Apprentissage de règles de réactions biochimiques à partir de propriétés en logique temporelle

Laurence Calzone, Nathalie Chabrier-Rivier, Francois Fages, Sylvain Soliman

Apprentissage de règles de réactions biochimiques à partir de propriétés en logique temporelle

Laurence Calzone Nathalie Chabrier-Rivier François Fages and Sylvain Soliman

INRIA Rocquencourt, BP 105, F-78153 Le Chesnay Cedex France
prenom.nom@inria.fr

Résumé *Avec le développement de langages formels pour modéliser les systèmes d'interactions biomoléculaires, la possibilité d'effectuer des calculs symboliques au delà des simulations numériques 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 les interactions biomoléculaires et un langage original 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 moléculaire à partir de propriétés temporelles observées. Dans ce contexte, le but est de corriger ou compléter les modèles BIOCHAM semi-automatiquement. Dans cet article, nous décrivons le système d'apprentissage automatique de BIOCHAM, qui permet, d'une part, de trouver de nouvelles règles d'interaction à partir d'un modèle partiel et de contraintes exprimées en logique temporelle, et d'autre part, d'estimer les valeurs de paramètres cinétiques à partir de propriétés formalisées en logique temporelle avec contraintes numériques sur les concentrations ou leurs dérivées.*

Keywords: Biologie systémique, apprentissage, logique temporelle, voies métaboliques, voies de signalisation.

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, soulève un besoin important de représentation formelle des systèmes biologiques. Les connaissances concernant des interactions de gènes et des chemins métaboliques sont intégrées dans des bases de données comme KEGG [19], BioCyc [20],. . .et présentées sous forme de diagrammes annotés. Les outils comme BioSpice [17], Copasi [28], GON, E-cell [27] etc. ont été développés pour faire des simulations sur ces bases de données lorsque les données numériques sont présentes.

Au delà de la simulation, la possibilité de faire des calculs symboliques sur les réseaux d'interactions moléculaires ouvre la voie vers une nouvelle sorte d'outils de raisonnement automatique pour les biologistes modélisateurs. Notre projet avec la Machine Abstraite Biochimique BIOCHAM[1] [16] , qui a débuté en 2002, est une tentative dans cette direction. BIOCHAM fournit une sémantique précise aux cartes d'interactions biomoléculaires qualitatives en tant que systèmes de transition concurrents [10]. Basé sur cette sémantique formelle, BIOCHAM propose :

- un langage compositionnel basé sur des règles de réactions dans le but de modéliser des systèmes biochimiques, utilisant des schémas et des expressions cinétiques lorsque ces dernières sont disponibles ;
- un simulateur booléen non-déterministe et un simulateur numérique ;

[1] http ://contraintes.inria.r/BIOCHAM/

– un langage de requête original basé sur la logique temporelle CTL [13] pour les modèles booléens, et LTL avec contraintes pour les modèles cinétiques. Ce langage de requête permet d'exprimer des propriétés biologiques sur l'accessibilité, les points de passage obligés, la stabilité ou les oscillations [8] ;
– une méthode d'apprentissage pour inférer des règles d'interactions et des paramètres cinétiques à partir de propriétés temporelles observées, qui constitue le principal sujet de cet article.

Nos premiers résultats expérimentaux montrant la faisabilité du requêtage en logique temporelle et son passage à l'échelle des grands systèmes, ont été obtenus sur un modèle qualitatif du contrôle du cycle cellulaire des mammifères développé d'après le diagramme de Kohn [21] qui implique environ 500 variables et 2700 règles de réaction [9]. D'autres utilisations de la logique temporelle dans ce domaine sont [24,5,4,15]

La méthode d'apprentissage dans BIOCHAM sur les modèles booléens permet de trouver des règles d'interactions à partir d'un modèle partiel et de contraintes décrivant le comportement du système [7]. Ces contraintes sont exprimées en logique temporelle CTL avec des formules qui expriment les propriétés attendues (exemples positifs) tout comme les propriétés interdites (exemples négatifs) du système. Le processus d'apprentissage peut être guidé par l'utilisateur en donnant des schémas pour limiter les types de réactions cherchées, telles que la complexation, la phosphorylation, etc. La méthode d'apprentissage s'applique de façon similaire aux modèles numériques en permettant d'apprendre la valeur de paramètres cinétiques à partir de spécifications en logique temporelle avec contraintes sur des quantités numériques.

Dans les travaux antérieurs, les techniques d'apprentissage automatique ont été utilisées comme la programmation logique inductive [25] ou la programmation génétique pour inférer des fonctions de gènes [6], des descriptions de chemins métaboliques [1,2,22] ou des interactions entre gènes [5]. Nos travaux peuvent être rapprochés des domaines de la découverte scientifique qualitative et numérique [23] et de la révision de théories [26,14]. Cependant, l'apprentissage structurel d'interactions biomoléculaires à partir de formules temporelles est un sujet relativement nouveau, que ce soit du point de vue de l'apprentissage automatique ou de la biologie des systèmes.

Dans ce papier, nous décrivons la méthode d'apprentissage à partir de formules de logique temporelle développées dans BIOCHAM. Nous présentons des exemples d'apprentissage de règles et de paramètres dans des modèles de processus intracellulaires, et montrons l'utilisation interactive du système d'apprentissage pour raffiner la spécification et le schéma de règle jusqu'à l'obtention d'une solution biologiquement satisfaisante.

2 Préliminaires sur BIOCHAM

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

2.1 Syntaxe

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

molecule = name | molecule-molecule |molecule$\sim${name,...,name}
reaction = solution => solution | kinetics for solution => solution
solution = _ | molecule | solution + solution

L'opérateur de liaison $-$ sert à représenter la complexation ou d'autres formes de liaison, et l'opérateur de modification $\sim$ permet d'attacher un ensemble de modifications tel que l'ensemble des sites phosphorylés.

Pour les règles de réaction, les abréviations suivantes peuvent être utilisées : $A = [C] => B$ pour la règle $A + C => B + C$ avec le catalyseur C, et $A <=> B$ pour les deux réactions symétriques. Par exemple, $Q + E1 => Q - E1$ représente une règle de complexation et $Q = [R] => Q \sim \{s\}$ une règle de phosphorylation catalysée par R.

BIOCHAM possède aussi un langage de schémas avec contraintes qui est utilisé pour spécifier des molécules et des ensembles de règles de réaction de façon concise ou pour indiquer la forme des règles à considérer pendant le processus d'apprentissage comme expliqué dans la section 3.

2.2 Sémantique

La sémantique de BIOCHAM est définie à deux niveaux d'abstraction : la sémantique des concentrations moléculaires, et la sémantique booléenne qui ne s'occupe que de la présence ou de l'absence des molécules. Cette dernière permet, sans connaître ni la valeur des concentrations ni les expressions cinétiques, de raisonner sur tous les comportements possibles du système.

La sémantique des concentrations moléculaires suppose quant à elle que chaque règle de réaction soit donnée avec une expression cinétique (loi d'action de masse, Michaelis-Menten, Hill, etc.). Ces règles peuvent être compilées en un système d'équations différentielles ordinaires (hautement non linéaire) qui, quand on fixe les concentrations initiales de molécules, rend l'évolution du système complètement déterministe.

Dans la sémantique booléenne, les règles de réaction sont interprétées par un système de transitions concurrent asynchrone [9]. Une règle comme $A + B => C + D$ 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, et 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 de façon non-déterministe afin de rendre compte de toutes les compétitions possibles.

Le trait le plus original de BIOCHAM est cependant l'utilisation de la logique temporelle comme langage de requêtage des propriétés biologiques d'un modèle. La logique temporelle des arbres de calcul (*Computation Tree Logic CTL**) [13] est une extension de la logique classique qui permet de raisonner sur un arbre de transition d'états à l'aide d'opérateurs sur le temps (transition d'états) et sur les choix de branches (non-déterminisme). Plusieurs opérateurs temporels sont introduits dans CTL* : $X\phi$ signifiant que ϕ est vraie à la transition suivante, $G\phi$ pour ϕ est toujours vraie, $F\phi$ pour ϕ finit par devenir vraie, et $\phi\ U\ \psi$ pour ϕ est vraie jusqu'à ce que ψ devienne vraie. Deux quantificateurs de chemins sont introduits pour raisonner sur le non-déterminisme : $A\phi$ qui signifie que ϕ est vraie sur tous les chemins, et $E\phi$, pour ϕ est vraie sur au moins un chemin. Dans cette logique, $F\phi$ est équivalent à $true\ U\ \phi$, $G\phi$ est équivalent à $\phi\ W\ false$ et nous avons les propriétés de dualité : $\neg EF(\phi) = AG(\neg\phi)$, $\neg E\phi\ U\ \psi = A\neg\psi\ W\ \neg\phi$ et $\neg E\phi\ W\ \psi = A\neg\psi\ U\ \neg\phi$

Dans le fragment CTL de CTL*, 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. Comme montré dans [8], le fragment CTL a une expression suffisante pour exprimer un très large éventail de propriétés biologiques :

Sur l'accessibilité. Existe-t-il un chemin pour produire (c-à-d 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 $\neg(E((\neg(Q)UP \sim \{s\})))$ est abrégée en `checkpoint(Q,P~{s})`.

Sur la stationnarité et la stabilité. Est-ce qu'un certain état s (même partiellement décrit) de la cellule est un état stationnaire (resp. stable) ? $s \Rightarrow EG(s)$ (resp. $s \Rightarrow AG(s)$).

Sur les oscillations. Le système peut-il exhiber un comportement cyclique sur la présence de P ? $EG((P \Rightarrow EF\ \neg P) \wedge (\neg P \Rightarrow EF\ P))$, abrégé en `loop(P, !P)`. Sans équité forte, cette formule n'est en fait qu'une approximation. En effet, elle est vraie lorsqu'il existe un chemin sur lequel à chaque point lorsque P est présent, il disparaît à un moment donné et lorsque P est absent il réapparaît à un moment donné.

Le langage de requête CTL pour les modèles booléens est implanté en BIOCHAM avec une interface au vérificateur de modèles symbolique NuSMV [12].

Une extension de la logique temporelle du temps linéaire (*Linear Time Logic*, LTL) avec contraintes arithmétiques est utilisée pour la sémantique (déterministe) des concentrations de molécules. Cette approche est similaire à celle de Antoniotti et al [3] utilisée dans BioSpice. LTL est une restriction de CTL qui utilise uniquement les opérateurs temporels (pas les quantificateurs de chemin), et est donc appropriée pour raisonner sur les systèmes déterministes comme les modèles cinétiques. Les mêmes propriétés temporelles sont exprimées en LTL, aux différences près qu'un seul chemin est considéré à la fois. De façon pratique, la simulation du modèle cinétique fournit une série de points. Chaque point décrit les valeurs des concentrations de chaque composant ainsi que leur dérivée. Cette série de points forme le modèle pour les requêtes LTL. Les formules de base à partir desquelles les requêtes LTL sont construites, sont faites avec des contraintes arithmétiques sur les concentrations ou leur dérivées (comme `[Q]` $>$ `[Q~{s}]` ou `d([P])/dt` $<$ `0`).

Les requêtes d'*accessibilité* sont formalisées par l'opérateur F, par exemple : La concentration de la protéine Q phosphorylée sur s peut-elle être dépassée par celle de la protéine Q ? $F([Q] > [Q \sim \{s\}])$. Les requêtes d'*oscillation* vérifient que la dérivée de la concentration d'une molécule P alterne entre positive et négative n fois avec $F((d[P]/dt > 0) \wedge F((d[P]/dt < 0) \wedge F((d[P]/dt > 0) \ldots)))$, abrégée par `oscil(P,n)`.

3 Apprentissage de règles de réaction à partir de formules CTL

Les biologistes des systèmes bâtissent des modèles décrivant les interactions biomoléculaires à partir d'expériences réalisées sur des organismes sauvages ou mutés. Ces expériences définissent les propriétés que leurs modèles doivent satisfaire.

Dans notre démarche, les propriétés biologiques du système sont formalisées en logique temporelle CTL [8]. Cet ensemble de formules CTL définissent la spécification du modèle. Ceci nous a conduit à développer des techniques d'apprentissage automatique originales permettant de proposer des règles à ajouter au modèle, ou à retirer, de façon à satisfaire la spécification. Un schéma de règles (le biais) décrivant les règles possibles à ajouter au modèle, est donné pour guider la recherche vers de nouvelles règles, éliminant d'avance les règles n'ayant pas de sens biologique.

Après des essais infructueux utilisant les outils de programmation logique inductive (difficultés liées à la complexité d'expression des propriétés temporelles les plus simples), nous avons développé un algorithme d'apprentissage énumératif ad-hoc et une version améliorée [11] dans la cadre conceptuel de la révision de théorie [14].

L'algorithme énumératif pour ajouter (resp. enlever) une règle consiste simplement à générer toutes les instances du schéma de règles à ajouter, et à essayer ces règles séquentiellement en les ajoutant (resp. enlevant) au modèle et en testant la spécification CTL par vérification symbolique du modèle. Les règles qui vérifient toutes les spécifications sont retournées en réponse et proposées à l'utilisateur pour ajout (resp. retrait). La commande `learn_one_rule(pattern_rule,...)` (resp. `learn_one_deletion(pattern_rule,...)`) utilise cet algorithme. Ces algorithmes ne permettent d'ajouter ou d'enlever qu'une seule règle. Souvent, nous avons besoin d'ajouter et d'enlever plusieurs règles.

L'algorithme d'apprentissage par révision de théorie utilise de façon active les contraintes temporelles de façon à limiter la phase d'énumération et à guider la recherche pour l'ajout ou bien le retrait de règles du modèle. A cette fin, nous avons introduit les notions de formules CTL positives et négatives [11] qui s'avère être des formules ECTL et ACTL [13] :

Definition 1 *p est une* **formule ECTL** *ou* **formule positive** *si et seulement si*
- *p est une formule propositionnelle,*
- *$p = c_1 \mid c_2$ où c_1 et c_2 sont des formules positives,*
- *$p = c_1 \,\&\, c_2$ où c_1 et c_2 sont des formules positives,*
- *$p = EX(c)$ où c est une formule positive,*
- *$p = EF(c)$ où c est une formule positive,*
- *$p = EG(c)$ où c est une formule positive,*
- *$p = E(c_1 \; U \; c_2)$ où c_1 et c_2 sont des formules positives,*
- *$p = E(c_1 \; W \; c_2)$ où c_1 et c_2 sont des formules positives.*

Les formules positives qui sont vraies restent vraies après l'ajout de règles, c'est-à-dire qu'elles se conservent par augmentation de la structure de Kripke [11]. A l'inverse les formules négatives qui sont vraies restent vraies après la suppression de règles, c'est-à-dire qu'elles se conservent par restriction de la structure de Kripke. On obtient donc par dualité la définition d'une **formule ACTL** ou **formule négative**.
Les formules qui ne sont pas classifiables comme $AG(EF(\phi))$ sont des **formules UCTL**.

Nous définissons l'algorithme de révision de théorie pour BIOCHAM par un système de transitions concurrent. L'état du système est défini par un triplet $< QT, Q, R >$ tel que
- R est le modèle (un ensemble de règles de réaction et son état initial).
- $QT = (\overline{E}, \overline{U}, \overline{A})$ représente l'ensemble des formules déjà traitées et satisfaites dans le modèle courant R triées en 3 groupes : $\overline{E}$ représente les formules ECTL, $\overline{U}$ les formules UCTL et $\overline{A}$ les formules ACTL.
- $Q = (E, U, A)$ représente l'ensemble des formules à traiter triées en 3 groupes : E représente les formules ECTL, U les formules UCTL et A les formules ACTL.

Dans la suite, nous noterons Ra (resp. Re) un ensemble de règles BIOCHAM qu'on ajoute (resp. enlève), e une formule ECTL, u une formule UCTL et a une formule ACTL. L'ensemble des règles à enlever Re est un sous-ensemble des règles du modèle R. Les règles de Ra sont de taille limitée (le nombre de réactants et le nombre de produits sont majorés), de plus la taille des polymères est limitée.

Il y a plusieurs ensembles de règles possibles à ajouter Ra ou à enlever Re, ce qui crée des points de choix, de même que le choix entre les transitions 2 et 3. Lorsque le système échoue il retourne au dernier point de choix.
État initial : $< (\overline{\varnothing}, \overline{\varnothing}, \overline{\varnothing}), (E, U, A), R >$
transition 1 : $< (\overline{E}, \overline{U}, \overline{A}), (E \cup \{e\}, U, A), R > \rightarrow < (\overline{E \cup \{e\}}, \overline{U}, \overline{A}), (E, U, A), R \cup Ra >$
transition 2 : $< (\overline{E}, \overline{U}, \overline{A}), (\varnothing, U \cup \{u\}, A), R > \rightarrow < (\overline{E}, \overline{U \cup \{u\}}, \overline{A}), (\varnothing, U, A), R \cup Ra >$
transition 3 : $< (\overline{E}, \overline{U}, \overline{A}), (\varnothing, U \cup \{u\}, A), R \cup Re > \rightarrow < (\overline{E}, U \cup \{u\}, \overline{A}), (\varnothing, U, A), R >$
transition 4 : $< (\overline{E \cup E_p}, \overline{U \cup U_p}, \overline{A}), (\varnothing, \varnothing, A \cup \{a\}), R \cup Re > \rightarrow < (\overline{E}, \overline{U}, A \cup \{a\}), (E_p, U_p, A), R >$
État final : $< (\overline{E}, \overline{U}, \overline{A}), (\varnothing, \varnothing, \varnothing), R >$

Dans BIOCHAM, par souci d'efficacité :
Ra est limité à une règle unique contenue dans le biais. Re a une cardinalité maximale de 3 et est inclus dans l'ensemble des chemins de contre-exemple de la formule qui est traitée.
La commande `theory_revision(pattern_rule,...)` utilise cet algorithme.

4 Apprentissage de paramètres cinétiques à partir de formules LTL avec contraintes

De la même manière que pour l'apprentissage de règles booléennes à partir de propriétés CTL, il est possible d'utiliser une spécification LTL avec contraintes arithmétiques pour l'apprentissage 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.

Par exemple, la commande `trace_get([kd1,kr1],[(400,4000),(100,1000)],20,oscil(P,3),40)` essaye pour chacun des deux paramètres (`kd1` et `kr1`) 20 valeurs différentes dans leur intervalle respectif $[400, 4000]$ et $[100, 1000]$, et ce de telle manière que `P` oscille au moins 3 fois en 40 unités de temps.

D'une certaine manière, le processus d'apprentissage répète ce que le modélisateur fait à la main, c'est-à-dire essayer différentes valeurs de paramètres qui correspondent à un intervalle que le modélisateur juge adéquat ou à des formes de courbes d'activité de protéines qu'il (ou elle) souhaite reproduire. L'algorithme de la méthode d'apprentissage permet de tester des espaces de paramètres plus rapidement après formalisation de la forme de la courbe en spécification LTL.

5 Exemple d'une boucle négative

Les méthodes d'apprentissage sont illustrées dans cette section à partir d'un modèle simple décrivant une boucle négative. Les deux méthodes d'apprentissage de règles et de paramètres sont utilisées pour arriver à un modèle cinétique qui correspond au comportement expérimental observé : avec une cinétique et des valeurs de paramètres appropriées, ce type de modèles oscille.

Un réseau simple composé de trois protéines est choisi. Les trois protéines sont présentes sous deux formes, active (`P`, `Q` et `R`) et inactive (`P~{s}`, ...). Les réactions connues sont telles que `P` (resp. `R`, `Q`) est responsable de l'inactivation de `R` (resp. `Q`, `P`).

Un modèle BIOCHAM est écrit de la manière la plus simple, en utilisant des lois d'action de masse et des valeurs de paramètres arbitraires :

```
rule1 : kax*[P~{s}]   for P~{s} => P.
rule2 : kix*[P]*[Q]   for P=[Q]=> P~{s}.

rule3 : kay*[Q~{s}]   for Q~{s} => Q.
rule4 : kiy*[Q]*[R]   for Q=[R]=> Q~{s}.

rule5 : kaz*[R~{s}]   for R~{s} => R.
rule6 : kiz*[R]*[P]   for R=[P]=> R~{s}.

parameter(kax,0.1).   parameter(kix,1.5).
parameter(kay,0.4).   parameter(kiy,1).
parameter(kaz,0.2).   parameter(kiz,1).
```

Pour simuler le modèle BIOCHAM, un ensemble de conditions initiales est donné. Le modèle est accompagné d'une liste de spécifications CTL et LTL qui prennent en compte les résultats expérimentaux. Le modèle est considéré comme correct quand toutes les spécifications sont vérifiées. Dans cet exemple, les spécifications indiquent que `P` est atteignable, que les formes actives et inactives de `P` s'alternent, et que `Q` est un checkpoint

dans le processus d'inhibition de `P` (de même pour `Q` et `R`) . Les spécifications LTL demandent que le système oscille (au moins 3 fois en 40s) avec une certaine amplitude.

```
present(P,1).  present(Q,1).  present(R,1).
absent(P~{s}). absent(Q~{s}). absent(R~{s}).
add_specs({
Ei(reachable(P)),     Ei(reachable(Q)),...
Ei(reachable(P~{s})), Ei(reachable(Q~{s})),...
Ai(loop(P,P~{s})),    Ai(loop(Q,Q~{s})),...
Ai(checkpoint(Q,P~{s})),
Ai(checkpoint(R,Q~{s})),
Ai(checkpoint(P,R~{s}))}).
```

Avec ces spécifications, le modèle booléen est en accord avec le comportement attendu, mais le modèle cinétique ne montre pas d'oscillations. Après une recherche dans l'espace de paramètres (utilisant `trace_get` pour `oscil(P,3)`), aucune valeur n'est trouvée (cf. Figure 1 à gauche). Ce résultat négatif suggère que les règles doivent être modifiées.

Afin de créer des oscillations, il semble nécessaire que l'expression cinétique de certaines interactions soit non linéaire. Une variable Q est choisie et on impose une étape intermédiaire à son inactivation telle que Q forme un complexe avec une enzyme $E1$ qu'on formalisera par la formule CTL `Ai(checkpoint(Q-E1,Q s))`. On complète l'état initial par : `present(E1)` et `absent(Q-E1)`. Il faut donc ajouter que le complexe Q-E1 est atteignable `Ei(reachable(Q-E1))` à la spécification du modèle. Le modèle ne satisfait plus sa spécification, pour le corriger nous lançons la commande :
`theory_revision(elementary_interaction_rules).`

```
...
add E1-Q=>E1+Q
specification as undefined formula added : Ai(AG(P->EF(P~{s})&(P~{s}->EF(P))))
...
delete Q=[R]=>Q~{s}
specification as negative formula added : Ai(!(E(!(Q-E1) U Q~{s})))
add Q=[E1-Q]=>Q~{s}
specification as undefined formula added : Ai(AG(Q->EF(Q~{s})&(Q~{s}->EF(Q))))
...
backtrack delete E1-Q=[Q]=>E1+Q~{s}
add E1-Q=[Zp]=>E1+Q~{s}
specification as undefined formula added : Ai(AG(Q->EF(Q~{s})&(Q~{s}->EF(Q))))
...
Time: 680.00 s
14 queries treated
All the specifications are true

rule1 : kax*[P~{s}] for P~{s}=>P.
rule2 : kix*[P]*[Q] for P=[Q]=> P~{s}.
rule3 : kay*[Q~{s}] for Q~{s}=>Q.
rule5 : kaz*[R~{s}] for R~{s}=>R.
rule6 : kiz*[R]*[P] for R=[P]=> R~{s}.
E1+Q=>E1-Q.
E1-Q=>E1+Q.
E1-Q=[R]=>E1+Q~{s}.
```

BIOCHAM a donc enlevé la règle 4 et ajouté une règle réversible `E1+Q<=>E1-Q` de complexation de la protéine `Q` avec l'enzyme `E1` et une règle qui inactive `Q`, `E1-Q=[R]=>E1+Q~{s}`. On peut dire que BIOCHAM a transformé la règle (4) de phosphorylation de `Q` en une règle en deux étapes.

Cette réaction en deux étapes ressemble à une cinétique de Michaelis-Menten. Comme première approximation, le modélisateur peut choisir des paramètres ainsi :

```
(1) (ka1*[Q]*[E1],kd1*[Q-E1]) for Q+E1 <=> Q-E1.
(2) kr1*[Q-E1]*[R]    for Q-E1 =[R]=> Q~{s}+E1.

parameter(ka1,5e6)).      parameter(kd1,4000).
parameter(kr1,1000).
present(E1,0.001).        absent(Q-E1).
```

Un appel à `trace_get` échoue cependant à nouveau, ne trouvant aucune valeur pour les paramètres qui fasse osciller le système.

Nous faisons donc l'hypothèse que la réaction inverse (l'activation de Q) nécessite aussi d'être non linéaire : Q doit être activée par une étape intermédiaire de la façon suivante. Une enzyme $E2$ est introduite :

```
(ka2*[Q~{s}]*[E2],kd2*[Q~{s}-E2])
                  for Q~{s}+E2 <=> Q~{s}-E2.
kr2*[Q~{s}-E2]    for Q~{s}-E2 => Q+E2.

present(E2,0.001).         absent(Q~{s}-E2).
```

La méthode d'apprentissage est également appliquée aux paramètres `kd2` et `kr2` avec les autres paramètres fixés :

```
biocham: parameter(ka2,2e6).
biocham: trace_get([kd2,kr2],[(1000,5000),(0,1000)],20,oscil(P,3),40).
Found parameters that make oscil(P,3) true:
parameter(kd2,1000).
parameter(kr2,250).
```

La solution proposée ne montrant pas une forte inactivation de P (cf. Figure 1, au centre), nous raffinons la spécification est raffinée en ajoutant `F([P]<0.2)`.

En un temps de l'ordre de la minute, les paramètres suivants sont proposés :

```
parameter(kd2,1000).    parameter(kr2,400).
```

Avec ces paramètres, le système oscille en accord avec la spécification (cf. Figure 1, à droite).

L'utilisation combinée de ces deux méthodes d'apprentissage assiste le modélisateur en trouvant les meilleurs règles et paramètres qui décrivent le comportement recherché. Cependant, le modélisateur reste actif pendant le processus de recherche de règles et de paramètres. De plus, une bonne connaissance de l'ordre de grandeur de la valeur des paramètres, est nécessaire pour avoir un temps de recherche de paramètres le plus court possible.

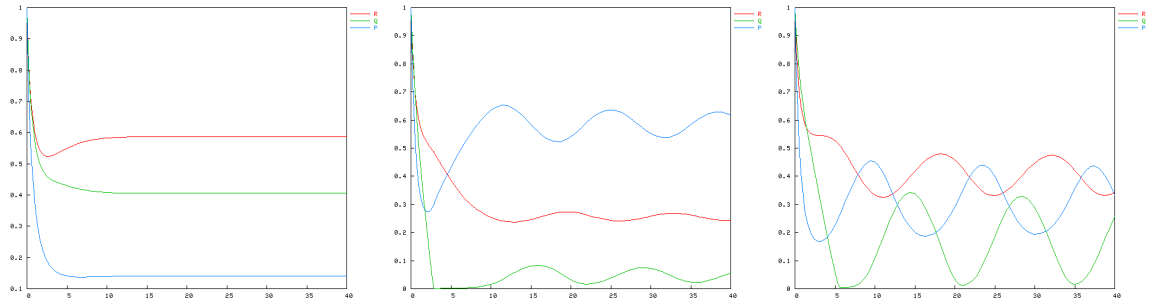

FIG. 1. Résultat des simulations avant et après l'apprentissage, puis après raffinement de la spécification.

6 Conclusion

Nous avons montré que dans la machine abstraite biochimique BIOCHAM, le langage de règles utilisé pour modéliser les interactions biomoléculaires, et la logique temporelle utilisée pour formaliser les propriétés biologiques du système, peuvent être combinés dans un processus d'apprentissage semi-automatique permettant de découvrir de nouvelles règles de réaction et d'estimer des valeurs de paramètres cinétiques.

Les exemples simples que nous avons montrés dans l'article sont des exemples pédagogiques. Cependant l'utilisation en cours de BIOCHAM sur des modèles couplés du cycle cellulaire, du cycle circadien et de la pharmacologie anticancéreuse confirme l'utilité du système d'apprentissage automatique sur des problèmes de modélisation en vraie grandeur. En particulier l'apprentissage des paramètres cinétiques à partir de la forme des courbes attendues, formalisée en logique temporelle avec contraintes, fournit une technique originale et relativement efficace d'estimation des paramètres.

Remerciements

Cette recherche s'effectue en partie dans le cadre du projet européen APRIL 2 `http ://www.aprill.org`.

Références

[1] N. Angelopoulos and S. H. Muggleton. Machine learning metabolic pathway descriptions using a probabilistic relational representation. *Electronic Transactions in Artificial Intelligence*, 7(9), 2002. also in Proceedings of Machine Intelligence 19.

[2] N. Angelopoulos and S. H. Muggleton. Slps for probabilistic pathways : Modeling and parameter estimation. Technical Report TR 2002/12, Department of Computing, Imperial College, London, UK, 2002.

[3] M. Antoniotti, A. Policriti, N. Ugel, and B. Mishra. Model building and model checking for biochemical processes. *Cell Biochemistry and Biophysics*, 38 :271–286, 2003.

[4] G. Batt, D. Bergamini, H. de Jong, H. Garavel, and R. Mateescu. Model checking genetic regulatory networks using gna and cadp. In *Proceedings of the 11th International SPIN Workshop on Model Checking of Software SPIN'2004*, Barcelona, Spain, Apr. 2004.

[5] G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. A fruitful application of formal methods to biological regulatory networks : Extending thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3) :339–347, 2004.

[6] C. H. Bryant, S. H. Muggleton, S. G. Oliver, D. B. Kell, P. G. K. Reiser, and R. D. King. Combining inductive logic programming, active learning and robotics to discover the function of genes. *Electronic Transactions in Artificial Intelligence*, 6(12), 2001.

[7] L. Calzone, N. Chabrier-Rivier, F. Fages, L. Gentils, and S. Soliman. Machine learning bio-molecular interactions from temporal logic properties. In G. Plotkin, editor, *CMSB'05 : Proceedings of the third Workshop on Computational Methods in Systems Biology*, 2005.

[8] N. Chabrier and F. Fages. Symbolic model checking of biochemical networks. In C. Priami, editor, *CMSB'03 : Proceedings of the first Workshop on Computational Methods in Systems Biology*, volume 2602 of *Lecture Notes in Computer Science*, pages 149–162, Rovereto, Italy, Mar. 2003. Springer-Verlag.

[9] N. Chabrier-Rivier, M. Chiaverini, V. Danos, F. Fages, and V. Schächter. Modeling and querying biochemical interaction networks. *Theoretical Computer Science*, 325(1) :25–44, Sept. 2004.

[10] N. Chabrier-Rivier, F. Fages, and S. Soliman. The biochemical abstract machine BIOCHAM. In V. Danos and V. Schächter, editors, *CMSB'04 : Proceedings of the second Workshop on Computational Methods in Systems Biology*, volume 3082 of *Lecture Notes in BioInformatics*, pages 172–191. Springer-Verlag, 2004.

[11] N. Chabrier-Rivier, F. Fages, S. Soliman, and L. Calzone. Learning transition rules from temporal logic properties. Research Report 5543, INRIA, Apr. 2005.

[12] A. Cimatti, E. Clarke, F. G. Enrico Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2 : An opensource tool for symbolic model checking. In *Proceedings of the International Conference on Computer-Aided Verification, CAV'02*, Copenhagen, Danmark, July 2002.

[13] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[14] L. de Raedt. *Interactive Theory Revision, an inductive Logic Programming Approach*. Knowledge-Based Systems. academic press, 1992.

[15] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and M. K. Sönmez. Pathway logic : Symbolic analysis of biological signaling. In *Proceedings of the seventh Pacific Symposium on Biocomputing*, pages 400–412, Jan. 2002.

[16] F. Fages, S. Soliman, and N. Chabrier-Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4(2) :64–73, Oct. 2004.

[17] G. Giaever et al. Functional profiling of the saccharomyces cerevisiae genome. *nature*, 418 :387–391, Jul 2002.

[18] M. Kanehisa and S. Goto. KEGG : Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1) :27–30, 2000.

[19] I. M. Keseler, J. Collado-Vides, S. Gama-Castro, J. Ingraham, S. Paley, I. T. Paulsen, M. Peralta-Gil, and P. D. Karp. EcoCyc : a comprehensive database resource for escherichia coli. *Nucleic Acids Research*, 33 :D334–D337, 2005.

[20] K. W. Kohn. Molecular interaction map of the mammalian cell cycle control and DNA repair systems. *Molecular Biology of the Cell*, 10(8) :703–2734, Aug. 1999.

[21] J. R. Koza, W. Mydlowec, G. Lanza, J. Yu, and M. A. Keane. Reverse engineering of metabolic pathways from observed data using genetic programming. In *Proceedings of the 6th Pacific Symposium on Biocomputing (PSB 2001)*, pages 434–445, Hawaii, USA, Jan. 2001.

[22] P. Langley, H. A. Simon, G. L. Bradshaw, and J. M. Zytkow. *Scientific Discovery : Computational Explorations of the Creative Processes*. MIT Press, Cambridge, MA, Feb. 1987.

[23] R. Mardare and C. Priami. Logical analysis of biological systems. *Fundamenta Informaticae*, 64 :271–285, 2005.

[24] S. H. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13 :245–286, 1995.

[25] L. Todorovski and S. Džeroski. Theory revision in equation discovery. In K. P. Jantke and A. Shinohara, editors, *Proceedings of the 4th International Conference on Discovery Science, DS 2001*, volume 2226 of *Lecture Notes in Artificial Intelligence*, pages 389–400, Washington, DC, USA, 2001. Springer-Verlag.

[26] M. Tomita et al. The E-CELL project : Towards integrative simulation of cellular processes. *New Generation Computing*, 1(18) :1–12, 2000.

[27] VBI and EML Research. *COPASI's manual*.