

## Langages formels dans la machine abstraite biochimique BIOCHAM

Laurence Calzone — Nathalie Chabrier-Rivier — François Fages  
Loïc Fosse — Sylvain Soliman

INRIA Rocquencourt - Projet CONTRAINTES  
Domaine de Voluceau, Rocquencourt, BP 105  
F-78153 Le Chesnay cedex  
{Prenom.Nom}@inria.fr

---

*RÉSUMÉ.* Le développement de langages formels pour modéliser les systèmes biologiques ouvre la voie à la conception de nouveaux outils de raisonnement automatique destinés au biologiste modélisateur. La machine abstraite biochimique BIOCHAM est un environnement logiciel qui offre un langage simple de règles pour modéliser des interactions biomoléculaires, et un langage puissant fondé sur la logique temporelle pour formaliser les propriétés biologiques du système. En s'appuyant sur ces deux langages formels, il devient possible d'utiliser des techniques d'apprentissage automatique pour inférer de nouvelles règles de réaction, estimer les valeurs des paramètres cinétiques, et corriger ou compléter les modèles semi-automatiquement. Dans cet article, nous décrivons les langages implantés dans BIOCHAM et illustrons l'utilisation du système d'apprentissage automatique sur un modèle simple du contrôle du cycle cellulaire.

*ABSTRACT.* With the advent of formal languages for modeling biological systems, the design of automated reasoning tools to assist the biologist becomes possible. The biochemical abstract machine BIOCHAM software environment offers a rule-based language to model bio-molecular interactions and a powerful temporal logic-based language to formalize the biological properties of the system. Building on these two formal languages, machine learning techniques can be used to infer new molecular interaction rules from temporal properties, or to estimate kinetic parameter values, in order to semi-automatically correct or complete models from observed biological properties of the system. In this article, we describe the formal languages of BIOCHAM and illustrate, on a simple cell cycle control model, the use of the machine learning system.

*MOTS-CLÉS :* biologie des systèmes, logique temporelle, apprentissage, cycle cellulaire.

*KEYWORDS:* systems biology, temporal logics, machine learning, cell cycle.

---

## 1. Introduction

La production en masse de données post-génomiques, telles que l'expression des ARN, la production de protéines et les interactions protéine-protéine, donne plus d'acuité aux besoins et aux difficultés, qu'il y a à raisonner à l'échelle d'un système complexe d'interactions biomoléculaires. Les connaissances concernant les réseaux d'influence géniques et les réseaux d'interactions protéiques sont intégrées dans des bases de données comme KEGG (Kanehisa *et al.*, 2000), BioCyc (Keseler *et al.*, 2005)... et présentées généralement sous forme de diagrammes annotés. Les outils comme BioSpice (Giaever *et al.*, 2002), Copasi (VBI, n.d.), GON, E-cell (Tomita *et al.*, 2000) etc. ont été développés pour faire des simulations numériques lorsque les données cinétiques sont connues.

Cependant, au-delà de la simulation, la possibilité de faire des calculs symboliques sur les réseaux d'interactions moléculaires ouvre la voie à la réalisation de nouveaux outils de raisonnement automatique destinés au biologiste modélisateur. Notre projet avec la Machine Abstraite Biochimique BIOCHAM<sup>1</sup> (Fages *et al.*, 2004), qui a débuté en 2002, est une tentative dans cette direction. BIOCHAM est fondé sur un langage de composés moléculaires et de règles de réaction, qui fournit des sémantiques précises aux diagrammes d'interactions biomoléculaires, à trois niveaux d'abstraction :

1) la sémantique booléenne, qui permet de raisonner sur la présence ou l'absence des molécules, associe à un ensemble de règles de réaction un système de transitions concurrent asynchrone ;

2) la sémantique des concentrations, qui permet de raisonner sur la dynamique continue du système, associe aux molécules des valeurs réelles représentant leur concentration, et aux règles un système d'équations différentielles (non linéaires) du premier ordre ;

3) la sémantique des populations, qui permet de raisonner sur les petits nombres de molécules, associe aux molécules une valeur entière représentant leur quantité, et aux règles une chaîne de Markov en temps continu.

Dans les trois cas, nous utilisons la logique temporelle comme langage de formalisation des propriétés biologiques du système. Dans la sémantique booléenne, la logique des arbres de calcul CTL (Clarke *et al.*, 1999) permet d'exprimer un large spectre de propriétés biologiques telles que l'accessibilité, les points de passage obligé, la stabilité ou les oscillations (Chabrier *et al.*, 2003). Nous avons montré le passage à l'échelle des algorithmes de *model-checking* dans ce contexte, sur un modèle booléen du contrôle du cycle cellulaire des mammifères, développé d'après le diagramme de Kohn (Kohn, 1999) et impliquant environ 500 variables et 800 règles de réaction (Chabrier-Rivier *et al.*, 2004a). Dans la sémantique des concentrations, la logique LTL (Clarke *et al.*, 1999) avec contraintes numériques permet d'exprimer des propriétés temporelles quantitatives. Dans la sémantique des populations, la logique

1. BIOCHAM est un logiciel distribué sous licence GPL à l'adresse <http://contraintes.inria.fr/BIOCHAM/>

PCTL (Hansson *et al.*, 1994; Kwiatkowska *et al.*, 2004) avec contraintes numériques permet d'exprimer les probabilités de réalisation de propriétés temporelles quantitatives.

La vérification automatique des propriétés d'un système biologique formalisées en logique temporelle, permet non seulement d'interroger un modèle, mais aussi d'aider à le corriger, en prenant ces propriétés comme une spécification et en cherchant automatiquement des règles et des paramètres permettant de la satisfaire (Calzone *et al.*, 2005). Dans cet article, nous décrivons les deux méthodes d'apprentissage implantées dans BIOCHAM. La première, qui utilise la sémantique booléenne, concerne la recherche de règles de réaction à ajouter ou retirer au modèle de façon à satisfaire une spécification CTL. La seconde, qui utilise la sémantique des concentrations, concerne la recherche de paramètres cinétiques permettant de satisfaire une spécification LTL avec contraintes. Les deux méthodes sont illustrées sur un modèle simple du contrôle du cycle cellulaire.

### **Travaux reliés**

Les travaux pionniers de (Regev *et al.*, 2001), sur l'utilisation du  $\pi$ -calcul de Milner pour modéliser les voies de signalisation dans la cellule, ont été la source d'inspiration de nombreux travaux ultérieurs dans la lignée des calculs de processus (Cardelli, 2004; Regev *et al.*, 2004; Danos *et al.*, 2004) et de leurs extensions stochastiques (Phillips *et al.*, to appear). Le langage de règles de BIOCHAM se veut une simplification de cette démarche utilisant les calculs de processus, au profit d'un formalisme simple de règles de réaction qui, d'une part est très naturel pour le biologiste, et d'autre part se prête mieux aux techniques d'analyse par *model-checking*. Dans la sémantique booléenne, ce langage de règles est similaire aux règles de réécriture de *Pathway Logic* implantées dans le système Maude (Eker *et al.*, 2002). Dans la sémantique des concentrations, ce langage est proche du *bio-calculus* (Nagasaki *et al.*, 1999; Nagasaki *et al.*, 2000).

L'utilisation de la logique temporelle et des techniques de *model-checking* en biologie des systèmes est quant à elle une idée récente. Elle apparaît dans (Eker *et al.*, 2002; Chabrier *et al.*, 2003) pour les modèles booléens, dans (Bernot *et al.*, 2004; Batt *et al.*, 2004; Calder *et al.*, 2005) pour les modèles discrets et dans (Antonioti *et al.*, 2003) pour les modèles continus.

Les techniques d'apprentissage automatique comme la programmation logique inductive (Muggleton, 1995) ou la programmation génétique, ont été utilisées pour inférer des fonctions de gènes (Bryant *et al.*, 2001), des descriptions de voies métaboliques (Angelopoulos *et al.*, 2002a; Angelopoulos *et al.*, 2002b; Koza *et al.*, 2001) ou des interactions entre gènes (Bernot *et al.*, 2004). Nos travaux se rapprochent du domaine de la découverte scientifique qualitative et numérique (Langley *et al.*, 1987) et de la révision de théories (Todorovski *et al.*, 2001; de Raedt, 1992). Cependant, l'appren-

tissage à partir de formules temporelles est un sujet nouveau, que ce soit du point de vue de l'apprentissage automatique, ou de celui de la biologie des systèmes.

## 2. Langage de description de processus biomoléculaires

Les règles de réaction BIOCHAM représentent des réactions entre des objets formels représentant des composés chimiques ou biochimiques, qui s'étendent des petites molécules aux gènes et aux protéines.

La syntaxe des molécules et des réactions peut être donnée par la grammaire (simplifiée) suivante :

```

objet      =  molecule | molecule : lieu
molecule =  nom | molecule-molecule | molecule~{nom,...,nom}
reaction   =  solution => solution | cinetique for solution => solution
solution   =  _ | objet | nombre*objet | solution + solution

```

L'objet de base est un composé moléculaire. Il peut être localisé avec l'opérateur : en indiquant un lieu, qui est un simple nom représentant un compartiment tel que le noyau, le cytoplasme, la membrane, etc. L'opérateur de liaison - sert à représenter la complexation ou d'autres formes de liaison de molécules. L'opérateur de modification ~ permet d'attacher un ensemble de modifications à une molécule, tel que l'ensemble des sites phosphorylés d'une protéine. Pour les règles de réaction, les abréviations suivantes peuvent être utilisées :  $A = [C] \Rightarrow B$  pour la règle  $A + C \Rightarrow B + C$  avec le catalyseur  $C$ , et  $A \Leftrightarrow B$  pour les deux réactions symétriques. Par exemple,  $M + N \Rightarrow M - N$  représente une règle de complexation,  $M = [N] \Rightarrow M \sim \{s\}$  une règle de phosphorylation catalysée par la kinase  $N$ , et  $M : L \Rightarrow M : L'$  représente une règle de transport de la molécule  $M$ , du lieu  $L$  en  $L'$ . La notation  $_$  représente la solution vide. Les règles de dégradation peuvent ainsi s'écrire  $A \Rightarrow _$  ou  $A = [C] \Rightarrow _$  et les règles de synthèse  $_ \Rightarrow A$  ou  $_ = [C] \Rightarrow A$ , selon le degré de détail (protéase, ARNm, gène, etc.) souhaité.

Dans tous les cas, les règles peuvent être données avec une expression cinétique, correspondant par exemple à la loi d'action de masse, aux fonctions de Michaelis-Menten, de Hill, ou à toute autre fonction mathématique comprenant éventuellement des expressions conditionnelles.

BIOCHAM possède aussi un langage de schémas avec contraintes (Chabrier-Rivier *et al.*, 2004b) pour dénoter de manière concise des ensembles de réactions. Un schéma de réaction est une règle de réaction dans laquelle certains objets ont été remplacés par des variables, notées \$A, \$B, équipée de contraintes sur les valeurs possibles de ces variables. Par exemple, la contrainte \$A diff \$B impose que les deux molécules soient différentes, \$A more\_phos\_than \$B impose que \$A soit plus phosphorylée que \$B. La description précise de ces schémas dépasse cependant le cadre de cet article et est détaillée dans (Chabrier *et al.*, 2003-2006). Ces schémas sont

entre autres utilisés pour indiquer la forme des règles à apprendre comme dans la section 4.1.

**Exemple 1.** *Le cycle cellulaire décrit le mécanisme biomoléculaire par lequel une cellule est amenée à se diviser et proliférer. Il s'agit d'une séquence d'événements qui inclut la réplication de l'ADN (phase S), la mitose (phase M) et l'apparition et la disparition successives de diverses protéines comme les cyclines. Pour être actives, ces cyclines s'associent à d'autres protéines, des kinases cyclines-dépendantes. Les complexes ainsi formés contrôlent le déroulement du cycle.*

*L'exemple BIOCHAM ci-dessous est un modèle simple du cycle cellulaire tiré de (Tyson, 1991). Il rend compte des interactions existant entre la kinase Cdc2 et une cycline. Les deux molécules se complexent en un hétérodimère qui s'active et s'inactive par des mécanismes de phosphorylation et de déphosphorylation. Si la cycline est de type B, la forme active du complexe, Cdc2-Cyclin<sup>p1</sup>, aussi appelée MPF pour M-phase Promoting Factor, joue un rôle central dans le cycle mitotique.*

```

R1: k1                                for _=>Cyclin.
R2: k2*[Cyclin]                        for Cyclin=>_.
R3: k3*[Cyclin]*[Cdc2p1] for Cyclin+Cdc2p1=>Cdc2p1-Cyclinp1.
R4: k4p*[Cdc2p1-Cyclinp1]
    for Cdc2p1-Cyclinp1=>Cdc2-Cyclinp1.
R5: k4*[Cdc2-Cyclinp1]2*[Cdc2p1-Cyclinp1]
    for Cdc2p1-Cyclinp1=[Cdc2-Cyclinp1]=>Cdc2-Cyclinp1.
R6: k5*[Cdc2-Cyclinp1] for Cdc2-Cyclinp1=>Cdc2p1-Cyclinp1.
R7: k6*[Cdc2-Cyclinp1] for Cdc2-Cyclinp1=>Cdc2+Cyclinp1.
R8: k7*[Cyclinp1] for Cyclinp1=>_.
R9: k8*[Cdc2] for Cdc2=>Cdc2p1.
R10: k9*[Cdc2p1] for Cdc2p1=>Cdc2.

parameter(k1,0.015). parameter(k2,0.015). parameter(k3,200).
parameter(k4p,0.018). parameter(k4,180). parameter(k5,0).
parameter(k6,1). parameter(k7,0.6). parameter(k8,100).
parameter(k9,100).

macro (YT,Cyclin+Cyclinp1+Cdc2p1-Cyclinp1+Cdc2-Cyclinp1).

present (Cdc2,1). make_absent_not_present.

```

*Le modèle ci-dessus inclut une liste de réactions biochimiques qui regroupe des règles de synthèse, dégradation, association, dissociation et (dé)phosphorylation de protéines. Chacune de ces règles est ici donnée avec une fonction cinétique et un nom (R1, R2, ...). Les règles décrivent les processus suivants : la cycline (Cyclin) est synthétisée à une vitesse k1 (R1) et dégradée à une vitesse k2 (R2); l'association entre Cyclin et la forme phosphorylée de Cdc2 (Cdc2<sup>p1</sup>) se fait avec une affinité k3 (R3), ..., le complexe actif Cdc2-Cyclin<sup>p1</sup> entraîne sa propre activation (R5), etc.*

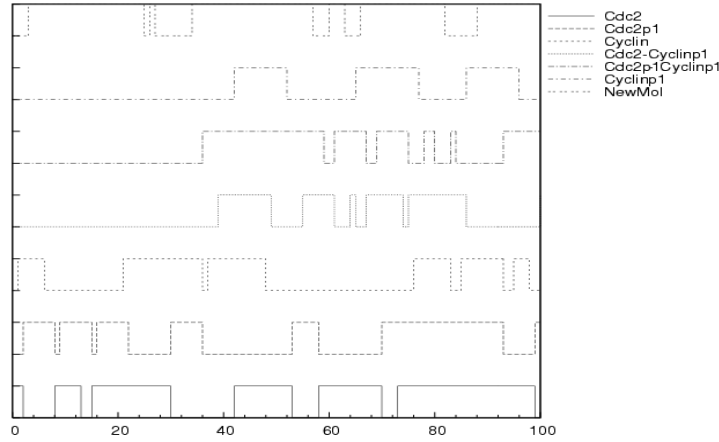
*La fonction `parameter` affecte des valeurs aux paramètres. La fonction `macro` définit ici `YT` comme étant égale à la somme de toutes les formes de `Cyclin`. Enfin l'état initial est défini par la présence de `Cdc2` (avec une concentration initiale égale à 1) et l'absence de tous les autres composés (la fonction `make_absent_not_present` mettant leur concentration initiale à zéro).*

La sémantique des règles BIOCHAM est définie à trois niveaux d'abstraction qui correspondent à trois formes de raisonnement sur les systèmes biologiques : la sémantique booléenne, la sémantique des concentrations en nombres réels, et la sémantique des populations de molécules en nombres entiers.

### 2.1. Sémantique booléenne

Dans la sémantique booléenne, on associe à chaque objet une variable booléenne qui représente sa présence ou absence dans le système, et les règles de réaction sont interprétées comme un système de transitions concurrent asynchrone. Une règle comme  $A+B \Rightarrow C+D$  définit quatre transitions possibles qui correspondent à la consommation totale ou non des réactifs  $A$  et  $B$  par la réaction. Pour que la règle soit applicable, les molécules  $A$  et  $B$  doivent être présentes dans l'état courant du système. Dans l'état suivant les molécules  $C$  et  $D$  sont présentes tandis que les molécules  $A$  et  $B$  peuvent être absentes (consommation totale) ou présentes (consommation partielle). Ceci se fait de façon non déterministe afin de rendre compte de tous les comportements possibles. Le système de transitions est asynchrone dans le sens où une seule transition est appliquée à la fois, ce qui permet de représenter les phénomènes biologiques fondamentaux de masquage d'une réaction par une autre, comme par exemple dans le cas de certaines inhibitions. A l'inverse, l'hypothèse de synchronie (dans laquelle toutes les règles applicables se déclenchent en même temps) ne permettrait pas de représenter correctement les phénomènes de compétition entre réactions.

Formellement, la sémantique booléenne d'un modèle BIOCHAM est définie par une *structure de Kripke*  $K = (S, R)$ , où  $S$  est l'ensemble des états définis par le vecteur de variables booléennes,  $R \subseteq S \times S$  est la relation de transition entre états supposée totale (c'est-à-dire que pour tout état  $s \in S$ , il existe un état  $s' \in S$  tel que  $(s, s') \in R$ ). Lorsque la relation de transition provenant uniquement des réactions BIOCHAM n'est pas totale, elle est automatiquement complétée par une transition qui ajoute une boucle sur les états n'ayant pas de successeur. Un chemin dans  $K$  partant de l'état  $s_0$  est une suite infinie d'états  $\pi = s_0, s_1, \dots$  telle que  $(s_i, s_{i+1}) \in R$  pour tout  $i \geq 0$ .  $\pi^k$  désignera le chemin  $s_k, s_{k+1}, \dots$ . L'évolution temporelle du système est modélisée par la succession des étapes de transition, et les différents comportements du système sont modélisés par le choix non déterministe des transitions. La figure 1 montre une simulation booléenne du modèle du cycle cellulaire.



**Figure 1.** Trace d'une simulation booléenne du modèle du cycle cellulaire de l'exemple 1

## 2.2. Sémantique des concentrations

Dans la sémantique des concentrations, on associe à chaque objet une variable réelle qui représente sa concentration, et les règles sont interprétées par un système d'équations différentielles. Formellement, à un ensemble de règles BIOCHAM  $E = \{e_i \text{ for } S_i \Rightarrow S'_i\}_{i=1,\dots,n}$  sur un ensemble de variables  $\{x_1, \dots, x_m\}$ , on associe le système d'équations différentielles

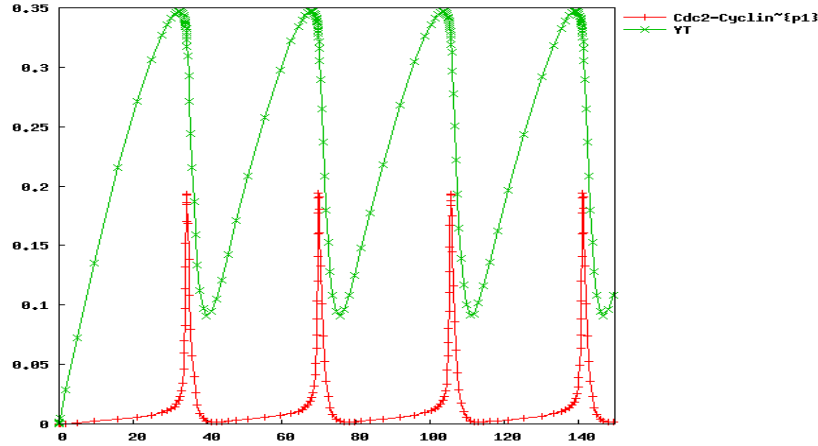
$$dx_k/dt = \sum_{i=1}^n r_i(x_k) * e_i - \sum_{j=1}^n l_j(x_k) * e_j$$

où  $r_i(x)$  (resp.  $l_i$ ) est le coefficient stoechiométrique de  $x$  dans le membre droit (resp. gauche) de la règle  $i$ .

Etant donné un état initial défini par les concentrations des objets, le comportement du système peut être simulé par intégration numérique du système différentiel. La figure 2 illustre, dans l'exemple 1, les oscillations de la concentration de Cdc2-Cyclin~{p1}, et de la cycline sous toutes ses formes (YT).

## 2.3. Sémantique des populations

Dans la sémantique des populations, on associe à chaque objet une variable entière qui représente le nombre de molécules, et les règles sont interprétées par une chaîne de Markov en temps continu dans laquelle les taux de transition sont définis par les expressions cinétiques des règles BIOCHAM.



**Figure 2.** Simulation numérique du modèle du cycle cellulaire de l'exemple 1, montrant la concentration de la forme active du complexe Cdc2-Cyclin<sup>p1</sup>, et la concentration totale de la cycline sous toutes ses formes (macro YT)

Les méthodes de simulation stochastique (Gillespie, 1976) calculent des réalisations du processus. Ces courbes sont en général des versions bruitées des courbes obtenues dans la sémantique des concentrations. Cependant il est préférable d'utiliser l'approche stochastique dans les modèles faisant intervenir des petits nombres de molécules. En effet, des comportements qualitativement différents peuvent être observés par rapport à l'approche continue. Un exemple classique de ce phénomène est le modèle du phage lambda dans lequel un petit nombre de molécules, facteurs de promotion de deux gènes, engendrent un phénomène de multiplication explosive (lyse) après une période d'attente passive plus ou moins longue (lysogénie) (Gibson *et al.*, 2000).

Dans la sémantique stochastique, pour un volume  $V$  donné, les concentrations sont donc converties en nombre de molécules ( $N = C \times V \times K$ ,  $K$  constante d'Avogadro). Les coefficients cinétiques  $k$  sont convertis en taux de transition  $\tau$  (fournissant une probabilité de transition après normalisation) comme suit (Gibson *et al.*, 2000) :

$$\tau_i = e_i \times (V_i \times K)^{(1 - \sum_{k=1}^m l_i(x_k))} \times \prod_{k=1}^m (l_i(x_k))$$

où  $l_i$  est le coefficient stoechiométrique du réactant  $x_k$  dans la réaction  $i$ . On a en particulier :

- $\tau = k$  pour les réactions du type  $A \Rightarrow \dots$ ,
- $\tau = \frac{k}{V \times K}$  pour les réactions du type  $A+B \Rightarrow \dots$ ,
- $\tau = 2 \times \frac{k}{V \times K}$  pour les réactions du type  $A+A \Rightarrow \dots$ ,

- $\tau = \frac{k}{(V \times K)^2}$  pour les réactions du type  $A+B+C \Rightarrow \dots$ ,
- etc.

### 3. Langage de description de propriétés biologiques

Les modèles biologiques sont construits à partir d'expériences réalisées sur des organismes sauvages ou mutés. Ces expériences définissent les propriétés biologiques que les modèles doivent satisfaire. Par exemple, les expériences sur les organismes mutés permettent de discriminer entre différentes hypothèses d'interaction.

Le trait le plus original de BIOCHAM est l'utilisation de la logique temporelle comme langage de formalisation de ces propriétés biologiques. Les logiques CTL booléenne, LTL et PCTL avec contraintes numériques sont utilisées respectivement dans chacun des trois niveaux sémantiques. Elles permettent de formaliser les propriétés importantes d'un modèle, qui aujourd'hui sont expliquées de façon informelle dans les articles de modélisation. Leur formalisation permet cependant de les considérer comme une spécification devant être préservée lors des opérations de raffinement, de simplification ou de composition de modèles. La possibilité de vérifier automatiquement une telle spécification permet d'envisager la comparaison et la réutilisation de différents modèles de façon beaucoup plus systématique que ce qui se pratique aujourd'hui dans le domaine.

#### 3.1. Logique CTL pour la sémantique booléenne

La logique temporelle des arbres de calcul (*Computation Tree Logic CTL\**) (Clarke *et al.*, 1999) est une extension de la logique classique qui permet de raisonner sur un arbre de transitions d'états à l'aide d'opérateurs sur les choix de branches (non-déterminisme) et sur le temps (transitions d'états). Deux quantificateurs de chemins sont introduits pour raisonner sur le non-déterminisme :  $A\phi$  qui signifie que  $\phi$  est vraie sur tous les chemins, et  $E\phi$ , pour  $\phi$  est vraie sur au moins un chemin. Plusieurs opérateurs temporels sont introduits :  $X\phi$  signifiant que  $\phi$  est vraie à la transition suivante,  $G\phi$  pour  $\phi$  est toujours vraie,  $F\phi$  pour  $\phi$  finit par devenir vraie,  $\phi U \psi$  pour  $\psi$  finit par devenir vraie et  $\phi$  est vraie jusqu'à ce que  $\psi$  soit vraie,  $\phi W \psi$  pour  $\phi$  est vraie jusqu'à ce qu'éventuellement  $\psi$  devienne vraie. Le tableau 1 définit la relation de vérité d'une formule dans une structure de Kripke  $K = (S, R)$  dont les états sont définis par des variables booléennes. Dans cette logique, en notant ! la négation et | la disjonction,  $F\phi$  est équivalent à  $true U \phi$ ,  $\phi W \psi$  est équivalent à  $(\phi U \psi) \mid G\phi$  et nous avons les propriétés de dualité suivantes :  $!EF(\phi) = AG(!\phi)$ ,  $!E\phi U \psi = A!\psi W !\phi$  et  $!E\phi W \psi = A!\psi U !\phi$ .

Pour des raisons de calculabilité pratique cependant, nous nous restreignons dans la suite au fragment CTL de CTL\*, dans lequel chaque opérateur temporel doit obligatoirement être précédé d'un quantificateur de chemin, et un quantificateur de chemin doit obligatoirement être suivi d'un opérateur temporel.

$s \models \alpha$	ssi	$\alpha$ est une formule propositionnelle satisfaite dans l'état $s$ ,
$s \models E\psi$	ssi	il existe un chemin $\pi$ partant de $s$ tel que $\pi \models \psi$ ,
$s \models A\psi$	ssi	pour tout chemin $\pi$ partant de $s$ , $\pi \models \psi$ ,
$s \models !\psi$	ssi	$s \not\models \psi$ ,
$s \models \psi \& \psi'$	ssi	$s \models \psi$ et $s \models \psi'$ ,
$s \models \psi \mid \psi'$	ssi	$s \models \psi$ ou $s \models \psi'$ ,
$s \models \psi \Rightarrow \psi'$	ssi	$s \models \psi'$ ou $s \not\models \psi$ ,
$\pi \models \psi$	ssi	$s \models \psi$ où $s$ est l'état de début de $\pi$ ,
$\pi \models X\psi$	ssi	$\pi^1 \models \psi$ ,
$\pi \models F\psi$	ssi	il existe $k \geq 0$ t. q. $\pi^k \models \psi$ ,
$\pi \models G\psi$	ssi	pour tout $k \geq 0$ , $\pi^k \models \psi$ ,
$\pi \models \psi U \psi'$	ssi	il existe $k \geq 0$ t. q. $\pi^k \models \psi'$ et $\pi^j \models \psi$ pour tout $0 \leq j < k$ .
$\pi \models \psi W \psi'$	ssi	il existe $k \geq 0$ t. q. $\pi^k \models \psi'$ et $\pi^j \models \psi$ pour tout $0 \leq j < k$ . ou pour tout $k \geq 0$ , $\pi^k \models \psi$ ,
$\pi \models \psi \& \psi'$	ssi	$\pi \models \psi$ et $\pi \models \psi'$ ,
$\pi \models \psi \mid \psi'$	ssi	$\pi \models \psi$ ou $\pi \models \psi'$ ,
$\pi \models !\psi$	ssi	$\pi \not\models \psi$ ,
$\pi \models \psi \Rightarrow \psi'$	ssi	$\pi \models \psi'$ ou $\pi \not\models \psi$ ,

**Tableau 1.** Définition inductive de la relation de vérité d'une formule CTL\* dans un état ou un chemin, dans une structure de Kripke  $K$

Un modèle BIOCHAM  $\mathcal{M}$  est constitué d'un ensemble de règles et d'une spécification (partielle) de l'ensemble des états initiaux. Les règles de  $\mathcal{M}$  définissent la structure de Kripke associé au modèle, et on note  $\mathcal{M} \models \phi$  si la formule  $\phi$  est vraie dans tous les états initiaux spécifiés par  $\mathcal{M}$ .

### 3.1.1. Expressivité

Comme montré dans (Chabrier *et al.*, 2003), le fragment CTL a une expressivité suffisante pour exprimer un très large éventail de propriétés biologiques :

– **sur l'accessibilité.** Existe-t-il un chemin pour produire (*i.e.* synthétiser, activer etc.) une protéine  $P$  ? Cette requête est formalisée par la formule CTL  $EF(P)$ , abrégée en `reachable(P)` dans BIOCHAM,

– **sur les voies.** Est-ce que l'état  $Q$  est un point de passage obligé pour atteindre les états où  $P \sim \{s\}$  est présent ? La formule correspondante  $!(E((!(Q)UP \sim \{s\})))$  est abrégée en `checkpoint(Q, P ~ {s})`,

– **sur la stationnarité et la stabilité.** Est-ce qu'un état (ou un ensemble d'états) de la cellule décrit par une formule  $s$  est un état stationnaire (resp. stable) ?  $s \Rightarrow EG(s)$  (resp.  $s \Rightarrow AG(s)$ ), abrégé en `steady(s)` (resp. `stable(s)`),

– **sur les oscillations.** Le système peut-il exhiber un comportement cyclique sur la présence de  $P$ ?  $EG((P \Rightarrow EF !P) \wedge (!P \Rightarrow EF P))$ , abrégé en  $oscil(P)$ <sup>2</sup>.

**Exemple 2.** *Le modèle du cycle cellulaire avec l'état initial de l'exemple précédent vérifie entre autres les propriétés suivantes :*

```

reachable(Cdc2~{p1})
reachable(Cyclin)
reachable(Cyclin~{p1})
reachable(Cdc2-Cyclin~{p1})
reachable(Cdc2~{p1}-Cyclin~{p1})
oscil(Cdc2)
oscil(Cdc2~{p1})
oscil(Cdc2~{p1}-Cyclin~{p1})
oscil(Cyclin)
AG((!(Cdc2-Cyclin~{p1})) ->
  checkpoint(Cdc2~{p1}-Cyclin~{p1}, Cdc2-Cyclin~{p1}))

```

*Toutes les molécules sont accessibles (reachable), leur activité oscille (oscil(A)), et sur tous les chemins possibles, à tout moment (AG), en l'absence du complexe actif (!(Cdc2-Cyclin~{p1})), il est nécessaire de produire la forme doublement phosphorylée Cdc2~{p1}-Cyclin~{p1} avant l'activation de Cdc2-Cyclin~{p1}.*

### 3.1.2. Algorithme de model-checking symbolique

Le langage de requête CTL pour les modèles booléens est implanté en BIOCHAM avec une interface au model-checker symbolique NuSMV (Cimatti *et al.*, 2002) qui permet de vérifier une formule CTL dans un état initial partiellement défini, en utilisant une représentation du problème par des diagrammes de décision binaires ordonnés.

Les performances obtenues sur un gros modèle du contrôle du cycle cellulaire suivant la carte de Kohn (Kohn, 1999), comprenant 800 règles et 500 variables, sont de l'ordre de quelques dizaines de secondes pour compiler le modèle, ou pour vérifier des formules CTL simples (Chabrier-Rivier *et al.*, 2004a). Cependant ces performances sont en deçà de celles obtenues classiquement par NuSMV avec des modèles fortement structurés pour la vérification de circuits ou de programmes. Les modèles booléens de BIOCHAM ont en effet la caractéristique d'être très fortement non déterministes, ce qui peut conduire à des problèmes de performance sur des modèles de plus petite taille comprenant un grand nombre de voies parallèles, comme c'est par exemple le cas dans le modèle de la voie de signalisation MAPK de (Schoeberl *et al.*, 2002).

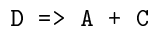
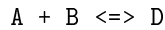
---

2. La formule CTL que nous associons aux oscillations n'est cependant qu'une approximation de cette propriété. La formule est en effet impliquée par l'existence d'oscillations mais n'implique pas l'alternance infinie d'états de types  $P$  et  $!P$  (sans hypothèse d'équité forte). La propriété d'oscillation s'exprime en CTL\* par la formule  $EG((P \Rightarrow F !P) \wedge (!P \Rightarrow F P))$  qui n'a pas d'équivalent dans CTL (Batt, 2006).

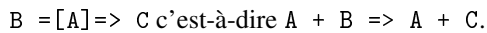
### 3.1.3. Équivalence CTL de modèles

Il existe, en général, de nombreuses manières de modéliser un même système, suivant le niveau de détail souhaité et l'information disponible. Même pour une simple réaction enzymatique, il existe déjà deux façons classiques de la coder en BIOCHAM :

– soit en détaillant la formation d'un complexe intermédiaire



– soit directement



Les deux propriétés ci-dessous montrent l'équivalence de ces deux modèles vis-à-vis de certaines propriétés CTL. Dans ces propositions, nous supposons que le complexe D n'apparaît que dans les deux règles du premier modèle. Les structures de Kripke associées à ces deux modèles définissent deux relations de vérité :  $\models_1$  et  $\models_2$ . On note  $I$  l'état initial.

**Proposition 1. (Accessibilité)** (Chabrier-Rivier et al., 2004b)

Soit  $\phi$  une formule CTL atomique, si  $I \models_2 EF(\phi)$  alors  $I \models_1 EF(\phi)$ . De plus, si A et B n'apparaissent pas négativement (i.e. sous un nombre impair de négations) dans  $\phi$  et si D n'apparaît pas du tout dans  $\phi$ , alors  $I \models_1 EF(\phi)$  implique  $I \models_2 EF(\phi)$ .

**Proposition 2. (Point de passage obligé)** (Chabrier-Rivier et al., 2004b)

Soit  $!E(!\phi U \psi)$  une formule de point de passage obligé (checkpoint) où  $\phi$  et  $\psi$  sont des formules atomiques décrivant des états. Si A et B n'apparaissent pas négativement dans  $\psi$  et D n'apparaît pas du tout dans  $\psi$ , alors  $I \models_2 !E(!\phi U \psi)$  implique  $I \models_1 !E(!\phi U \psi)$ .

De plus, si A et B n'apparaissent pas négativement dans  $\phi$  et D n'apparaît pas du tout dans  $\phi$ , alors  $I \models_1 !E(!\phi U \psi)$  implique  $I \models_2 !E(!\phi U \psi)$ .

## 3.2. LTL avec contraintes numériques pour la sémantique des concentrations

Une extension de la logique temporelle du temps linéaire (*Linear Time Logic*, LTL) avec contraintes arithmétiques est utilisée pour la sémantique des concentrations moléculaires. Cette approche est similaire à celle décrite dans (Antonioti et al., 2003) et utilisée dans le projet DARPA BioSpice.

LTL est un fragment de la logique temporelle, qui utilise uniquement les opérateurs temporels (pas les quantificateurs de chemin), et est donc appropriée pour raisonner sur les systèmes déterministes que sont les modèles cinétiques. Dans LTL avec contraintes arithmétiques, les formules atomiques sont des formules de logique du premier ordre avec égalité, inégalité et opérations arithmétiques. Ces formules portent sur les valeurs réelles des concentrations, par exemple  $[P] > [P \sim \{s\}]$ , ainsi que celles de leurs dérivées, par exemple  $d([P])/dt < 0$ .

### 3.2.1. Expressivité

Les propriétés d'*accessibilité* sont formalisées par l'opérateur  $F$ . Par exemple, la propriété que la concentration de la protéine  $P$  phosphorylée sur  $s$  peut être, à un moment, dépassée par celle de la protéine  $P$ , s'exprime par la formule  $F([P] > [P \sim \{s\}])$ . La propriété que la concentration de  $P$  atteint et reste au-dessus d'un certain seuil  $c$  s'exprime par  $FG([P] > c)$ .

Les propriétés d'*oscillation* d'une concentration  $[P]$  un nombre  $n$  de fois, portent sur le nombre d'alternances du signe de la dérivée. Elles s'expriment par la formule

$$F((d[P]/dt > 0) \wedge F((d[P]/dt < 0) \wedge F((d[P]/dt > 0) \dots)))$$

abrégée par `oscil(P, n)`.

### 3.2.2. Algorithme de model-checking avec contraintes

Sous l'hypothèse que les conditions initiales sont complètement déterminées, les méthodes d'intégration numérique du système d'équations différentielles associées aux règles BIOCHAM, fournissent une trace de simulation discrète (à pas de temps variable). Une telle trace peut être vue comme une structure de Kripke linéaire (*i.e.* ayant un seul chemin partant de chaque état), dans laquelle il est possible d'interpréter les formules LTL.

De façon à poser des contraintes sur non seulement les valeurs des concentrations mais aussi sur leurs dérivées, on considère des traces de simulation de la forme

$$\langle t_0, x_0, dx_0/dt \rangle, \langle t_1, x_1, dx_1/dt \rangle, \dots$$

qui contiennent pour chaque point de temps  $t_i$  (calculé par l'intégration numérique), la valeur des concentrations  $x_i$  et de leurs dérivées  $dx_i/dt$ .

Les traces de simulation étant cependant calculées sur un horizon fini, les propriétés temporelles sont vérifiées sur ce même horizon. L'algorithme de *model-checking* avec contraintes pour vérifier une formule  $\phi$  consiste à effectuer les opérations suivantes :

- 1) calculer une trace de simulation finie,
- 2) étiqueter les points de la trace par les sous-formules atomiques de  $\phi$  qui sont vraies en ce point ;
- 3) ajouter les sous-formules de la forme  $F\phi_1$  aux points prédécesseurs des points étiquetés par  $\phi_1$ ,
- 4) ajouter les sous-formules de la forme  $\phi_1 U \phi_2$  aux points prédécesseurs des points étiquetés par  $\phi_2$  tant qu'ils satisfont  $\phi_1$

Enfin, la propriété de dualité  $G\phi_1 = !F(!\phi_1)$ , nous conduit ici à étiqueter le dernier état par  $G\phi_1$  s'il est étiqueté par  $\phi_1$ , puis à étiqueter les prédécesseurs d'un état étiqueté par  $G\phi_1$ , s'ils sont eux-mêmes étiquetés par  $\phi_1$ .

Pour des raisons d'intégration au système BIOCHAM, cet algorithme est implanté en Prolog, de même que les méthodes d'intégration numérique (méthode de Runge-Kutta à pas variable, et méthode implicite de Rosenbrock pour les systèmes raides).

### 3.3. PCTL avec contraintes arithmétiques pour la sémantique des populations

La logique PCTL (Hansson *et al.*, 1994) remplace essentiellement les opérateurs de choix des chemins  $E$  et  $A$  de la logique CTL, par un opérateur  $P_{\succ p}$ , désignant la probabilité de réalisation de la formule sous sa portée. Par exemple,  $A(\psi U \psi')$  devient  $P_{\geq 1}(\psi U \psi')$  imposant que la probabilité que  $\psi U \psi'$  se réalise soit égale à 1. Comme dans la section précédente, on s'intéresse à des formules atomiques du premier ordre avec contraintes arithmétiques portant, cette fois-ci, sur des nombres entiers de molécules.

Les algorithmes de *model-checking* probabiliste existants, comme celui du système PRISM (Kwiatkowska *et al.*, 2004), ne permettent cependant pas de traiter les exemples fortement non déterministes ou pour lesquels les variables ont un grand domaine, ce qui est le cas pour la sémantique de population de BIOCHAM. De meilleurs résultats ont été obtenus avec un algorithme de type Monte-Carlo semblable à celui utilisé dans le système APMC (Hérault *et al.*, 2004). Cet algorithme permet de vérifier une formule LTL avec contraintes, et fournit une probabilité approchée de réalisation de cette formule. Pour ce faire, un ensemble de simulations stochastiques est effectué, en étiquetant les traces de chaque simulation par l'algorithme pour LTL avec contraintes. La probabilité de réalisation est alors estimée par comptage sur l'échantillon. Dans ce cadre, l'opérateur  $P_{\succ p}$  n'est autorisé qu'en tête de la formule. Ce ne sont donc pas des formules PCTL générales qui sont vérifiées, mais des formules LTL en fournissant une probabilité de réalisation. Le résultat est donc une probabilité, et non pas la vérification d'une formule comprenant un ensemble de probabilités imbriquées.

## 4. Apprentissage de modèles

Dans BIOCHAM, l'utilisation de langages formels pour, à la fois décrire le système (langage de règles), et décrire ses propriétés (spécification en logique temporelle), permet d'appliquer des algorithmes d'apprentissage automatique pour aider à corriger ou compléter un modèle en fonction d'une nouvelle spécification.

### 4.1. Apprentissage de règles

En ce qui concerne la recherche de règles à ajouter ou supprimer au modèle, nous utilisons la sémantique booléenne de BIOCHAM, et une spécification CTL des propriétés attendues du modèle.

Les formules CTL sont partitionnées en trois classes de façon à guider la recherche pour l'ajout ou la suppression de règles. La classe ECTL est constituée des formules CTL qui ne contiennent pas l'opérateur  $A$  (c'est-à-dire qui ne contiennent pas d'occurrence positive de  $A$  ni d'occurrence négative de  $E$ ). La classe ACTL est constituée des formules CTL qui ne contiennent pas l'opérateur  $E$ . On note UCTL l'ensemble des autres formules.

**Proposition 3. (Conservation des propriétés CTL)** *Soient  $K = (S, R)$  et  $K' = (S, R')$  deux structures de Kripke telles que  $R \subseteq R'$ . Soit  $s$  un état de  $S$ . Pour toute formule ECTL  $\phi$ , si  $s \not\models_{K'} \phi$  alors  $s \not\models_K \phi$ . Pour toute formule ACTL  $\phi$ , si  $s \not\models_K \phi$  alors  $s \not\models_{K'} \phi$ .*

**Preuve 1.** *Commençons par le cas où  $\phi$  est une formule ECTL et prouvons la proposition par la contraposée, c'est-à-dire si  $s \models_K \phi$  alors  $s \models_{K'} \phi$ . Raisonnons par induction sur la taille de la preuve de  $s \models_K \phi$ .  $\phi$  sera supposée sans négation, autre que dans les formules propositionnelles, et n'utilisant que les opérateurs temporels  $X$ ,  $G$  and  $U$ , grâce aux équivalences et dualités données en section 3.1.*

*Si  $\phi = \alpha$  est propositionnelle,  $\alpha$  est vraie dans l'état  $s$ , or  $s$  est aussi un état de  $K'$  donc  $s \models_{K'} \phi$ .*

*Si  $\phi = \psi_1 \& \psi_2$  (resp.  $\psi_1 \mid \psi_2$ ) alors par définition  $s \models_K \psi_1$  et (resp. ou)  $s \models_K \psi_2$ . Par hypothèse d'induction on a  $s \models_{K'} \psi_1$  et (resp. ou)  $s \models_{K'} \psi_2$  on peut donc conclure que  $s \models_{K'} \phi$ .*

*Le seul cas restant est celui où  $\phi = E\psi$  avec  $\psi$  commençant par un opérateur temporel. Il existe alors un chemin  $\pi$  de  $K$  tel que  $\pi \models_K \psi$ . Notons que puisque  $R \subseteq R'$ ,  $\pi$  est aussi un chemin de  $K'$ . On a :*

*– si  $\psi = X\psi'$ , alors  $\pi^1 \models_K \psi'$ , par hypothèse d'induction,  $\pi^1 \models_{K'} \psi'$  et puisque  $\pi$  est un chemin de  $K'$ ,  $\pi \models_{K'} X\psi'$ , c.q.f.d.*

*– si  $\psi = G\psi'$ , alors  $\pi^n \models_K \psi'$  pour tout  $n \geq 0$ . En appliquant l'hypothèse d'induction à tous les états  $\pi^n$ , on obtient  $\pi^n \models_{K'} \psi'$  pour tout  $n \geq 0$ , et comme  $\pi$  est un chemin de  $K'$  on a  $\pi \models_{K'} G\psi'$ , c.q.f.d.*

*– si  $\psi = \psi' U \psi''$ , on a  $\pi^n \models_K \psi''$  pour un  $n \geq 0$  et  $\pi^m \models_K \psi'$  pour tout  $0 \leq m < n$ . En appliquant l'hypothèse d'induction à tous les  $\pi^m$  pour  $0 \leq m \leq n$  on obtient,  $\pi^n \models_{K'} \psi''$ , et  $\pi^m \models_{K'} \psi'$  pour tout  $0 \leq m < n$ , donc  $\pi \models_{K'} \psi' U \psi''$ , c.q.f.d.*

*Pour une formule ACTL  $\phi$ , on prouve la proposition de la même manière par la contraposée puis une induction sur la preuve de  $s \models_{K'} \phi$ . La seule différence est que quand on arrive au cas  $\phi = A\psi$ , on doit alors prouver que  $\pi \models_K \psi$  pour tous les chemins  $\pi$  commençant en  $s$  dans  $K$ . Comme ils sont tous des chemins de  $K'$  on peut appliquer l'hypothèse d'induction et le même raisonnement que ci-dessus et obtenir  $s \models_K \phi$ , c.q.f.d.  $\square$*

Cette proposition indique s'il faut ajouter ou retirer des règles d'un modèle ne satisfaisant par une formule CTL. Pour une formule ACTL fausse, la proposition montre qu'il est nécessaire de supprimer des règles au modèle, et que cela peut invalider uniquement des formules ECTL ou UCTL. Inversement pour une formule ECTL fausse, la proposition montre qu'il est nécessaire de rajouter des règles, et que cela peut invalider uniquement des formules ACTL ou UCTL. Ces propriétés sont utilisées pour guider l'algorithme de révision du modèle dans son choix d'essais d'ajout ou de retrait de règle.

La commande

```
learn_one_addition(reaction_pattern, spec_CTL)
```

énumère chaque instance du schéma de règles fourni en premier argument qui, ajoutée au modèle, permet de satisfaire la formule CTL fournie en second argument (ou implicitement avec la commande `add_spec`). La commande

```
learn_one_deletion(spec_CTL)
```

énumère chaque instance de règle qu'il suffit de retirer au modèle pour satisfaire la formule CTL.

**Exemple 3.** *Dans cet exemple, on retire la règle R3 du modèle de Tyson :*  
`delete_rule(Cyclin+Cdc2~{p1}=>Cdc2~{p1}-Cyclin~{p1}).`

*Le complexe inactif  $Cdc2\sim\{p1\}-Cyclin\sim\{p1\}$  ne pouvant plus se former, certaines molécules ne sont plus accessibles. La spécification donnée dans l'exemple 2 n'est donc plus vérifiée.*

*L'algorithme de recherche de règles propose quatre règles parmi 587 testées. L'ajout de l'une d'elles permet de satisfaire à nouveau la spécification :*

```
biocham: learn_one_addition(elementary_interaction_rules).
```

- (1)  $Cyclin+Cdc2\sim\{p1\}=[Cdc2]=>Cdc2\sim\{p1\}-Cyclin\sim\{p1\}$
- (2)  $Cyclin+Cdc2\sim\{p1\}=[Cyclin]=>Cdc2\sim\{p1\}-Cyclin\sim\{p1\}$
- (3)  $Cyclin+Cdc2\sim\{p1\}=>Cdc2\sim\{p1\}-Cyclin\sim\{p1\}$
- (4)  $Cyclin+Cdc2\sim\{p1\}=[Cdc2\sim\{p1\}]=>Cdc2\sim\{p1\}-Cyclin\sim\{p1\}$

*La troisième règle correspond à la règle retirée. les autres règles ne sont pas biologiquement plausibles et auraient pu être éliminées d'avance en fournissant plus d'information dans le schéma de règles à chercher.*

On définit un algorithme de révision du modèle permettant d'ajouter et retirer plusieurs règles, en fixant une stratégie de recherche des modifications de règles permettant de satisfaire la spécification. La commande

```
revise_model(reaction_pattern, spec_CTL)
```

énumère de façon heuristique des ensembles d'ajouts et de retraits de règles qui permettent de satisfaire une spécification CTL fournie en argument.

L'algorithme traite les formules CTL de la spécification une par une. Afin de rendre vraie une formule ECTL insatisfaite, l'algorithme essaye d'ajouter une règle. Pour rendre vraie une formule UCTL, l'algorithme essaye d'ajouter ou de retirer des règles. Enfin pour rendre vraie une formule ACTL, l'algorithme essaye de retirer des règles provenant du contre-exemple fourni par le *model-checker*. Les formules ECTL sont traitées en premier puis les UCTL et enfin les ACTL. Le traitement d'une formule ACTL est cependant autorisé à invalider les formules ECTL et UCTL, qui seront alors retraitées avec la contrainte de conserver la satisfaction des formules ACTL déjà traitées. Comme à chaque traitement de formule plusieurs choix d'ajouts ou de retraits sont possibles, un système de retour en arrière (backtrack) permet de tester toutes ces possibilités afin de trouver une solution et ne pas rester bloqué sur un échec.

Formellement, l'algorithme est défini par un système de transitions non déterministe dont tous les choix sont explorés par retour en arrière. L'état d'une recherche est représenté par un triplet  $[Q_t, Q, R]$  où :

- $R$  est l'ensemble courant des règles de réaction,
- $Q_t = (E_t, U_t, A_t)$  est l'ensemble des formules déjà traitées et satisfaites dans  $R$ . Ces formules sont triées en 3 groupes :  $E_t$  représente les formules ECTL,  $U_t$  les formules non classifiables (UCTL) et  $A_t$  les formules ACTL,
- $Q = (E, U, A)$  est l'ensemble des formules à traiter triées de la même manière en formules ECTL, UCTL et ACTL.

Par un léger abus de notation, on note  $R \models \phi$  la satisfaction de la formule  $\phi$  par (la structure de Kripke associée à) l'ensemble de règles  $R$ . Les ajouts sont recherchés en considérant seulement les instances du schéma de règles fourni en argument. La recherche des règles à retirer  $Re$  (dans les transitions U'' et A' ci-dessous) ne se fait que dans l'ensemble des règles apparaissant dans le contre-exemple fourni par le model-checker.

La configuration initiale est :  $[(\emptyset, \emptyset, \emptyset), (E, U, A), R]$ , l'algorithme explore ensuite les transitions suivantes :

- E** :  $[(E_t, U_t, A_t), (E \cup \{e\}, U, A), R] \rightarrow [(E_t \cup \{e\}, U_t, A_t), (E, U, A), R]$   
si  $R \models e$
- E'** :  $[(E_t, U_t, A_t), (E \cup \{e\}, U, A), R] \rightarrow [(E_t \cup \{e\}, U_t, A_t), (E, U, A), R \cup \{r\}]$   
si  $R \not\models e$  et  $\forall f \in \{e\} \cup E_t \cup U_t \cup A_t \quad R \cup \{r\} \models f$
- U** :  $[(E_t, U_t, A_t), (\emptyset, U \cup \{u\}, A), R] \rightarrow [(E_t, U_t \cup \{u\}, A_t), (\emptyset, U, A), R]$   
si  $R \models u$
- U'** :  $[(E_t, U_t, A_t), (\emptyset, U \cup \{u\}, A), R] \rightarrow [(E_t, U_t \cup \{u\}, A_t), (\emptyset, U, A), R \cup \{r\}]$   
si  $R \not\models u$  et  $\forall f \in \{u\} \cup E_t \cup U_t \cup A_t \quad R \cup \{r\} \models f$
- U''** :  $[(E_t, U_t, A_t), (\emptyset, U \cup \{u\}, A), R \cup Re] \rightarrow [(E_t, U_t \cup \{u\}, A_t), (\emptyset, U, A), R]$   
si  $K : s_i \not\models u$  et  $\forall f \in \{u\} \cup E_t \cup U_t \cup A_t \quad R \models f$

$$\begin{aligned}
\mathbf{A} : & [(E_t, U_t, A_t), (\emptyset, \emptyset, A \cup \{a\}), R] \rightarrow [(E_t, U_t, A_t \cup \{a\}), (E_p, U_p, A), R] \\
& \text{si } R \models a \\
\mathbf{A}' : & [(E_t \cup E'_t, U_t \cup U'_t, A_t), (\emptyset, \emptyset, A \cup \{a\}), R \cup Re] \rightarrow [(E_t, U_t, A_t \cup \{a\}), \\
& (E'_t, U'_t, A), R] \\
& \text{si } R \cup Re \not\models a, \forall f \in \{a\} \cup E_t \cup U_t \cup A_t \quad R \models f, \text{ et } \forall g \in E'_t \cup U'_t, R \not\models g.
\end{aligned}$$

**Proposition 4. (Correction et terminaison)** *L'algorithme de révision termine et si sa configuration terminale est de la forme  $[(E_t, U_t, A_t), (\emptyset, \emptyset, \emptyset), R]$  alors le modèle  $R$  vérifie la spécification initiale  $(E, U, A)$  équivalente à  $(E_t, U_t, A_t)$ .*

**Preuve 2.** *La correction de l'algorithme découle du fait que chaque transition préserve l'ensemble des propriétés en les faisant passer entre les ensembles vérifiés et non-vérifiés et ne conserve dans l'ensemble vérifié que des propriétés vraies, ce qui peut être vérifié de façon triviale pour chaque transition.*

*La terminaison se montre en considérant l'ordre lexicographique sur le couple  $\langle a, n \rangle$  où  $a$  est le nombre de propriétés ACTL non vérifiées, et  $n$  est le nombre de propriétés ECTL ou UCTL non vérifiées. Chaque transition diminue strictement soit  $a$  (transitions  $\mathbf{A}$  et  $\mathbf{A}'$ ), soit  $n$  en laissant  $a$  inchangé (transitions  $\mathbf{E}$ ,  $\mathbf{E}'$ ,  $\mathbf{U}$ ,  $\mathbf{U}'$  et  $\mathbf{U}''$ ). On a donc décroissance stricte de  $\langle a, n \rangle$  à chaque transition, ce qui interdit l'existence de suites infinies de transitions. De plus, l'arbre de recherche est à branchement fini : en effet les délétions se font dans un ensemble fini, et de même les ajouts se font dans l'ensemble fini des instances du schéma fourni. On a donc bien terminaison par le lemme de Koenig.*

Cet algorithme est toutefois incomplet pour deux raisons. D'une part, la satisfaction d'une formule ECTL ou UCTL n'est recherchée qu'en ajoutant une seule règle au modèle (transitions  $\mathbf{E}'$  et  $\mathbf{U}'$ ). D'autre part, la complétion de la relation d'accessibilité en une relation totale dans la structure de Kripke associée au modèle, ne garantit pas que les conditions de la proposition 3. sont vérifiées après l'ajout ou le retrait d'une règle. Il s'agit donc d'un algorithme de révision heuristique qui peut échouer dans des cas où la spécification CTL est satisfaisable.

**Exemple 4.** *L'algorithme de révision peut être utilisé pour raffiner un modèle, de façon par exemple à prendre en compte une nouvelle molécule donnée en fournissant uniquement une spécification de ses propriétés biologiques. Dans l'exemple du cycle cellulaire, on peut ainsi s'intéresser à la recherche d'une modification du modèle, qui préserve la spécification précédente, et fait intervenir une molécule  $X$  telle que (1)  $X$  ne peut être activée sans la présence de  $\text{Cdc2-Cyclin}\{p1\}$ , et (2) sans  $X$ ,  $\text{Cyclin}\{p1\}$  n'est jamais inactivée ou dégradée :*

- (1)  $\text{AG}(\!(X)\!) \rightarrow \text{checkpoint}(\text{Cdc2-Cyclin}\{p1\}, X)$ ,
- (2)  $\text{AG}(\!(\text{Cyclin}\{p1\}) \rightarrow \text{checkpoint}(X, \!(\text{Cyclin}\{p1\}))\!)$ .

*A cette spécification, sont également rajoutées des formules d'accessibilité et d'oscillation relatives à  $X$  :*

```

    reachable(X),
    reachable(!X),
    oscil(X).

```

Par défaut, on associe à la molécule  $X$  une dynamique arbitraire avec deux règles simples de synthèse et de dégradation :  $\_ \Rightarrow X$  et  $X \Rightarrow \_$ .

La spécification n'est plus vérifiée par le modèle. La commande `revise_model`, limitée à des règles faisant intervenir  $X$  pour l'ajout de règles, permet de trouver des solutions pour le corriger. La première solution trouvée :

```

    Deletion(s):
    \_ \Rightarrow X.
    Cyclin~{p1} \Rightarrow \_.
    Addition(s):
    \_ = [Cdc2-Cyclin~{p1}] \Rightarrow X.
    X + Cyclin~{p1} \Rightarrow X - Cyclin~{p1}.

```

propose non seulement que la synthèse de  $X$  dépende directement de  $Cdc2-Cyclin\sim\{p1\}$  mais aussi que  $X$  se complexe à  $Cyclin\sim\{p1\}$  afin de l'inactiver. Ces modifications apportées au modèle (retraits et ajouts des nouvelles règles) sont assurées de respecter la spécification.

D'autres solutions sont également proposées, comme par exemple :

```

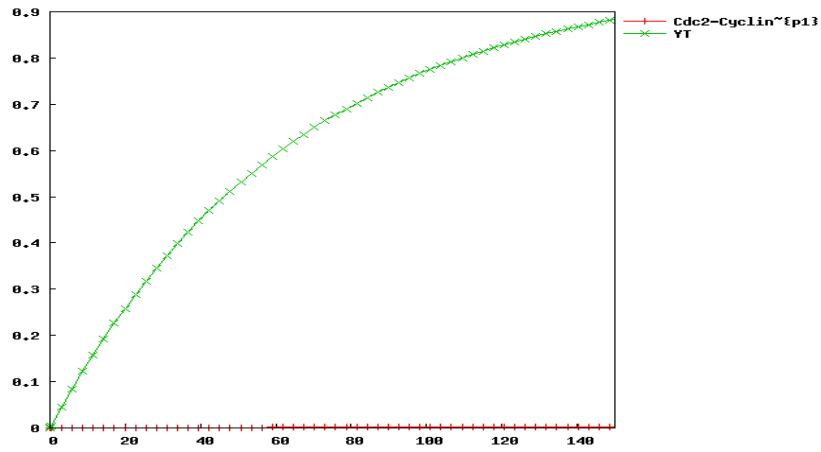
    Deletion(s):
    \_ \Rightarrow X.
    Cyclin~{p1} \Rightarrow \_.
    Addition(s):
    \_ = [Cdc2-Cyclin~{p1}] \Rightarrow X.
    Cyclin~{p1} = [X] \Rightarrow \_.

```

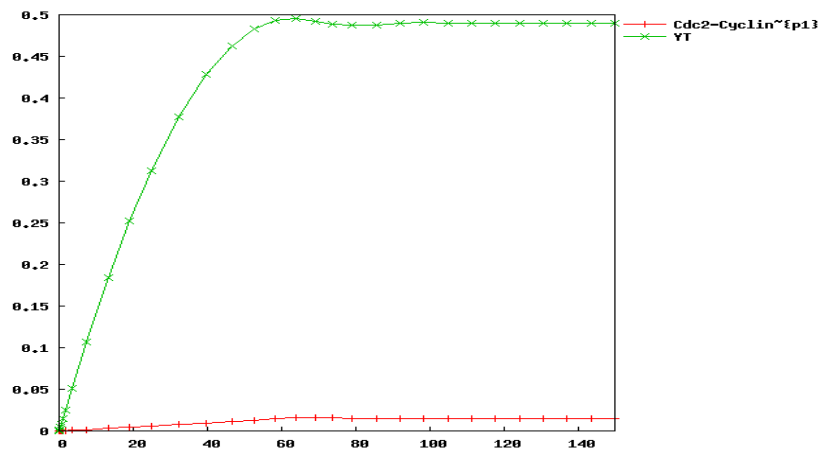
où la synthèse de  $X$  est à nouveau assurée par  $Cdc2-Cyclin\sim\{p1\}$  mais où  $Cyclin\sim\{p1\}$  est directement dégradée par  $X$ .

#### 4.2. Recherche de paramètres

De la même manière que pour l'apprentissage de règles d'interaction à partir de propriétés CTL, il est possible d'utiliser une spécification LTL avec contraintes arithmétiques pour l'apprentissage de valeurs de paramètres cinétiques. Nous avons développé une méthode énumérative dans laquelle l'espace de recherche, fourni par des bornes sur les valeurs des paramètres, est exploré avec une précision spécifiée par le modélisateur. Pour chaque jeu de valeurs testées, le modèle est simulé et confronté à la spécification LTL. Le nombre d'appels au *model-checker* est donc borné par  $r^n$  où  $n$  est le nombre de paramètres à chercher en même temps et  $r$  est le nombre maximum de valeurs à essayer pour chaque paramètre.

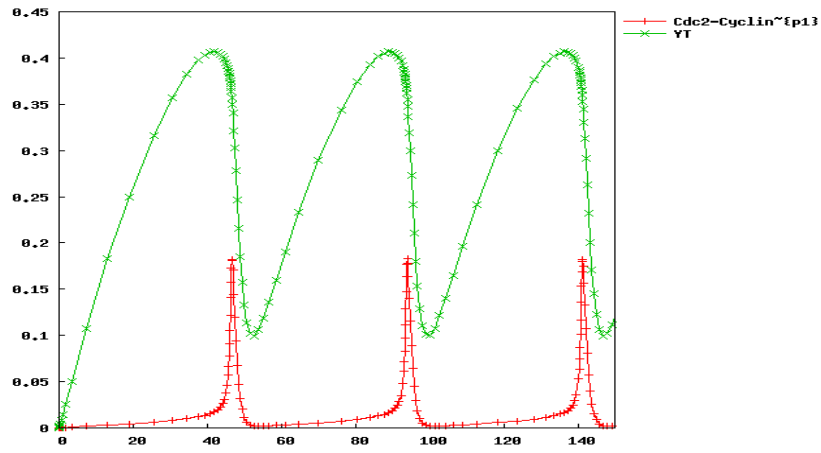


**Figure 3.** Simulation numérique avec  $k_3 = 0.01$  et  $k_4 = 0.01$

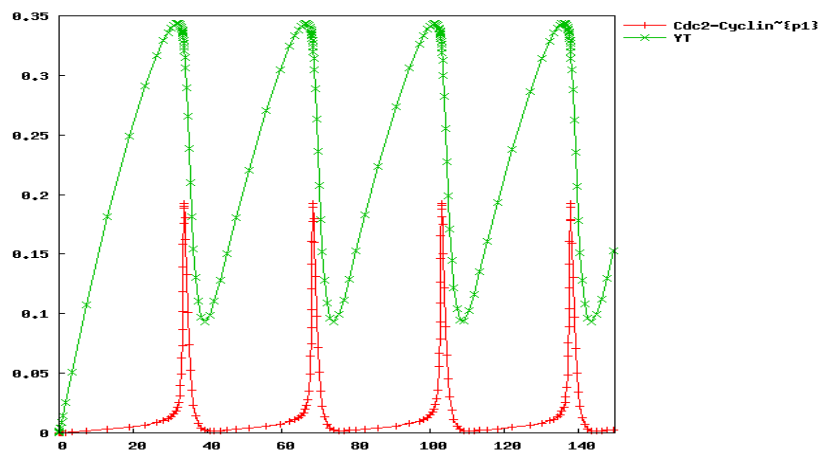


**Figure 4.** Simulation numérique avec  $k_3 = 10$  et  $k_4 = 70$  obtenue en réponse à la requête `oscil(Cdc2-Cyclin~{p1},3)` sur 150 unités de temps

La recherche de paramètres peut être illustrée à partir du modèle de l'exemple 1. Deux paramètres sont choisis,  $k_3$  et  $k_4$ , auxquels sont assignées les valeurs arbitraires 0.01 et 0.01. La simulation montre que le système atteint un état stationnaire (figure 3) or il est censé osciller.



**Figure 5.** Simulation numérique avec  $k_3 = 10$  et  $k_4 = 120$  obtenue en réponse à la requête `oscil(Cdc2-Cyclin~{p1},3,0.1)`.



**Figure 6.** Simulation numérique avec  $k_3 = 10$  et  $k_4 = 190$  obtenue en réponse à la requête `period(Cdc2-Cyclin~{p1},35)`.

Pour y remédier, la requête

```
learn_parameters([k3, k4], [(0, 200), (0, 200)], 20,
                 oscil(Cdc2-Cyclin~{p1},3),150).
```

lance la recherche de valeurs des paramètres  $k_3$  et  $k_4$  dans l'intervalle  $[0, 200]$ , telles que Cdc2-Cyclin~{p1} oscille au moins 3 fois en 150 unités de temps. BIOCHAM propose une première solution ( $k_3 = 10$ ,  $k_4 = 70$ ) qui n'est toutefois pas satisfaisante car Cdc2-Cyclin~{p1} oscille de façon amortie avec des valeurs faibles (figure 4). L'ajout d'une contrainte d'existence de valeurs de concentration supérieures à 0.1 dans les oscillations de ce complexe produit les valeurs de paramètres  $k_3 = 10$  et  $k_4 = 120$  et la courbe de la figure 5.

La spécification peut être encore raffinée, en ajoutant la contrainte `period(Cdc2-Cyclin~{p1},35)` qui indique que le complexe doit osciller avec une période de 35 unités de temps comme indiqué dans (Tyson, 1991). L'algorithme de recherche de paramètres trouve alors les valeurs  $k_3 = 10$  et  $k_4 = 190$ . Les formes des courbes obtenues dans la simulation de la figure 6 se superposent presque parfaitement à celles correspondant aux paramètres publiés  $k_3 = 200$  et  $k_4 = 180$  (figure 2), montrant ainsi que la spécification est bien vérifiée. La valeur du paramètre  $k_3$  obtenue avec BIOCHAM est cependant très différent de celle publiée. La raison en est simple :  $k_3$  est un paramètre d'association entre la cycline et la kinase et doit être comparé au paramètre de dissociation  $k_4p = 0.018$ . Plus le rapport entre ces deux paramètres est élevé, plus l'affinité entre ces protéines est grande. Pour  $k_3 = 10$ , le complexe est déjà fortement lié. Augmenter la valeur de ce paramètre ne modifiera que très peu les courbes.

Rappelons que BIOCHAM donne la première valeur pour laquelle la spécification est vérifiée. Les valeurs supérieures à  $k_3 = 10$  sont donc susceptibles, elles aussi, de reproduire le comportement recherché, ce que l'on peut déterminer en calculant l'ensemble de toutes les valeurs de paramètres satisfaisant la spécification.

D'une certaine manière, le processus d'apprentissage automatise une partie de ce que le modélisateur fait à la main, c'est-à-dire essayer différentes valeurs de paramètres qui correspondent à des formes de courbes d'activité de protéines qu'il souhaite reproduire, et ce dans un intervalle de valeurs de paramètres jugé réaliste. L'algorithme d'apprentissage permet de tester rapidement des espaces de paramètres après formalisation des aspects qualitatifs et quantitatifs de la courbe en spécification LTL. Notons cependant que le nombre de paramètres pour lesquels une valeur est cherchée ainsi que la précision de la recherche influencent le temps de calcul. Cette méthode est en fait complémentaire d'autres méthodes d'estimation de paramètres telles que l'utilisation de diagrammes de bifurcation qui permettent de visualiser la stabilité du modèle pour toutes les valeurs d'un paramètre et de choisir une valeur qui reproduit un comportement précis (oscillation, bistabilité, etc.).

## 5. Conclusion et perspectives

Le projet BIOCHAM compte parmi les premières expériences d'application des méthodes formelles de vérification et d'apprentissage automatique en modélisation de systèmes biologiques. La sémantique booléenne est utilisée d'une part, pour exprimer des propriétés d'accessibilité, de points de passage obligés, de stationnarité et d'oscillation dans les réseaux d'interaction, et d'autre part, pour inférer des règles de réaction à partir d'une telle spécification. La sémantique des concentrations permet de développer des modèles à dynamique continue. Elle permet aussi de fournir des valeurs de paramètres cinétiques à partir des propriétés des courbes numériques correspondant aux observations biologiques formalisées en logique temporelle.

Le cadre formel développé dans BIOCHAM fournit aussi une base pour étudier les questions d'abstraction et de composition de modèles en biologie des systèmes. En particulier nous envisageons d'explorer les techniques d'interprétation abstraite pour formaliser les questions de simplification et de raffinement de modèles, préservant une spécification des propriétés devant être conservées. De même, si la composition de deux modèles n'est pas une notion bien définie aujourd'hui, la formalisation des propriétés biologiques fournit un cadre d'étude des phénomènes de conservation et d'émergence de propriétés dans le modèle composé. Ces questions sont actuellement abordées pour l'étude de modèles couplés du cycle cellulaire et du cycle circadien avec l'objectif de contribuer à formuler de nouvelles hypothèses biologiques par ces méthodes<sup>3</sup>.

## 6. Bibliographie

- Angelopoulos N., Muggleton S. H., « Machine Learning Metabolic Pathway descriptions using a Probabilistic Relational Representation », *Electronic Transactions in Artificial Intelligence*, 2002a. also in Proceedings of Machine Intelligence 19.
- Angelopoulos N., Muggleton S. H., Slps for Probabilistic Pathways : Modeling and Parameter estimation, Technical Report n° TR 2002/12, Department of Computing, Imperial College, London, UK, 2002b.
- Antoniotti M., Policriti A., Ugel N., Mishra B., « Model Building and Model Checking for Biochemical Processes », *Cell Biochemistry and Biophysics*, vol. 38, p. 271-286, 2003.
- Batt G., Validation de modèles qualitatifs de réseaux de régulation génique : une méthode basée sur des techniques de vérification formelle, PhD thesis, Université Joseph Fourier - Grenoble I, February, 2006.
- Batt G., Bergamini D., de Jong H., Garavel H., Mateescu R., « Model Checking Genetic Regulatory Networks using GNA and CADP », *Proceedings of the 11th International SPIN Workshop on Model Checking of Software SPIN'2004*, Barcelona, Spain, April, 2004.
- Bernot G., Comet J.-P., Richard A., Guespin J., « A Fruitful Application of Formal Methods to Biological Regulatory Networks : Extending Thomas' Asynchronous Logical Approach with Temporal Logic », *Journal of Theoretical Biology*, vol. 229, n° 3, p. 339-347, 2004.

3. <http://contraintes.inria.fr/moca/>

- Bryant C. H., Muggleton S. H., Oliver S. G., Kell D. B., Reiser P. G. K., King R. D., « Combining Inductive Logic Programming, Active Learning and Robotics to Discover the Function of Genes », *Electronic Transactions in Artificial Intelligence*, 2001.
- Calder M., Vyshemirsky V., Gilbert D., Orton R., « Analysis of Signalling Pathways using the Prism Model Checker », in G. Plotkin (ed.), *CMSB'05 : Proceedings of the third Workshop on Computational Methods in Systems Biology*, 2005.
- Calzone L., Chabrier-Rivier N., Fages F., Gentils L., Soliman S., « Machine learning biomolecular interactions from temporal logic properties », in G. Plotkin (ed.), *CMSB'05 : Proceedings of the third Workshop on Computational Methods in Systems Biology*, 2005.
- Cardelli L., « Brane Calculi - Interactions of Biological Membranes », in V. Danos, V. Schächter (eds), *CMSB'04 : Proceedings of the second Workshop on Computational Methods in Systems Biology*, vol. 3082 of *Lecture Notes in BioInformatics*, Springer-Verlag, p. 257-280, 2004.
- Chabrier N., Fages F., « Symbolic model checking of biochemical networks », in C. Priami (ed.), *CMSB'03 : Proceedings of the first Workshop on Computational Methods in Systems Biology*, vol. 2602 of *Lecture Notes in Computer Science*, Springer-Verlag, Rovereto, Italy, p. 149-162, March, 2003.
- Chabrier N., Fages F., Soliman S., *BIOCHAM's user manual*, INRIA. 2003–2006.
- Chabrier-Rivier N., Chiaverini M., Danos V., Fages F., Schächter V., « Modeling and querying biochemical interaction networks », *Theoretical Computer Science*, vol. 325, n° 1, p. 25-44, September, 2004a.
- Chabrier-Rivier N., Fages F., Soliman S., « The Biochemical Abstract Machine BIOCHAM », in V. Danos, V. Schächter (eds), *CMSB'04 : Proceedings of the second Workshop on Computational Methods in Systems Biology*, vol. 3082 of *Lecture Notes in BioInformatics*, Springer-Verlag, p. 172-191, 2004b.
- Cimatti A., Clarke E., Enrico Giunchiglia F. G., Pistore M., Roveri M., Sebastiani R., Tacchella A., « NuSMV 2 : An OpenSource Tool for Symbolic Model Checking », *Proceedings of the International Conference on Computer-Aided Verification, CAV'02*, Copenhagen, Danmark, July, 2002.
- Clarke E. M., Grumberg O., Peled D. A., *Model Checking*, MIT Press, 1999.
- Danos V., Laneve C., « Formal Molecular Biology », *Theoretical Computer Science*, vol. 325, n° 1, p. 69-110, 2004.
- de Raedt L., *Interactive Theory Revision, an inductive Logic Programming Approach*, Knowledge-Based Systems, academic press, 1992.
- Eker S., Knapp M., Laderoute K., Lincoln P., Meseguer J., Sönmez M. K., « Pathway Logic : Symbolic Analysis of Biological Signaling », *Proceedings of the seventh Pacific Symposium on Biocomputing*, p. 400-412, January, 2002.
- Fages F., Soliman S., Chabrier-Rivier N., « Modelling and Querying Interaction Networks in the Biochemical Abstract Machine BIOCHAM », *Journal of Biological Physics and Chemistry*, vol. 4, n° 2, p. 64-73, October, 2004.
- Giaever G. et al., « Functional profiling of the *Saccharomyces cerevisiae* genome », *nature*, vol. 418, p. 387-391, Jul, 2002.
- Gibson M. A., Bruck J., « A probabilistic model of a prokaryotic gene and its regulation », in H. Bolouri, J. Bower (eds), *Computational Methods in Molecular Biology : From Genotype to Phenotype*, MIT press, chapter 2, 2000.

- Gillespie D. T., « General Method for Numerically Simulating Stochastic Time Evolution of Coupled Chemical-Reactions », *Journal of Computational Physics*, vol. 22, p. 403-434, 1976.
- Hansson H., Jonsson B., « A Logic for Reasoning about Time and Reliability », *Formal Aspects of Computing*, vol. 6, n° 5, p. 512-535, 1994.
- Hérault T., Lassaigne R., Magniette F., Peyronnet S., « Approximate Probabilistic Model Checking », *Proceedings of the 5th Verification, Model Checking and Abstract Interpretation (VMCAI 2004)*, vol. 2937 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 73-84, 2004.
- Kanehisa M., Goto S., « KEGG : Kyoto Encyclopedia of Genes and Genomes », *Nucleic Acids Research*, vol. 28, n° 1, p. 27-30, 2000.
- Keseler I. M., Collado-Vides J., Gama-Castro S., Ingraham J., Paley S., Paulsen I. T., Peralt-Gil M., Karp P. D., « EcoCyc : a comprehensive database resource for Escherichia coli », *Nucleic Acids Research*, vol. 33, p. D334-D337, 2005.
- Kohn K. W., « Molecular Interaction Map of the Mammalian Cell Cycle Control and DNA Repair Systems », *Molecular Biology of the Cell*, vol. 10, n° 8, p. 2703-2734, August, 1999.
- Koza J. R., Mydlowec W., Lanza G., Yu J., Keane M. A., « Reverse Engineering of Metabolic Pathways from Observed Data Using Genetic Programming », *Proceedings of the 6th Pacific Symposium on Biocomputing (PSB 2001)*, Hawaii, USA, p. 434-445, January, 2001.
- Kwiatkowska M. Z., Norman G., Parker D., « PRISM 2.0 : A Tool for Probabilistic Model Checking », *st International Conference on Quantitative Evaluation of Systems (QEST 2004)*, IEEE Computer Society, p. 322-323, 2004.
- Langley P., Simon H. A., Bradshaw G. L., Zytkow J. M., *Scientific Discovery : Computational Explorations of the Creative Processes*, MIT Press, Cambridge, MA, February, 1987.
- Muggleton S. H., « Inverse Entailment and Progol », *New Generation Computing*, vol. 13, p. 245-286, 1995.
- Nagasaki M., Onami S., Miyano S., Kitano H., « Bio-calculus : Its Concept and Molecular Interaction », *Proceedings of the Workshop on Genome Informatics*, vol. 10, p. 133-143, 1999.
- Nagasaki M., Onami S., Miyano S., Kitano H., « Bio-calculus : Its Concept, and an Application for Molecular Interaction », *Currents in Computational Molecular Biology*, vol. 30 of *Frontiers Science Series*, Universal Academy Press, Inc., 2000. This book is a collection of poster papers presented at the RECOMB 2000 Poster Session.
- Phillips A., Cardelli L., « A Correct Abstract Machine for the Stochastic Pi-calculus », *Transactions on Computational Systems Biology*, to appear. Special issue of BioConcur 2004.
- Regev A., Panina E. M., Silverman W., Cardelli L., Shapiro E., « BioAmbients : An Abstraction for Biological Compartments », *Theoretical Computer Science*, vol. 325, n° 1, p. 141-167, September, 2004.
- Regev A., Silverman W., Shapiro E. Y., « Representation and simulation of biochemical processes using the pi-calculus process algebra. », *Proceedings of the sixth Pacific Symposium of Biocomputing*, p. 459-470, 2001.
- Schoeberl B., Eichler-Jonsson C., Gilles E., Muller G., « Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors », *Nature Biotechnology*, vol. 20, n° 4, p. 370-375, 2002.

- Todorovski L., Džeroski S., « Theory Revision in Equation Discovery », in K. P. Jantke, A. Shinohara (eds), *Proceedings of the 4th International Conference on Discovery Science, DS 2001*, vol. 2226 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Washington, DC, USA, p. 389-400, 2001.
- Tomita M. et al., « The E-CELL Project : Towards Integrative Simulation of Cellular Processes », *New Generation Computing*, vol. 1, n° 18, p. 1-12, 2000.
- Tyson J. J., « Modeling the cell division cycle : cdc2 and cyclin interactions », *Proceedings of the National Academy of Sciences*, vol. 88, n° 16, p. 7328-7332, August, 1991.
- VBI, *COPASI's manual*. n.d.

Article reçu le 10 octobre 2005  
Accepté après révision le 18 mai 2006

**Laurence Calzone** est actuellement post-doctorante à l'INRIA Rocquencourt dans le projet Contraintes. Elle a soutenu sa thèse à Virginia Polytechnic Institute and State University aux Etats-Unis en 2003 sous la direction de John. J. Tyson et a effectué un premier séjour post-doctoral à Budapest en Hongrie dans le groupe de modélisation de Belà Novák. Ses travaux de recherche se sont concentrés sur la modélisation mathématique de systèmes biologiques et plus précisément ceux concernant le cycle cellulaire.

**Nathalie Chabrier-Rivier** est doctorante dans le projet Contraintes sous la direction de François Fages, où elle achève sa thèse sur la Machine Abstraite Biochimique BIOCHAM.

**François Fages** est directeur de recherche à l'INRIA Rocquencourt depuis 1999 où il dirige le projet Contraintes. Après la soutenance de sa thèse en 1983 sur la théorie de l'unification en démonstration automatique, il fait une carrière de chercheur au CNRS à l'Ecole Normale Supérieure, enseigne à l'Ecole Polytechnique et est consultant au Laboratoire Central de Recherches de Thalès. Logicien spécialiste des langages de programmation déclarative, il s'intéresse depuis 2001 à l'application des concepts informatiques à la biologie des systèmes et est le créateur avec Nathalie Chabrier de la Machine Abstraite Biochimique BIOCHAM.

**Loïc Fosse** a été stagiaire de l'EPITA dans le projet Contraintes où il a développé et implanté la sémantique stochastique de BIOCHAM.

**Sylvain Soliman** est depuis 2002 chercheur à l'INRIA Rocquencourt, au sein du projet Contraintes. Après une thèse soutenue en 2001 sur les interactions entre logique linéaire et programmation par contraintes, il a travaillé un an et demi dans un laboratoire de la DGA sur le traitement des langues. Depuis son arrivée à l'INRIA sa recherche s'articule autour de la bioinformatique, à travers des travaux sur la Machine Abstraite Biochimique (BIOCHAM), et sur la sémantique des langages linéaires concurrents avec contraintes, à travers un soutien pour la conception du langage SiLCC.