Analyse de Séries Temporelles par Résolution de Contraintes de Logique Temporelle

Francois Fages, Aurélien Rizk

Analyse de Séries Temporelles par Résolution de Contraintes de Logique Temporelle

François Fages **Aurélien Rizk**

Project-team Contraintes, INRIA Paris-Rocquencourt
BP105, 78153 Le Chesnay Cedex, France.
{Francois.Fages,Aurelien.Rizk}@inria.fr

Résumé

La logique temporelle et le model-checking ont fait leurs preuves pour formaliser des propriétés biologiques de systèmes biochimiques complexes et vérifier automatiquement leur satisfaction sur des modèles quantitatifs ou qualitatifs. Dans cet article, nous allons au delà du model-checking en présentant un algorithme de résolution de contraintes pour les formules de logique temporelle du premier ordre sans quantificateur avec contraintes sur les réels. Cet algorithme calcule le domaine des variables réelles apparaissant dans une formule pour lequel la formule est satisfaite pour un modèle donné. Nous illustrons cette approche pour la génération automatique de spécification en logique temporelle à partir de données biologiques. Nous fournissons un ensemble de motifs de formules biologiquement pertinentes et les appliquons sur des données simulées d'un modèle du cycle cellulaire. Nous montrons sur ces exemples que cette approche permet d'inférer automatiquement, des propriétés semi-qualitatives, semi-quantitatives sur les seuils de concentration, l'amplitude d'oscillations, les propriétés de stabilité, de point de passage et d'influence entre molécules.

1 Introduction

La logique temporelle et les algorithmes de model-checking [11] ont fait leurs preuves pour formaliser des propriétés biologiques de systèmes biochimiques complexes et vérifier automatiquement leur satisfaction sur des modèles quantitatifs ou qualitatifs, c'est à dire booléens [13, 8, 9], discrets [3, 2], stochastiques [4, 18], ou continus [5, 1, 8].

Cette approche repose sur un paradigme logique pour la biologie des sytèmes qui consiste à faire les identifications suivantes [14] :

modèle biologique = système de transition

propriété biologique = formule de logique temporelle
validation biologique = model-checking

Dans cette approche, les connaissances biologiques expérimentales sont formalisées dans un langage basé sur la logique temporelle. Le système biologique est représenté par un système de transition qui peut être défini par des règles [13, 16, 6], des réseaux de Petri [22, 17], ou des algèbres de processus, [24, 7, 23, 12, 21], etc....

Décrire de manière formelle un modèle biologique mais aussi ses propriétés biologiques ouvre de grandes possibilités de conception d'outils automatisés, inspirés de la vérification de programmes, aidant le modélisateur à concevoir, maintenir et valider ses modèles [15].

Cependant, la formalisation des propriétés biologiques en une spécification en logique temporelle est une étape délicate et freine l'utilisation de cette approche.

Dans ce travail, nous explorons l'utilisation de ce paradigme logique pour l'analyse de séries temporelles, et l'inférence automatique de spécifications en logique temporelle.

Des travaux précédents abordent l'inférence de corrélations et d'influences, positives ou négatives, entre molécules à partir de séries temporelles, particulièrement à partir de données d'expression génique [25, 19]. Cependant, à notre connaissance, l'inférence de formules de logique temporelle à variables réelles à partir de séries temporelles est nouveau.

Du point de vue de la logique temporelle, notre travail revient à généraliser les algorithmes de model-checking à des algorithmes de résolution de contraintes, pour vérifier la satisfiabilité (au lieu de la validité) de formules de logique temporelle sur une

structure de Kripke linéaire, comme une trace de simulation.

A notre connaissance, cette généralisation est nouvelle.

Des travaux précédents dans cette direction appliquent les méthodes de model-checking aux traces de simulation [20, 5, 1] mais pas les méthodes de résolution de contraintes permettant de calculer le domaine des solutions des variables.

Dans cet article, nous généralisons l'algorithme de model-checking de traces décrit dans [5] et rappelé dans la section suivante à un algorithme de résolution de contraintes pour le fragment des formules LTL sans quantificateurs avec variables réelles.

Cette approche premier-ordre permet de calculer les instanciations des formules qui sont satisfaites pour une trace donnée, en donnant le domaine des variables apparaissant dans la formule pour lequel elle est satisfaite. Un résultat de complétude forte montrant que le domaine calculé correspond exactement à l'ensemble des solutions ainsi que la complexité en temps de l'algorithme sont présentés Sec. 3.

Puis, nous illustrons la pertinence de cette approche à l'analyse de séries temporelles de données biologiques. Nous fournissons un ensemble de formules biologiquement judicieuses Sec. 4, et les évaluons sur des traces de simulation du cycle cellulaire Sec. 5. Nous montrons sur ces exemples que cette approche permet d'inférer automatiquement, des propriétés semi-qualitatives, semi-quantitatives sur les seuils de concentration, l'amplitude d'oscillations, les propriétés de stabilité, de point de passage et d'influence entre molécules.

Nous concluons sur la pertinence de cette généralisation du model-checking à la résolution de contraintes temporelles pour la modélisation de systèmes biologiques, les résultats obtenus, et les perspectives pour les travaux futurs.

2 Préliminaires sur le Model-checking de Formules LTL avec Contraintes sur les Réels

2.1 LTL avec Contraintes sur les Réels

La logique LTL *Linear Time Logic* [11] étend la logique propositionnelle par des opérateurs déterminant à quels instants une formule est vérifiée dans un arbre d'états, appelée une structure de Kripke.

Ces opérateurs temporels sont X ("next", pour l'instant suivant), F ("finally", pour un instant dans le futur), G ("globally", pour tous les instants dans le futur), et U ("until", jusqu'à ce que).

Ces opérateurs vérifient les propriétés de dualité suivantes : $\neg X\phi = X\neg\phi$, $\neg F\phi = G\neg\phi$, $\neg G\phi = F\neg\phi$, $\neg(\psi\ U\ \phi) = (\neg\phi\ W\ \neg\psi)$, $\neg(\psi\ W\ \phi) = (\neg\psi\ U\ \neg\phi)$, et $F\phi = \text{true}\ U\ \phi$, $G\phi = \phi\ W\ \text{false}$.

Une structure de Kripke (cf par exemple [11]) est un couple $K = (S, R)$ où S est un ensemble d'états sur lesquels les propositions atomiques sont évaluées et où $R \subseteq S \times S$ est la relation de transition entre états, supposée totale ($\forall s \in S, \exists s' \in S$ s.t. $(s, s') \in R$) Un chemin de K, ayant pour origine s_0 est une séquence d'états infinie $\pi = s_0, s_1, \cdots$ tel que $(s_i, s_{i+1}) \in R$ pour tout $i \geq 0$. Dans la suite, on utilisera la notation π^k pour représenter le chemin $s_k, s_{k+1}, \cdots$.

La sémantique d'une formule LTL en un état s ou un chemin π sur une structure de Kripke K est indiqée dans le tableau 2.1.

$s \models \alpha$	ssi	α est une formule propositionnelle vraie en l'état s,
$s \models \psi$	ssi	pour tous les chemins π commençant en s, $\pi \models \psi$,
$\pi \models \phi$	ssi	$s \models \phi$ où s est le premier état de π,
$\pi \models X\psi$	ssi	$\pi^1 \models \psi$,
$\pi \models \psi\ U\ \psi'$	ssi	il existe $k \geq 0$ tel que $\pi^k \models \psi'$ and $\pi^j \models \psi$ pour tout $0 \leq j < k$.
$\pi \models \psi\ W\ \psi'$	ssi	soit pour tout $k \geq 0$, $\pi^k \models \psi$. soit il existe $k \geq 0$ tel que $\pi^k \models \psi \& \psi'$ et pour tout $0 \leq j < k$, $\pi^j \models \psi$.
$\pi \models !\psi$	ssi	$\pi \not\models \psi$,
$\pi \models \psi\ \&\ \psi'$	ssi	$\pi \models \psi$ et $\pi \models \psi'$,
$\pi \models \psi \mid \psi'$	ssi	$\pi \models \psi$ ou $\pi \models \psi'$,
$\pi \models \psi \Rightarrow \psi'$	ssi	$\pi \models \psi'$ ou $\pi \not\models \psi$,

TAB. 1 – Définition inductive de la valeur de vérité d'une formule propositionnelle LTL sur un état s ou sur un chemin π, pour une structure de Kripke donnée K.

Une version de LTL avec contraintes sur les réels, appelée Constraint-LTL, est utilisée dans Biocham [5] pour exprimer des propriétés temporelles sur des concentrations de molécules. Une approche similaire est utilisée dans le projet Darpa BioSpice [1].

La logique LTL avec contraintes considère des formules atomiques du premier ordre avec égalités, inégalités et opérateurs arithmétiques sur les valeurs réelles des concentrations et de leurs dérivées.

Par exemple `F([A]>10)` exprime le fait que la concentration de `A` dépasse la valeur 10 à au moins un moment dans le futur. `G([A]+[B]<[C])` signifie que la concentration de C est toujours supérieure à la somme des concentrations de A et B. Les propriétés d'oscillation, abrégées par `oscil(M,K)` sont définies comme une succession de changements de signes de la dérivée

de M au moins K fois :
`F((d[M]/dt > 0) & F((d[M]/dt < 0)`
`& F((d[M]/dt > 0)...)))`
La formule `oscil(M,K,V)` ajoute la contrainte que la concentration maximum de M doit être supérieure au seuil V au moins K fois.

Dans ce contexte, les structures de Kripke dans lesquelles les formules LTL sont interprétées sont des structures de Kripke linéaires qui représentent une série temporelle, expérimentale ou simulée, complétée par une boucle sur le dernier état. Ainsi, dans un modèle décrit par un système d'équations différentielles ordinaires (ODE), et en supposant que l'état initial est complètement défini, des méthodes d'intégration numérique (comme les méthodes de Runge-Kutta ou Rosenbrock) permettent d'obtenir une simulation discrète de la trace. La trace forme une structure de Kripke linéaire sur laquelle les formules LTL avec contraintes peuvent être interprétées. Comme les contraintes portent sur les concentrations mais aussi sur leurs dérivées nous considérons des traces de la forme

$$(< t_0, \vec{x}_0, d\vec{x}_0/dt, d^2\vec{x}_0/dt^2 >,$$
$$< t_1, \vec{x}_1, d\vec{x}_1/dt, d^2\vec{x}_1/dt^2 >, ...)$$

A chaque instant t_i la trace associe les valeurs de concentrations x_i des variables ainsi que les valeurs des dérivées premières et secondes dx_i/dt et d^2x_i/dt^2. Ce choix de dérivées est justifié dans la section 4 pour permettre de représenter les influences positives et négatives entre les molécules. Il est intéressant de remarquer que dans les méthodes d'intégration à pas de temps adaptatif, le pas de temps $t_{i+1} - t_i$ n'est pas constant et est déterminé par une estimation de l'erreur causée par la discrétisation.

2.2 Algorithme de Model-Checking de Formules LTL avec Contraintes

Supposons une structure de Kripke linéaire finie, c'est à dire une chaîne d'états contenant une boucle sur le dernier état. Pour de telles structures, les algorithmes standards de model-checking [11] peuvent être aisément adaptés aux formules LTL avec contraintes de la manière suivante :

Algorithme 1 ***(Model-checking de formules LTL avec contraintes)****[5, 1]*

1. *assigner à chaque instant de la trace les sous formules atomiques de ϕ qui sont satisfaites à cet instant ;*
2. *assigner les sous formules de la forme resp. $X\phi$ au prédécesseur immédiat d'un instant étiqueté par ϕ ;*
3. *assigner les sous formules de la forme ϕ_1 U ϕ_2 aux instants précédant ceux étiquetés par ϕ_2 et cela tant que ϕ_1 est vrai.*
4. *assigner les sous formules de la forme ϕ_1 W ϕ_2 au dernier instant de la trace s'il est étiqueté par ϕ_1, et aux prédécesseurs des instants étiquetés par ϕ_1 W ϕ_2 tant que ϕ_1 est vrai et assigner les sous formules de la forme ϕ_1 W ϕ_2 aux instants précédant un instant étiqueté par $\phi_1 \wedge \phi_2$ tant que ϕ_1 est vrai ;*
5. *retourner les instants étiquetés par ϕ.*

En particulier, étant donné un système d'ODEs et une propriété temporelle ϕ à vérifier sur un horizon de temps fini, le calcul d'une trace de simulation finie par intégration numérique fournit une structure de Kripke linéaire dans laquelle le dernier état est complété par une boucle.

La notion d'état suivant (opérateur X) se réfère à l'instant suivant dans la trace discrétisée, et n'implique donc pas forcément un voisinage réel dans le temps.

Le raisonnement justifiant cet algorithme est que la trace discrétisée comporte suffisamment de points, en particulier aux endroits où la dérivée évolue rapidement, pour évaluer correctement les formules de logique temporelle. Ceci a été vérifié en pratique pour plusieurs exemples de modèles mathématiques publiés [5].

3 Résolution de Contraintes de Logique Temporelle pour les Formules LTL du Premier Ordre sur les Réels sans Quantificateur

3.1 Formules LTL du Premier Ordre sur les Réels sans Quantificateur

Nous considérons ici le fragment sans quantificateur des formules LTL du premier ordre sur les réels, nommé QFLTL($\mathbb{R}$), c'est à dire les formules LTL avec contraintes avec la possibilité d'avoir des variables réelles dans les contraintes. Plus précisément, le langage des formules QFLTL($\mathbb{R}$) considéré dans cet article est défini par la grammaire donnée dans le tableau 3.1.

Les négations et les implications peuvent être éliminées en propageant les négations vers les contraintes atomiques d'une formule.

A partir de maintenant, nous supposerons que toutes les formules QFLTL($\mathbb{R}$) sont sous forme normale sans négation.

$$
\begin{aligned}
Qfltl = \\
&Atom \\
&\mid X(Qfltl) \\
&\mid (Qfltl)\ U\ (Qfltl) \\
&\mid (Qfltl)\ W\ (Qfltl) \\
&\mid (Qfltl) \wedge (Qfltl) \\
&\mid (Qfltl) \vee (Qfltl) \\
&\mid (Qfltl) \Rightarrow (Qfltl) \\
&\mid \neg(Qfltl) \\
Atom = \\
&Valeur\ Op\ Variable \mid Valeur\ Op\ Valeur \\
Op = \\
&< \mid > \mid \leq \mid \geq \\
Valeur = \\
&float \mid [molecule] \mid d[molecule]/dt \\
&\mid d^2[molecule]/dt^2 \mid Time \\
&\mid Valeur + Valeur \mid Valeur - Valeur \\
&\mid -Valeur \mid Valeur \times Valeur \\
&\mid Valeur/Valeur \mid Valeur^{Valeur}
\end{aligned}
$$

TAB. 2 – Grammaire des formules QFLTL(ℝ).

3.2 Algorithme de Résolution de Contraintes QFLTL(ℝ)

Etant donné une structure de Kripke $\mathcal{K}$ avec des états à valeurs réelles, et une formule QFLTL(ℝ) $\phi(\vec{x})$ contenant n variables réelles, le *problème de satisfaction de contraintes*, $\exists \vec{x} \in \mathbb{R}^n\ (\phi(\vec{x}))$, est le problème consistant à déterminer une assignation $\vec{v}$ des variables pour laquelle la formule ϕ est vraie. Autrement dit, on recherche le domaine de validité $\mathcal{D}_\phi \subset \mathbb{R}^n$ tel que $\mathcal{K} \models_{LTL} \forall \vec{v} \in \mathcal{D}_\phi\ (\phi(\vec{v}))$.

Le domaine de validité $\mathcal{D}_\phi$ de ϕ peut être calculé par un algorithme similaire à l'algorithme de model-checking de la section 2.2 :

Algorithme 2 ***(Résolution de contraintes QFLTL(ℝ))***

1. *étiqueter chaque instant de la trace par les sous formules atomiques de ϕ et par leur domaine de validité de la manière suivante :*
   - *pour une formule atomique ψ sans variables, étiqueter chaque instant t_i par : $(\psi, \mathcal{D}_\psi(t_i) = \mathbb{R}^n)$ si ψ est vrai à l'instant t_i et par $(\psi, \mathcal{D}_\psi(t_i) = \varnothing)$ sinon ;*
   - *pour une formule atomique de la forme $[A] \geq p$ (c'est à dire de la forme value $\geq$ variable) étiqueter chaque point t_i par $([A] \geq p, \mathcal{D}_{[A]\geq p}(t_i))$ où $\mathcal{D}_{[A]\geq p}(t_i)$ est le demi espace de $\mathbb{R}^n$ défini par $p \leq [A](t_i)$;*
   - *procéder de la même façon pour les autres formules atomiques contenant des variables ;*
2. *étiqueter chaque instant t_i par la sous formule $\psi_1 \vee \psi_2$ et par son domaine de validité $\mathcal{D}_{\psi_1 \vee \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cup \mathcal{D}_{\psi_2}(t_i)$;*
3. *étiqueter chaque instant t_i par la sous formule $\psi_1 \wedge \psi_2$ et par son domaine de validité $\mathcal{D}_{\psi_1 \wedge \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cap \mathcal{D}_{\psi_2}(t_i)$;*
4. *étiqueter chaque instant t_i par la sous formule $X\psi$ et par son domaine de validité $\mathcal{D}_{X\psi}(t_i) = \mathcal{D}_\psi(t_{i+1})$;*
5. *étiqueter le dernier instant de la trace t_n par la sous formule $\psi_1\ U\ \psi_2$ et par son domaine de validité $\mathcal{D}_{\psi_1\ U\ \psi_2}(t_n) = \mathcal{D}_{\psi_2}(t_n)$. En commençant par l'instant t_{n-1}, étiqueter chaque instant t_i par la sous formule $\psi_1 U \psi_2$ et par son domaine de validité :*
   $\mathcal{D}_{\psi_1\ U\ \psi_2}(t_i) = \mathcal{D}_{\psi_2}(t_i) \cup (\mathcal{D}_{\psi_1 U \psi_2}(t_{i+1}) \cap \mathcal{D}_{\psi_1}(t_i))$;
6. *étiqueter le dernier instant de la trace t_n par la sous formule $\psi_1\ W\ \psi_2$ et par son domaine de validité $\mathcal{D}_{\psi_1\ W\ \psi_2}(t_n) = \mathcal{D}_{\psi_1}(t_n)$. En commençant par l'instant t_{n-1}, étiqueter chaque instant t_i par la sous formule $\psi_1 W \psi_2$ et par son domaine de validité :*
   $\mathcal{D}_{\psi_1\ W\ \psi_2}(t_i) = (\mathcal{D}_{\psi_1}(t_i) \cap \mathcal{D}_{\psi_2}(t_i)) \cup (\mathcal{D}_{\psi_1 W \psi_2}(t_{i+1}) \cap \mathcal{D}_{\psi_1}(t_i))$;
7. *retourner le domaine $\mathcal{D}_\phi(t_i)$ pour tous les points t_i où il n'est pas vide.*

En particulier, étant donné un système d'ODEs et une propriété temporelle ϕ avec variables, à vérifier sur un horizon de temps fini, le calcul d'une trace de simulation finie par intégration numérique fournit une structure de Kripke linéaire à laquelle peut être appliqué l'algorithme de résolution de contraintes pour déterminer le domaine de validité des variables rendant ϕ vraie.

Rappelons qu'un orthotope de $\mathbb{R}^n$ est le produit cartésien de n intervalles de $\mathbb{R}$. Les propositions suivantes montrent que le domaine de validité calculé par l'algorithme est une union finie d'orthotopes, et décrit exactement l'espace des solutions pour le fragment choisi des contraintes sur les réels.

Proposition 1 *Le domaine calculé par l'algorithme de résolution de contraintes QFLTL(ℝ) est une union finie d'orthotopes.*

Preuve. Dans le cas de base des formules atomiques, l'algorithme calcule des orthotopes, et dans les autres

cas, applique des opérations d'intersection et d'union finie sur les domaines calculés. Comme une intersection finie d'orthotope est une union finie d'orthotopes, le domaine calculé est toujours une union finie d'orthotopes. □

Théorème 1 (Complétude forte) *L'algorithme de résolution de contraintes est correct et complet : une valuation $\vec{v}$ rend une formule QFLTL(ℝ) ϕ vraie à l'instant t_i, $T, t_i \models_{LTL} (\phi(\vec{v}))$, si et seulement si $\vec{v}$ est dans le domaine calculé pour ϕ à t_i, $\vec{v} \in \mathcal{D}_\phi(t_i)$.*

Preuve. Montrons inductivement sur la structure d'une formule QFLTL(ℝ) que pour tout instant t, toute formule QFLTL ϕ et tout instanciation $\vec{v}$ des variables,

si $\phi(\vec{v}, t_i)$ est vrai alors $\vec{v} \in \mathcal{D}_\phi(t_i)$ et si $\vec{v} \in \mathcal{D}_\phi(t_i)$alors $\phi(\vec{v}, t_i)$ est vrai :

- Les formules atomiques QFLTL considérées sont de la forme $Value\ Op\ Variable$ ou$Value\ Op\ Value$ où $Value$ et une expression arithmetique évaluable et Op un opérateur d'inégalité. Pour toutes ces formules atomiques l'algorithme retourne le domaine de validité exact. Par exemple, la formule $([A] \leq p)(t_i)$ est vraie si et seulement si p est supérieur ou égal à $[A](t_i)$ et le domaine de validité retourné est défini par $p \geq [A](t_i)$;
- $\phi_1 \wedge \phi_2$. Par construction dans l'algorithme $\mathcal{D}_{\phi_1 \wedge \phi_2}(t_i) = \mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i)$ d'où : $\vec{v} \in \mathcal{D}_{\phi_1 \wedge \phi_2}(t_i) \Leftrightarrow \vec{v} \in \mathcal{D}_{\phi_1}(t_i)$ *et* $\vec{v} \in \mathcal{D}_{\phi_2}(t_i) \Leftrightarrow \phi_1(\vec{v}, t_i) \wedge \phi_2(\vec{v}, t_i) \Leftrightarrow (\phi_1 \wedge \phi_2)(\vec{v}, t_i)$;
- $\phi_1 \vee \phi_2$. Par construction $\mathcal{D}_{\phi_1 \vee \phi_2}(t_i) = \mathcal{D}_{\phi_1}(t_i) \cup \mathcal{D}_{\phi_2}(t_i)$ d'où : $\vec{v} \in \mathcal{D}_{\phi_1 \vee \phi_2}(t_i) \Leftrightarrow \vec{v} \in \mathcal{D}_{\phi_1}(t_i)$ *ou* $\vec{v} \in \mathcal{D}_{\phi_2}(t_i) \Leftrightarrow \phi_1(\vec{v}, t_i) \vee \phi_2(\vec{v}, t_i) \Leftrightarrow (\phi_1 \vee \phi_2)(\vec{v}, t_i)$;
- $X(\phi)$. Par construction $\mathcal{D}_{X(\phi)}(t_i) = \mathcal{D}_\phi(t_{i+1})$ d'où : $\vec{v} \in \mathcal{D}_{X(\phi)}(t_i) \Leftrightarrow \vec{v} \in \mathcal{D}_\phi(t_{i+1}) \Leftrightarrow X(\phi)(\vec{v}, t_i)$;
- $\phi_1\ U\ \phi_2$. Par construction : $\mathcal{D}_{\phi_1 U \phi_2}(t_i) = \mathcal{D}_{\phi_2}(t_i) \cup (\mathcal{D}_{\phi_1 U \phi_2}(t_{i+1}) \cap \mathcal{D}_{\phi_1}(t_i))$ d'où : $\vec{v} \in \mathcal{D}_{(\phi_1 U \phi_2)}(t_i) \Leftrightarrow \phi_2 \vee (\phi_1 \wedge X(\phi_1 U \phi_2))(\vec{v}, t_i)$. Or la formule $(\phi_1 U \phi_2)$ peut être écrite sous la forme $(\phi_1 U \phi_2) = \phi_2 \vee (\phi_1 \wedge X(\phi_1 U \phi_2))$;
- $\phi_1\ W\ \phi_2$. Par construction : $\mathcal{D}_{\phi_1\ W\ \phi_2}(t_i) = (\mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i)) \cup (\mathcal{D}_{\phi_1\ W\ \phi_2}(t_{i+1}) \cap \mathcal{D}_{\phi_1}(t_i))$ d'pù : $\vec{v} \in \mathcal{D}_{(\phi_1 W \phi_2)}(t_i) \Leftrightarrow (\phi_1 \wedge \phi_2) \vee (\phi_2 \wedge X(\phi_1 W \phi_2))(\vec{v}, t_i)$. Or la formule$(\phi_1\ W\ \phi_2)$ peut être écrite sous la forme $(\phi_1\ W\ \phi_2) = (\phi_1 \wedge \phi_2) \vee (\phi_1 \wedge X(\phi_1\ W\ \phi_2))$.

□

La taille d'une formule QFLTL est le nombre de symboles contenus dans la formule. Définissons la taille d'une union fine d'orthotopes $\mathcal{D}$, comme le plus petit entier k tel que $\mathcal{D} = \bigcup_{i=1}^{k} \mathcal{R}_i$ où les R_i sont des orthotopes.

Théorème 2 (complexité du domaine des solutions) *Le domaine de validité d'une formule QFLTL de taille f contenant v variable sur une trace de longeur n est une union d'orthotopes de taille inférieure à $(nf)^{2v}$.*

Preuve.

Considérons le nombre possible de bornes apparaissant dans le domaine de validité de $\mathcal{D}_\phi$ d'une formule ϕ pour une seule variable x.

Nous considérons le nombre possible de bornes générées par les formules atomiques. Chaque occurrence de la variable x dans ϕ est dans une contrainte de la forme $Value(t_i)\ Op\ Variable$. Une telle contrainte peut éventuellement être évaluée sur chaque instant de la trace et ainsi créer au plus n bornes pur x. Le nombre maximum de bornes pour la variable x est donc n fois le nombre d'occurence de x dans ϕ qui est inférieur à $n \times f$. Ce nombre maximum de bornes est atteint par exemple dans la formule

$F([A] = u \vee [A] + 1 = u \vee \cdots \vee [A] + f = u)$.

En réécrivant les opérateurs U et W sous la forme : $\phi_1\ U\ \phi_2(t_i) = \bigvee_{j \geq i}(\phi_2(t_j) \wedge \bigwedge_{i \leq k < j} \phi_1(t_k))$ et $\phi_1\ W\ \phi_2(t_i) = \bigwedge_{j \geq i}(\phi_1(t_j) \vee \bigvee_{i \leq k < j} \phi_2(t_k))$ nous remarquons que les formules QFLTL peuvent être réécrites sous une forme ne contenant que des $\vee$ et des $\wedge$ sans changer leur ensemble de solutions.

Si $\mathcal{B}_v(\phi)$ est l'ensemble des bornes possibles pour la variable x dans ϕ et si ϕ_1 et ϕ_2 sont des sous formules de ϕ, nous avons : $\mathcal{B}_v(\phi_1 \vee \phi_2) \subset \mathcal{B}_v(\phi)$ et $\mathcal{B}_v(\phi_1 \wedge \phi_2) \subset \mathcal{B}_v(\phi)$. Les intersections et les unions d'orthotopes ne génèrent pas de nouvelles bornes.

Comme un orthotope est un produit cartésien d'intervalles, il est défini par deux bornes pour chaque variable. Avec moins de $n \times f$ bornes par variable, on peut donc former au plus $(nf)^{2v}$ orthotopes. Ainsi, le domaine calculé par l'algorithme est une union finie d'orthotopes (Prop. 1) de taille inférieure à $(nf)^{2v}$.

□

La formule $F([A_1] = X_1 \vee [A_1] + 1 = X_1 \vee ... \vee [A_1] + f = X_1) \wedge ...$
$\wedge\ F([A_v] = X_v \vee [A_v] + 1 = X_v \vee ... \vee [A_v] + f = X_v)$ a par exemple un domaine solution de taille $(nf)^v$ sur une trace de n valeurs pour les $[A_i]$s telle que les valeurs des $[A_i] + k$ son toutes différentes pour $1 \leq i \leq v$ et $0 \leq k \leq f$.

4 Motifs de Formules QFLTL($\mathbb{R}$) Formalisant des Propriétés Biologiques

La logique temporelle est suffisamment expressive pour formaliser un large panel de propriétés biologiques connues à partir d'expériences menées sous diverses conditions.

L'algorithme de résolution de contraintes donné pour les formules QFLTL($\mathbb{R}$) rend possible l'analyse de traces de concentration et l'obtention de propriétés semi-quantitatives formalisées en formules QFLTL($\mathbb{R}$). En particulier, le pendant quantitatif des propriétés CTL purement qualitatives étudiées dans [8] peuvent être exprimées de la manière suivante, les variables étant écrites en caractères minuscules :

Atteignabilité : `F([A]>=p)`, quel seuil p la molécule A atteint dans la trace ?

Point de passage : `not (([A]<p1)U([B]>p2))`, pour quels seuils $p1$ et $p2$ est-il faux que $[A]$ est inférieur à $p1$ avant que $[B]$ ne soit supérieur à $p2$, c'est à dire pour quels $p1$ et $p2$ `[A] >= p1` est obligatoire pour avoir de `[B]>p2` ?

Stabilité : `G([A]=<p1 & [A]>=p2)`, pour formaliser l'intervalles de valeurs prises par $[A]$. Cet intervalle de valeurs peut être recherché dans un contexte défini par une condition comme par exemple `G(Time>10 -> ([A]<p1 & [A]>p2))`.

Oscillation : `F((d([A])/dt>0 & [A]>v1) & (F((d([A])/dt<0 & [A]<v2))))`, quelle amplitude $(v1 - v2)$ est atteinte par au moins une oscillation ? Une oscillation est définie comme un changement de signe de la dérivée. Cette formule peut être étendue à un plus grand nombre d'oscillations et est notée `oscil(M,K,v1,v2)`. Cette abréviation signifie que M doit avoir une amplitude d'au moins $v1 - v2$ dans au moins K oscillations. En utilisant cette formule pour différentes valeurs de K, en commençant par $K = 1$, on peut déterminer le nombre maximal d'oscillations dans la trace et l'amplitude minimale atteinte par K oscillations pour tout K.

Influence : `G(d[A]/dt>p1 -> d2[B]/dt2>=0)`, à partir de quels seuil la dérivée de A a-t-elle une influence sur la dérivée de B ? L'influence est positive si une valeur élevée de `d[A]/dt` a pour conséquence une dérivée seconde positive de `[B]`. Etant donné que plusieurs molécules peuvent influencer B, cette formule indique seulement une corrélation entre les valeurs de la dérivée de A et de la dérivée seconde de B et ne donne pas de preuve de l'influence directe de A sur B.

5 Application à l'Inférence de Propriétés Temporelles à Partir de Séries Temporelles Biologiques

Nous présentons dans cette section l'application de l'algorithme de résolution de contraintes aux données du cycle cellulaire de la levure. Pour évaluer la méthode, nous n'utilisons pas de données expérimentales mais des données simulées obtenues à partir du modèle de [10]. Les traces de concentrations sont obtenues en simulant le modèle du cycle cellulaire dans Biocham. Nous essayons ensuite de retrouver les principales propriétés du modèle en analysant automatiquement les traces.

Les règles de réaction du modèle sont les suivantes :

```
(1) _=>Cyclin.
(2) Cyclin+Cdc2~{p1}     => Cdc2-Cyclin~{p1,p2}
(3) Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}
(4) Cdc2-Cyclin~{p1,p2} =[Cdc2-Cyclin~{p1}]=>
    Cdc2-Cyclin~{p1}
(5) Cdc2-Cyclin~{p1}     => Cyclin~{p1}+Cdc2
(6) Cyclin~{p1} =>_
(7) Cdc2         => Cdc2~{p1}
(8) Cdc2~{p1}    => Cdc2
```

Les notations `~{p1}` et `{p1,p2}` représentent les formes phosphorylées des molécules. La figure 1 donne les traces simulées obtenues pour quatre molécules de ce modèle.

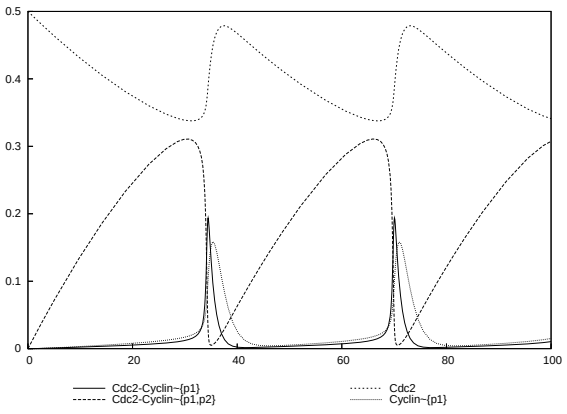

FIG. 1 – Trace de simulation du cycle cellulaire de la levure sur 100 unités de temps. La trace est constituée de 94 points.

Ces traces sont très informatives en elles mêmes, mais pour y automatiser le raisonnement, nous proposons d'utiliser des requêtes QFLTL($\mathbb{R}$).

Atteignabilité

Molécule	Atteignabilité	Stabilité
Cdc2	0.500	(0.338,0.500)
Cdc2-Cyclin~{p1,p2}	0.311	(0.000,0.311)
Cdc2-Cyclin~{p1}	0.194	(0.000,0.194)
Cyclin~{p1}	0.159	(0.000,0.159)

TAB. 3 – Résultats d'atteignabilité (valeur maximale atteinte) et de stabilité (valeurs minimales et maximales atteintes)

	Amplitude d'au moins n oscillations	
Molécules	$n = 1$	$n = 2$
Cdc2	0.141	0.138
Cdc2-Cyclin~{p1,p2}	0.306	0.306
Cdc2-Cyclin~{p1}	0.192	0.192
Cyclin~{p1}	0.155	0.154

TAB. 4 – Résultats des requêtes d'amplitude d'au moins n oscillations.

Une requête d'atteignabilité indique la concentration maximale atteinte par une molécule :

```
biocham: trace_analyze(F([Cdc2-Cyclin~{p1}]>=v)).
[[v=<0.194]]
```

Le résultat retourné est une liste de domaines représentés par une liste de contraintes sur les variables. Ici un domaine unique est retourné avec une contrainte unique sur v. Plusieurs types d'informations peuvent être obtenues à partir des domaines retournées en résultat. Dans cet exemple, la plus grande valeur de `v` dans le domaine, est la concentration maximale de `Cdc2-Cyclin~{p1}` dans la trace. Sa valeur est ici `0.194`. La table 3 donne les résultats des requêtes d'atteignabilité pour les quatre espèces dont la trace est donnée dans la Figure 1.

Stabilité

Pour déterminer la stabilité, calculons l'intervalle de valeurs prises par `[Cdc2]` :

```
biocham: trace_analyze(G([Cdc2]=<v1 & [Cdc2]>=v2)).
[[v1>=0.500, v2=<0.338]]
```

Le domaine est défini par la conjonction des deux contraintes `v1 >= 0.500` et `v2 =< 0.338`. Ces valeurs sont les valeurs maximales et minimales atteintes par `[Cdc2]`. Les résultats pour les autres molécules sont données dans le tableau 3.

Oscillation

Une requête d'oscillation peut retourner un domaine composé de plusieurs orthotopes :

```
biocham: trace_analyze(oscil(Cdc2,1)).
[[v2>=0.338, v1=<0.479], [v2>=0.341, v1=<0.479]]
```

Le résultat est l'union de deux orthotopes.

Ici nous pouvons extraire l'amplitude maximale $v1-v2$. Le maximum de $v1-v2$ est atteint dans le domaine pour $v1 - v2 = 0.479 - 0.338 = 0.141$. Ce résultat indique qu'au moins une oscillation de `Cdc2` a une amplitude supérieure ou égale à 0.141. Le nombre d'oscillations demandé est ensuite incrémenté jusqu'à obtenir un domaine de validité vide. Il est obtenu pour `Cdc2` avec la requête `oscil(Cdc2,3,v1,v2)`, indiquant qu'il n'y a que deux oscillations de `Cdc2` dans la trace.

Les résultats pour les autres molécules sont donnés dans le tableau 4. Obtenir l'amplitude des oscillations est utile pour distinguer des oscillations d'amplitudes diverses dans la trace. Par exemple, dans des données bruitées l'amplitude peut être utilisée pour compter le nombre d'oscillations sans tenir compte des petites oscillations dues au bruit.

Point de passage

Pour savoir si `Cdc2-Cyclin~{p1,p2}` est un point de passage pour `Cdc2-Cyclin~{p1}` on peut utiliser la requête suivante :

```
not([Cdc2-Cyclin~{p1,p2}]<v1 U
    [Cdc2-Cyclin~{p1}]   >v2
    )
```

Le domaine retourné est une union de dix orthotopes. Il est nécessaire d'examiner chacun des orthotopes pour interpréter ce domaine. Les requêtes de point de passage sont donc plus délicates et moins adpatées à une approche d'analyse automatique. Dans cet exemple, les valeurs $v1 = 0.311$ et $v2 = 0.014$ sont dans le domaine, indiquant que `Cdc2-Cyclin~{p1,p2}` n'est pas toujours inférieur à 0.311 avant que `Cdc2-Cyclin~{p1}` dépasse 0.014. En d'autres termes `Cdc2-Cyclin~{p1,p2}` dépasse 0.311 avant que `Cdc2-Cyclin~{p1}` dépasse 0.014 montrant que `Cdc2-Cyclin~{p1,p2}` est en effet un point de passage.

Influence

L'influence d'une molécule A sur une molécule B est recherchée avec la formule `G(d[A]/dt>p1 -> d2[B]/dt2>0)`. L'idée justifiant cette formule est que si une molécule B est présente dans une seule réaction de la forme $A \to B$ avec une loi d'action de masse, les formules suivantes sont vérifiées : $G(d[A]/dt > 0 \Rightarrow d^2[B]/dt^2 > 0)$ et $G(d[A]/dt < 0 \Rightarrow d^2[B]/dt^2 < 0)$.

Dans un système biologique usuel la concentration de chaque molécule est le résultat de l'action combinée de plusieurs autres molécules. La résolution de contrainte sur les formules QFLTL($\mathbb{R}$) détermine pour quels seuils ces formules sont vérifiées, c'est à dire les domaines de validité des variables `v1` et `v2`.

Molécules	Cdc2	Cdc2-Cyclin~{p1,p2}
Cdc2	0.00	0.11
Cdc2~{p1}	0.01	0.12
Cyclin	0.00	**0.34**
Cdc2-Cyclin~{p1,p2}	0.00	0.02
Cdc2-Cyclin~{p1}	**0.90**	0.00
Cyclin~{p1}	0.50	0.09

TAB. 5 – Scores d'influence positives de toutes le s molécules sur Cdc2 et Cdc2-Cyclin~{p1,p2}. Les molécules apparaissant sur les lignes (resp. les colonnes) jouent le rôle de la molécule A (resp. B) dans les formules $G(d[A]/dt > v1 \Rightarrow d^2[B]/dt^2 > 0)$ et $G(d[A]/dt < v2 \Rightarrow d^2[B]/dt^2 < 0)$.

En comparant les domaines de validité trouvés aux valeurs prises par $d[A]/dt$, un score $s \in [0, 1]$ est obtenu indiquant l'influence de la dérivée de [A] sur la dérivée seconde de [B]. Plus précisément, si le domaine de validité est $v1 \geq 0$ cela signifie que la formule est satisfaite pour toute valeur positive de la dérivée de $d[A]/dt$ induisant un score maximal d'influence de 1. Si le domaine de validité est $v1 >= \frac{\max(d[A]/dt)}{2}$ cela signifie que la formule est satisfaite pour la moitié des valeurs positives de la dérivée de $d[A]/dt$, le score d'influence est dans ce cas de 0.5. Le tableau 5 fournit les scores d'influence sur les molécules Cdc2 et Cdc2-Cyclin~{p1,p2} calculés selon cette méthode.

Selon les règles de réaction, la seule espèce ayant une influence positive sur [Cdc2] est [Cdc2-Cyclin~{p1}] (réaction (5)). Les scores d'influence calculés reflètent cela correctement. Le score proche obtenu par Cyclin~{p1} est du à son comportement très similaire à celui de [Cdc2-Cyclin~{p1,p2}] comme on peut le voir dans la trace. Ces deux molécules ont une augmentation de concentration coïncidant avec l'augmentation de la concentration de [Cdc2]. Les scores d'influence retournés permettent néanmoins de distinguer les deux molécules et de voir que c'est [Cdc2-Cyclin~{p1}] qui a une influence positive sur Cdc2.

Selon les règles de réaction, les deux molécules [Cyclin] et [Cdc2~{p1}] sont les seules à avoir une influence positive sur Cdc2-Cyclin~{p1,p2}(réaction (2)). On peut remarquer que, comme plus de molécules influencent Cdc2-Cyclin~{p1,p2} que Cdc2, il est plus difficile de trouver des corrélations entre une molécule unique et Cdc2-Cyclin~{p1,p2}. Les scores d'influence sont donc globalement plus faibles dans ce cas. En dépit de cela, les deux scores les plus élevés sont bien obtenus pour [Cyclin] et [Cdc2~{p1}].

6 Conclusion

Etant donné la difficulté de spécifier en logique temporelle les propriétés biologiques d'un système à partir de données expérimentales, nous avons proposé un algorithme pour inférer des formules LTL avec contraintes à partir de séries temporelles. Pour cela, l'algorithme de model-checking des formules LTL avec contraintes décrit dans [5] a été généralisé en un algorithme de résolution de contraintes dans le fragment sans quantificateur des formules LTL du premier ordre avec contraintes numériques sur les réels. La complétude forte de l'algorithme, indiquant que le domaine de validité des variables calculé représente l'espace exact des solutions a été démontré. Une borne supérieure de la complexité en temps de l'algorithme a également été montrée. La complexité est $\mathcal{O}((nf)^{2v})$ où n est le nombre de points de la série temporelle, f la taille de la formule et v le nombre de variables.

Afin d'évaluer la méthode nous avons utilisé des séries temporelles issues de simulations de modèles. Nous prévoyons d'appliquer la méthode à l'analyse de données expérimentales temporelles de protéines de la voie de signalisation FSH et à des données d'expression génique du cycle circadien issues du projet européen Tempo sur les chronothérapies du cancer dans le cadre du projet européen Tempo[1].

Une généralisation possible de ce travail serait de considérer un fragment plus important des contraintes sur les réels, en obtenant ainsi un résultat de complétude faible indiquant que le domaine calculé est une surapproximation de l'ensemble des solutions au lieu de lui être égal. Une autre généralisation serait de lever la restriction de la linéarité des structures de Kripke.

Financement

Ce travail a été financé en partie par le projet Tempo EU FP7 STREP.

Références

[1] Marco Antoniotti, Alberto Policriti, Nadia Ugel, and Bud Mishra. Model building and model checking for biochemical processes. *Cell Biochemistry and Biophysics*, 38 :271–286, 2003.

[2] Grégory Batt, Damien Bergamini, Hidde de Jong, Hubert Garavel, and Radu 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, April 2004.

[1] http ://www.chrono-tempo.org/

[3] Gilles Bernot, Jean-Paul Comet, Adrien 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.

[4] Muffy Calder, Vladislav Vyshemirsky, David Gilbert, and Richard Orton. Analysis of signalling pathways using the continuous time markow chains. In Gordon Plotkin, editor, *Transactions on Computational Systems Biology VI*, volume 4220 of *Lecture Notes in BioInformatics*, pages 44–67. Springer-Verlag, November 2006. CMSB'05 Special Issue.

[5] Laurence Calzone, Nathalie Chabrier-Rivier, François Fages, and Sylvain Soliman. Machine learning biochemical networks from temporal logic properties. In Gordon Plotkin, editor, *Transactions on Computational Systems Biology VI*, volume 4220 of *Lecture Notes in BioInformatics*, pages 68–94. Springer-Verlag, November 2006. CMSB'05 Special Issue.

[6] Laurence Calzone, François Fages, and Sylvain Soliman. BIOCHAM : An environment for modeling biological systems and formalizing experimental knowledge. *BioInformatics*, 22(14) :1805–1807, 2006.

[7] Luca Cardelli. Brane calculi - interactions of biological membranes. In Vincent Danos and Vincent Schächter, editors, *CMSB'04 : Proceedings of the second international workshop on Computational Methods in Systems Biology*, volume 3082 of *Lecture Notes in BioInformatics*, pages 257–280. Springer-Verlag, 2004.

[8] Nathalie Chabrier and François Fages. Symbolic model cheking of biochemical networks. In Corrado 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, March 2003. Springer-Verlag.

[9] Nathalie Chabrier-Rivier, Marc Chiaverini, Vincent Danos, François Fages, and Vincent Schächter. Modeling and querying biochemical interaction networks. *Theoretical Computer Science*, 325(1) :25–44, September 2004.

[10] Katherine C. Chen, Attila Csikász-Nagy, Bela Györffy, John Val, Bela Novàk, and John J. Tyson. Kinetic analysis of a molecular model of the budding yeast cell cycle. *Molecular Biology of the Cell*, 11 :396–391, 2000.

[11] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

[12] Vincent Danos and Cosimo Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1) :69–110, 2004.

[13] Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, José Meseguer, and M. Kemal Sönmez. Pathway logic : Symbolic analysis of biological signaling. In *Proceedings of the seventh Pacific Symposium on Biocomputing*, pages 400–412, January 2002.

[14] François Fages. Temporal logic constraints in the biochemical abstract machine biocham (invited talk). In Springer-Verlag, editor, *Proceedings of Logic Based Program Synthesis and Transformation, LOPSTR'05*, number 3901 in Lecture Notes in Computer Science, London, UK, September 2005.

[15] François Fages. From syntax to semantics in systems biology - towards automated reasoning tools. *Transactions on Computational Systems Biology IV*, 3939 :68–70, December 2006.

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

[17] David Gilbert, Monika Heiner, and Sebastian Lehrack. A unifying framework for modelling and analysing biochemical pathways using petri nets. In *CMSB'07 : Proceedings of the fifth international conference on Computational Methods in Systems Biology*, volume 4695 of *Lecture Notes in Computer Science*. Springer-Verlag, 2007.

[18] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. In *Proc. Computational Methods in Systems Biology (CMSB'06)*, volume 4210 of *Lecture Notes in Computer Science*, pages 32–47. Springer-Verlag, 2006.

[19] Iftach Nachman, Aviv Regev, and Nir Friedman. Inferring quantitative models of regulatory networks from expression data. In *ISMB/ECCB (Supplement of Bioinformatics)*, pages 248–256, 2004.

[20] Dejan Nickovic and Oded Maler. Amt : a property-based monitoring tool for analog systems. In *Proceedings of 5th International Conference on Formal Modelling and Analysis of Times Systems, FORMATS'07*, Lecture Notes in Computer Science, Salzburg, Austria, 2007. Springer-Verlag.

[21] Andrew Phillips and Luca Cardelli. A correct abstract machine for the stochastic pi-calculus.

Transactions on Computational Systems Biology, to appear. Special issue of BioConcur 2004.

[22] V. N. Reddy, M. L. Mavrovouniotis, and M. N. Liebman. Petri net representations in metabolic pathways. In L. Hunter, D. B. Searls, and J. W. Shavlik, editors, *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 328–336. AAAI Press, 1993.

[23] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Shapiro. Bioambients : An abstraction for biological compartments. *Theoretical Computer Science*, 325(1) :141–167, September 2004.

[24] Aviv Regev, William Silverman, and Ehud Y. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Proceedings of the sixth Pacific Symposium of Biocomputing*, pages 459–470, 2001.

[25] Rui Xu, Xiao Hu, and Donald C. Wunsch II. Inference of genetic regulatory networks from time series gene expression data. In *JCNN*, volume 2, pages 1215–1220, 2004.