"Model Checking" symbolique de réseaux biochimiques

Nathalie Chabrier, François Fages*

9 mars 2003

Résumé

Le "model checking" est une méthode automatique permettant de décider si un circuit ou un programme, exprimé comme un système de transitions concurent, satisfait un ensemble de propriétés exprimées dans une logique temporelle telle que CTL. Dans ce papier nous montrons que l'utilisation du "model checking" symbolique apporte des avantages par rapport à la simulation pour questionner et valider des modèles formels de processus biologiques. Nous reportons nos expériences d'utilisisation du "model checker" symbolique NuSMV et du "model checker" avec contraintes DMC pour modéliser et questionner deux processus biologiques : un modèle qualitatif du contrôle du cycle cellulaire des mammifères d'après les diagrammes de Kohn,et un modèle qualitatif sur la régulation de l'expression des gènes.

1 Introduction

Ces dernières années, la biologie a clairement engagé un travail d'élucidation des processus biologiques de haut niveau en termes de leur base biochimique au niveau moléculaire. La production en masse des données post-génomiques, telles que l'expression d'ARN, la production de protéines et les interactions protéine-protéine, soulève le besoin en parallèle d'un réel effort sur la représentation formelle des *processus* biologiques. Les réseaux de métabolisme, les voies de signalisation extracellulaires et intracellulaires, ainsi que les réseaux de régulation d'expression de gènes, sont des systèmes dynamiques très complexes. Les bases de données avec des informations qualitatives et quantitatives sur la dynamique des systèmes biologiques, ne seront pas suffisantes pour intégrer et employer efficacement les connaissances actuelles au sujet de ces systèmes. La conception des outils formels pour *modéliser* des processus biomoléculaires et pour raisonner au sujet de leur dynamique semble être une étape obligatoire de recherches auquelle le champ de la vérification formelle en informatique peut énormément contribuer. On a proposé ces dernières années plusieurs formalismes pour modéliser des réseaux biochimiques. Regev et Shapiro [?] étaient les premiers à proposer l'utilisation d'un langage concurrant formel, à savoir le

* Cet article est une version révisée de [?]. Le travail a été réalisé dans le cadre de l'Action Coopérative de Recherche "Calculs de Processus et processus biologiques", ARC CPBIO, `http://contraintes.inria.fr/cpbio`

π-calculus de Milner, pour modéliser un processus biochimique tel que la cascade de RTK/MAPK. Le bio-calcul de [?] présente une syntaxe orientée biologie pour un calcul semblable. Plus récemment, des modèles quantitatifs de processus biochimiques ont été développés avec des réseaux de Pétri hybrides [?, ?], des langages concurrants avec contrainte hybrides [?], et des automates hybrides [?, ?].

Dans cet article nous proposons d'aller au delà de la simulation et de nous concentrer sur la question de fournir des méthodes automatisées pour *questionner et valider les modèles formels* en biologie des systèmes. Plus spécifiquement, nous proposons,

- l'utilisation de la logique temporelle CTL comme langage d'interrogation des modèles de processus biologiques,
- l'utilisation des systèmes de transitions concurrents pour modéliser ces processus biologiques.
- l'utilisation des techniques de "model checking" symbolique pour évaluer automatiquement les requêtes CTL sur des modèles qualitatifs ou quantitatifs,

Notre démarche sera illustrée par deux exemples: un modèle qualitatif du contôle du cycle cellulaire des mammifères d'après les diagrammes de Kohn [?, ?], et un modèle quantitatif de l'expression gènique.

1.1 Exemple 1: Le Contrôle du Cycle Cellulaire des Mammifères

Dans cet exemple, les acteurs principaux sont les gènes, les protéines avec leurs sites de phosphorylation, les complexes multimoléculaires, et les membranes qui compartimentent la cellule. Les molécules interagissent pour produire de nouvelles protéines (synthèse), former des complexes multimoléculaires (complexation), modifier des protéines (phosphorylation et dephosphorylation), dégrader ou transporter des molécules.

Le cycle cellulaire des eukaryotes est divisé en deux phases principales. La phase de division cellulaire (phase de mitose) pendant la quelle la cellule mère se divise en deux cellules fille qui possèdent le même patrimoine gènétique que leur parente. La seconde phase principale est l'interphase qui occupe 90% du temps du cycle cellulaire. Cette phase est découpée en trois sous-phases. Une phase de croissance cellulaire (la phase G1) pendant laquelle elle peut cesser d'évoluer et rester dans un état quiescent nommé G0. Une phase de synthèse (la phase S) pendant laquelle le patrimoine gènétique de la cellule est dupliqué. Et une phase de vérification (la phase G2) pendant laquelle s'il y a eu déterioration de l'ADN, il y a réparation de l'ADN, ou si cela s'avère impossible, il y a mort cellulaire.

Chaque phase est caractérisée par l'activité de deux principaux types de protéines: les cyclines et cyclin-dependant-kinase (Cdk). L'activité des Cdk nécessite la complexation avec à une cycline. Puis cette activité est contrôlée par des inhibiteurs ou activateurs spécifiques grâce à des phosphorylations, déphosporylations ou complexations. Les complexes cdk-cycline peuvent aussi influencer (directement ou indirectement) leurs activateurs et inhibiteurs. On parle de boucles de rétrocontrôle positifs ou négatifs selon les effets de la boucle.

Un état de la cellule est défini par des données générales comme le pH et la température et par la valeur de ces acteurs. Cette valeur peut être juste la présence ou l'absence des molécules, ou leur nombre, ou leur concentration dans chaque partie de la cellule. Notez qu'un ensemble d'états peut être représenté juste par une information partielle sur les valeurs réelles des variables d'état, comme par exemple des intervalles ou des contraintes entre les variables.

Dans la suite, nous nous intéressons à différents types de questions biologiques :

Sur l'accéssibilité :

1 Soit un état initial *init*, existe-t-il un chemin pour synthétiser une protéine P?

2 Quels sont les états initiaux à partir desquels un ensemble de protéines S peuvent être produites?

Sur les chemins :

3 La cellule peut-elle atteindre un état s sans passer par un autre état s_2?

4 L'état s_2 est-il une étape indispensable ("checkpoint") pour atteindre l'état s?

5 La cellule, peut-elle atteindre l'état s sans violer certaines contraintes c?

6 A partir d'un état initial *init*, est-il possible de synthétiser une protéine P sans créer ni utiliser la protéine Q?

Sur les états stables :

Par "état stable" on entend état permanent ou état stationnaire selon si l'on prend le sens fort ou non.

7 Est-ce qu'un certain (partiellement décrit) état s de la cellule est un état stable?

8 La cellule peut-elle atteindre l'état stable s à partir de l'état initial *init*?

9 La cellule doit-elle atteindre l'état stable s à partir de l'état initial *init*?

10 Quels sont les états stables?

Sur le temps :

11 Combien de temps faut-il pour qu'une molécule soit activée?

12 En un temps donné, combien de cyclines A peuvent être accumilées?

13 Combien de temps dure une phase donnée du cycle?

Sur la correction du modèle :

14 *Peut-on voir les incohérences du modèle et les corriger?*

1.2 Exemple 2: La Régulation de l'Expression Génique

Les gènes sont les acteurs qui permettent la synthèse de protéines. On dit qu'un gène est actif lorsque sa production à dépasser un certain seuil. On dit qu'un gène est activé (resp inhibé) lorsque qu'une molécule l'aide (resp le gène) à synthétiser. Les gènes, via leur production, se régulent entre eux par activation et inhibition. On parle de réseaux de régulation génique.

Comme noté dans [?, ?], la dynamique des réseaux de régulation de gènes peut être modélisée par un système d'équations différentielles de la forme:

$$\dot{x}_i = f_i(\vec{x}) - g_i(\vec{x}) * x_i,\ x_i \geq 0,\ 1 \leq i \leq n,$$

où $\vec{x}$ est un vecteur de variables représentant des concentrations cellulaires de la production des gènes (protéines et ARN messagers), ou bien des variables exogènes (Ph, température,...), $g_i(\vec{x})$ est le taux de dégradation de la protéine x_i, et f_i est une fonction fortement non linéaire qui exprime l'effet (activation, inhibition) des autres variables sur la synthèse de x_i. L'action d'un gène sur un autre gène met en évidence l'importance des paliers d'activation pour qu'il y ai influence. Les variables exogènes sont définies en plaçant $\dot{x}_i = 0$.

Pour cet exemple, nous nous intéressons aussi aux types de questions suivantes:

Sur l'activation :

15 Un protéine x peut-elle atteindre une concentration supérieur à un certain seuil de valeur τ?

16 Quelles états peuvent produire pour x une concentration supérieur au palier de valeur τ?

Sur les invariants :

17 Est-ce que le rapport c entre des concentrations peut toujours être satisfait?

1.3 Plan du papier

La section suivante présente la logique temporelle CTL que nous proposons d'utiliser. Cette démarche est illustrée par la formalisation des questions biologiques données précédement pour les deux cas étudiés dans ce papier. Quelques limites de CTL sont discutées à travers des questions biologiques non directement traduites en CTL.

Dans la section 3, nous nous focalisons sur un formalisme de systèmes de transitions concurrents pour la modélisation en base de règles des réseaux biochimiques. Ceci est illustré par un système de transitions sur des variables booléennes pour une partie (la boîte des cyclines) du contrôle du cycle cellulaire des mammifères, et par un système de transitions sur des nombres réels et des contraintes linéaires pour un exemple pédagogique d'intéraction entre gènes.

La section 4 présente l'algorithme de base du "model checking" ainsi que les variantes, comme le "model checking" symbolique et celui basé sur les contraintes, que nous utilisons pour questionner nos modèles biologiques. Cette section fournit quelques performances qui montre l'applicabilité de la méthode.

La section 5 donne quelques informations et références supplémentaires sur le travail effectué.

La dernière section présente notre conclusion.

2 La Logique Temporelle CTL comme une Langage de Requête pour les Modèles Biochimiques

2.1 Préliminaires sur CTL

La logique temporelle CTL, "Computation Tree Logic", est une logique pour décrire les propriétés des arbres de calcul et des systèmes de transitions non-déterministes [**?**]. CTL est une logique temporelle qui fait abstraction de la durée et qui décrit l'occurence des événements dans les deux dimensions du système: le temps(ordonnancement) et le non-déterminisme. CTL prolonge fondamentalement la logique *propositionelle* ou celle du *premier ordre* (FO) [**?**], avec deux quantifieurs de chemins pour le non déterminisme : A (All), signifie "pour tous les chemins de transitions", and E(Exist), signifie "pour au moins un chemin de transitions", et avec plusieurs opérateurs temporels : X (neXt), signifie "au temps suivant", F (Finaly) signifie "à un moment dans le futur", G (Generally) signifie "tout le temps", U (Until) signifie "jusqu'à ce que".

Une propriété de *"sûreté"*, indiquant qu'une certain situation décrite par une formule ϕ ne doit jamais se produire, est exprimé par la formule CTL $AG\neg\phi$, c.a.d. sur tous les chemins,ϕ est toujours (à chaque moment) fausse.

Une propriété de *vivacité"*, indiquant qu'une bonne situation décrite par la formule ψ se produira par la suite, est exprimée par la formule CTL $AF\psi$, c.à.d. sur tous les chemins ψ sera vraie à un moment donné. Notez que par dualité nous avons $EF\phi = \neg AG\neg\phi$ et $EG\phi = \neg AF\neg\phi$ pour n'importe quelle formule ϕ.

Formellement, les formules CTL* sont divisées entre les formules des états et celles des chemins. Soit AP un ensemble de propositions atomiques décrivant des états. Une *formule d'état* est une proposition atomique, ou une formule de chemin préfixé par un quantifieur de chemin, ou une combinaison logique de telles formules. L'ensemble des formules de chemin est la fermeture de l'ensemble des formules d'états par des opérateurs temporels et des connecteurs logiques. Les formules d'état et de chemin sont des formules CTL*. La logique CTL est un fragment syntaxique de CTL* dans lequel tout opérateur temporel doit être précédé par un quantifieur de chemin. Par exemple, $A(FG\phi)$ et $E(F\phi \wedge G\psi)$ sont des formules CTL* et ne sont pas exprimable en CTL. Dans cet article, nous ne concidéreront que le fragment des formules CTL.

La sémantique de CTL est donnée par une structure de Kripke. Une *structure de Kripke* K est un triplet (S,R,L) où S est un ensemble d'états, $R \subseteq S \times S$ est une relation (de transition) et $L : S \to 2^{AP}$ est une fonction qui associe à chaque état l'ensemble des propositions atomiques vraies dans cet état. Un chemin dans K à partir d'un état s_0 est une suite infinie d'états $\pi = s_0,s_1,...$ tel que $(s_i,s_{i+1}) \in R$ pour tout $i \geq 0$. Nous dénotons par π^i le suffixe de π commençant à s_i. Ci suit, la définition inductive de la relation de vérité posant qu'une formule CTL ϕ est vraie dans K à l'état s, noté $K,s \models \phi$, ou vrai dans K le long du chemin π, noté $K,\pi \models \phi$ (les règles standard pour les connections logique sont omises):

- $K,s \models \phi$ ssi $s \models \phi$, si ϕ est une formule d'état,
- $K,s \models E\phi$ ssi il y a un chemin π à partir de s tel que $K,\pi \models \phi$,
- $K,s \models A\phi$ ssi pour tout les chemins π à partir de s, $K,\pi \models \phi$,
- $K,\pi \models \phi$ ssi $s \models \phi$ où s est le premier état de π, si ϕ est une formule

d'état,

- $K,\pi \models X\phi$ ssi $K,\pi^1 \models \phi$,
- $K,\pi \models F\phi$ ssi il existe $k \geq 0$ tel que $K,\pi^k \models \phi$,
- $K,\pi \models G\phi$ ssi pour tout $k \geq 0$, $K,\pi^k \models \phi$,
- $K,\pi \models \phi U \psi$ ssi il existe $k \geq 0$ tel que $K,\pi^k \models \psi$ and $K,\pi^j \models \phi$ pour tout $0 \leq j < k$.

Suivant [?], étant donnée une structure de Kripke K, on peut identifier une formule CTL ϕ à l'ensemble des états qui la satisfont, c.à.d. $\{s \in S | K,s \models \phi\}$.

2.2 Exemple 1: Le Contrôle du Cycle Cellulaire des Mammifères

Suit l'exemple des questions biologiques listée dans la section précédente traduites en formule CTL :

Sur l'accéssibilité :

1 $init \Rightarrow EF(P)$,

2 $EF(S)$, la formule CTL est en effet une représentation de tous les états la satisfaisant, Les outils de "model checking" permette d'énumérer explicitement ces états.

Sur les chemins :

3 $EF(s_2 \wedge EFs)$,

4 $\neg E((\neg s_2)\ U\ s)$,

5 $E(c\ U\ s)$,

6 $init \Rightarrow E(\neg Q\ U\ P)$,

Sur les états stables :

7 $s \Rightarrow AG(s)$, un état stable dans le sens fort est un état dans lequel la cellule reste indéfiniment sans la possibilité de s'échapper, on parle d'état permanent. Un état stationnaire (dans lequel la cellule peut rester indéfiniment mais peut s'échapper), peut être modélisé par $s \Rightarrow EG(s)$,

8 $init \Rightarrow EF(AGs)$, ou $init \Rightarrow EF(EGs)$ au sens faible de l'état stable, on remarque que la requête au sens fort ne s'exprimerait pas en logique LTL,

9 $init \Rightarrow AF(AGs)$, ou $init \Rightarrow AF(EGs)$ au sens faible,

10 L'ensemble des états stables d'un système ne peut pas être représenté par une question en CTL. Dans CTL il est seulement possible de vérifier si un état (partiellement décrit) donné est un état stable. Une approche possible pour calculer l'ensemble des états stables (ou de points de contrôle (checkpoint), etc...) d'un réseau biochimique serait de combiner les méthodes de model checking avec des méthodes de recherche. Cette hybridation est un problème ouvert intéressant.

Sur le temps :

Le temps dans la logique temporelle CTL est une notion purement qualitative, basée sur une relation simple de précédence. Le raisonnement au sujet des durées n'est ainsi pas exprimable avec les opérateurs temporels de CTL. Néanmoins, si la logique CTL sous-jacente de la description de

l'état n'est pas propositionelle mais du premier ordre, il est toujours possible dans FO de modéliser des intervalles de temps en ajoutant à toutes les propositions atomiques des arguments numériques supplémentaires représentant le début et la durée de l'action. Le model checking basé sur les contraintes présenté dans la section 4.1 fournit une méthode automatique pour évaluer de telles questions.

Sur la correction du modèle : Quand une propriété prévue n'est pas vérifiée, le chemin montrer pour le contre exemple peut aider l'utilisateur à rafiner le modèle. De même, lorsqu'une propriété inattendue est vérifiée, le chemin servant de témoin aide l'utilisateur à rafiner son modèle en ajoutant des conditions supplémentaires aux règles, ou si la propriété n'est pas connue par les biologistes comme vraie ni comme fausse, le témoin peut suggérer des expériences biologique dans le but de valider ou d'invalider cette propriété du modèle. En biologie, la boucle standard entre la modélisation et la validation du modèle est en fait une boucle en trois point : modéliser, questionner le modèle et expérimenter biologiquement.

2.3 Exemple 2: La Régulation de l'Expression Génique

La seconde série de questions sur l'exemple d'interraction entre gènes peut être traduite en CTL comme suit: Il est important de noter que pour cette seconde série d'exemples, l'utilisation de la logique du premier ordre est utile pour exprimer les contraintes dans les requêtes CTL [?, ?].

Sur l'activation :

15 $init \Rightarrow EF(x > \tau)$,

16 $EF(x > \tau)$,

Sur les invariants :

17 $init \Rightarrow AG(c)$.

La remarque précédente, sur les outils pour corriger le modèle ou suggérer des expériences biologiques, s'applique tout aussi bien.

3 Modéliser des Réseaux Biochimiques avec des Systèmes de transitions concurrents

Les systèmes de transitions concurrents ont été introduits dans [?] pour raisonner sur des programes concurrents. Les systèmes de transitions concurents offrent une manière directe pour spécifier une structure de Kripke par des règles de réaction et nous les emploierons donc pour cette raison afin de modéliser des réseaux biochimiques.

Un *système de transitions concurent* est une structure de Kripke présentée comme un triplet $(\vec{x},I,R)$ où $\vec{x}$ est un nuplet de variables (de donnée ou de contrôle), I est une formule sur $\vec{x}$ qui exprime les conditions initiales comme un ensemble de valeurs pour toutes les variables, et R est un ensemble de règles condition-action. Les règles ont la syntaxe suivante:

$$condition\ \phi(\vec{x})\quad action\ \vec{x}' = \rho(\vec{x})$$

où $\phi(\vec{x})$ est les conditions sous lesquelles la règle est applicable, et la version prime des variables dénote les nouvelles valeurs $\rho(\vec{x})$ des variables après l'application de la règle. Par convention, les variables qui ne sont modifiées dans le coté droit de la règle garde leur valeur inchangée.

Clairement, un système de transition concurrent définit une structure de Kripke, où l'ensemble des états est l'ensemble de tout les nuplets des valeurs des variables, l'état initial est un nuplet de valeurs qui satisfont la condition initial, et la relation de transition est l'union [1] (c.a.d. la disjonction) des relations entre les états de toutes les instances des règles condition-action.

Dans la suite, nous associerons une variable de donnée à chaque molécule (protéine ou gène). La valeur d'une variable sera soit une valeur numérique exprimant la concentration de la molécule, soit une valeur booléene exprimant simplement la présence ou l'absence de la molécule. L'évolution temporelle du système sera modélisée par les pas de transition. Les différents chemins de transition modèliseront le comportement indéterministe du système.

3.1 Exemple 1 : Le contrôle du cycle cellulaire des mammifères

Dans [?], Kohn fournit, par diagramme anoté, un plan des intéractions moléculaires des systèmes du contrôle du cycle cellulaire et de réparation de l'ADN. La partie concernant le contrôle du cycle cellulaire et avec plus de détails la "Cyclin box" et la "E2F box" ont été modélisées dans l'ARC CPBIO par M. Chiaverini et V. Danos [?], en un ensemble de 732 règles de réaction sur 165 protéines, et 532 variables qui tiennent compte des différentes formes des protéines. La beauté de ce modèle est que chaque règle est une instance d'un de ces cinq schémas de règles suivants:

1. La Complexation : $A + B \rightarrow AB$,
   Deux molécules A et B se lient pour former un complexe multimoléculaire AB;
2. La Phosphorylation : $A + B \rightarrow Ap + B$,
   La molecule A est modifiée par l'action d'un catalyseur B, A est transformée en une forme phosphorylée Ap,
3. La Déphosphorylation : $Ap + B \rightarrow A + B$,
   La molécule phosphorylée Ap perd un élément phosphate par l'action du catalyseur B;
4. La Synthèse : $A \rightarrow A + B$,
   La molécule B est synthétisée à partir d'un gène dont le promoteur A est activé;
5. La Dégradation : $A + B \rightarrow A$,
   La molécule B est dégradée par la molécule A.

Dans ce modèle, chaque type de molécule est modélisé par une variable. Par manque de donnée quantitative, les variables associées aux molécules sont toutes booléenes. Elles expriment simplement la présence ou l'absence de la molécule dans la cellule. Ce modèle du contrôle du cycle cellulaire des mammifères est un

1. les systèmes de transitions concurents sont asynchrones dans le sens qu'une seule règle est exécutée par pas de temps (sémantique d'entrelacement), d'où la relation de transition est l'union des relations associées aux règles. Par ailleurs, on ne considère pas les programmes synchrones dans ce papier, mais, leur relation de transitions est définie par intersection.

modèle purement logique. Par exemple, CycH, Cdk7 et CycH-cdk7 sont trois variables représentant respectivement la cycline H, la "cyclin dependent kinase" sept et le dimer cycline H - cdk 7. Dans le schéma de la règle de compléxation, AB représente une variable propositionelle qui dénote le complexe multimoléculaire qui est le résultat de la liaison des molécules dénotées par A et B. Une instance de ce schéma est la règle CycH + Cdk7 → CycH-Cdk7. Les trimers, les teramers et plus généralement les polymers peuvent être formés en appliquant le schéma de complexation à un dimers, ect. A noter qu'il est possible de distinguer les complexes (AB)C et A(BC) s'il y a des raisons biologiques de faire cette distinction.

De façon similaire, on peut introduire une variable nommées Cdk1(phosphorylé sur la Thr14)-cyclinB, pour représenter la forme phosphorylée du dimer Cdk1-Cyclin B sur le site Thr 14 de la Cdk 1. La phosphorylation de ce dimer par la Mytosine 1 est modélisée par l'intance de règle: Cdk1-CycB + Myt1 → Cdk1(phosphorylé sur la Thr14)-CycB + Myt1. Une instance de la règle de déphosphorylation est: Cdk1(phosphorylé sur la Thr14 et sur la Tyr15)-CycB + Cdc25C(phosphorylé sur le domain N-terminal) → Cdk1-cyclinB + Cdc25C(phosphorylé sur le domaine N-terminal).

Afin d'être concis, nous utilisons, dans un schéma de règle pour dénoter les règles de condition-action, la convention suivante: La partie gauche d'une règle est juste sa condition. La partie droite est une formule qui exprime quelles variables deviennent vraies après l'action, avec la convention que les variables qui n'apparaissent pas dans l'instance de la règle restent inchangées, et que la variables qui n'apparaissent que dans la partie gauche de la règle prennent des *valeur arbitraire*. Le schéma de la règle de compléxation est par conséquant une abréviation des quatres règles de condition-action suivantes:

condition $A + B$ *action* $AB' = true, A' = true, B' = true$
condition $A + B$ *action* $AB' = true, A' = false, B' = true$
condition $A + B$ *action* $AB' = true, A' = true, B' = false$
condition $A + B$ *action* $AB' = true, A' = false, B' = false$

Les règles de condition-action permettent explicitement la disparition (la consommation) des molécules A et B par complexation.

3.2 Exemple 2 : La Régulation de l'Expression Génique

Suivant la méthode de Euler pour résoudre les équations différentielles numériquement, on peut associer un système de transitions discret, en gardant l'état infini, à un système d'équations différentielles.

Nous utiliserons l'exemple pédagogique pris dans [?] d'intéraction entre deux gènes suivant:

$\dot{y} = 0.01 * x$,
$\dot{x} = 0.01 - 0.02 * x$ si $y < 0.08$,
$\dot{x} = -0.02 * x$ si $y \geq 0.08$.

Le gène x active le gène y, mais à partir d'un certain palier, le gène y inhibe l'expression du gène x.

Une discrétisation par palier d'une unité de temps dt (e.g. $dt = 1$) permet d'avoir le système simple de transitions suivant:

condition $y < 0.8$ *action* $x' = x + (0.01 - 0.02 * x) * dt$, $y' = y + 0.01 * x * dt$
condition $y \geq 0.8$ *action* $x' = x - 0.02 * x * dt$, $y' = y + 0.01 * x * dt$

Le système de transitions de cet exemple est déterministe, mais il est important

de noter que ce n'est pas requi par la méthode. Noter que les dérivées peuvent être ajouter aux états du système dans le cas où leur expression explicite est requise. Une discrétisation dynamique est possible en ajoutant le palier de temps dt comme une variable d'état, similairement à la simulation à plusieurs vitesses dans les systèmes hybrides.

4 Le "Model checking" pour les Systèmes Biologiques

4.1 Préliminaires sur le "Model Checking"

Le "Model checking" est un algorythme pour calculer , dans une structure de Kripke K, l'ensemble des états qui satisfont une formule CTL donnée, c.à.d. l'ensemble $\{s \in S | K, s \models \phi\}$. Par soucis de simplicité, nous ne considérons que le fragment CTL de CTL*, et nous utilisons le fait que (par dualité) toutes les formules CTLpeuvent s'exprimer en termes de $\neg$, $\vee$, EX, EU et EG.

Quand K a un ensemble d'états fini, l'algorithme de "model checking", dans sa plus simple forme, travaille avec une représentation explicite de K comme un graphe de transitions, et étiquette chaque état avec un ensemble de sous-formules de ϕ qui est vraie pour cet état.

Premièrement, les états sont étiquettés avec les propositions atomiques de ϕ qui sont vraies dans ces états. L'étiquetage plus complexe est fait itérativement, suivant la syntaxe de la sous-formule de ϕ. Les formules de la forme $\neg\phi$ marquent les états qui ne sont marqués par ϕ. Les formules de la forme $\phi \vee \psi$ sont ajoutées aux étiquettes des états marqués par ϕ ou ψ. Les formules $EX\phi$ sont ajoutées aux étiquettes des états directement prédecesseurs des états marqués par ϕ Les formules $E(\phi U \psi)$ sont ajoutées aux états prédécesseurs de ψ tantqu'ils satisfont ϕ. Les formules $EG\phi$ demande le calcul des composantes fortement connextes du sous-graphe des transitions restreint aux états satisfaisant ϕ. Les états étiquetés par $EG\phi$ sont les états du sous-graphe pour lesquels il existe un chemin menant à un état dans une composante fortement connexte non triviale. La complexité de l'algorithme est $O(|\phi| * (|S| + |R|))$ où $|\phi|$ est la taille de la formule, $|S|$ est le nombre d'états, et $|R|$ est le nombre de transitions [**?**].

Le modèle checking symbolique est un algorithme plus efficace qui utilise une reprèsentation symbolique de la structure de Kripke finie avec des formules booléenes. En particulier, la relation de transition est entièrement codée par une seule formule booléene (disjonctive), les ensembles d'états sont encodés par des formules booléennes, et les diagrammes de décisions binaires ordonés (ordered binary decision diagrams (OBDDs)) sont utilisés comme formes canoniques pour les formules booléenes. L'algorithme de "model checking" symbolique calcul la représentation de l'ensemble des états en OBDD qui satisfont une formule CTL donnée. Ce calcul nécessite un calcul itératif du plus petit point fixe (pour EF) et du plus grand point fixe (pour EG) d'un opérateur simple de transformation des prédicats associé aux connecteurs temporels [**?**]. Dans nos expériences reportées plus bas, nous utilisons le "model checker" symbolique à-l'état-de-l'art NuSMV [**?**].

Le "model checking" basé sur les contraintes est appliqué sur les structures de Kripke à état infini, tel que les structures de Kripke avec des variables à intervales non bornés ou aux domaines numériques continus. Un états contraint

est une représentation finie utilisant les contraintes d'un ensemble fini ou infini d'états. Dans la méthode de Delzanno et Podelski [?], les structures de Kripke à état infini sont représentées par des programmes logiques avec contraintes, et les formules CTL, qui sont basées sur un fragment de la logique du premier-ordre, sont identifiées comme le plus petit point fixe et le plus grand point fixe de tels programes. Dans nos expériences reportées plus loin sur le modèle de régulation de l'expression génique, nous utilisons l'implémentation en Sicstus Prolog avec contraintes sur les domaines finis et les nombres réel (algorithme du simplex) du "model checker" DMC [?].

4.2 "Model Checking" symbolique sur les Modèles Logiques

Requête	Profondeur Recherche	Profondeur Preuve	DMC états	DMC temps (s)	NuSMV temps (s)
compilation	-	-	-	-	47.5
EF(CycE)	6	3	279	440	2
EF(SL1p)	10	5	2107	9950	29
EF(CycA)	6	2	1072	7632	1.9
EF(PCNA-CycD)	6	4	245	850	1.7
¬E(¬Cdc25c-a U Cdk1-CycB-a	-	4	-	-	2.2

TAB. 1 – *Evaluation de requête CTL dans le contrôle du cycle cellulaire des mammifères avec DMC et NuSMV.*

Le tableau 1 quelques idées de performance sur l'évaluation des requètes CTL dans le modèle du cycle cellulaire des mammifères. La première colonne indiquent les requêtes. La seconde colone indique la longueur du (plus court) chemin entre le premier état qui vérifie la propriété et et l'état le plus éloigné qui la vérifie aussi. La troisième colonne indique la longueur du chemin (contre-exemple ou témoin). La quatrième colonne indique le nombre d'états contraints calculés par DMC. Les deux dernières colonnes indiquent le temps CPU en seconde (mesuré sur un pentium 4 à 660Mhz pour DMC et un atlon 1800 pour NuSMV) mis pour calculer la réponse à la requête. Les temps d'exécution de DMC (ceux qui inclus la construction du chemin) montrent l'efficacité de la représentation des états en OBDD, comparée à la simple représentation des états en Prolog utilisée dans DMC.

4.3 Le "Modèle Checking" avec contraintes sur les modèles quantitatifs

Constraint-based model checking of quantitative models must be contrasted with symbolic model checking techniques which use finite domain abstraction techniques to deal with quantitative models [?]. The constraint-based model checker DMC [?] performs a backward reachability analysis, starting from a constrained state expressing the CTL property to prove, up to the computation of a state expression containing the initial state. Safety (resp. liveness) properties involve the computation of the least (resp. greatest) fixpoint of a constraint logic program.

It is worth noting that this kind of reasoning mixes symbolic computation on set of states described by *numerical constraints*, with a form of reasoning by induction. In the simple example of gene regulation, the query $EF(x \geq 0.5)$ immediately evaluates to false, as any predecessor state of a state described by the constraint $x \geq 0.5$ again satisfies that constraint and is thus subsumed.

The table below show some performance figures in the Prolog implementation of DMC. Better performance results could be obtained by using other discretizations of the problem, other translations involving Runge-Kutta method, and other implementations of the constraints.

Type	Request	Pathway length	Number of states	Computing time (sec)
16	EF(x≥0.5)	0	1	0.02
16	EF(x≥0.2)	28	29	3.65
16	EF(x≥0.45)	116	117	59
16	EF(y≥0.8)	212	213	256
16	EF(x+y≥1.3)	178	179	173
16	EF(x+y≥1.2)	194	195	206
17	AG(x>0.5)	1	2	0.03
17	AG(x<0.1)	14	127	8
17	AG(y<0.8)	211	421	7

TAB. 2 – *Examples of CTL queries in the example of gene interaction.*

5 Discussion and Related Work

The experiments reported in this paper should be read as a proof-of-concept rather than as providing an already usable accurate modeling of biological systems. Some errors or ambiguities were corrected with the biologists of the ARC CPBIO, but more work with biologists is needed to validate further the formal modeling of Kohn's diagram as a concurrent transition system, and incorporate more knowledge in the model.

The pathway logic of S. Eker et al. [**?**] is tightly related to our approach. The modeling of biochemical networks with concurrent transition systems is of a somewhat lower level than with pathway logic. Pathway logic is indeed more expressive as it can express algebraic properties of the components, such as the commutativity and associativity of complexation. This capability can be used to infer the possible reactions of molecules from their logical structure. It is worth considering however, that the interaction capabilities of a protein are often not related to the ones of its components, as they depend on the 3D structure of the proteins which is obviously impractical to take into account in a global modeling [**?**].

One limitation to the modeling of biological systems with concurrent transition systems is the necessity to encode all the parameters of the system in a finite vector of data and control variables. In order to get rid of this difficulty, we are currently investigating the use of other model checkers based on linear logic or multiset rewriting, that dont't have this restriction and make it possible to reason about systems within an arbitrary context [**?**].

Another obvious limitation in the experiments reported here is the absence of stochastic data. There are however stochastic model checking methods [?, ?] which can be investigated to circumvent this limitation.

The performances of our model checker can be improved in many ways as we have already shown with the use of DMC and NuSMV for the logical model of the mammalian cell cycle control. Similar improvements will be necessary to show the scalability of this approach for quantitative models. In this respect, constraint-based model checking is thightly related to hybrid systems methods and to hybrid verification tools such as for example Hytech [?] or d/dt [?]. Approximation techniques coming from hybrid automata can be imported in constraint-based model checking with the framework of abstract interpretation. On the other hand, constraint-based model checking provides a method for generalizing hybrid verification tools, going from pure reachability analysis towards more general CTL query evaluation.

6 Conclusion

We have applied symbolic model checking techniques to the querying and validation of both quantitative and qualitative models of biomolecular systems. Our first experiments show some advantages over simulation. Constraint reasoning makes it possible to group large or infinite sets of states into small constrained state expressions which provide formal proofs of reachability, pathway, checkpoint and stability properties. In some cases the properties can be checked by computing a fewer number of states than by simulation. It is also possible to reason with infinite sets of initial states finitely represented by constraints. Moreover the proof method applies to non-deterministic systems, for which simulations may be unfeasible.

We have also shown that constraint-based model checking can be applied in quantitative models described by differential equations. In our experiments we used a symbolic variant of Euler's method, but the use of the more accurate Runge-Kutta method, of non-linear constraint solving by interval propagation, and the use of abstraction techniques open many ways for improvement.

For all these reasons, we believe that, beyond simulation, verification tools such as model checking will become indispensable for querying and validating complex models in systems biology.

6.0.1 Acknowledgement :

We gratefully acknowledge the interactions we had with our colleagues of the ARC CPBIO, especially with Magali Roux-Rouquié, Julien Renner and Grégory Sautejeau from Institut Pasteur, for interesting discussions on Systems Biology and relevant bits of Biomolecular Biology, Vincent Danos and Marc Chiaverini from CNRS PPS Lab. for their beautiful transcription of Kohn's diagrams in their core modeling language, Vincent Schächter at Genoscope Evry, for his insights on the validation of biological models, Alexander Bockmayr, Arnaud Courtois and Damien Eveillard from the ModBio group at LORIA Nancy, for fruitful discussions on quantitative models.