P3M : une plate-forme logicielle pour la modélisation et la manipulation des cartes d'interactions moléculaires

J.M. Alliot[1] M. Cialdea[2] R. Demolombe[1] M. Dieguez[1]
L. Farinas[1] G. Favre[3] J.C. Faye[3] O. Sordet[3]

[1] IRIT
[2] Università degli Studi Roma
[3] INSERM

jean-marc.alliot@irit.fr

Résumé

Les réseaux métaboliques, composés d'une série de voies métaboliques, consistent en un ensemble de réactions cellulaires et extracellulaires qui déterminent les propriétés biochimiques d'une cellule, régulées par des interactions complexes d'activation et d'inhibition. Ces réseaux sont le plus souvent représentés sous la forme de graphe qui décrivent les liaisons entre les différents points de contrôle du cycle cellulaire. Dans cet article, nous décrivons une plate-forme logicielle, partant de la description des réactions sous forme de graphe, de leur modélisation en logique temporelle et aboutissant à un outil permettant de vérifier la cohérence du modèle avec les résultats expérimentaux, de trouver l'ensemble de conditions initiales aboutissant à un état donné, ou de réconcilier des données expérimentales avec le graphe.

Abstract

Metabolic networks, formed by a series of metabolic pathways, are made of intracellular and extracellular reactions that determine the biochemical properties of a cell, and by a set of interactions that guide and regulate the activity of these reactions. Most of these pathways are formed by an intricate and complex network of chain reactions, and can be represented in a human readable form using graphs (often called Molecular Interaction Maps) which describe the cell cycle checkpoint pathways. In this article, we describe a software workflow, starting from the graph description of these interactions, from their modeling in temporal logic and leading to a tool that can model reactions, find the initial conditions leading to a given final state, or modify the graph in order to correctly model the observed behaviour of the cell during real experiments.

Keywords : logique temporelle, médecine, biologie

1 Introduction

Les réseaux métaboliques sont formés par un ensemble de voies métaboliques, et consistent en un ensemble

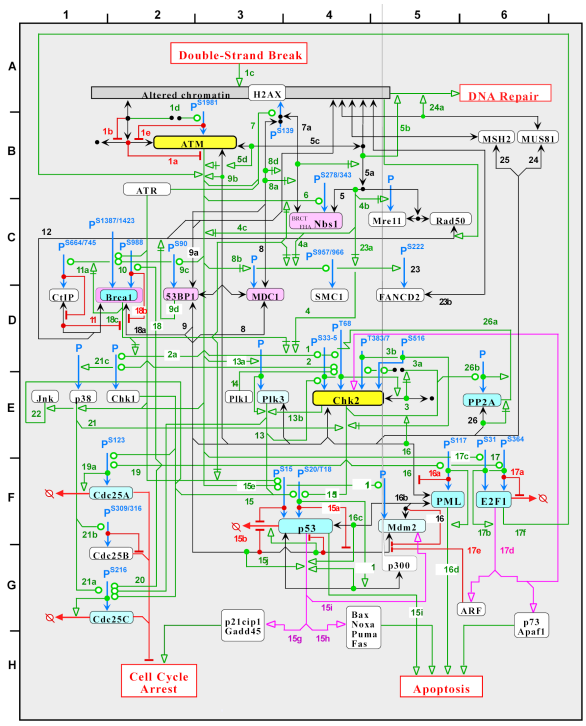

FIGURE 1 – atm-chk2/atr-chk1 molecular interaction map.

de de réactions cellulaires et extra-cellulaires qui déterminent les propriétés biochimiques d'une cellule, régulées par des interactions complexes d'activation et d'inhibition [KAWP06]. Ces réactions sont à la base du fonctionnement cellulaire et sont régulées par d'autres protéines, qui peuvent les activer ou les inhiber.

Ces réseaux sont le plus souvent représentés sous la forme de graphes qui décrivent les liaisons entre les différents points de contrôle du cycle cellulaire. Ces graphes peuvent devenir extrêmement importants et complexes (voir figure 1 [KP05]), et bien qu'essentiels à la formalisation de

la connaissance du fonctionnement de la cellule, ils sont difficiles à utiliser :

— Leur lecture est difficile en raison de leur taille.

— Ils contiennent souvent de la connaissance implicite, ce qui en modifie la compréhension suivant le niveau d'expertise de celui qui l'écrit et celui qui le lit.

— Ils peuvent contenir des incohérences, ou des manques, souvent difficiles, voire impossibles à détecter.

Dans cet article, nous présentons un logiciel permettant de formaliser, de vérifier, et de répondre à des questions complexes concernant ce type de graphe. Ce logiciel est basé sur l'utilisation de la logique temporelle linéaire pour modéliser le fonctionnement cellulaire, suivi de la transformation des formules temporelles en formules de la logique propositionnelle par réification, et enfin par l'utilisation d'un solveur SAT pour la résolution [1].

Dans la section 2, nous présentons un exemple classique (l'opéron Lac) sur lequel nous baserons la plupart de nos exemples, dans la section 3 nous montrons comment représenter une série de voies métaboliques dans notre formalisme et nous présentons l'ensemble des opérations, et les types de variable que notre modèle peut traiter, ainsi que le mécanisme général d'évolution de l'automate, la section 4 présente l'état actuel de l'implantation de l'outil la section 5 présente des exemples d'utilisation, et la section 6 présente les travaux actuellement en cours pour améliorer l'outil.

2 L'opéron lac

Nous allons tout d'abord présenter un graphe simple, qui décrit la régulation de l'opéron lactose, ou opéron *lac* [2]. Cet opéron a été le premier mécanisme de régulation génétique à être compris clairement et est maintenant un exemple "standard" de tous les cours de biologie.

L'opéron lac est un opéron utilisé par la bactérie pour permettre le transport et le métabolisme du lactose. En effet, si le glucose est la source de carbone "préférée" de la plupart des bactéries, l'opéron lac permet la digestion du lactose quand le glucose n'est pas disponible. L'opéron lac est une séquence de trois gènes (lacZ, lacY et lacA) qui encodent trois enzymes qui, à leur tour, permettent la transformation du lactose en glucose. Nous allons nous concentrer ici sur le gène lacZ qui encode la β-galactosidase, qui permet de cliver le lactose en glucose et galactose.

L'opéron lac utilise un système de contrôle en deux parties, pour s'assurer que la cellule n'utilise que l'énergie nécessaire. En l'absence de lactose, le répresseur lac arrête la production des enzymes encodés par l'opéron ; en présence de glucose, la production de l'adénosine monophosphate cyclique (CAMP) à partir de l'adénosine triphosphate (ATP) est bloquée.

1. Pour d'autres approches, on peut se référer à [CRFS04] et plus généralement à [FI14].

2. Le prix Nobel fut attribué à Monod, Jacob and Lwoff in 1965 en partie pour la découverte de l'opéron lac par Monod et Jacob [JM61]

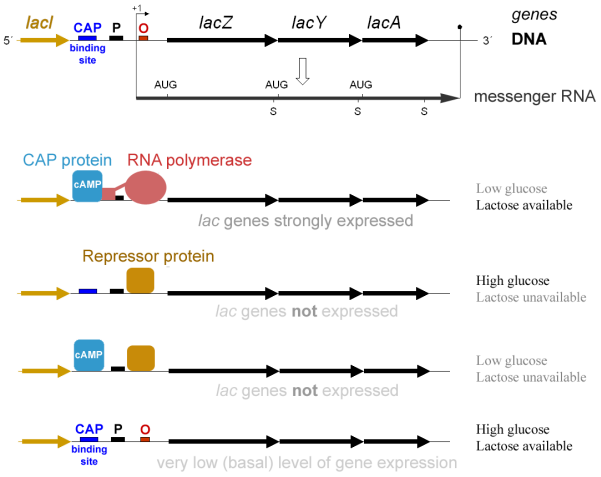

FIGURE 2 – Opéron Lac

La figure 2 décrit ce mécanisme. L'expression du gène lacZ n'est possible que si l'ARN polymérase (en rose) peut se lier au site promoteur (P, en noir) en amont du gène. Cette liaison est aidée par la molécule CAMP (en bleu) qui se lie avant le promoteur sur le site CAP (bleu foncé).

Le gène lacI (en jaune) est toujours exprimé, et produit une protéine répresseur LacI, qui se lie sur le site promoteur de l'ARN polymérase quand le lactose est disponible, empêchant l'ARN polymérase de s'y fixer, et bloquant ainsi l'expression des gènes suivants (LacZ, lacY, lacA) : il s'agit là d'une *régulation négative*, ou d'une *inhibition*, car elle bloque la production des protéines. Mais quand le lactose est présent, un de ses isomères (l'allolactose) se lie avec la protéine répresseur LacI, et le composé résultant ne peut plus se lier sur le site promoteur. L'ARN polymérase. peut alors se lier au site promoteur et permettre l'expression du gène lacZ *si* CAMP est liée à CAP.

CAMP exerce une *régulation positive*, ou activation, car sa présence est nécessaire à l'expression du gène lacZ. Mais CAMP est elle-même régulée négativement : quand du glucose est présent, CAMP n'est plus produite et ne peut pas se lier sur le site CAP, empêchant l'expression de lacZ.

3 Modélisation

Le mécanisme décrit dans la section précédente est résumé dans le graphe de la figure 3 [ADFdC16, ADD$^+$16]. Cet exemple contient toutes les relations et tous les types d'acteur que nous utilisons dans notre modèle, et nous allons les détailler maintenant, ainsi que leur mode d'évolution temporelle, et le type de problème que notre logiciel peut traiter.

3.1 Relations

Il existe deux grands types de relations : les productions et les régulations.

Productions. Les **productions** peuvent prendre deux formes différentes, suivant que les réactifs sont consom-

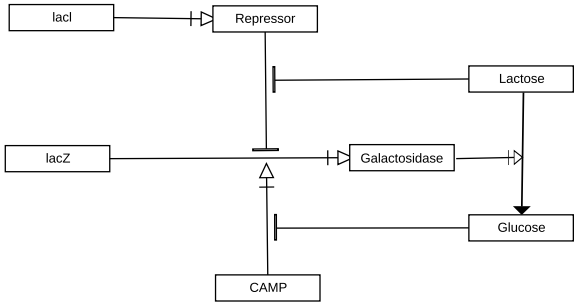

FIGURE 3 – Représentation fonctionnelle de l'opéron lac

més, ou non consommés [3] :

— Dans la figure 3, le lactose produit du glucose, et est consommé, ce que l'on note : ($lactose \rightarrow glucose$).

— A l'opposé, l'expression du gène lacZ pour produire la galactosidase (ou du gène lacI pour produire la protéine LacI) ne consomme pas le gène, et nous le notons : ($lacZ \rightarrow galactosidase$).

Définition :

— Si une réaction consomme ses réactifs, nous écrivons : $a_1, a_2 ... a_n \rightarrow b$. Ici la production de b consomme a_1,...,a_n

— Si les réactifs ne sont pas consommés, nous écrivons : $a_1, a_2, ... a_n \rightarrow b$. Ici b est produit mais a_1, a_2 ... a_n sont toujours présents après la production de b.

Régulations. Les **régulations** peuvent également prendre deux formes différentes : une réaction peut-être soit *inhibé* soit *activée* par d'autres protéines, ou d'autres conditions.

— Dans l'exemple ci-dessus, la production de la galactosidase à partir de l'expression du gène lacZ est activée par CAMP (nous écrivons $CAMP \rightarrow$ pour exprimer l'activation)

— D'autre part la même production de galactosidase est bloquée (ou inhibée) par la protéine répresseur LacI (ce que l'on note $Repressor \dashv$).

Définition :

— nous écrivons $a_1, a_2, ... a_n \rightarrow$ si la présence simultanée de $a_1, a_2, ... a_n$ active une production ou une autre régulation.

— nous écrivons $a_1, a_2, ... a_n \dashv$ si la présence simultanée de $a_1, a_2, ... a_n$ inhibe une production ou une autre régulation.

3. Dans les graphes Pathvisio nous employons $\dashrightarrow$ à la place de $\rightarrow$ pour des raisons techniques mais les deux opérateurs sont équivalents.

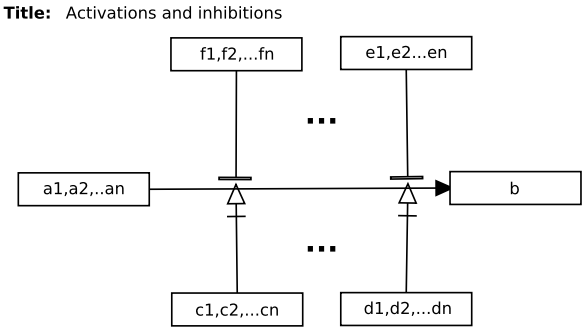

FIGURE 4 – Activations/Inhibitions

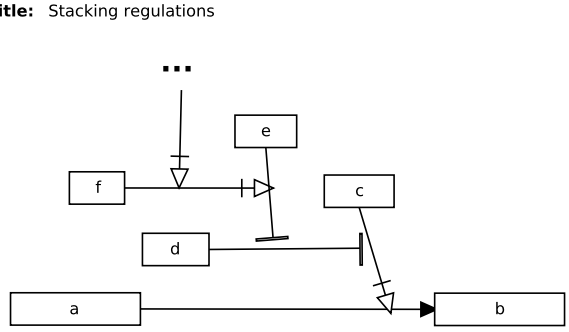

FIGURE 5 – Empilage

Forme générale d'un graphe. Sur la figure 4, nous avons un résumé de la plupart des cas possibles : la production de b à partir de $a_1, \cdots, a_n$ est activée par la présence simultanée de $c_1, \cdots, c_n$ **ou** par la présence simultanée de $d_1, \cdots, d_n$, et inhibée par la présence simultanée de $e_1, \cdots, e_n$ **ou** par la présence simultanée de $f_1, \cdots, f_n$. Les régulations sont souvent "empilées" sur plusieurs niveaux (voir figure 5). Par exemple, dans la figure 3, l'inhibition de la production de galactosidase par le répresseur LacI peut elle-même être inhibée par la présence de lactose, et l'activation de cette même production par CAMP est elle-même inhibée par la présence de glucose.

3.2 Types d'acteur

Après avoir détaillé les différents types d'interactions possibles entre les acteurs, nous allons maintenant détailler les deux différents types d'acteurs présents dans ces graphes :

Acteurs exogènes : un acteur *exogène* a sa valeur fixée par des conditions externes (température, présence ou absence d'un gène, protéine fournie en quantité suffisante par l'environnement) et sa valeur n'évolue jamais dans le temps. Ils sont fixés par les conditions expérimentales, et leur valeur n'évolue jamais dans le temps. S'ils sont consommés, l'environnement en fournira toujours en quantité suffisante. S'ils sont produits, ils sont supposés consommés instantanément par l'environnement.

Acteurs endogènes : à l'opposé les acteurs endogènes peuvent évoluer dans le temps en fonction de la dynamique du graphe. Ils peuvent apparaître s'ils sont produits, disparaître s'ils sont consommés. Leur valeur initiale est fixée par l'utilisateur.

Le statut d'un acteur est fixé par le biologiste en fonction de sa compréhension du processus biologique décrit par le graphe. En général, il existe des règles de bon sens permettant d'initialiser le type des acteurs : les acteurs exogènes n'apparaissent quasiment jamais à droite d'une règle de production (ils ne sont généralement pas produits), alors que les acteurs endogènes apparaissent eux à droite de ce même type de règles (ils sont produits).
Par exemple, dans la figure 3, les deux types d'acteurs sont présents : lacl, lacZ, CAMP et le lactose sont initialisés comme acteurs exogènes en fonction de la règle "de bon sens" énoncés ci-dessus ; toujours suivant la même règle, la galactosidase, la protéine répresseur et le glucose sont classés comme endogènes. Ces règles "de bon sens" permettant de classifier les variables en endogène ou exogène ne servent qu'à initialiser "simplement" les types de variables, mais ils doivent pouvoir être modifiés ultérieurement par le biologiste suivant les conditions expérimentales et le type de l'étude. Le lactose peut par exemple être considéré comme une variable exogène ou une variable endogène, suivant le type de dynamique que le biologiste veut étudier : soit un environnement saturé en lactose, soit un environnement où le lactose est présent à l'état initial, mais peut disparaître par consommation par la bactérie. De la même façon le glucose pourrait être considéré comme exogène si l'on veut simuler un environnement saturé en glucose.
Il faut également comprendre que les graphes décrivent les réactions que les biologistes estiment importantes pour décrire un mécanisme particulier du fonctionnement cellulaire. Ils peuvent être écrits de multiples façons, en fonction par exemple des blocs fonctionnels que l'on veut dégager, et certaines réactions ou relations ne sont simplement pas décrites parce qu'elles ne sont pas fondamentales pour l'objet de l'étude.

3.3 Évolution temporelle

Un graphe peut être considéré comme un automate qui produit une séquence d'états des acteurs. Les formules pour décrire les séquences d'état sont celles de la logique temporelle linéaire. Le temps est supposé discret, et toutes les relation de production/consommation qui peuvent s'exécuter s'exécutent simultanément en un pas de temps. Un acteur peut donc être vu comme une variable booléenne pouvant prendre les valeurs 0 (absent) ou 1 (présent). Lorsqu'un acteur est consommé, il devient absent ; lorsqu'il est produit, il devient présent. Lorsqu'un acteur (déjà présent) est à la fois consommé et produit, il reste présent par défaut. Ce fonctionnement peut sembler réducteur, puisqu'il ne prend en compte ni les quantités, ni les vitesses de réaction, mais nous allons voir qu'il permet déjà de traiter de nombreux problèmes.
Les valeurs des variables sont initialisées également à partir de règles de "bon sens". Les variables exogènes sont initialisées à *libre* : ce sont généralement les variables sur lesquelles on va "raisonner". Les variables endogènes sont initialisées à *absent*. Là aussi, cette initialisation sera modifiée par l'utilisateur à travers l'interface graphique en fonction de l'étude qu'il poursuit (conditions initiales, etc).

3.4 Types de problèmes traités

Le logiciel est capable de traiter trois grands types de "problème" sur un graphe donné :

Cohérence : le logiciel peut vérifier la cohérence du graphe avec des résultats expérimentaux. L'utilisateur fournit les conditions initiales et les conditions finales d'une expérimentation, et le système vérifie que les résultats expérimentaux sont en accord avec la modélisation.

Abduction : étant donné un ensemble de conditions finales, le logiciel peut trouver le ou les ensembles de conditions initiales qui permettent d'y aboutir. Dans ce cas l'utilisateur initialise les valeurs des variables dont il veut fixer la valeur, et laisse libres les valeurs initiales des variables sur lesquelles il veut raisonner. La complexité du système croit exponentiellement avec le nombre de variables libres : lorsque n variables sont libres, il y a 2^n chemins possibles pour l'automate. Nous verrons un exemple détaillé dans la section 5.1.

Mise à jour : lorsque les conditions finales d'une expérimentation ne sont pas en accord avec le modèle, le logiciel peut proposer des modifications du graphe permettant de mettre en cohérence les résultats observés avec le modèle. Cela peut se faire soit en fixant toutes les valeurs initiales des variables, soit en laissant certaines variables libres et en recherchant des solutions vérifiant certaines conditions sur ces variables. Dans ce cas, la mise à jour se combine avec l'abduction. Si N est le nombre total de variables, n le nombre total de variables libres, r le nombre de relations et p le nombre de relations nouvelles que nous cherchons à ajouter au graphe,la complexité du problème à résoudre croit comme $2^n(N(r+N))^p$, Nous verrons un exemple détaillé de mise à jour de graphe dans la section 5.2.

4 Implantation

Nous allons décrire ici la plate-forme logicielle que nous utilisons pour la représentation, l'interprétation et l'utilisation de nos graphes. Le mécanisme général, en quatre parties, est décrit dans la figure 6.

4.1 Saisie des graphes

Les graphes sont construits en utilisant un outil standard, *Pathvisio* [vIKP+08]. Il s'agit d'un logiciel du domaine public, bien connu dans la communauté des biologistes.

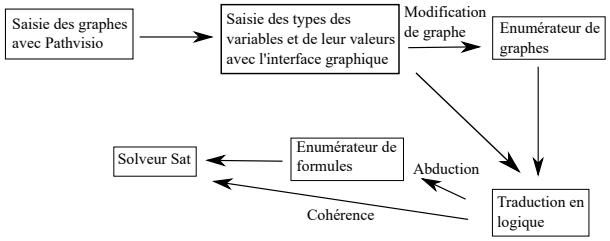

FIGURE 6 – Implantation

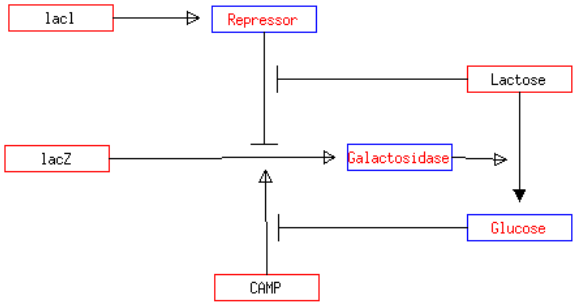

FIGURE 7 – Opéron lac après initialisation "simple" des types de variables et de leurs valeurs

Utiliser Pathvisio nous permet également de récupérer des graphes déjà écrits dans d'autres contextes et de les utiliser pour nos besoins propres sans avoir à les saisir à nouveau. Pathvisio permet de saisir des informations plus riches que celles utilisées par notre modélisation, mais elles ne sont pas utilisées pour l'instant. Les graphes sont sauvegardés en format XML par Pathvisio, et peuvent donc être relus par n'importe quel parseur XML.

4.2 Saisie des types des variables et de leurs valeurs avec l'interface graphique

Le fichier XML produit par Pathvisio est directement relu par notre logiciel. A travers une interface graphique, l'utilisateur peut modifier le type des variables (exogène / endogène) ainsi que leurs valeurs initiales. On peut voir sur la figure 7 comment le logiciel initialise les types et les valeurs des variables à partir des règles "de bon sens" décrites dans la section précédentes. Les boites entourant lacI, lacZ, CAMP et Lactose sont *rouges*, ce qui indiquent qu'elles sont *exogènes*, les boîtes entourant le glucose, la galactosidase et le répresseur sont en *bleues*, indiquant qu'il s'agit de variables *endogènes*. Les valeurs des variables sont initialisées à *absent* (*rouge*) pour les variables endogènes et à *libre* (*noir*) pour les variables exogènes.
Sur la figure 8, on voit comment l'utilisateur a modifié les valeurs des variables. lacI, lacZ et CAMP sont en *verts*, indiquant qu'ils *sont présents* au début (et le resteront, puis-

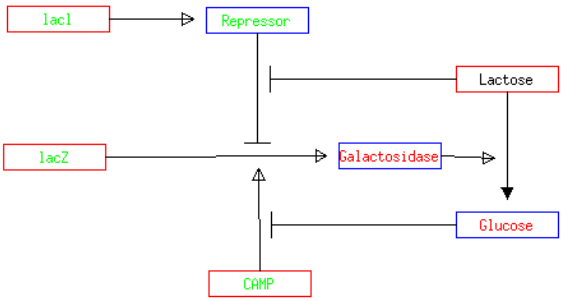

FIGURE 8 – Opéron lac après modifications

qu'il s'agit de variables exogènes). Le répresseur est *vert*, en effet la protéine répresseur est toujours présente dans la cellule, en particulier dans les conditions initiales. Le lactose reste *noir* car il s'agit de la variable *libre*, sur laquelle l'utilisateur va "poser des questions" au système.
Les questions à poser au système ainsi que les différents paramètres à saisir (nombre de pas de temps, ou nombre de modifications à effectuer sur le graphe pour les problèmes de mise à jour) doivent être posés au système à travers une interface textuelle pour l'instant.

4.3 Moteur de résolution

Le fonctionnement du moteur de résolution dépend du type de questions posées :

Cohérence : on appelle directement un solveur SAT standard (ici minisat [ES03]) qui vérifie la cohérence des formules décrivant le graphe et les conditions initiales avec les conditions finales de la question.

Abduction : on construit l'ensemble des conditions initiales possibles, et pour chacune d'entre elles, on appelle le solveur SAT afin de vérifier la cohérence avec les conditions finales.

Mise à jour de graphe : on construit l'ensemble des graphes possibles et on appelle le solveur SAT pour en vérifier la cohérence. S'il existe des variables libres, on ajoute l'étape d'abduction avant l'appel du solveur SAT.

La traduction en logique temporelle linéaire du graphe, puis la traduction de cette représentation en logique propositionnelle par réécriture et réification en associant à chaque état du modèle temporel un nombre naturel, s'effectue une fois que le graphe est fixé, c'est à dire après l'appel de l'énumérateur de graphes s'il a lieu ; dans ce cas, il faudra traduire successivement chacun des graphes énumérés. Les fondements logiques et le détail de la méthode de traduction sont décrits dans [ADFdC16, ADD$^+$16, CMD17], nous n'y reviendrons pas ici.

5 Exemples d'utilisation

Nous allons nous concentrer ici sur les deux types les plus intéressants et les plus complexes à traiter : l'abduction et la mise à jour de graphe.

5.1 Abduction

Dans cette section, nous allons nous intéresser à un exemple complexe ; nous allons reprendre une fraction significative de la carte présentée dans la figure 1 et la modéliser sous forme de graphe ; nous allons nous intéresser à la voie métabolique atm-chk2, qui amène à l'apoptose cellulaire lorsqu'une rupture du double brin d'ADN se produit.
La rupture du double brin d'ADN (Double strand break ou *dsb*) est une cause majeure de cancer, et les recherches médicales et pharmaceutiques [KP05, GMP$^+$11] ont montré que la rupture du double brin d'ADN peut apparaître dans une cellule comme la résultante d'une pathologie dans une voie métabolique.
Ce type de carte est utilisée pour étudier les déterminants moléculaires de la réponse tumorale aux cancers. Les paramètres moléculaires incluent la voies métabolique de la réparation de l'ADN, celle de l'apoptose programmée de la cellule et celle des points de contrôle du cycle cellulaire [PSR$^+$05, KP05, GMP$^+$11, LKLC07, PZL$^+$11]. Lorsque l'ADN est endommagée, les points de contrôle du cycle cellulaire sont activés et peuvent rapidement tuer la cellule par apoptose, ou arrêter la progression du cycle cellulaire pour permettre la réparation de l'ADN avant la reproduction ou la division cellulaire. Deux de ces points de contrôle sont les voies métaboliques atm-$chk2$ (que nous allons utiliser comme exemple ici) et atr-$chk2$ [PSR$^+$05].
Sur la figure 9 (construite à partir de la partie de la carte de la figure 1 représentant la voie atm-$chk2$) nous avons le graphe représentant la voie métabolique atm-$chk2$ qui conduit de trois façons différentes à l'apoptose cellulaire. Il y a dans ce graphe six variables exogènes : atm, dsb, $chk2$, $mdm2$, pml et $p53$. Toutes les autres variables sont endogènes. Certaines de ces variables sont des protéines, alors que d'autres comme dsb qui représente la rupture du double brin d'ADN, ou $apoptose$ qui représente la mort cellulaire, sont des conditions ou des états.
La possibilité de faire de l'abduction sur ce type de graphe dépend du nombre de pas de temps considérés ; sur cet exemple, si nous utilisons t pas de temps, nous avons $20 + 42t$ variables et $14 + 145t$ clauses. Cependant, si le nombre de variables libres reste relativement bas (ce qui est le cas ici), les temps de réponse du système sont extrêmement rapides. En effet, si nous avons n variables libres, nous ne devons réaliser que 2^n appels au solveur SAT, car seules les valeurs des variables libres ont un intérêt pour l'expérimentateur. Dans le cas présent, en utilisant par exemple 11 pas temporels (482 variables et 1609 clauses), le système répond en quelques millisecondes à la question lui demandant de trouver les conditions initiales menant à l'apoptose cellulaire [4].

4. L'utilisation d'un algorithme d'abduction "pur" comme celui de Tsiknis [KT90] amène en revanche à des temps de résolution très élevés. La raison est que l'algorithme de Tsiknis travaille sur l'ensemble des variables, alors que les seules valeurs intéressant les biologistes sont les valeurs des variables exogènes.

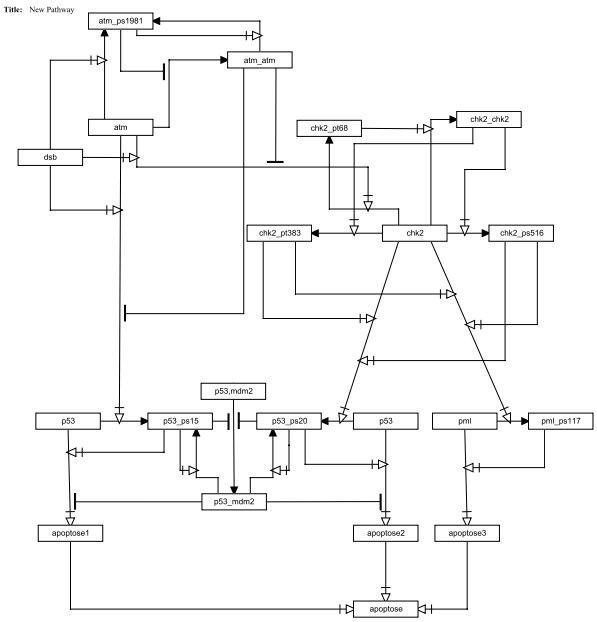

FIGURE 9 – carte d'interaction moléculaire atm-$chk2$ (avant initialisation des variables)

Il est même possible de raffiner les questions en demandant par exemple combien de temps il faut pour atteindre l'apoptose cellulaire sur chacun des trois voies métaboliques, et quelles sont les conditions initiales qui y mènent. Il suffit de poser les questions $apoptose1_i$, $apoptose2_i$ et $apoptose3_i$ pour tous les i en partant de $i = 0$.

- $apoptose1$ est la voie la plus courte et $apoptose1_2$ (second pas de temps) est vraie si atm, dsb and $p53$ sont présents, $mdm2$ est absent et les valeurs de pml et $chk2$ sont indifférentes (la réponse est correcte et peut facilement être vérifiée sur la carte). Pour $i \geq 3$, la réponse à $apoptose1_i$ est la même sauf que $mdm2$ devient aussi indifférent ce qui est cohérent avec le graphe ($p53\_mdm2$ est dissocié à l'étape 2).
- $apoptose2$ est le chemin le plus court et la première réponse positive est obtenue pour $apoptose2_5$; atm, $chk2$, dsb, $p53$ doivent être présents, et $mdm2$ et pml sont indifférents.
- $apoptose3$ demande le même nombre d'étapes que $apoptose2$ mais les conditions sont différentes : atm, $chk2$, dsb, et pml doivent être présents, $mdm2$ et $p53$ sont indifférents.

5.2 Mise à jour de graphe

Dans la figure 10 nous voyons à nouveau la représentation de l'opéron lac mais nous avons retiré l'inhibition exercée par le lactose sur la régulation négative du répresseur sur

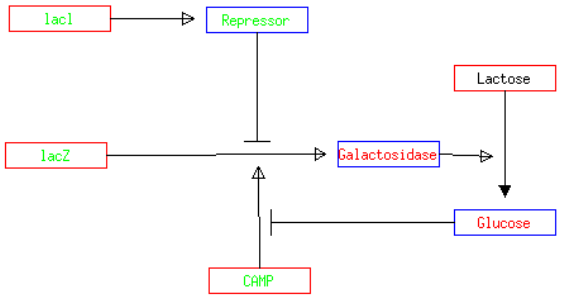

FIGURE 10 – Opéron lac sans l'inhibition par le lactose

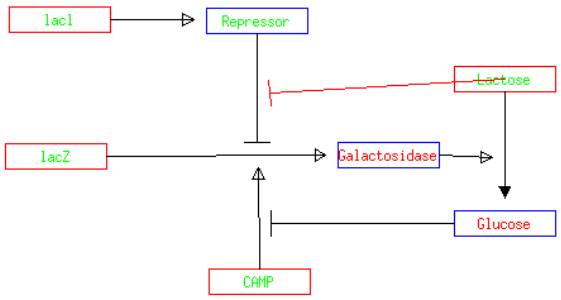

FIGURE 11 – Solution correcte

la production de la galactosidase. Dans ce cas, le glucose n'est pas produit en présence de lactose.

L'utilisateur peut demander au système quels ajouts peuvent être faits sur le graphe pour que le glucose soit produit. La solution exacte est trouvée instantanément (figure 11) avec une dizaine d'autres. Certaines sont sans intérêt, comme par exemple la production directe de glucose par un des gènes lacZ ou lacI.

En revanche le système propose aussi d'autres solutions, comme celle représentée sur la figure 12. Ici le glucose est nécessaire pour activer l'action inhibitrice de la protéine répresseur sur la réaction de production de la galactosidase. Donc, en présence de glucose, la production de galactosidase est stoppée, et elle est effectuée en l'absence de glucose. Il faut cependant noter que la nature a choisi la solution la plus "économique", puisque dans le cas proposé ici la galactosidase sera produite dès qu'il n'y a pas de glucose, qu'il y ait du lactose ou pas, alors que dans la solution standard, la galactosidase n'est produite qu'en l'absence de glucose et en présence de lactose.

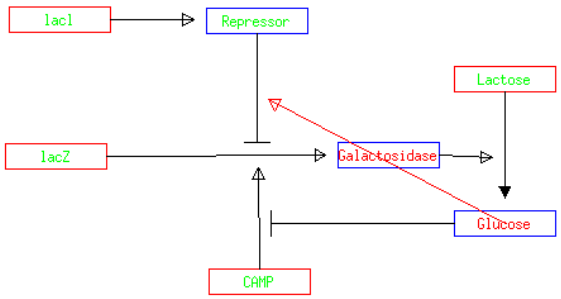

FIGURE 12 – Autre solution "intéressante"

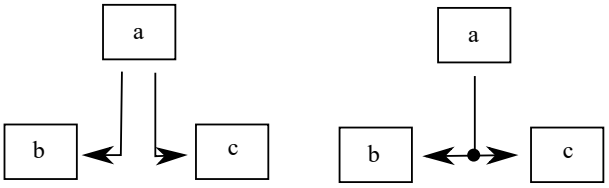

FIGURE 13 – Opérateur •

6 Conclusion

Nous avons présenté dans cet article une méthode pour traduire des graphes représentant des systèmes biologiques en logique temporelle, et un logiciel permettant de répondre à des questions complexes sur ces graphes. Le logiciel a aujourd'hui atteint le stade d'un prototype qui peut être testé sur des exemples réalistes de grande taille.

Il reste cependant un certain nombre de points à résoudre :

— Les graphes que nous utilisons sont limités par les relations élémentaires (productions, activation, inhibition) et la syntaxe limitée que nous autorisons dans les graphes. Ces relations élémentaires permettent de traduire des relations plus complexes (comme l'opérateur • qui permet par exemple de dupliquer une relation de production, d'activation ou d'inhibition, voir figure 13), mais dans l'état actuel du logiciel c'est l'utilisateur qui doit lui même décomposer les relations complexes en relations élémentaires. L'implantation d'un plus grand nombre de relations améliorerait l'expérience utilisateur.

— Les graphes sont créés par des biologistes, qui s'appuient sur leur "connaissance métier". Cette connaissance comprend souvent un ensemble de non-dits et d'implicites, qu'ils n'incluent pas dans les graphes. L'expérience montre que la construction formelle d'un graphe complexe comme celui de $atm\text{-}chk2$ (figure 9) a demandé de nombreuses itérations entre biologistes et informaticiens pour saisir la totalité de la connaissance nécessaire pour saisir la logique complète du système. Résoudre ce problème pourrait peut-être se faire à travers l'utilisation d'une interface graphique permettant de modifier directement le graphe et de le tester interactivement, alors qu'aujourd'hui toute modification de graphe impose de recourir à Pathvisio.

— La saisie des questions se fait à travers une interface

textuelle, et demande un minimum de compréhension de fonctionnement du système au niveau logique pour poser les "bonnes" questions. Là encore, le développement d'un outil graphique permettant de saisir les questions dans un format plus naturel serait largement appréciable.

— La vitesse des réactions n'est pas prise en compte. Une solution en cours d'étude est le passage au dual : au lieu de représenter la vitesse d'une réaction, nous pensons représenter la durée nécessaire pour qu'une réaction se déclenche.

— Le dernier point, plus complexe à résoudre, est que notre système repose sur l'hypothèse du "tout ou rien". Nous ne savons pas représenter les quantités intermédiaires, un acteur est soit présent, soit absent. Il s'agit à aussi d'un axe d'amélioration sur lequel nous travaillons.

Remerciements

Ce travail a bénéficié du soutien du projet ANR-11-LABX-0040-CIMI au sein du programme ANR-11-IDEX-0002-02, de l'IREP Associated European Laboratory et du projet CLE de la région Midi-Pyrénées .

Références

[ADD+16] Jean-Marc Alliot, Robert Demolombe, Martín Diéguez, Luis Fariñas del Cerro, Gilles Favre, Jean-Charles Faye, Naji Obeid, and Olivier Sordet. Temporal logic modeling of biological systems. In Seiki Akama, editor, *Towards Paraconsistent Engineering*, pages 205–226. Springer International Publishing, 2016.

[ADFdC16] Jean-Marc Alliot, Martín Diéguez, and Luis Fariñas del Cerro. Metabolic pathways as temporal logic programs. In Loizos Michael and Antonis Kakas, editors, *Logics in Artificial Intelligence*, pages 3–17. Springer International Publishing, 2016.

[CMD17] Serenella Cerrito, Marta Cialdea Mayer, and Robert Demolombe. Temporal abductive reasoning about biochemical reactions. *Journal of Applied Non-Classical Logics*, 27(3-4) :269–291, 2017.

[CRFS04] Nathalie Chabrier-Rivier, Francois Fages, and Sylvain Soliman. The Biochemical Abstract Machine BIOCHAM. In Vincent Danos and Vincent Schächter, editors, *CMSB'04 : Proceedings of the second Workshop on Computational Methods in Systems Biology*, volume 3082, pages 172–191, Paris, 2004. Springer-Verlag.

[ES03] N. Een and N. Sörensson. An extensible satsolver. In *SAT'03*, pages 502–518, 2003.

[FI14] L. Farinas and K. Inoue, editors. *Logical Modeling of Biological Systems*. John Wiley & Sons, 2014.

[GMP+11] V. Glorian, G. Maillot, S. Poles, J. S. Iacovoni, G. Favre, and S. Vagner. Hur-dependent loading of mirna risc to the mrna encoding the ras-related small gtpase rhob controls its translation during uv-induced apoptosis. *Cell Death & Differentiation*, 18(11) :1692–70, 2011.

[JM61] F. Jacob and J. Monod. Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology*, 3 :318–356, 1961.

[KAWP06] K. W. Kohn, M. I. Aladjem, J. N. Weinstein, and Y. Pommier. Molecular interaction maps of bioregulatory networks : A general rubric for systems biology. *Molecular Biology of the Cell*, 17(1) :1–13, 2006.

[KP05] K. W. Kohn and Y. Pommier. Molecular interaction map of the p53 and mdm2 logic elements, which control the off-on switch of p53 response to dna damage. *Biochemical and biophysical research communications*, 331(3) :816–27, 2005.

[KT90] A. Kean and G. Tsiknis. An incremental method for generating prime implicants/implicates. *Journal of Symbolic Computing*, 9 :185–206, 1990.

[LKLC07] W. Lee, D. Kim, M. Lee, and K. Choi. Identification of proteins interacting with the catalytic subunit of pp2a by proteomics. *Proteomics*, 7(2) :206–214, 2007.

[PSR+05] Y. Pommier, O. Sordet, V. A. Rao, H. Zhang, and K. W. Kohn. Targeting chk2 kinase : molecular interaction maps and therapeutic rationale. *Current pharmaceutical design*, 11(22) :2855–72, 2005.

[PZL+11] H. Pei, L. Zhang, K. Luo, Y. Qin, M. Chesi, F. Fei, P. L. Bergsagel, L. Wang, Z. You, and Z. Lou. MMSET regulates histone H4K20 methylation and 53BP1 accumulation at DNA damage sites. *Nature*, 470(7332) :124–128, 2011.

[vIKP+08] M. P. van Iersel, T. Kelder, A. R. Pico, K. Hanspers, S. Coort, B. R. Conklin, and C. Evelo. Presenting and exploring biological pathways with pathvisio. *BMC Bioinformatics*, page 399, 2008.