\neg$, $\wedge$, $\vee$ denoting NOT, AND and OR operators, respectively). TCR stands for $TCR = 1$, $Tbet_1$ and $Tbet_2$ stand for $Tbet = 1$ and $Tbet = 2$ respectively (the same for remaining variables). The rules are given for the non-zero target values (target value is 0 for the complementary statement).

cross-inhibitions of these transcription factors constitute the switch enabling a stable differentiation of Th1 and Th2 subtypes.

The model encompasses 11 regulatory circuits. Five of these circuits play crucial roles:

- The negative circuit involving ZAP70 and cCbl is functional when the TCR is active. In this context, it triggers oscillations of the activity levels of these two components, further driving oscillations of downstream targets.
- The positive, cross-inhibitory circuit involving GATA3 and Tbet is functional in the presence of IFNγ and IL4. This circuit prevents the cell from reaching a chimeric state.
- The positive circuit involving GATA3 is functional in the absence of Tbet and IL4.
- The positive circuit involving medium Tbet is functional in the absence of GATA3 and IFNγ.
- The positive circuit involving high Tbet is functional in the absence of GATA3 and for low or medium IFNγ.

The auto-regulations of GATA3 and Tbet enable the memorisation of their activation (maintenance) above the corresponding thresholds. However, the coexistence of Tbet and GATA3 is forbidden

by their cross-inhibitory effects, thus leading to four stable states characterised by the exclusive expression of one of these factors, or yet of none of them.

6.2 MRPN representation

We have generated the MRPN corresponding to the LRG as defined in Figure 10. It encompasses 16 places and is covered by 8 P-invariants, which correspond to the 8 pairs of complementary places (one pair for each regulatory component):

$$M(IFNg) + M(\widetilde{IFNg}) = 2, \qquad M(Tbet) + M(\widetilde{Tbet}) = 2,$$
$$M(IL4) + M(\widetilde{IL4}) = 1, \qquad M(GATA3) + M(\widetilde{GATA3}) = 1,$$
$$M(cCbl) + M(\widetilde{cCbl}) = 1, \qquad M(ZAP70) + M(\widetilde{ZAP70}) = 1,$$
$$M(NFAT) + M(\widetilde{NFAT}) = 2, \qquad M(TCR) + M(\widetilde{TCR}) = 1.$$

As previously mentioned, the number of transitions can vary, depending on the ordering of the variables (see Figure 5). For example, considering the ordering ($IFN\gamma$, $Tbet$, $IL4$, $GATA3$, $cCbl$, $ZAP70$, $NFAT$, TCR), the MRPN has 28 transitions and 16 T-invariants. In this case, 7 transitions govern the evolution of $Tbet$. Now, if we change the ordering to ($GATA3$, $IFN\gamma$, $Tbet$, $IL4$, $cCbl$, $ZAP70$, $NFAT$, TCR), the MRPN has 25 transitions and 7 T-invariants, and only 3 transitions governing $Tbet$ evolution. This raises the question of finding an optimal ordering for each LRG node, seeking a lower number of transitions.

Figure 11 illustrates the part of the MRPN dealing with the regulation of $Tbet$. Here, we have two T-invariants: $(t^+_{\Phi_1}, t^-_{\Phi_3})$ and $(t^+_{\Phi_2}, t^-_{\Phi_3})$.

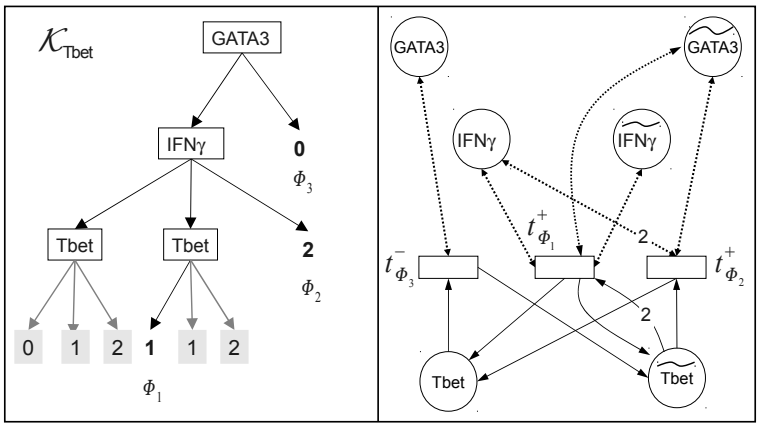
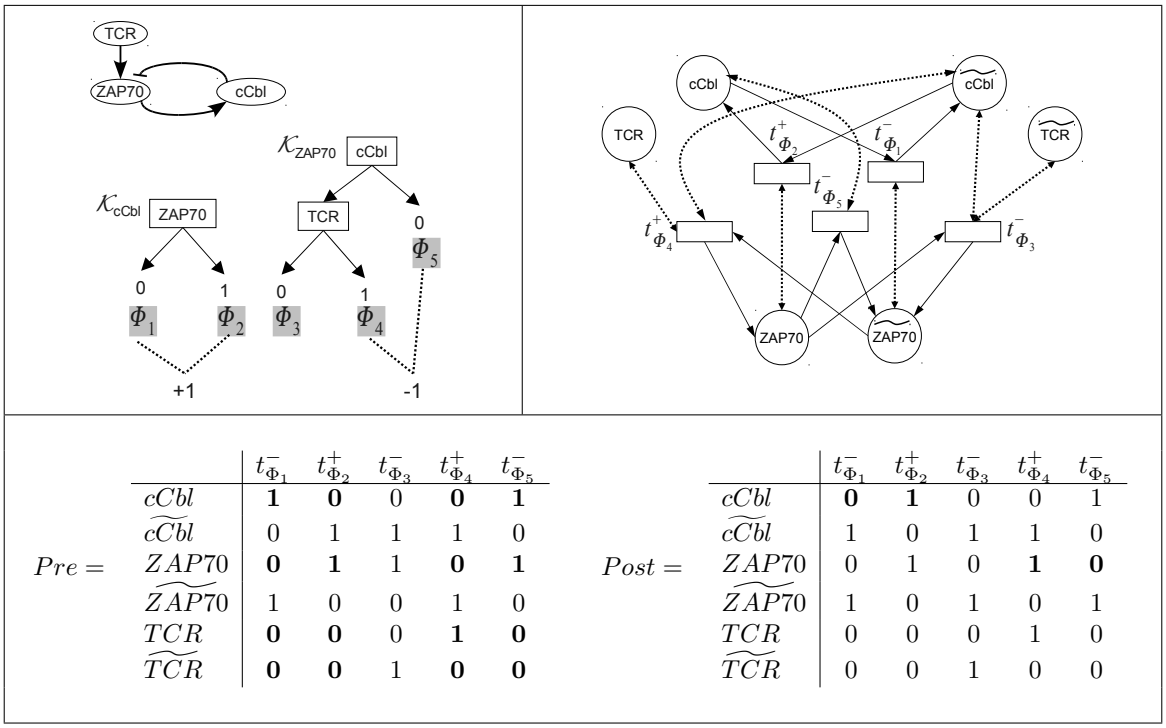


Figure 11: Determination of the MRPN representation for $Tbet$. The **left panel** gives the MDD for $\mathcal{K}_{Tbet}$, which governs the behaviour of $Tbet$. Path names label relevant leaves (Φ_1,Φ_2 and Φ_3), whereas leaves corresponding to situations where $Tbet$ is not called to change (because of its self-regulation) are greyed out. The **right panel** illustrates the MRPN obtained for the regulation rules of $Tbet$, with three transitions.

We have checked that, with an initial marking corresponding to a level 1 for TCR and 0 for

all other components, the marking graph encompasses 336 markings, including four dead markings corresponding to the four cellular states described above.

Focusing on the Boolean regulatory circuit between *cCbl* and $ZAP70$, we now exemplify the determination of functionality contexts (see Figure 12). First, we apply the method proposed in [30], which applies only in the Boolean case and consists in comparing the effects of pairs of transitions related to the components involved in the circuit (see [30] for further details). The relevant pairs of transitions are determined from both the matrix Pre (giving the arcs between places and transitions) and the matrix $Post$ (giving the arcs from transitions to places).



$Pre =$

	$t^-_{\Phi_1}$	$t^+_{\Phi_2}$	$t^-_{\Phi_3}$	$t^+_{\Phi_4}$	$t^-_{\Phi_5}$
$cCbl$	**1**	**0**	0	**0**	**1**
$\widetilde{cCbl}$	0	1	1	1	0
$ZAP70$	**0**	**1**	1	**0**	**1**
$\widetilde{ZAP70}$	1	0	0	1	0
TCR	**0**	**0**	0	**1**	**0**
$\widetilde{TCR}$	**0**	**0**	1	**0**	**0**

$Post =$

	$t^-_{\Phi_1}$	$t^+_{\Phi_2}$	$t^-_{\Phi_3}$	$t^+_{\Phi_4}$	$t^-_{\Phi_5}$
$cCbl$	**0**	**1**	0	0	1
$\widetilde{cCbl}$	1	0	1	1	0
$ZAP70$	0	1	0	**1**	**0**
$\widetilde{ZAP70}$	1	0	1	0	1
TCR	0	0	0	1	0
$\widetilde{TCR}$	0	0	1	0	0

Figure 12: Analysis of the $ZAP70$-*cCbl* circuit. The **top left panel** gives the Boolean circuit encompassing $ZAP70$ and *cCbl* in the LRG of Figure 10. The MDDs associated to functions $\mathcal{K}$ are shown. On these MDDs, we can verify that the interaction from $ZAP70$ to *cCbl* is always functional and positive, whereas the interaction from *cCbl* to $ZAP70$ is functional and negative in the presence of TCR (dotted lines join the situations that have to be compared). The **top right panel** displays the corresponding MRPN. The **bottom panel** shows the submatrices Pre and $Post$ for the three relevant nodes: *cCbl*, $ZAP70$ and TCR. Using these matrices, we can check the functionality context of the interaction from *cCbl* to $ZAP70$: we first select $t^-_{\Phi_3}$, $t^+_{\Phi_4}$ and $t^-_{\Phi_5}$, which change the marking of $ZAP70$. Then, we consider the pair $(t^+_{\Phi_4}, t^-_{\Phi_5})$ because these transitions are differently constrained by *cClb* and have different effects on $ZAP70$ (in contrast with the pair $(t^-_{\Phi_3}, t^-_{\Phi_5})$). Since $t^+_{\Phi_4}$ requires the presence of TCR, whereas $t^+_{\Phi_5}$ is not constrained by TCR, we conclude that the interaction is functional in the presence of TCR. The interaction from $ZAP70$ to *cCbl* is always functional. Elements in bold in the matrices allow us to select the relevant pairs of transitions and to conclude on the functionality of the interactions.

For example, to determine the functionality context of the interaction from *cCbl* towards $ZAP70$, we have to check if it has an effect on $ZAP70$ for fixed levels of TCR (the only external regulator of the circuit, acting on $ZAP70$). For this purpose, we select the pairs of transitions (t, t') such that:

1. $Pre(ZAP70, t) - Post(ZAP70, t) \neq 0$ and $Pre(ZAP70, t') - Post(ZAP70, t') \neq 0$: both t and t' change the value of $ZAP70$;
2. $Pre(cCbl, t) \neq Pre(cCbl, t')$: t and t' account for different constraints on *cCbl* (presence *versus* absence of *cCbl*);
3. $Pre(TCR, t) = Pre(TCR, t')$ or $Pre(\widetilde{TCR}, t) = Pre(\widetilde{TCR}, t')$: t and t' account for compatible constraints on TCR.

For such a pair (t, t'), we further check if $Post(t, ZAP70) \neq Post(t', ZAP70)$. If it is the case, the interaction is functional (the effect on $ZAP70$ is different) for specific values of TCR (given by the constraint on TCR for both t and t').

Here, the pair of transitions satisfying the conditions listed above is $(t^+_{\Phi_4}, t^-_{\Phi_5})$ (see bold elements in the matrices in Figure 12). Since $Pre(TCR, t^+_{\Phi_4}) = 1$, the interaction from *cCbl* towards $ZAP70$ is functional in the presence of TCR.

Figure 12 displays the relevant part of the MRPN encompassing the regulatory circuit between *cCbl* and $ZAP70$, subject to a regulation by TCR. The MDDs are given for $\mathcal{K}_{cCbl}$ and $\mathcal{K}_{ZAP70}$ and, as in Figure 9, dotted lines connect the values that must be compared, and the sign of the interaction is given. This illustrates how the functionality contexts are computed from the MDDs representing the logical functions. Figure 12 also shows the Pre and $Post$ submatrices relevant for the analysis of the $ZAP70 - cCbl$ circuit.

7 Discussion

This article reviews a series of results allowing the translation of logical regulatory models into discrete Petri nets and, thereby, the combination of complementary analytical methods and computational tools (reachability analysis, PN invariants, regulatory circuit analysis, etc.).

A different rewriting of LRGs into Coloured PNs (CPNs) is proposed in [7], which comprises just one place (accounting for the whole state of the LRG) and one transition (accounting for the dynamics of the LRG). Leaning on this CPN representation, the authors further propose a method to automatically generate sets of logical rules (or *logical parameters*) compatible with the topology of a regulatory graph and temporal logic formulae capturing dynamical properties of the corresponding biological system.

Here, we lean on CPN rewriting to further transpose one of the most original aspect of the logical modelling method, namely the analysis of the dynamical roles of the regulatory circuits embedded in complex networks. For a given CRPN, this approach enables the identification of positive or negative circuits at the basis of multistationary properties or sustained oscillatory behaviour, respectively. This is facilitated by the fact that in CRPNs, for each LRG node there is one transition governing its behaviour, contrary to the rewriting proposed in [7], where a unique transition accounts for the behaviours of all the nodes. The proposed method enables the delineation of definite constraints on the marking of the places. These constraints are derived from the logical rules

associated with the transitions feeding the places involved in a circuit. However, further work is required to clarify how interconnected circuits can cooperate to generate more complex behaviours.

We have previously shown how the combination of logical and PN formalisms can be applied to the dynamical modelling and analysis of regulated metabolic pathways ([37]; see [32] for a complementary approach). In this context, the circuit analysis delineated here complements existing PN methods (*e.g.* computation of dead markings, P- and T-invariants, reachability analysis) to better cope with regulated metabolic networks, *i.e.* by providing an abstraction, which is relevant from a dynamical point of view.

We have illustrated PN rewriting using a simplified logical model for the activation and differentiation of T lymphocytes. We are currently developing a full-fledged model encompassing detailed signalling transduction cascades and additional differentiation pathways. The analysis of the resulting network, which currently involves over 60 regulatory components, will clearly benefits from the exploitation of the full set of tools associated with logical (LRG) and PN frameworks. In particular, adequate model reduction methods should help to determine important dynamical properties for large networks (cf. [28] for a first step in this direction, using the logical framework).

Although time is often implicitely considered in logical models, the use of priorities or time delays enables the specification of temporal constraints associated with concurrent processes (cf. Section 4). In this respect, the PN framework offers a variety of refined temporisation methods, from the definition of delay intervals to stochastic firing laws (cf. [21] for an application of timed-PN to model a biological signalling pathway, and [24] for a SPN modelling the yeast cell cycle control).

The modelling and analysis of complex biological regulatory networks implies the capacity to handle and combine different levels of abstractions or details, to identify and compose relevant network modules, and to delineate the associated essential dynamical properties. Whenever experimental data are sufficiently abundant and precise, qualitative PN models (or sub-models) can be refined to generate predictive quantitative models, taking advantage of timed, stochastic or hybrid PN versions (cf., *e.g.*, [39, 25, 9, 21, 24]). In this respect, the progressive integration of methods transposed from other dynamical modelling frameworks and newly developed PN analysis tools should ease the definition of hierarchical or modular models (see *e.g.* [32, 16]).

Acknowledgments.

A.N. has been supported by a PhD grant from the French Ministry of Research and Technology. C.C. acknowledges the support provided by the Calouste Gulbenkian Foundation. This work was further supported by research grants from the French National Agency (projects ANR-06-BYOS-0006 and ANR-08-SYSC-003), and from the Belgian Science Policy Office (IAP BioMaGNet).

Notes

[1]Multigraphs are also called *pseudographs*
[2]$\mathbb{N}^*$ is the set of non-zero natural numbers
[3]For $x \in \mathbb{Z}$, $sign(x) = +1$ if $x > 0$, -1 if $x < 0$, 0 otherwise

References

[1] J. Ahmad, A. Richard, G. Bernot, J-P. Comet, O. Roux. Delays in biological regulatory networks (BRN). *Lecture Notes in Computer Sciences*, 3992: 887–94, 2006.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter. *Molecular Biology of the Cell*. Fifth Edition, Garland Science, Taylor & Francis, 2008.

[3] C. Chaouiya, E. Remy, B. Mossé, D. Thieffry. Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework. *Lecture Notes in Control and Information Sciences*, 294: 119–26, 2003.

[4] C. Chaouiya, E. Remy, P. Ruet, D. Thieffry. Qualitative modelling of genetic networks: From logical regulatory graphs to standard Petri nets. *Lecture Notes in Computer Science*, 3099: 137–56, 2004.

[5] C. Chaouiya, E. Remy, D. Thieffry. Qualitative Petri net modelling of genetic networks. *Lecture Notes in Computer Science*, 4220: 95–112, 2006.

[6] C. Chaouiya, E. Remy, D. Thieffry. Petri net modelling of biological regulatory networks. *Journal of Discrete Algorithms*, 6(2):165-77, 2008.

[7] J.-P. Comet, H. Klaudel, S. Liauzu. Modeling multi-valued genetic regulatory networks using high-level Petri nets. *Lecture Notes in Computer Science*, 3536: 208–27.

[8] H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology*, 1: 67–103, 2002.

[9] A. Doi, M. Nagasaki, H. Matsuno, S. Miyano. Simulation-based validation of the p53 transcriptional activity with hybrid functional Petri net. *In Silico Biology*, 6, 2006.

[10] A. Fauré, A. Naldi, C. Chaouiya, D. Thieffry. Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22: 124–31, 2006.

[11] A. Fauré, A. Naldi, F. Lopez, C. Chaouiya, A. Ciliberto, D. Thieffry. Modular Logical Modelling of the Budding Yeast Cell Cycle. *Molecular BioSystems*, [Epub ahead of print], 2009.

[12] A. Garg, I. Xenarios, L. Mendoza and G. DeMicheli. An Efficient Method for Dynamic Analysis of Gene Regulatory Networks and in-silico Gene Perturbation Experiments. *Lecture Notes in Computer Science*, 4453: 62–76, 2007.

[13] GINsim web page: http://gin.univ-mrs.fr/GINsim/

[14] A. González, C. Chaouiya, D. Thieffry. Logical modelling of the role of the Hh pathway in the patterning of the Drosophila wing disc. *Bioinformatics*, 24: i234-40, 2008.

[15] P. J. Goss, J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proceedings of the National Academy of Sciences of the U. S. A.*, 95: 6750–5, 1998.

[16] E. Grafahrend-Belau, F. Schreiber, M. Heiner, A. Sackmann, B.H. Junker, S. Grunwald, A. Speer , K. Winder , I. Koch. Modularization of biochemical networks based on classification of Petri net t-invariants. *BMC Bioinformatics*, 9: 90, 2008.

[17] INA, Integrated Net Analyzer, tool for the analysis of (Coloured) PNs: www.informatik.hu-berlin.de/~starke/ina.html

[18] T. Kam, T. Villa, R. K. Brayton, A. L. Sangiovanni-Vincentelli. Multi-Valued Decision Diagrams: Theory and Applications. *International Journal on Multiple-Valued Logic*, 4: 9–62, 1998.

[19] S. Kauffman. *The origins of order: Self-organization and selection in evolution.* Oxford University Press, 1993.

[20] S. Klamt, J. Saez-Rodriguez, J. A. Lindquist, L. Simeoni, and E. D. Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics* 7: 56, 2006.

[21] C. Li, Q.W. Ge, M. Nakata, H. Matsuno, S. Miyano. Modelling and simulation of signal transductions in an apoptosis pathway by using timed Petri nets. *Journal of Biosciences*, 32: 113–27, 2007.

[22] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* John Wiley, 1994.

[23] L. Mendoza. A network model for the control of the differentiation process in Th cells. *Biosystems* 84: 101–114, 2006.

[24] I. Mura, A. Csikasz-Nagy. Stochastic Petri Net extension of a yeast cell cycle model. *J Theor Biol* 254(4): 850–860, 2008.

[25] M. Nagasaki, A. Doi, H. Matsuno, S. Miyano. A versatile Petri net based architecture for modeling and simulation of complex biological processes. *Genome Informatics*, 15: 180–97, 2004.

[26] A. Naldi, D. Thieffry, C. Chaouiya. Decision diagrams for the representation of logical models of regulatory networks. *Lecture Notes in Bioinformatics*, 4695: 233–47, 2007.

[27] A. Naldi, D. Berenguier, A.Fauré, F. Lopez, D. Thieffry, C. Chaouiya. Logical modelling of regulatory networks with GINsim 2.3. *Biosystems*, 97(2):134-139, 2009.

[28] A. Naldi, E. Remy, D. Thieffry, C. Chaouiya. A reduction method for logical regulatory graphs preserving essential dynamical properties. *Lecture Notes in Bioinformatics*, 5688: 266–80, 2009.

[29] E. Remy, P. Ruet, D. Thieffry. Positive or negative regulatory circuit inference from multilevel dynamics. *Lecture Notes in Control and Information Sciences*, 341: 263–70, 2006.

[30] E. Remy, P. Ruet, L. Mendoza, D. Thieffry, C. Chaouiya. From Logical Regulatory Graphs to Standard Petri Nets: Dynamical Roles and Functionality of Feedback Circuits. *Lecture Notes in Computer Science*, 4230: 55–72, 2006.

[31] A. Richard, J.-P. Comet. Necessary conditions for multistationarity in discrete dynamical systems. *Discrete Applied Mathematics*, 155(18):2403–13, 2007.

[32] A. Sackmann, M. Heiner, I. Koch. Application of Petri net based analysis techniques to signal transduction pathways. *BMC Bioinformatics*, 7: 482, 2006.

[33] L. Sánchez, C. Chaouiya, D. Thieffry. Segmenting the fly embryo: a logical analysis of the segment polarity cross-regulatory module. *Int Journal of Developmental Biology* 52(8):1059-75, 2008.

[34] T. Schlitt, A. Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8: S9, 2007.

[35] H. Siebert, A. Bockmayr. Incorporating time delays into the logical analysis of gene regulatory networks. *Lecture Notes in Computer Science*, 4210: 169–83, 2006.

[36] H. Siebert, A. Bockmayr. Context sensitivity in logical modeling with time delays. *Lecture Notes in Computer Science*, 4695: 64–79, 2007.

[37] E. Simão, E. Remy, D. Thieffry, and C. Chaouiya. Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in *E. coli*. *Bioinformatics*, 21: ii190–6, 2005.

[38] C. Soulé. Mathematical approaches to gene regulation and differentiation. *Comptes Rendus de l'Académie des Sciences de Paris (Biologie)*, 329: 13–20, 2006.

[39] R. Srivastava, M. S. Peterson, W. E. Bentley. Stochastic kinetic analysis of the escherichia coli stress circuit using σ^{32}-targeted antisense. *Biotechnology and Bioengineering*, 75: 120–9, 2001.

[40] L.J. Steggles, R. Banks, O. Shaw, A. Wipat. Qualitatively modelling and analysing genetic regulatory networks: a Petri net approach. *Bioinformatics*, 23: 336–43, 2007.

[41] D. Thieffry. Dynamical roles of biological regulatory circuits. *Briefings in Bioinformatics*, 8: 220–5, 2007.

[42] R. Thomas, R. D'Ari. *Biological Feedback.* CRC Press, 1990.

[43] R. Thomas. Regulatory networks seen as asynchronous automata: a logical description. *Journal of Theoretical Biology*, 153, 1–23, 1991.

[44] R. Thomas, D. Thieffry, M. Kaufman. Dynamical behaviour of biological regulatory networks – I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bulletin of Mathematical Biology*, 57: 247–76, 1995.

A.4 Modélisation modulaire du cycle cellulaire

Cette publication introduit un modèle logique du contrôle du cycle cellulaire chez la levure. Ce modèle a été obtenu en composant les traductions dans le formalisme logique de trois modèles différentiels.

L’article inclus ici est sous presse (publication électronique anticipée).

Modular logical modelling of the budding yeast cell cycle†‡

Adrien Fauré,*[a] Aurélien Naldi,[a] Fabrice Lopez,[a] Claudine Chaouiya,[ab] Andrea Ciliberto[c] and Denis Thieffry*[ad]

Received 21st May 2009, Accepted 5th June 2009
First published as an Advance Article on the web
DOI: 10.1039/b910101m

Systems biologists are facing the difficult challenge of modelling and analysing regulatory networks encompassing numerous and diverse components and interactions. Furthermore, available data sets are often qualitative, which complicates the definition of truly quantitative models. In order to build comprehensive and predictive models, there is clearly a need for incremental strategies, enabling the progression from relatively small to large scale models. Leaning on former models, we have defined a logical model for three regulatory modules involved in the control of the mitotic cell cycle in budding yeast, namely the core cell cycle module, the morphogenetic checkpoint, and a module controlling the exit from mitosis. Consistency with available data has been assessed through a systematic analysis of model behaviours for various genetic backgrounds and other perturbations. Next, we take advantage of compositional facilities of the logical formalism to combine these three models in order to generate a single comprehensive model involving over thirty regulatory components. The resulting logical model preserves all relevant characteristics of the original modules, while enabling the simulation of more sophisticated experiments.

1. Introduction

The development of modelling frameworks enabling the definition and analysis of large and complex regulatory networks is a challenging problem. One promising strategy to tackle this issue relies on the modularity of regulatory networks, which allows the identification of functional modules to be separately modelled and analysed. Such models have then to be composed to obtain comprehensive networks. Within the logical framework, such a modular approach has been used to build multicellular models from non-overlapping, cellular network modules.[1,2] Here we explore the coupling of partly overlapping modules involved in cell cycle control.

The cell cycle is the process by which a cell replicates its genetic material and divides itself into two daughter cells. This regulatory system relies on an intricate molecular network that is highly conserved among eukaryotes. The core cycling engine is completed by a set of checkpoints placing cell division under external control and ensuring that every single step has been completed before the next one begins, such that, for example, sister chromatids are not separated until chromosomes are correctly aligned on the metaphase plate.

Most of these mechanisms have been well characterised in the budding yeast. In mammals (and multicellular organisms in general), an additional layer of controls coordinates the cell growth and division with the needs of the whole organism, and disruptions of these checkpoints constitute important steps towards cancer. The considerable amount of data available on molecular details and mutant phenotypes makes budding yeast an appealing system to develop a model of the eukaryotic cell cycle control.

Indeed, the cell cycle core engine has been modelled in great detail, most notably by the groups of Béla Novák and John Tyson, using a differential formalism. Several models focusing on different regulatory modules have been developed.[3–12] In parallel, the lack of supporting quantitative data that hampers the development of differential models, as well as the numerical instabilities inherent to large non-linear systems, motivated the development of simplified Boolean models of the core cycling engine.[13–15] However, Boolean modelling appears too crude to properly account for several subtle aspects of cell cycle control, such as the effect of cellular mass. The use of a multi-level logical formalism offers a good compromise between Boolean drastic simplification and daunting quantitative models. Moreover, the use of a logical formalism facilitates the development of more integrated models, through the articulation of control modules with the core cell cycle engine.

Hereafter, we present three logical (multi-level) models for the core cycling engine (from now on, the *core model*), the morphogenetic checkpoint (*MCP module*), and a module rewiring the FEAR (Cdc Fourteen Early Anaphase Release) and MEN (Mitotic Exit Network) networks (*exit module*). These three models heavily rely on previous modelling studies by the groups of Novák and Tyson,[16–18] which constitute a challenging benchmark for our modular approach. Once validated through a systematic analysis of their behaviours for wild-type and numerous mutant situations, these modules

[a] *Université de la Méditerranée & INSERM U928 - TAGC, Marseille, France. E-mail: faure@tagc.univ-mrs.fr, thieffry@tagc.univ-mrs.fr*
[b] *Instituto Gulbenkian de Ciência, Oeiras, Portugal*
[c] *IFOM-FIRC, Milano, Italy*
[d] *CONTRAINTES Project, INRIA Paris-Rocquencourt, Le Chesnay, France*

† This article is part of a *Molecular BioSystems* themed issue on Computational and Systems Biology.

‡ Electronic supplementary information (ESI) available: Detailed descriptions of the models. See DOI: 10.1039/b910101m

are then integrated into a comprehensive logical model (from now on, the *coupled model*) through a simple formal procedure. The resulting model preserves the main dynamical features of each of the three modules. Furthermore, this model allows the recovery of additional reported properties involving components belonging to different modules.

2. Results

2.1 Logical modelling of the core cycling engine

In a first step, we have defined a logical version of the model for the core cell cycle engine of the budding yeast published by the groups of Novák and Tyson.[16] This first step was also a benchmark for our modelling approach to the cell cycle, as building a qualitatively valid logical adaptation of such a complex differential model is not a trivial task (see materials and methods). Presented in Fig. 1, the resulting logical model is thus based on the antagonism between mitotic cyclins (Clb2 and Clb5) and G1 stabilizers (Cdh1 and the CKI). When mass increases above a certain threshold, G1 cyclins accumulate high enough to inhibit the G1 stabilizers, and thus indirectly activate the mitotic cyclins, thereby promoting entry into the S phase and mitosis. Clb2 triggers exit from mitosis and its own destruction, by activating Cdc20, and indirectly activates the G1 stabilizers through the release of the phosphatase Cdc14. In addition, the model integrates a checkpoint mechanism that monitors DNA replication and spindle formation.

Following the principles described in section 4 and using well-defined logical rules for the updating of each considered regulatory node (*cf*. examples in Tables 3 and 4, and the ESI‡ for a complete listing of these rules), the simulation of this model qualitatively recapitulates the wild-type sequence of events the cell has to go through to be considered viable: firing of the origins of replication (ORI goes up), spindle alignment (SPN goes up), separase activation (Esp1 goes up), division (CYTOKINESIS goes up to level 2) after the formation of a bud (BUD must have reached level 1, although it may go down afterwards before cell division), and origin relicensing (ORI goes down). The different phases of the cycle are defined in terms of the levels of activity of key components, as follows: low Clb5 and Clb2 activity (either = 0 or sequestered by Sic1/Cdc6) define a G0/G1 phase; high Clb5 activity (*i.e*. not sequestered by the CKI) and low Clb2 activity define S/G2 phase; high Clb2 activity defines M phase, which can be further divided into prophase/metaphase (low Esp1 activity) and anaphase/telophase (high Esp1).

Many different mutants (gene knock-outs, partial loss-of-functions, temperature-sensitive mutations, ectopic and over-expressions) or perturbations (*e.g*. culture in the presence of drugs such as nocodazole) have been analysed experimentally and reported in the literature. Over 130 of these mutants have been considered by Chen *et al*. In the present article, we set out to reproduce their results with our logical version of the model. Mutant dynamical behaviours have been systematically computed on the basis of the wild-type model, using a functionality of our software *GINsim* enabling the definition and recording of multiple mutants in terms of logical rules. Mutant simulations are evaluated qualitatively regarding viability (according to the above sequence of events), or arrest in a particular phase of the cell cycle. As shown in the ESI‡, dynamical analysis of most of the individual or multiple perturbations leads to results qualitatively consistent with reported phenotypes, as well as with the results published in ref. 16.

Only four cases led to major discrepancies: the mutant cln1Δcln2Δcln3Δcdh1Δ, which should be arrested in telophase, appears to be viable in our simulations; so do the pds1Δcdc20Δ and CLB5-dbΔpds1Δcdc20Δ mutants. Finally, in our simulations, the mutant CLB2-dbΔclb5Δ on galactose appears to be arrested in telophase, as when grown in glucose, whereas it should be viable in the slow-growth rate medium. The problems encountered here are linked to the importance of synthesis rates in the phenotypes of these mutants. This is clearly exemplified by the case of the CLB2-dbΔclb5Δ mutant, whose viability depends on the growth medium. Such characteristics are hard to account for in the logical formalism. Other minor discrepancies relate to timing issues for some mutants, especially over the issue of BUD formation, which in the Chen *et al*. model heavily relies on bud synthesis rate; more elaborate priority rules might help solve these problems in a logical context (see the ESI‡ for a detailed presentation of the mutants).

2.2 Logical modelling of the morphogenesis checkpoint module

Leaning on the differential model published by Ciliberto *et al*.,[17] we have delineated a logical model for the regulatory network monitoring the formation of the bud (BUD), called the morphogenetic checkpoint (MCP). Presented in Fig. 2, this model accounts for the fact that the cell cycle is temporarily blocked in the G2 phase in the case of budding defect. This G2 blocking can be bypassed in the presence of a high Clb2 activity level, which, according to Ciliberto *et al*., correlates with the growth of the cell. Consequently, nuclear division occurs without cell division, thereby giving rise to dinucleate cells. To properly model this phenomenon, we have considered a second threshold for the MASS component, which denotes a mass large enough to bypass G2 arrest (ref. 17 and references therein).

Our logical model of the MCP recapitulates the wild-type and knockout phenotypes considered by Ciliberto *et al*.,[17] as well as three additional knockout mutants described by Harrison *et al*.[19] As this model focuses on Clb2 activation depending on the mass of the cell, its dynamics are analysed in terms of stable states for each of the possible values of MASS, when BUD formation is allowed or impaired (six different situations in total, all described in the corresponding section of the ESI‡). In a wild-type context, Clb2 can be fully activated ($Clb2 = 2$) for $MASS = 1$, whereas when budding is impaired ($BUD = 0$), $MASS$ must cross its second threshold and reach level 2 to force the activation of Clb2. In this latest situation, we do obtain a second stable state with $Clb2 = 2$ and $MASS = 1$, which can be considered an artefact of the isolation of the MCP module. In fact, all paths leading to this stable state involve an activation of Clb2 before the activations of MBF and Swe1, which does not happen in a normal cycle. As we shall see, this artefactual stable state can be eliminated through

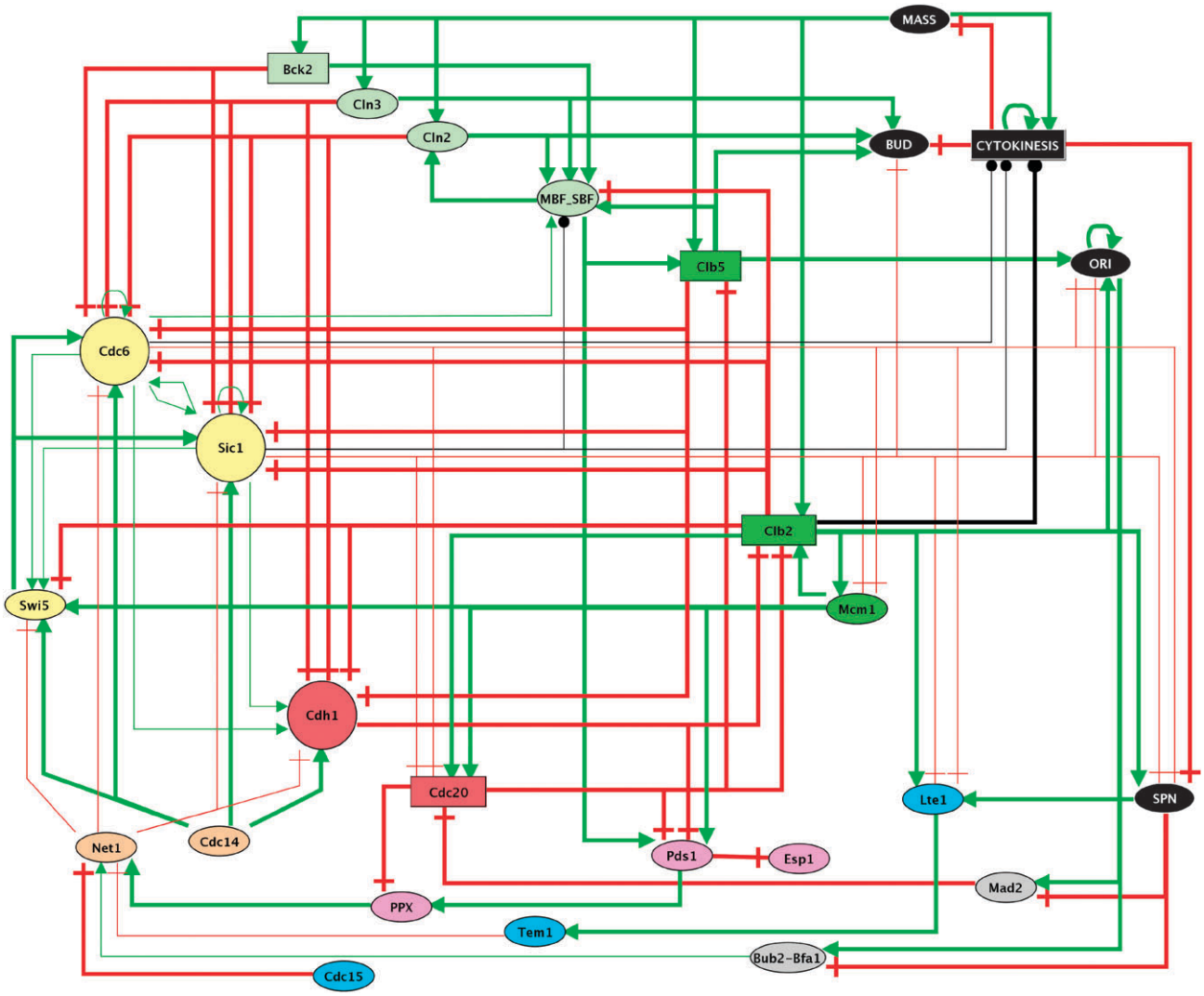


Fig. 1 Logical regulatory graph for the core engine controlling cell cycle in the budding yeast derived from ref. 16. In this model, Cln2 represents both Cln1 and Cln2, Clb5 both Clb5 and Clb6, and Clb2 both Clb1 and Clb2; Cdc28, the kinase partner of the cyclins, is implicit. A newborn daughter cell is stuck in a G1 state characterised by an absence of cyclin activity, which is kept low by the CDK inhibitors (CKI) Sic1 and Cdc6, and by the APC activator Cdh1. The cell must grow up to a certain size (MASS) in order to produce enough Cln3 and Bck2 and turn the MBF and SBF transcription factors on. Their cyclin targets Cln2 and Clb5 then begins to accumulate. Finally, Cln2 activates the synthesis of the bud (BUD), but Clb5 activity is impaired by the CKI Sic1. Once this inhibition is relieved by the action of Cln3, Bck2 and Cln2, Clb5 will help Cln2 and Cln3 to inhibit Cdh1. In the meantime, Clb5 initiates the replication (ORI). Once Cdh1 is off, Clb2 is activated in a positive feedback loop with its transcription factor Mcm1. Clb2 shuts the MBF and SBF off, reinforces the inhibition of the CKI and Cdh1, and plays a role in the assembly and maintenance of the mitotic spindle. Once the spindle is complete, the checkpoint is relieved, and activated Cdc20 initiates mitosis. Cdc20 degrades Pds1, while free Esp1 separates sister chromatids; Cdc20 also degrades Clb5 and initiates the degradation of Clb2, an important step for Cdh1 and the CKI reactivation. Cdc20 would also promote the degradation of a (hypothetical) phosphatase PPX that plays a role in Net1 activation. As the spindle checkpoint is relieved, Net1 is also inhibited by Cdc15. In the absence of its competitive inhibitor, Cdc14 can re-activate Cdh1 and CKI, which resets the cell to its initial G1 state by inhibiting the cyclins. Graphical conventions: Boolean and multi-level nodes are denoted by circles and rectangles, respectively; activatory *versus* inhibitory interactions are denoted by green normal *versus* red T-headed arrows; black, circle-headed arrows denote context-depending signs; thick and thin arcs as in Fig. 5.

a proper coupling of the MCP module with the core engine model.

We have then coupled the MCP module to the core model, following the procedure detailed in the materials and methods section. The coupled model preserves all the properties mentioned above for the wild-type and loss-of-function backgrounds of the core and MCP modules. As hinted above, the artefactual activation of Clb2 before BUD formation for $MASS = 1$ observed in the isolated MCP module disappears under the updating assumption considered for the coupled model. Furthermore, the behaviour of the MCP module now constrains that of the core model, in particular regarding the timing of bud formation. Such behaviour is difficult to assess with a logical model; nonetheless, in the coupled model, as long as MASS is kept in the lowest priority class, the activation of Clb2, and thus the G2/M transition, is conditioned

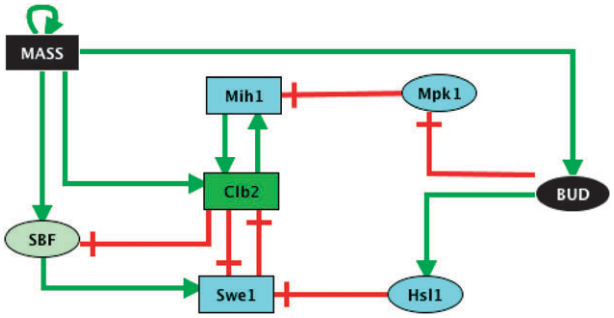


Fig. 2 Logical regulatory graph for the morphogenesis checkpoint. When budding is impaired, the checkpoint destabilises Hsl1, which is partly responsible for the Swe1 kinase degradation. In the absence of its inhibitors, Swe1 inhibits Cdc28/Clb2, while an alternative pathway inhibits Mih1, whose role is to remove the inhibitory phosphate from Clb2. In these conditions, Clb2 cannot be activated and the cell is stuck in a G2 state. It takes a further increase in mass to force Clb2 activation and allow the cell to enter mitosis. Graphical conventions as in Fig. 1.

to bud formation, which was not the case in the core model, as hinted by Chen *et al.*[16] (see also Table 2).

2.3 Logical modelling of the mitotic exit module

In their differential model of the core engine, Chen *et al.* introduced the hypothetical PPX phosphatase to account for the sequential activation of Cdc20 and Cdh1.[16] Recent evidence about the FEAR pathway was mentioned in the article, but had come out too late to be integrated in their model. Recently, Queralt *et al.* proposed a model accounting for the role of separase in the initiation of mitotic exit.[18] We rely on their model to propose a logical model of mitotic exit and replace the PPX mechanism in our logical model of the core engine with this more detailed version.

Briefly, when securin is degraded by Cdc20 at anaphase onset, free separase cleaves the cohesin that maintains sister chromatids together and downregulates $PP2A^{Cdc55}$, a phosphatase that opposes Net1 phosphorylation by Clb2. Downregulation of $PP2A^{Cdc55}$ also participates in MEN activation by facilitating phosphorylation of Bfa1 by the Polo-like Cdc5 kinase. This mechanism accounts for the two-step release of the Cdc14 phosphatase from its competitive inhibitor Net1.

Our model (Fig. 3) qualitatively recapitulates the behaviours of the wild-type situation and of nine reported mutants (*cf.* ESI‡). In each case, we have tested mitotic exit, *i.e.* we have checked whether a stable state corresponding to G1 (with active Cdh1 and low Clb2) can be reached from the state corresponding to metaphase (high Clb2, no Cdh1).

The procedure to couple the exit module to the core engine is described in the materials and methods section. The coupled model (Fig. 4) preserves the behaviour observed for the wild-type core module, as well as the properties of numerous mutations, such as those implied in the G1/S transition and in the MCP (Table 1 displays a simulation of the wild-type cell cycle). The coupled model now further fits the data that were used to build the exit module. In particular, inactivation of

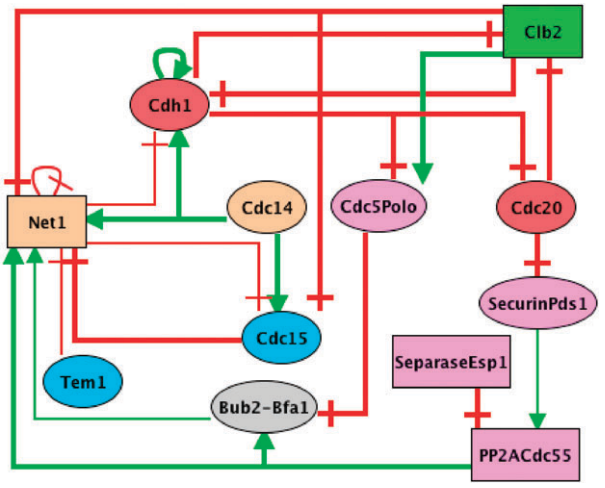


Fig. 3 Logical regulatory graph for the module controlling the exit from mitosis. At anaphase onset, Cdc20 is activated. It degrades the securin (Pds1), setting the separase (Esp1) free, and initiates the degradation of Clb2. Free separase then inactivates the PP2A phosphatase, initiating the phosphorylation of Net1, a competitive inhibitor of the Cdc14 phosphatase. This partial release of Cdc14 allows the activation of Cdc15, which together with Tem1 fulfills the inhibition of Net1, achieving the complete release of Cdc14. Free Cdc14 then activates Cdh1, which completes the degradation of Clb2, thus achieving mitotic exit. Graphical conventions as in Fig. 1.

separase (the *esp1ts* mutant mentioned by Chen *et al.*[16]) now provokes an arrest in telophase, which is consistent with the results reported by Queralt *et al.*[18] Selected mutant simulations are listed and commented upon in Table 2, while a summary of the results of all mutant simulations can be found in the ESI.‡

3. Discussion

Although the appropriateness of a modular strategy is widely advocated, modelling frameworks supporting systematic composition are still lacking. A first step in this direction can be found in ref. 20, where the authors present a tool to ease model composition, with an application to the generic model published by Tyson and Novák.[21] Leaning on previous dynamical modelling efforts by the groups of Novák and Tyson, the present article focuses on the development of a modular modelling approach and its application to the integration of three multi-level logical models for regulatory modules involved in the control of the yeast cell cycle.

The first of these models is the most comprehensive and already implements regulatory mechanisms involved in the spindle checkpoint, the G1/S transition, as well as a preliminary version of the exit control module. The morphogenesis checkpoint module constrains activation of Clb2, and thus entry into mitosis, with respect to bud formation. Lastly, the exit module integrates recent data on the role of separase and securin in the release of Cdc14, thereby driving mitotic exit. Although these three differential models have been developed by the same groups, they were not yet explicitly integrated into a unique model. Indeed, such an integration appears really

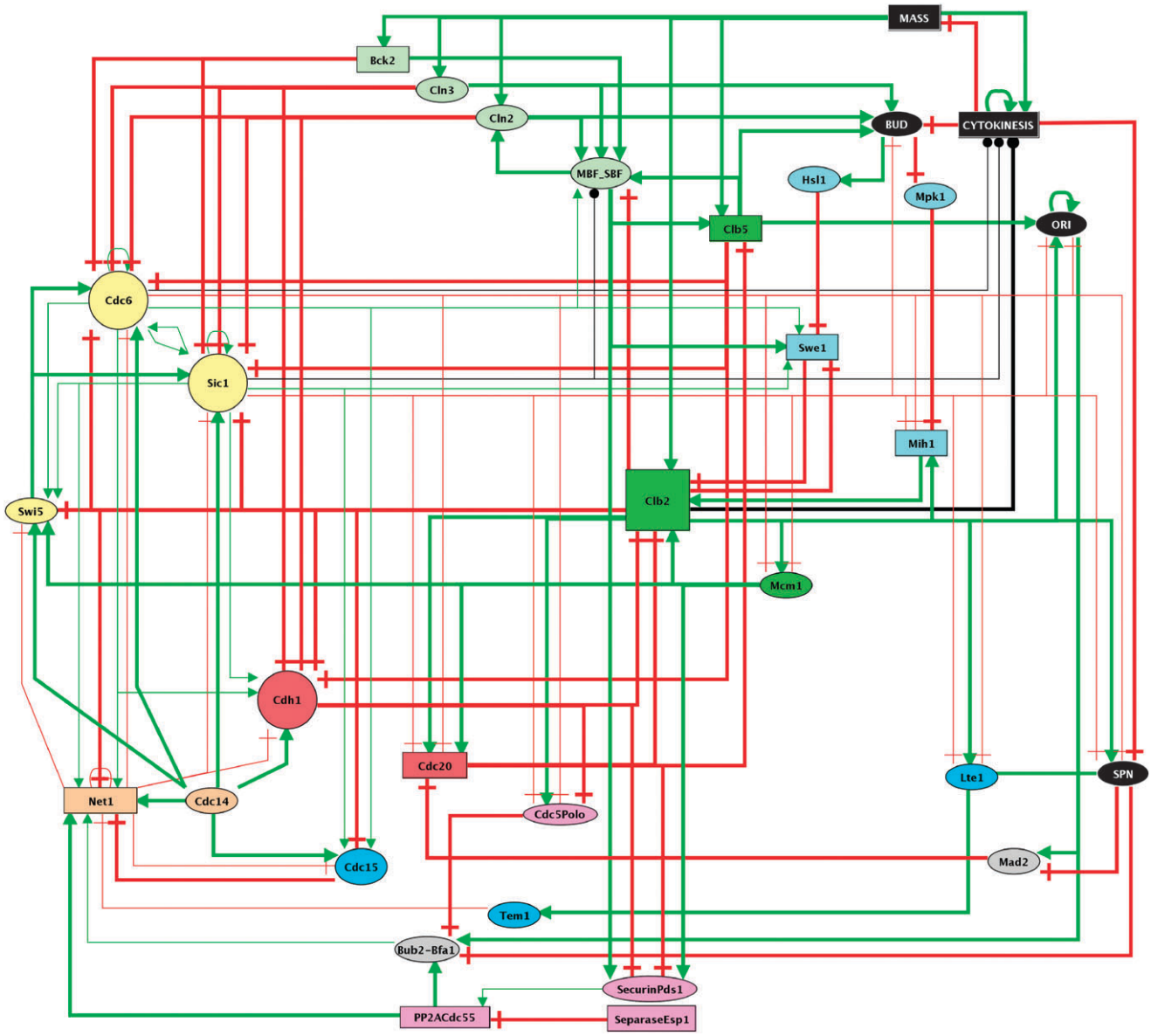


Fig. 4 Logical regulatory graph for the coupled model, encompassing the core engine (Fig. 1), the MCP module (Fig. 2) and the exit module (Fig. 3). Compare with Fig. 1. Graphical conventions as in Fig. 1.

challenging in the differential framework, as numerical instabilities arise when the number of variables and the number of nonlinearities increase.

When applying the composition method described in subsection 4.3, the resulting comprehensive model preserves the essential properties of the three modules, while enabling the simulation of perturbations involving components that belong to several modules. Provided that these modules have been modelled with the prospect of their integration in mind, a systematic integration method can be defined. Our logical framework is thus suited for incremental modelling strategies and eases the concerted refinement of the different modules composed to build a comprehensive model. Module integration is further facilitated through the development of novel *GINsim* functionalities enabling the copying and pasting of (sub)networks (along with thresholds and logical rules) from one file to another, or yet the automatic computation of stable states for all pre-defined perturbations.

For each module, logical rules have been derived on a case-by-case basis, using differential equations as a template to determine positive and negative regulatory influences and infer distinct qualitative levels. Based on this experience, it would be particularly interesting to delineate generic rules to ease the translation of differential models into logical ones. A first step in this direction can be found in ref. 22, where the authors introduce a systematic Boolean transposition of an ODE model, which recapitulates its most salient qualitative dynamical properties.

The results presented here confirm that the qualitative knowledge of the regulatory network of the system is sufficient to explain most of its dynamical properties, thus pointing once more to cell cycle network robustness.[13,16] However, detailed

Table 1 Simulation of the coupled model in the wild-type condition. The model is simulated using a set of synchronous priority classes starting from the initial state corresponding to the first row of the table. Successive rows give the successive states obtained in the simulation. Colour code: white = 0, light gray = 1, gray = 2, black = 3 (*cf.* the ESI‡ and model file for further details)

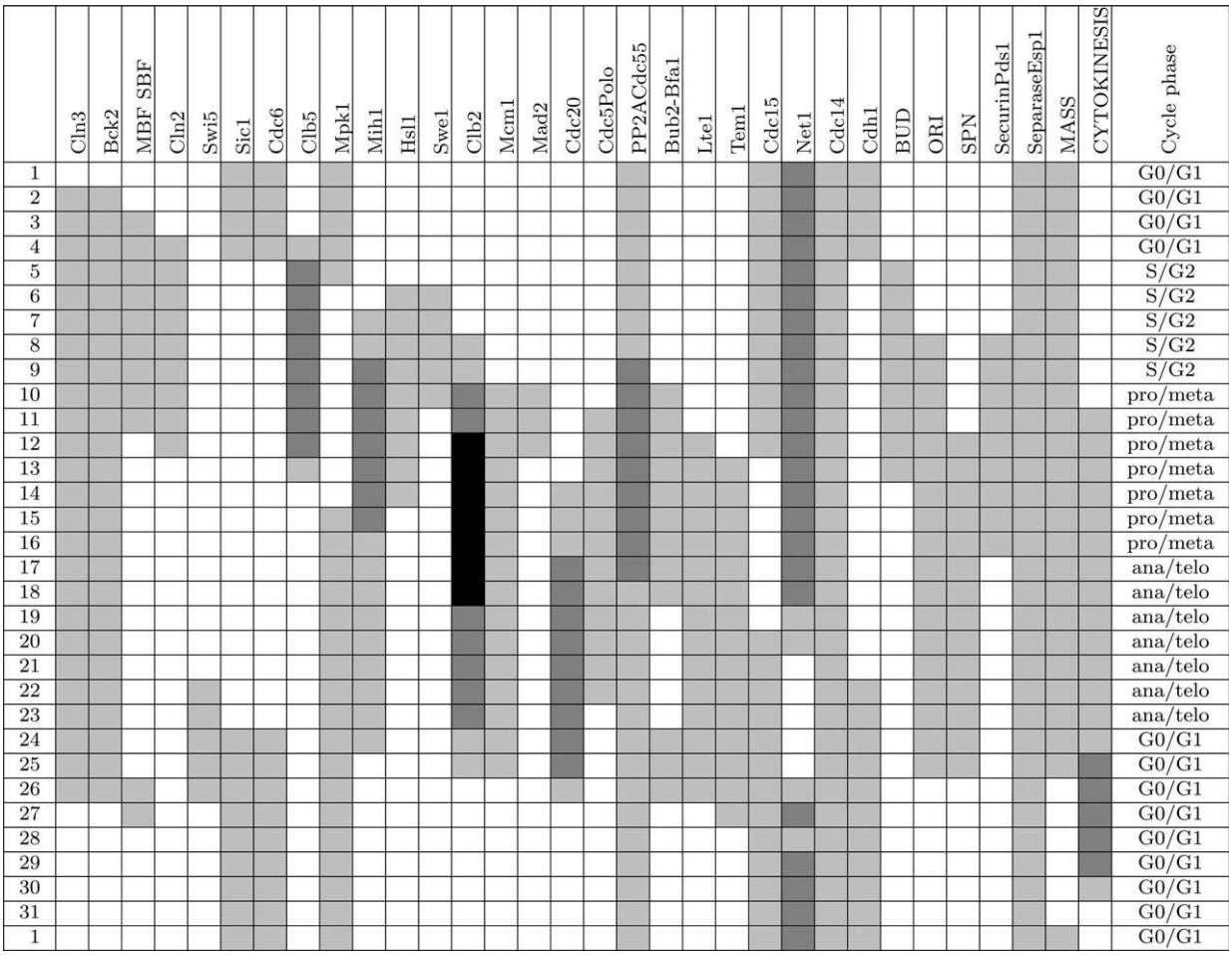


Table 2 Examples of mutant simulations where the coupled model shows improved consistency with biological data

Mutant	Comments
GAL-CLB2 GAL-CLB2 sic1Δ GAL-CLN2 cln1Δ cln2Δ cdh1Δ	In the core model, there is no constraint on BUD formation, and these mutants can divide despite the absence of bud. In the coupled model, the morphogenesis checkpoint delays entry into mitosis long enough for the cell to grow a bud.
Separase inactivation	In the core model, the *esp1ts* mutant displayed sustained oscillations, consistent with the simulations reported in ref. 16. In the coupled model, this mutant arrests in telophase, consistent with the results reported in ref. 18.
CLB1 clb2*Delta*	In the core model, this mutant could exit mitosis with sister chromatids still attached, consistent with the simulations presented in ref. 16. In the coupled model, due to the constraints on Cdc55 introduced by the exit module, activation of separase is required for mitotic exit.

analysis of the mutants reveals that, in some cases, incorporation of refined information about expression or activity levels, or yet synthesis or growth rates may be necessary to fully reproduce some properties of the system. Mainly relying on qualitative information, the logical formalism used here offers a flexible means to integrate this information into discrete dynamical models, including the possibility to use multi-level components whenever this is functionally justified. Whether the definition of subtler updating rules (priorities, delays) could enable the modelling of finer kinetic behaviours remain to be assessed.

Finally, although logical models can be used independently to gain understanding of the articulation of the different modules in comprehensive models and predict the concerted behaviour of components belonging to different modules, they can serve in turn as scaffolds to develop more quantitative (differential or stochastic) models. In this respect, *GINsim* enables the conversion of logical models into Petri nets, thereby providing an access to complementary analysis

methods (*e.g.* model checking tools), or yet to hybrid or stochastic extensions (*cf.* ref. 23 for an application of Stochastic Petri nets to a simplified model of the budding yeast cell cycle).

4. Materials and methods

4.1 The logical modelling framework

The main features of the logical approach used are defined in ref. 24, 25 and 26. Briefly, a *logical model* is defined by a *regulatory graph*, where the nodes and arcs represent the regulatory components and interactions, respectively. To each node is associated an integer interval (from 0 to max) defining discrete levels of activity, and each arc is accordingly associated to an interval defining the set of values for which its source may influence its target. The dynamical behaviour of each node is then defined by *logical functions* (also represented in terms of *logical parameters*), which associate a target value for this node with each combination of regulators.

The dynamics of the system is represented in terms of a *state transition graph*, where the nodes denote states of the system (*i.e.*, vectors giving the levels of activity of all the variables), and the arcs denote state transitions (*i.e.*, changes in the value of one or several variables, depending on the values of the relevant logical functions). When several transitions are enabled, the resulting dynamical pathway depends on the updating assumption selected: synchronous, asynchronous, or user-defined through the setting of priority classes.[14] Time is thus implicitly defined by the sequence of transitions.

For several years, our group has been developing a software suite, *GINsim*, to facilitate the definition and the dynamical analysis of such logical models.[27] In particular, this software enables the definition and storage of different initial states and alternative genetic backgrounds (*i.e.* mutants). For more details, see the *GINsim* web site at the url http://gin.univ-mrs.fr/GINsim.

4.2 Logical modelling of cell cycle control mechanisms

Direct activations or inhibitions (post/transcriptional regulations, including protein de/phosphorylations or degradations) are simply represented by regulatory arcs from the regulator to the target gene or protein, together with the definition of consistent logical rules for the regulated target.

Although the logical formalism is particularly well suited to represent regulatory interactions (activations, inhibitions), it is less adapted to the representation of mass flow, and in particular of multiproteic complex formation. Consequently, we usually represent the complexes implicitly: a complex is considered present if all its components are present; all components, whether regulatory or enzymatic subunits, regulate the targets of the enzymatic member, with a logical AND rule—or AND NOT in the case of sequestration (*cf.* Fig. 5). In the models presented here, this is the case for the sequestration of cdc14 by Net1, the inhibition of Net1 by Cdc15 and Tem1, the sequestration of Tem1 by Bub2-Bfa1, the sequestration of separase by securin in the exit module, and the sequestration of Clb5 and Clb2 by Cdc6 and Sic1. In the latter case, for example, we have represented inhibition of the Clbs by introducing arcs from the CKI towards the Clbs' targets, so that inhibition of the Clbs is represented by an inhibition of the components they activate and by an activation of the components they inhibit. In addition, for the sake of simplicity, complex subunits that are not dynamically regulated are usually not represented (*e.g.*, the different CDKs), unless we want to test mutations (as in the case of Cdc14).

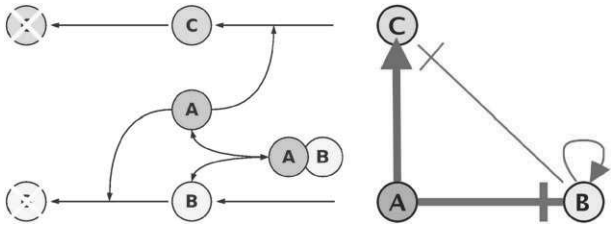


Fig. 5 Complex formation. Left: wiring diagram representing the sequestration of a molecule A by a molecule B. Free A activates the synthesis of C and the degradation of B; in complex with B, A is inactive. Right: the corresponding regulatory graph. A directly activates C and inhibits B (thick arrows); B sequesters A, which is represented by thin arrows of the opposite sign directed to A's targets.

Regarding the *number of thresholds* considered, we started with Boolean models,[14] which were progressively refined through the consideration of a minimal number of additional levels of activity for some regulatory components (rectangular nodes in Fig. 1–4) in order to solve discrepancies between some mutant behaviours and published data.

Beyond the representation of specific molecular actors (regulatory proteins and complexes), the models considered encompass a series of phenomenological variables: MASS, CYTOKINESIS, BUD, ORI, SPN, denoting cellular growth, cell division, the formation of the bud, the firing of the origins of replication and the formation of the spindle, respectively. Here, different activity levels are associated with qualitatively different effects. Although the evolution of these components is still incremental (*i.e.* one can not go directly from level 0 to level 2), they do not directly represent quantitative information.

In ref. 16, the mass of the cell is hypothesised to drive the cycle by amplifying synthesis, thus increasing the concentration of the cyclins that belong to the nuclear compartment, whose volume is constant. In our models, MASS is represented by a multi-level node. A first threshold represents the minimal level above which a normal cell can go through the G0/G1 transition (which involves the inhibition of the CKI, depending on the activation of Cln3 and Bck2). A second threshold is considered in the context of the implementation of the morphogenetic checkpoint that controls the G2/M transition, to represent the requirement of higher cellular mass to bypass the checkpoint when budding is impaired.

Following Chen *et al.*, we consider that mitosis is triggered when the Clb2 level decreases below a specific threshold.[16] In our model, a high level (levels 2 or 3) of Clb2 thus enables the node CYTOKINESIS to reach the value 1. The CYTOKINESIS node will reach its highest level 2 only after Clb2 inactivation. The MASS, BUD and SPN nodes are then reset to zero, before the CYTOKINESIS node also reaches its bottom level.

A similar mechanism is implemented for the control of the relicensing of replication origins (ORI).

As our focus in this paper lies in the delineation and validation of an efficient module composition method, we have kept the treatment of concurrent molecular processes as simple as possible, while preserving a qualitative matching between model simulation and the order of events documented in the literature or explicated in previous models. Consequently, all transitions have been treated synchronously, excepting the special cases of CYTOKINESIS and MASS updatings. Indeed, a correct sequential in/activation of these two phenomenological nodes requires the definition of specific priority classes. The resulting model thus implements four priority classes: the first priority class (highest priority) contains the CYTOKINESIS up-regulation as the unique member; the third class contains CYTOKINESIS down-regulation; the fourth class (lowest priority) contains MASS up-regulation; finally, all the other transitions are grouped into the second class and are treated synchronously.

For each model, mutants are defined as sets of alternative logical rules for particular components of the system. These alternative rules can then be used in replacement of the default (*i.e*. *wild-type*) rules of the model. This allows us to analyse the behaviour of the system for various alternative genetic backgrounds, encompassing gene knock-outs, over-expressions, ectopic expressions, or yet non-degradable forms of one or several components. The logical rules for wild-type and all mutants for each model are described in the ESI.‡

4.3 Coupling procedure

4.3.1 Principles. Leaning on the properties of the logical formalism, we have delineated a systematic method to merge logical modules (or submodels) into comprehensive models.

The first step consists in the definition of the comprehensive regulatory graph. In the case of the coupling of a completely novel module (*e.g*. the MCP module), this regulatory graph simply encompasses all the components and interactions from the original modules (MCP module and core engine). In the case of the replacement of part of the model with a more detailed, or updated version (*e.g*. the exit module), the corresponding nodes and interactions of the original model (*e.g*. the core engine) are replaced by the more detailed graph.

Next, the logical rules associated with the nodes of the new graph are defined as follows. The rules of the nodes that are present in only one model are left unchanged, as their activities are already consistently defined. In contrast, the rules associated to nodes that are present in two modules must be modified to integrate the effects of all regulators. In the models presented, several components of the core, MCP and exit modules amount to simplifications of their counterparts in another module (for example, MBF and BUD in the MCP module, Cdc15 in the core model, or Clb2, Cdc20 and Cdh1 in the exit module). In these cases, it is the most detailed rule that is conserved in the coupled model. Moreover, when a component receives input from regulators that belong to different modules, as in the case of Clb2 present in both the MCP module and the core engine, or in the case of Net1 and Bub2-Bfa1 present in both the exit module and the core engine, novel logical formulae are systematically defined through proper combinations of the original ones. We proceed in two steps: (i) the thresholds are first interpolated; (ii) the logical formulae associated with each logical value are then combined (logical AND). Finally, accordingly with our modelling of complex formation (*cf*. materials and methods), we have to take into account both partners of a complex in the regulation of their targets, as in the case with the sequestration of Clb2 by Sic1 and Cdc6.

4.3.2 Coupling of the MCP module to the core engine. In order to define a comprehensive model encompassing the two logical modules already defined, we proceed in the following way. First, all the components considered in only one module are simply conserved as such, *i.e*. with identical logical rules, level numbers and interactions, with the exception of Clb2 targets. In this case, we have to take into account the sequestration of Clb2 by the CKIs, which were not present in our MCP module. This means that, in the new logical formulae for Swe1 and Mih1, *Clb2* (presence of Clb2) has to be replaced by $Clb2 \wedge \overline{(Sic1 \vee Cdc6)}$ (*i.e*., "*Clb2* AND NOT (*Sic*1 OR *Cdc*6)", meaning that Clb2 has an enzymatic action on its targets only when not sequestered by Sic1 or Cdc6). Four components are involved in both modules: MASS, MBF (MBF_SBF in the core model), BUD and Clb2. BUD and MBF are Boolean components in both models, and their regulation in the MCP model is a simplification of their regulation in the core model; accordingly, we retain the more detailed definitions of the core engine in the coupled model.

The case of MASS is slightly more complex, as it is Boolean in the core engine model, while it has an additional level in the MCP model. Consequently, MASS is also endowed with two significant levels of activity in the coupled model. This means that MASS can still regulate Clb2 differentially at levels 1 and 2, as it does in the MCP module, and will have the same effect on its other targets all along the[11,26] interval. In the MCP model, MASS is only self-regulated. As in the core model, MASS in the coupled model is regulated by the CYTOKINESIS component, and takes its maximum value (1 in the original core model, 2 in the coupled model) when CYTOKINESIS is below its threshold. In addition, as we put MASS growth in the lowest priority class, it will increase only if growth is necessary for cell cycle progression.

The case of Clb2 is more complex. In the core model, it receives input from MASS, Cdh1, Cdc20, and Mcm1, and can take three non-zero values. In the MCP model, Clb2 receives input from MASS, Swe1 and Mih1, and can take two non-zero values. First, we have to establish a correspondence between Clb2 levels of activity in the core and in the MCP models. We use the inhibition of MBF, which is common to both models, as a reference to set the threshold of activity of Clb2 on Swe1 in the coupled model at level 2. Consequently, the threshold of activity of Clb2 on Mih1 is set at level 1. In the coupled model, Clb2 thus has three levels of activity, as in the core model. The logical rules for Clb2 are then determined, for levels 2 and 3, by a logical AND between the rules for levels 2 and 3 of the core and level 2 of the MCP models (see Table 3, rules (e) and (g)).

Definition of the rule for level 1 was more delicate, as it was not defined in the core model; based on the role of the

Table 3 Definition of the logical rules enabling the activation of the multi-level node Clb2 for the core engine (Fig. 1), the MCP module (Fig. 2) and the coupled model (Fig. 4). Clb2 has two (non-zero) activation levels in the MCP module, and three (values 1 to 3) in the core engine and in the coupled model. Specific logical formulae are associated with values 1, 2 and 3 for the different models, thereby defining the conditions enabling the activation of Clb2 up to the corresponding levels. All other situations lead per default to the inactivation of Clb2 (value 0). Note that the value 1 is omitted for the core engine and the coupled model, meaning that this Clb2 level can only transiently occur for these models. In the logical formulae, atomic terms are denoted by the name of the component along with the corresponding interval of values (*e.g.* $Cdh1^{[1]}$ denotes $Cdh1 = 1$; bracketed values are omitted in the case of Boolean nodes, or when all non-zero values are considered). The symbols –, $\wedge$, and $\vee$ denote the classical operators NOT, AND and OR, respectively. For example rule (c) states that the target value of Clb2 is 2 when $MASS = 1$ and $Cdh1 = 0$ and $Cdc20$ and $Mcm1$ are both 0 or 1. For the coupled model, the rule (b) (target value 1) is defined as a conjunction of (a) (target value 1) and the disjunction of rules (c) (target value 2) and (f) (target value 3) , while rules (e) and (g) are defined as the conjunction of (d) and (c) and the conjunction of (d) and (f), respectively

Core engine	MCP module	Coupled model
Incoming interactions		
$Cdh1$, $Mcm1$, $Cdc20^{[2,3]}$, $MASS$	$MASS^{[1,2]}$, $Mih1^{[1,2]}$, $Swe1^{[1,2]}$	$Cdh1$, $Mcm1$, $Cdc20^{[2,3]}$, $MASS^{[1,2]}$, $Mih1^{[1,2]}$, $Swe1^{[1,2]}$
Logical rules for the value 1		
	(a) $(MASS^{[1]} \wedge Swe1 \wedge \overline{Mih1^{[2]}}) \vee (MASS^{[2]} \wedge Swe1^{[2]} \wedge \overline{Mih1})$	(b) $((MASS^{[1]} \wedge Swe1 \wedge \overline{Mih1^{[2]}}) \vee (MASS^{[2]} \wedge Swe1^{[2]} \wedge \overline{Mih1})) \wedge (\overline{Cdh1} \wedge \overline{Cdc20^{[3]}} \wedge (Cdc20^{[2]} \vee Mcm1))$
Logical rules for the value 2		
(c) $MASS \wedge \overline{Cdh1} \wedge ((Cdc20 \wedge Mcm1) \vee (Cdc20^{[2]} \wedge \overline{Mcm1}))$	(d) $(MASS^{[1]} \wedge (\overline{Swe1} \vee \overline{Mih1^{[2]}})) \vee (MASS^{[2]} \wedge (Swe1^{[2]} \vee Mih1))$	(e) $((MASS^{[1]} \wedge (\overline{Swe1} \vee Mih1^{[2]})) \vee (MASS^{[2]} \wedge (Swe1^{[2]} \vee Mih1))) \wedge (\overline{Cdh1} \wedge ((Cdc20^{[2]} \wedge Mcm1) \vee (\overline{Cdc20} \wedge \overline{Mcm1})))$
Logical rules for the value 3		
(f) $MASS \wedge \overline{Cdh1} \wedge \overline{Cdc20} \wedge Mcm1$		(g) $((MASS^{[1]} \wedge (\overline{Swe1} \vee Mih1^{[2]})) \vee (MASS^{[2]} \wedge (Swe1^{[2]} \vee Mih1))) \wedge (\overline{Cdh1} \wedge \overline{Cdc20} \wedge Mcm1)$

inhibitors Cdc20 and Cdh1, we defined the rule for level 1 of Clb2 as a logical AND between the rule for level 1 in the MCP model and the union (logical OR) of the rules for levels 2 and 3 in the core (see Table 3, rule (b)).

4.3.3 Integration of the exit module within the core model. The integration of the mitotic exit module within the core model essentially can be achieved through a re-wiring of a specific part of the core model. The hypothetical phosphatase PPX has to be replaced by the mechanism proposed in the exit module. All the other nodes from the core model can be conserved, and left unchanged, as well as the regulation of PP2A and Cdc5Polo, which are defined only in the exit module.

The regulation of several components of the exit module is a simplification of their regulation in the core model. This is the case for Cdc20, Cdh1, Clb2, Pds1 and Tem1, and we have thus kept the original, detailed rule of the core model for the regulation of these components. In the case of Clb2, which has different threshold configurations in the two modules, a mapping of these thresholds can be determined similarly to what has been done for the coupling of the MCP module. Note that the regulatory rules associated with Pds1 and Tem1 in the core model already encompass the rules defined in the exit module. Furthermore, the Cdc14 node is not affected by the coupling process, as the associated logical rules are identical in both models.

In the coupled model Cdc15 becomes regulated by Cdc14 (and thus Cdc14 competitive inhibitor Net1) and Clb2, according to the rules defined in the exit module. The regulation of Net1 is similarly redefined.

The sequestration of separase by securin, which was treated as a direct inhibition in the core model (since in this model Esp1 did not have any target), is now represented by an opposing effect of securin over the separase target, PP2ACdc55.

In the core model, the Bub2-Bfa1 complex is activated by the spindle assembly checkpoint: *i.e.*, it takes the value 1 as long as there are replicated chromosomes ($ORI = 1$) that are not aligned on the metaphase plate ($SPN = 0$) (see Table 4).

The checkpoint is lifted, and Bub2-Bfa1 is inhibited, when $SPN = 1$. In the exit module, Bub2-Bfa1 is the target of another checkpoint mechanism that monitors the separation of sister chromatids: as long as separase has not been released, Bub2-Bfa1 is kept active by the PP2A phosphatase, in spite of the Polo-like kinase Cdc5. To determine the rule for the coupled model, we have to take into account the fact that, in the system represented in the exit module, chromosomes are replicated and aligned on the metaphase plate: in other terms, the condition $ORI = 1$ and $SPN = 1$ is implicit in the rule for Bub2-Bfa1 activation. Thus, in the coupled model, as long as $ORI = 1$, presence of PP2A or absence of Cdc5Polo is sufficient to keep Bub2-Bfa1 active, whatever the level of SPN.

Finally, we have to take into account the sequestration of Clb2 by the CKI in the new logical formulae for Cdc5Polo, Cdc15 and Net1, similarly to what has been done for Swe1 and Mih1 in the previous section.

4.4 Availability

The software *GINsim* implementing the logical formalism and the composition method is freely available to academic groups. Furthermore, we provide the XML files (GINML format) containing the four models (for the three modules plus their composition) on a dedicated webpage (http://gin.univ-mrs.fr/GINsim). This website further summarizes experimental data supporting the interactions considered. Finally, results of our dynamical analyses are provided for about a hundred different conditions, along with corresponding experimental observations.

Table 4 Definition of the logical rules for the coupled model (Fig. 4) combining the core and exit modules (*cf*. Fig. 1 and 3). Only the situations enabling the activation of the nodes Bub2-Bfa1 and Net1 are displayed. A complete listing of the rules defined for the three modules and for the coupled model is provided in the ESI‡. Notations of the logical formulae as in Table 3

Core engine	Exit module	Coupled model
Incoming interactions for Bub2-Bfa1		
ORI, *SPN*	*Cdc5Polo*, *PP2A*	*ORI*, *SPN*, *Cdc5Polo*, *PP2A*
Logical rules for the activation of Bub2-Bfa1 (value 1)		
$ORI \wedge \overline{SPN}$	$PP2A \vee \overline{Cdc5Polo}$	$ORI \wedge (\overline{SPN} \vee PP2A \vee \overline{Cdc5Polo})$
Incoming interactions for Net1		
PPX, *Cdc*15, *Tem*1, *Bub*2-*Bfa*1	*Cdc*15, *Bub*2-*Bfa*1, $PP2ACdc55^{[1,2]}$, $Clb2^{[1,2]}$, *Tem*1, *Net*1, *Cdc*14	*Net*1, *Cdc*6, *Bub*2-*Bfa*1, $Clb2^{[2,3]}$, $PP2ACdc55^{[1,2]}$, *Cdc*15, *Cdc*14, *Sic*1, *Tem*1
Logical rules for the activation of Net1 at value 1		
$((Cdc15 \wedge Tem1) \vee Cdc15^{[2]}) \vee Bub2\text{-}Bfa1 \vee PPX$	$\overline{(Cdc15 \wedge Tem1 \wedge \overline{Bub2\text{-}Bfa1})} \wedge \overline{((Cdc14 \wedge Net1 \wedge \overline{Clb2} \wedge \overline{PP2ACdc55}) \vee (PP2ACdc55^{[1]} \wedge ((\overline{Cdc14} \wedge Clb2^{[1]}) \vee (Cdc14 \wedge Clb2^{[1]}))))}$	$\overline{(((Cdc15 \wedge Tem1) \vee Cdc15^{[2]}) \vee Bub2\text{-}Bfa1) \wedge ((Cdc14 \wedge Net1 \wedge \overline{(\overline{Clb2} \vee Sic1 \vee Cdc6)} \wedge \overline{PP2ACdc55}) \vee (PP2ACdc55^{[1]} \wedge ((\overline{Cdc14} \wedge Clb2^{[2]} \wedge \overline{(Sic1 \vee Cdc6)}) \vee (Cdc14 \wedge Clb2^{[2]} \wedge (Sic1 \vee Cdc6)))))}$
Logical rules for the activation of Net1 at value 2		
	$\overline{(Cdc15 \wedge Tem1 \wedge \overline{Bub2\text{-}Bfa1})} \wedge ((\overline{Clb2} \wedge ((Cdc14 \wedge \overline{Net1}) \vee PP2ACdc55)) \vee PP2ACdc55^{[2]})$	$\overline{(((Cdc15 \wedge Tem1) \vee Cdc15^{[2]}) \vee Bub2\text{-}Bfa1) \wedge (((\overline{Clb2} \vee Sic1 \vee Sic6) \wedge ((Cdc14 \wedge \overline{Net1}) \vee PP2ACdc55)) \vee PP2ACdc55^{[2]})}$

Acknowledgements

This work has been supported by the European Commission through the DIAMONDS FP6 STREP (LSHG-CT-2004-503568), the Action de Recherche Concertée INRIA MOCA (2006-2007), and the Association pour la Recherche sur le Cancer (PhD grant to AF).

References

1 M. A. Schaub, T. A. Henzinger and J. Fisher, Qualitative networks: a symbolic approach to analyze biological signaling networks, *BMC Syst. Biol.*, 2007, **1**, 4.

2 L. Sánchez, C. Chaouiya and D. Thieffry, Segmenting the fly embryo: logical analysis of the role of the segment polarity cross-regulatory module, *Int. J. Dev. Biol.*, 2008, **52**(8), 1059–1075.

3 A. Goldbeter, A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase, *Proc. Natl. Acad. Sci. U. S. A.*, 1991, **88**(20), 9107–9111.

4 J. J. Tyson, Modeling the cell division cycle: cdc2 and cyclin interactions, *Proc. Natl. Acad. Sci. U. S. A.*, 1991, **88**(16), 7328–7332.

5 P. C. Romond, M. Rustici, D. Gonze and A. Goldbeter, Alternating oscillations and chaos in a model of two coupled biochemical oscillators driving successive phases of the cell cycle, *Ann. N. Y. Acad. Sci.*, 1999, **879**, 180–193.

6 J. J. Tyson, K. Chen and B. Novák, Network dynamics and cell physiology, *Nat. Rev. Mol. Cell Biol.*, 2001, **2**(12), 908–916.

7 J. R. Pomerening, E. D. Sontag and J. E. Ferrell, Building a cell cycle oscillator: hysteresis and bistability in the activation of cdc2, *Nat. Cell Biol.*, 2003, **5**(4), 346–351.

8 A. Ciliberto, A. Lukács, A. Tóth, J. J. Tyson and B. Novák, Rewiring the exit from mitosis, *Cell Cycle*, 2005, **4**(8), 1107–1112.

9 J. R. Pomerening, S. Y. Kim and J. E. Ferrell, Systems-level dissection of the cell-cycle oscillator: bypassing positive feedback produces damped oscillations, *Cell*, 2005, **122**(4), 565–578.

10 A. Csikász-Nagy, D. Battogtokh, K. C. Chen, B. Novák and J. J. Tyson, Analysis of a generic model of eukaryotic cell-cycle regulation, *Biophys. J.*, 2006, **90**(12), 4361–4379.

11 L. Calzone, D. Thieffry, J. J. Tyson and B. Novák, Dynamical modeling of syncytial mitotic cycles in drosophila embryos, *Mol. Syst. Biol.*, 2007, **3**, 131.

12 B. Novák, J. J. Tyson, B. Gyorffy and A. Csikasz-Nagy, Irreversible cell-cycle transitions are due to systems-level feedback, *Nat. Cell Biol.*, 2007, **9**(7), 724–728.

13 F. Li, T. Long, Y. Lu, Q. Ouyang and C. Tang, The yeast cell-cycle network is robustly designed, *Proc. Natl. Acad. Sci. U. S. A.*, 2004, **101**(14), 4781–4786.

14 A. Fauré, A. Naldi, C. Chaouiya and D. Thieffry, Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle, *Bioinformatics*, 2006, **22**(14), e124–e131.

15 M. I. Davidich and S. Bornholdt, Boolean network model predicts cell cycle sequence of fission yeast, *PLoS ONE*, 2008, **3**(2), e1672.

16 K. Chen, L. Calzone, A. Csikasz-Nagy, F. Cross, B. Novák and J. Tyson, Integrative analysis of cell cycle control in budding yeast, *Mol. Biol. Cell*, 2004, **15**(8), 3841–3862.

17 A. Ciliberto, B. Novák and J. Tyson, Mathematical model of the morphogenesis checkpoint in budding yeast, *J. Cell Biol.*, 2003, **163**(6), 1243–1254.

18 E. Queralt, C. Lehane, B. Novák and F. Uhlmann, Downregulation of pp2a(cdc55) phosphatase by separase initiates mitotic exit in budding yeast, *Cell*, 2006, **125**(4), 719–732.

19 J. Harrison, E. Bardes, Y. Ohya and D. Lew, A role for the pkc1p/mpk1p kinase cascade in the morphogenesis checkpoint, *Nat. Cell Biol.*, 2001, **3**(4), 417–420.

20 R. Randhawa, C. Shaffer and J. Tyson, Model Composition for Macromolecular Regulatory Networks, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2008, accepted for future publication.

21 J. J. Tyson and B. Novák, Regulation of the eukaryotic cell cycle: molecular antagonism hysteresis and irreversible transitions, *J. Theor. Biol.*, 2001, **210**(2), 249–263.

22 M. Davidich and S. Bornholdt, The transition from differential equations to boolean networks: a case study in simplifying a regulatory network model, *J. Theor. Biol.*, 2008, **255**(3), 269–277.

23 I. Mura and A. Csikász-Nagy, Stochastic petri net extension of a yeast cell cycle model, *J. Theor. Biol.*, 2008, **254**(4), 850–860.

24 R. Thomas and R. D'Ari, *Biological Feedback*, CRC Press, 1990.

25 R. Thomas, D. Thieffry and M. Kaufman, Dynamical behaviour of biological regulatory networks–i. biological role of feedback loops and practical use of the concept of the loop-characteristic state, *Bull. Math. Biol.*, 1995, **57**(2), 247–276.

26 C. Chaouiya, É. Remy, B. Mossé and D. Thieffry, Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework, *Lecture Notes in Control and Information Sciences*, 2003, **294**, 119–126.

27 A. González, A. Naldi, L. Sánchez, D. Thieffry and C. Chaouiya, GINsim: a software suite for the qualitative modelling simulation, and analysis of regulatory networks, *Biosystems*, 2006, **84**(2), 91–100.

Annexe B

Matériel supplémentaire

B.1 Le format GINML

GINsim stocke les graphes de régulation et les graphes de transitions d'états dans des fichiers au format GINML. Ce format XML est une extension du format GXL[1], auquel il ajoute en particulier la définition de fonctions logiques associées aux composants de régulation. Le fichier ci-dessous illustre la définition d'un graphe de régulation dans ce format.

Il est également possible d'ajouter des attributs graphiques et des annotations à chaque élément (graphe, noeuds et arcs). Des exemples plus complets sont disponibles dans le dépôt de modèles du site web de GINsim.

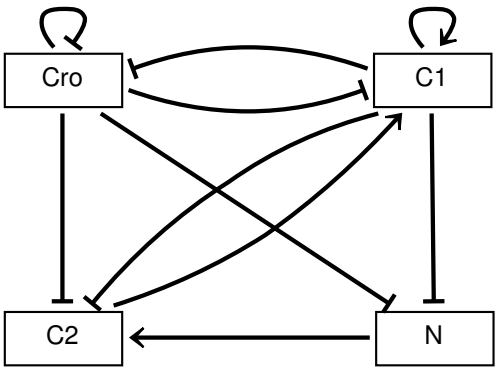


Figure B.1: Graphe de régulation

[1] Graph eXchange Language : www.gupro.de/GXL

```
<?xml version="1.0"?>
<!DOCTYPE gxl SYSTEM "http://gin.univ-mrs.fr/GINsim/GINML_2_1.dtd">
<gxl xmlns:xlink="http://www.w3.org/1999/xlink">
   <graph id="phage4" class="regulatory" nodeorder="C1 Cro C2 N">
      <node id="Cro" basevalue="3" maxvalue="3">
         <parameter idActiveInteractions="Cro_Cro_0" val="2"/>
      </node>
      <node id="N" basevalue="1" maxvalue="1">
      </node>
      <node id="C1" basevalue="2" maxvalue="2">
         <parameter idActiveInteractions="C2_C1_0" val="2"/>
         <parameter idActiveInteractions="C1_C1_0" val="2"/>
         <parameter idActiveInteractions="C1_C1_0 C2_C1_0" val="2"/>
      </node>
      <node id="C2" basevalue="0" maxvalue="1">
         <parameter idActiveInteractions="N_C2_0" val="1"/>
      </node>
      <edge id="Cro_Cro_0" from="Cro" to="Cro"
            minvalue="3" maxvalue="3" sign="negative" />
      <edge id="Cro_C1_0" from="Cro" to="C1"
            minvalue="1" maxvalue="3" sign="negative" />
      <edge id="Cro_N_0" from="Cro" to="N"
            minvalue="2" maxvalue="3" sign="negative" />
      <edge id="Cro_C2_0" from="Cro" to="C2"
            minvalue="1" maxvalue="3" sign="negative" />
      <edge id="N_C2_0" from="N" to="C2"
            minvalue="1" maxvalue="1" sign="positive" />
      <edge id="C1_C2_0" from="C1" to="C2"
            minvalue="2" maxvalue="2" sign="negative" />
      <edge id="C1_N_0" from="C1" to="N"
            minvalue="1" maxvalue="2" sign="negative" />
      <edge id="C1_Cro_0" from="C1" to="Cro"
            minvalue="2" maxvalue="2" sign="negative" />
      <edge id="C1_C1_0" from="C1" to="C1"
            minvalue="2" maxvalue="2" sign="positive" />
      <edge id="C2_C1_0" from="C2" to="C1"
            minvalue="1" maxvalue="1" sign="positive" />
   </graph>
</gxl>
```

GINsim supporte également des fichiers au format zginml qui consistent en une archive (zip) contenant un fichier GINML et des fichiers annexes (définition de mutants, états initiaux, paramètres de simulation, de réduction, ...).

B.2 Exemples de scripts

J'ai récemment introduit dans GINsim un mode "script". Ce mode, utilisant l'excellent Jython[2], permet d'accéder aux fonctionnalités de GINsim de manière automatique. L'API n'étant pas encore fixe, son utilisation n'est pas encore encouragée (c'est plutôt une preuve de concept utilisée en interne).

Ce petit exemple de script donne un aperçu des possibilités.

```
# GINsim-jython script ; run it with:
# java -cp jython.jar:GINsim.jar org.python.util.jython <script.py>

import GINsim as gs
import sys

from java.io import FileOutputStream, OutputStreamWriter

# open a model
graph = gs.open("/path/to/a/file.zginml")

# apply a predefined reduction, identified by its name
reduced_graph = gs.reduce(graph, "reduction")

# export the file as SVG
gs/export_SVG(graph, "/path/to/file.svg")

# find and print the stable states
for state in gs.get_stable_states(graph):
    print state
# the same with a predefined mutant
for state in gs.get_stable_states(graph, "mutant_name"):
    print state


# launch a predefined simulation on the reduced graph
simulator = gs.SimulationReporter(graph)
stg = simulator.run("simulation_name")

# compute the strongly connected components of this STG
scc = gs.get_scc(stg)
```

[2] www.jython.org

L'ensemble des API de GINsim, Java et Python sont accessibles depuis ces scripts, il est donc possible d'aller plus loin. En particulier, GINsim propose une API pour la création de document, très pratique pour générer automatiquement un rapport. Voici par exemple une version un peu allégée du script utilisé pour lancer les simulations en cascade et générer une partie des tables de résultats de [82].

```
# GINsim-jython script ; run it with:
# java -cp jython.jar:GINsim.jar org.python.util.jython <script.py>

import GINsim as gs
import sys

from java.io import FileOutputStream, OutputStreamWriter

# some parameters
extension = "html"  # we used "tex" for the paper
styles = {0:"ko", 1:"ok", 2:"okb"}
state_order = ("Th0", "Th1", "Th17", "Th2", "Treg")
colors = {0:"", 1:"\\TCL", 2:"\\TCM", 3:"\\TC"}

# function called for each reached stable state
def stable_handler(item):
    try:
        item = item.state
    except:
        pass
    state, i = [], 0
    for v in ref_state:
        if v: state.append(0)
        else: state.append(item[i])
        i += 1
    # make sure that the new state is listed in done
    if state not in done:
        queue.append(state)
        done[state] = {}

    # add this simulation to the results
    if current_simulation:
        simu, istate, iname = current_simulation
        previous = done[state]
        if istate in previous:
            previous[istate].append(current_simulation)
        else:
            previous[istate] = [current_simulation]

def add_name_in_map(name, value, m):
    if not name: name = "NONAME"
    if name in m: mname = m[name]
    else:
        mname = {}
        m[name] = mname
    mname[value] = None
```

```
def esc(s):
    return "␣".join(str(s).split("_"))

##############################################################################
# main part
##############################################################################

# open and reduce the graph and create the reference state
graph = gs.open("Th_differentiation.zginml").reduce("reduction")

node_order = [ str(node)      for node in graph.getNodeOrder() ]
ref_state =  [ node.isInput() for node in graph.getNodeOrder() ]

# initialise data structures
queue = []
done = {}
current_simulation = None
stable_handler([ 0 for v in ref_state ])

simu = gs.get_simulations(graph, "simulation")

# the reporter in charge of the simulations
sr = gs.SimulationReporter(graph, stable_handler)

#short names for input configs:
inputs = gs.get_inputs(graph)
inputNames, next = {}, ord('A')
for i in inputs:
    inputNames[i.getName()] = chr(next)
    next += 1

while queue:
    next_state = queue.pop()
    print "next␣state:␣", next_state
    for i in inputs:
        iname = i.getName()
        current_simulation = (simu, next_state, iname)
        sr.run(str(simu), next_state, i)

# create a report
###################
shortNext = {}
namedStates = {}
# compact view of the results
named, unnamed = {}, {}
for state, nexts in done.items():
    sname = gs.name_state(state, graph)
    add_name_in_map(sname, state, namedStates)
    for next, simulations in nexts.items():
        for param, init, inpconfig in simulations:
            iname = gs.name_state(init, graph)
            add_name_in_map(iname, init, namedStates)
            inpname = inputNames[inpconfig]
```

```
            try:
                inext = shortNext[iname]
            except:
                inext = {}
                shortNext[iname] = inext
            if sname in inext:
                inext_s = inext[sname]
                if inpname not in inext_s:
                    inext_s.append(inpname)
            else:
                inext[sname] = [inpname]

# first, created a custom-sorted list of reached states
rstates = [s for s in shortNext ]
rstates.sort()
states = []
for prefix in state_order:
    for state in rstates:
        if state = prefix or state.startswith("%s_" % prefix):
            states.append(state)
for state in rstates:
    if state not in states:
        states.append(state)

# set some metadata for styling
nb_items = len(states)
properties = {"title":"report"}
if extension = "tex":
    properties["sty"] = "style.sty"
elif extension = "html":
    properties["css"] = "style.css"

# create the document
dw = gs.create_report("reports/my_report.%s" % extension, properties)
dw.openTable("", "", ["" for it in xrange(nb_items+1)])
dw.openTableCell(1,1, "", True)

index = 0
for state in states:
    dw.openTableCell(1,1, state, "side", True)
for cur in states:
    nexts = shortNext[cur]
    dw.openTableRow()
    dw.openTableCell(1,1, cur, True)
    for next_state in states:
        if next_state in nexts:
            shortInputs = nexts[next_state]
            shortInputs.sort()
            value = "".join(shortInputs)
            dw.openTableCell(1,1, value, "mcp", False)
        else:
            dw.openTableCell(1,1, "", False)
dw.closeTable()
```

```
dw.openHeader(2,"Legend",None)
dw.openHeader(3,"Simulations",None)

# states and their names
dw.openHeader(3,"States",None)
normal_nodes = [ pos
                 for pos in xrange(len(ref_state))
                 if not ref_state[pos] ]
dw.openTable("", "", ["" for it in xrange(len(normal_nodes)+1)])
dw.openTableCell(1,1,"",True)
for pos in normal_nodes:
    dw.openTableCell(1,1,node_order[pos],"side",True)

nbcol = len(normal_nodes) + 1
for name in states:
    states = namedStates[name]
    first = True
    for s in states:
        dw.openTableRow()
        if first:
            dw.openTableCell(1,len(states),name,True)
            first = False
        for pos in normal_nodes:
            value = s[pos]
            dw.openTableCell(1,1,"", styles[value], False)
dw.close()
```

Modélisation logique de la différenciation des lymphocytes T auxiliaires

Résumé. La réponse immunitaire adaptative implique la collaboration de plusieurs lignées cellulaires, dont plusieurs sont régulées par les lymphocytes T auxiliaires (Th). Les Th ont longtemps été divisés en sous-types Th1 et Th2, activant respectivement la “réponse cellulaire” et la “réponse humorale” (production d’anti-corps). Cette dichotomie a été récemment revue suite à la découverte des cellules régulatrices (Treg) et des Th17, responsables respectivement de l’arrêt de la réponse et de l’activation de l’inflammation. Afin de mieux comprendre le fonctionnement du réseau de régulation sous-jacent, nous avons utilisé une méthode de modélisation dynamique qualitative. En nous appuyant sur les données publiées et sur de précédents modèles, nous avons construit un réseau de régulation intégrant les interactions croisées entre 65 composants de régulation. Pour analyser les propriétés d’un tel réseau, nous avons développé des méthodes informatiques basées sur les diagrammes de décision. Implémentées dans le logiciel de modélisation logique GINsim, ces méthodes permettent de déterminer tous les états stables, ainsi que les circuits de régulation dynamiquement cruciaux au sein d’un modèle. Enfin, nous avons mis au point une méthode itérative de réduction de la taille d’un modèle qui conserve d’importantes propriétés dynamiques. Le modèle logique proposé récapitule les quatre voies de différenciation des cellules Th connues, mais prédit en outre l’existence de lignées mixtes, ainsi que diverses transitions possibles entre ces lignées en fonction des stimuli (activation TCR, cytokines) fournis par l’environnement cellulaire.
Mots-clefs : modélisation dynamique, formalisme logique, lymphocytes T, diagrammes de décision, circuits de régulation, réduction de modèle, différenciation cellulaire

---

Logical Modelling of Helper T Cell Differentiation

Abstract. The adaptive immune response relies on the collaboration between complementary cell lineages, several of which are regulated by Helper T (Th) lymphocytes. Th cells have long been partitioned into Th1 and Th2 subtypes, which respectively activate the “cellular response” and the “humoral response” (antibody production). This dichotomy has been recently revised following the discovery of regulatory T cells (Treg) and Th17, responsible for the arrest of the immune response and for the activation of the inflammatory response, respectively. To gain insight into the functioning of the complex signalling, regulatory networks involved in Th cell activation and differentiation, we have used a qualitative modelling approach. Leaning on published data and previous models, we have built a comprehensive logical model encompassing 65 regulatory components. To cope with such a large network, we have developed several novel computational analysis methods based on decision diagrams. Implemented into a dedicated modelling software, GINsim, these methods enable the efficient determination of all stable states, as well as the delineation of functional regulatory circuits in a model. Furthermore, we have designed an iterative method to reduce the size of a model while preserving its most salient dynamical properties. Our logical model recapitulates the four known Th cell differentiation pathways, but also predicts various mixed lineages, as well as potential transitions between them, depending on the inputs (TCR activation,cyokines) provided by the cellular environment.
Keywords: dynamical modelling, logical formalism, T lymphocytes, decision diagrams, regulatory circuits, model reduction, cellular differentiation

---

TAGC INSERM-U928, 163 Avenue de Luminy, 13288 MARSEILLE