université PARIS DIDEROT PARIS 7

UFR D'INFORMATIQUE
DENIS-DIDEROT

THÈSE

Présentée pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ PARIS 7
DENIS DIDEROT

Spécialité : Informatique

par

Aurélien Rizk

Résolution de Contraintes Temporelles pour l'Analyse de Systèmes Biologiques

Présentée le 6 juin 2011 devant le jury composé de :

M.	Oded Maler	(Rapporteur)
M.	Gilles Bernot	(Rapporteur)
M.	Eugène Asarin	(Président du jury)
Mme	Jane Hillston	(Examinateur)
M.	Nicolas Le Novère	(Examinateur)
M.	François Fages	(Directeur de thèse)

Thèse préparée à
l'INRIA Paris-Rocquencourt
EPI CONTRAINTES
Domaine de Voluceau, Rocquencourt, BP 105
78 153 LE CHESNAY CEDEX

Table des matières

Introduction

Cadre

La biologie moléculaire à révélé et décrit les propriétés d'une grande quantité de données biologiques, comme les séquences des génomes ou les propriétés chimiques des protéines. Cela n'est pas suffisant pour interpréter les systèmes biologiques. Les systèmes biologiques sont faits de composants dont les interactions ont été définies par l'évolution ; une grande quantité d'éléments divers, souvent multifonctionnels, interagissent entre eux pour produire des comportements cohérents. La compréhension des systèmes biologiques doit ainsi se faire à l'échelle du système : c'est la *biologie des systèmes*.

En raison de leur complexité, la compréhension des systèmes biologiques ne peut se faire de manière uniquement intuitive. Il est nécessaire de combiner des approches expérimentales et numériques. La simulation de modèles et la comparaison de ces simulations aux observations expérimentales permet de tester la validité des connaissances sur les systèmes biologiques. Les modèles validés peuvent ensuite être utilisés pour faire des prédictions qui seront testées expérimentalement et aussi pour explorer des problèmes qu'il n'est pas possible d'étudier de manière expérimentale. La modélisation et la simulation permettent ainsi d'améliorer la compréhension des interactions et du comportement dynamique des systèmes.

On s'intéresse dans ce travail aux réseaux d'interactions biochimiques (réseaux métaboliques, réseaux de signalisation, réseaux d'interaction génique). De nombreux formalismes ont été proposés pour la modélisation de ces réseaux. Ces modélisations peuvent être booléennes [1, 2, 3], discrètes [4], stochastiques [5, 6] ou continues [7, 8, 2, 9].

L'élaboration d'un modèle est un problème difficile. La construction d'un modèle peut comporter le choix entre plusieurs topologies, le choix de paramètres et de formules cinétiques et de valeurs initiales des composants du système. Ces choix se font en comparant les simulations du modèle à des données expérimentales. La difficulté de la conception de modèle est renforcée par l'incertitude et l'imprécision fréquentes des données biologiques. Ainsi, il est nécessaire de développer des outils automatiques d'aide à la modélisation pouvant aborder la complexité des systèmes biologiques tout en étant capables de s'accommoder du caractère flou et incertain des données biologiques.

La logique temporelle [10, 11] a été utilisée avec succès pour décrire le comportement d'une grande variété de systèmes complexes allant des circuits électroniques aux logiciels et plus récemment pour décrire le comportement des systèmes biologiques [4]. La logique temporelle permet d'exprimer des propriétés qualitatives (par exemple une protéine sera produite), et quantitatives (par exemple une concentration dépassera la valeur 10) et fournit donc un langage de spécification puissant en comparaison des pro-

priétés essentiellement qualitatives considérées dans la théorie des systèmes dynamiques (multi-stabilité, présence d'oscillations) ou de propriétés quantitatives exactes et complètes considérées habituellement en optimisation (curve-fitting). La logique temporelle permet ainsi de représenter des propriétés de haut niveau des systèmes biologiques sans s'attacher forcément à certaines valeurs particulières, souvent bruitées, des données expérimentales.

L'utilisation de la logique temporelle pour la biologie des systèmes repose sur un paradigme logique qui consiste à faire les identifications suivantes [12] :

$$\begin{gathered} \textit{modèle biologique} = \textit{système de transition} \\ \textit{propriété biologique} = \textit{formule de logique temporelle} \\ \textit{validation biologique} = \textit{model-checking} \end{gathered}$$

Dans cette approche, les connaissances biologiques expérimentales sont formalisées dans un langage basé sur la logique temporelle. La validation d'une propriété biologique se fait par model-checking de formules de logique temporelle. Une propriété est soit vraie soit fausse sur un modèle donné. 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 [13].

Cette approche a été utilisée dans de nombreuses applications, comme langage de requête sur de larges réseaux d'interaction comme la carte de Kohn du cycle cellulaire [3, 14] ou sur des réseaux de régulation génique [9] ou comme langage de spécification de propriétés biologiques et utilisées pour valider des modèles, proposer de nouvelles expériences [4], trouver des valeurs de paramètres [7] ou estimer la robustesse [15].

Ce paradigme logique a en particulier montré son efficacité dans le cas de modèles booléens pour vérifier rapidement de larges réseaux d'interaction vis à vis de propriétés de haut niveau exprimées en logique temporelle. Cependant l'évaluation booléenne classique des formules de logique temporelle n'est pas adaptée à de nombreux problèmes. Par exemple, dans le cas de la recherche de paramètres de modèles continus, on cherche des paramètres qui satisfont une spécification formalisée en logique temporelle. Des réponses uniquement négatives ne permettent pas de déterminer les directions les plus prometteuses à explorer dans l'espace de recherche. Seules des méthodes inefficaces (échantillonnage exhaustif ou aléatoire), utilisables uniquement pour un petit nombre de paramètres inconnus, sont alors possibles. Enfin une évaluation booléenne de propriétés biologiques est mal adaptée aux analyses quantitatives, comme l'analyse de robustesse ou l'analyse de sensibilité.

Sujet

L'objectif de ce travail est la généralisation du model-checking en un problème de résolution de contraintes et son application à la biologie des systèmes. Etant donné une formule de logique temporelle avec variables libres et une structure sur laquelle interpréter la formule, la résolution de contraintes temporelles est le calcul des valeurs des variables libres rendant la formule vraie sur la structure.

On ajoute au paradigme logique pour la biologie des systèmes présenté ci-dessus l'identification :

$$\textit{inférence de propriétés} = \textit{résolution de contraintes temporelles}$$

On établira que la résolution de contraintes temporelles permet l'inférence de propriétés biologiques et la définition d'un degré d'évaluation continu de formules de logique temporelle en comparant les propriétés inférées à une propriété voulue. On montrera que ce degré d'évaluation continu de formules de logique temporelle permet l'utilisation de méthodes d'optimisation continues pour la recherche de paramètres et le développement de méthodes d'analyse quantitatives de modèles biologiques vis à vis de propriétés biologiques formalisées en logique temporelle et que ceci rend possible l'application efficace de la logique temporelle et des méthodes de model-checking à des modèles continus de réseaux d'interaction biochimiques.

On développera ces outils au sein de l'environnement de modélisation Biocham, basé sur le paradigme logique pour la biologie des systèmes décrit ci-dessus (formalisation du système biologique et de ses propriétés temporelles) et on évaluera ces méthodes en les appliquant à plusieurs problèmes biologiques.

Plan

Chapitre 1 Ce chapitre décrit le paradigme logique pour la biologie des systèmes consistant à identifier un modèle biologique à un système de transition, une propriété biologique à une formule de logique temporelle et sa validation à un problème de model-checking. On introduit les structures de Kripke sur lesquelles les propriétés temporelles sont évaluées, le langage de logique temporel LTL($\mathbb{R}$) utilisé pour formaliser les propriétés biologiques et l'algorithme de model-checking permettant d'évaluer ces formules sur les structures de Kripke.

Chapitre 2 Dans ce chapitre on définit le problème de résolution de contraintes temporelles. D'abord on présente l'introduction de variables libres dans les formules LTL($\mathbb{R}$), ensuite on décrit un algorithme calculant le domaine de ces variables rendant une formule vraie. Ce domaine est utilisé pour définir un degré de violation et un degré de satisfaction continu de formules. Enfin on étend la résolution de contraintes temporelles à la logique temporelle avec opérateur de branchement CTL.

Chapitre 3 La résolution de contraintes de formules de logique temporelle et le degré de satisfaction continu de formules sont appliquées pour développer des méthodes de construction et d'analyse de modèles biologiques. D'abord on montre comment on peut utiliser la résolution de contraintes temporelles pour extraire automatiquement de données expérimentales temporelles des propriétés formalisées en LTL. Ensuite on utilise le degré de violation continu comme fonction de coût d'une méthode d'optimisation continue pour constituer une méthode de recherche de paramètres. On présente une méthode générique d'optimisation puis la méthode d'optimisation retenue pour la recherche de paramètres, la stratégie d'évolution par adaptation de matrice de covariance

CMA-ES. Enfin le degré de satisfaction continu est employé pour fournir des méthodes d'analyse de robustesse et d'analyse de sensibilité vis à vis de propriétés formalisées en logique temporelle.

Chapitre 4 Ce chapitre décrit l'implémentation des méthodes définies dans le chapitre 3. Ces méthodes sont implémentées dans l'environnement de modélisation Biocham. Le domaine de validité d'une formule est représenté par une union d'orthotopes ou de polyèdres selon le nombre de variables apparaissant dans les contraintes atomiques d'une formule. Pour la gestion de domaines polyédriques on utilise la bibliothèque de gestion de polyèdres Parma Polyhedra Library (PPL) [16]. Dans le cas d'un domaine représentable par union d'orthotopes une implémentation en C de la représentation et de la gestion des domaines a été réalisée. La méthode d'optimisation CMA-ES est intégrée à Biocham pour fournir une méthode de recherche de paramètres. Enfin on décrit la parallélisation de la méthode de recherche de paramètres sur un cluster composé de milliers de processeurs.

Chapitre 5 Les méthodes dont les implémentations sont décrites chapitre 4 sont appliquées à des problèmes biologiques. On considère d'abord l'inférence de propriétés temporelles du cycle cellulaire à partir de séries temporelles. Ensuite on applique la méthode de recherche de paramètres sur des modèles du cycle cellulaire et de la cascade de signalisation MAPK pour une première évaluation de cette méthode, sur un réseau de signalisation induit par l'angiotensine pour lequel des spécifications sont connues dans différentes conditions biologiques pour illustrer l'utilisation de la méthode dans le cas de conditions multiples, et enfin sur un problème de couplage de modèles et d'optimisation de planning de traitement par un anticancéreux pour montrer comment on peux optimiser une loi de contrôle par la méthode de recherche de paramètres. Les méthodes d'analyse de robustesse et d'analyse de sensibilité sont appliquées à un système de biologie synthétique, une cascade d'inhibitions génique.

Chapitre 1

Paradigme logique pour la biologie systémique

Les trois sections de ce chapitre décrivent le paradigme logique pour la biologie des systèmes présenté dans l'introduction : (i) l'identification d'un modèle biologique à un système de transition (ii) la représentation des propriétés biologiques par des formules de logique temporelle et (iii) la vérification de ces propriétés par model-checking.

1.1 Système de transition

Afin de pouvoir y appliquer des méthodes automatiques d'analyse on identifie un modèle biologique à un système de transition. Plus précisément un système biologique est représenté par un ensemble de réactions chimiques. Dans un second temps cet ensemble de réactions chimiques est simulé sur un horizon temporel fini pour produire une trace de simulation décrivant l'évolution au cours du temps des concentrations des espèces du système. C'est cette trace de simulation qui est représentée par une structure de Kripke et sur laquelle sont évaluées les formules de logique temporelle. Cela correspond au domaine de la vérification temps réel (*runtime verification* [17]) où l'on vérifie un comportement d'un système dynamique plutôt que tous ses comportements possibles. On définit dans un premier ces structures avant de décrire les modèles biologiques étudiés et leur simulation numérique produisant une trace de simulation.

1.1.1 Structure de Kripke

Une structure de Kripke est un automate fini non déterministe utilisé pour représenter le comportement d'un système. Une structure de Kripke est constituée d'états (description instantanée du système), un ensemble de transitions entre états (évolution du système) et une fonction qui étiquette chaque états par les propriétés vraies dans cet état.

Définition. *Soit AP un ensemble de propositions atomiques. Une structure de Kripke est un 4-uplet $K = (S, S_0, R, L)$ où S est un ensemble d'états sur lesquels les propositions atomiques sont évaluées, $S_0 \subset S$ est l'ensemble des états initiaux, $R \subseteq S \times S$ est*

la relation de transition entre états, supposée totale ($\forall s \in S, \exists s' \in S, (s, s') \in R$) et $L : S \to 2^{AP}$ est une fonction qui étiquette chaque état par l'ensemble des propositions vraies dans cet état.

Un chemin de K ayant pour origine s_0 est une séquence d'états infinie $T = s_0, s_1, \cdots$ tel que $(s_i, s_{i+1}) \in R$ pour tout $i \geq 0$.

Les structures de Kripke, bien que simples, sont suffisamment expressives pour capturer des comportements complexes. Les logiques modales sont interprétées sur ces structures.

1.1.2 Modèle de réactions chimiques

On se place dans le cadre de systèmes de réactions biochimiques au sein d'une cellule modélisés à un niveau d'abstraction continu, c'est à dire qu'on considère les valeurs de concentrations continues des espèces de ces systèmes. La concentration d'une molécule A est notée [A].

Ces systèmes peuvent être modélisés par un système d'équations différentielles ordinaires. Leur représentation peut se faire sous la forme de règles de réactions chimiques associées à leur cinétique de réaction (modèle SBML [18] ou modèle Biocham [19]). Un système d'équations différentielles ordinaires peut être obtenu automatiquement d'un ensemble de règles de réactions. Ces modèles peuvent représenter par exemple des réseaux métaboliques, des réseaux d'interaction génique ou des réseaux de signalisation.

1.1.3 Simulation numérique d'un système de réactions chimiques

Etant donné un état initial (valeurs de concentrations initiales des espèces du système) l'évolution du système est déterministe et des méthodes d'intégration numérique (intégration d'un système d'équations différentielles ordinaires) peuvent être utilisées pour obtenir une série temporelle de l'évolution au cours du temps des espèces du système sur un horizon borné. Les systèmes biologiques étudiés pouvant être fortement non linéaires il convient d'utiliser des méthodes pour systèmes raides. Biocham repose sur la méthode implicite à pas d'intégration variable de résolution pour systèmes raides de Rosenbrock.

La série temporelle de simulation est une structure de Kripke particuliere : c'est une structure linéaire finie. L'état initial est défini par les valeurs initiales des espèces du système. La relation de transition est la relation qui lie un état à son état immédiatement suivant dans la trace de simulation obtenue par intégration numérique. 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.

Lorsque l'intégration se fait par une méthode d'intégration a pas variable (comme la méthode de Rosenbrock) ces transitions représentent ainsi un intervalle de temps variable.

Plus généralement toute modélisation pouvant produire par simulation une trace

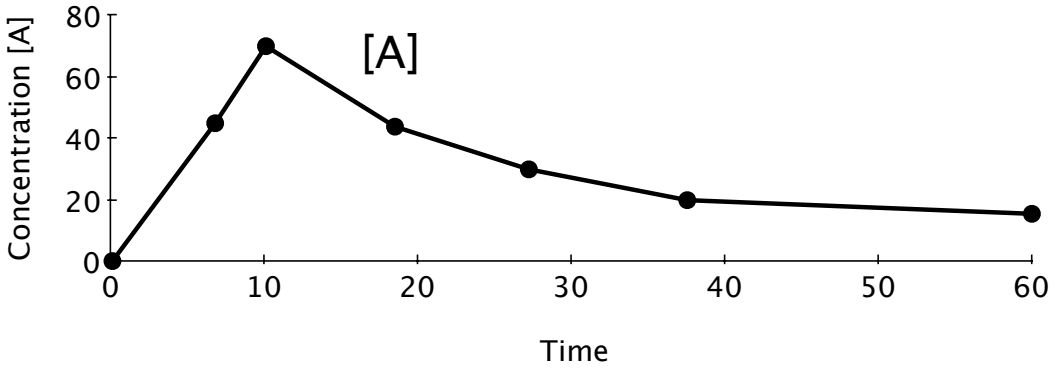


FIGURE 1.1 – Exemple de série temporelle représentant l'évolution au cours du temps des concentrations de la molécule A.

de l'évolution au cours du temps des espèces du système peut être utilisée (équations différentielle ordinaire ou stochastique [20, 21], langage de règle comme SBML [18] ou Biocham [22, 23], réseau de Petri hybride [24, 25], processus de calcul stochastique [26, 27]) (dans le cas de modélisation stochastiques il conviendrait de considérer un ensemble de traces). Les séries temporelles d'évolution au cours du temps d'espèces d'un système peuvent également provenir de données expérimentales. Les chapitres 2 et 3 se placent dans ce cadre général ou le comportement du système est représenté par une structure de Kripke. Dans les chapitres 4 (implémentation) et 5 (application aux problèmes biologiques) on considère uniquement des traces obtenues par simulation de modèles écrits sous forme de règles de réactions Biocham ou extraites de données expérimentales.

1.2 Langage de propriétés : LTL avec contraintes sur les réels

Un des défis de la biologie de systèmes est le développement d'outils s'accommodant de l'imprécision de certaines données expérimentales. De plus les propriétés biologiques considérées pour valider un modèle sont habituellement des propriétés de haut niveau qui ne s'attachent pas forcément aux valeurs particulières des espèces à des instants précis.

La Figure 1.1 donne un exemple de série temporelle représentant l'évolution au cours du temps des concentrations d'une molécule A. Les informations pertinentes à retenir de cette trace peuvent varier selon le problème biologique considéré. Aussi l'imprécision sur les mesures incite à décrire la trace de manière semi-quantitative. Par exemple on peut décrire la trace par :

- [A] dépasse la valeur 60
- [A] dépasse la valeur 60 puis reste au dessus de 20
- [A] augmente puis diminue (et rien d'autre ne se produit)
- [A] atteint un maximum local de valeur 70

Ces propriétés sont à la fois quantitatives (valeurs de seuil ou de temps) et qualita-

tives (existence d'un maximum local). Ce niveau de description, beaucoup plus simple que par une simple courbe, doit souvent être considéré pour des données expérimentales temporelles et il peut être formalisé par des formules de logique temporelles. Aussi dans le cadre de la biologie synthétique, on est amené à décrire les spécifications souhaitées d'un système en construction. Ces spécifications ne sont pas dans ce cas extraites d'observations expérimentales et sont également des propriétés de haut niveau de ce type.

La logique temporelle adaptée à la formalisation de propriétés sur une trace est la logique LTL *Linear Time Logic* [11]. La logique LTL étend la logique propositionnelle par des opérateurs déterminant à quels instants une formule est vérifiée sur un chemin.

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), U ("until", jusqu'à ce que), et W ("weak until").

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 $F\phi = \text{true}\ U\ \phi$, $G\phi = \phi\ W$ false.

Afin d'exprimer des propriétés temporelles portant sur les valeurs de concentration continues des espèces d'un système on considère une version de LTL avec contraintes sur les réels notée LTL($\mathbb{R}$). 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. La syntaxe des formules LTL($\mathbb{R}$) est donnée Table 1.1. On se restreint aux formules sans négation : en pratique on réécrit les formules avec négation en formule sans négation en propageant les négations aux formules atomiques.

Formule : :=	Atome | Formule $\wedge$ Formule | Formule $\vee$ Formule
	| X Formule | F Formule | G Formule
	| Formule U Formule | Formule W Formule
Atome : :=	Terme Op Terme
Op : :=	$< \mid > \mid \leq \mid \geq$
Terme : :=	Valeur | Terme + Terme | Terme - Terme | - Terme
	| Terme $\times$ Terme | Terme / Terme | Terme ^ Terme
Valeur : :=	flottant | [molécule] | d[molécule]/dt | Temps

TABLE 1.1 – Syntaxe des formules LTL($\mathbb{R}$).

La sémantique standard de formules LTL est habituellement définie vis à vis de traces infinies. Dans notre cas, les traces considérées étant des traces finies, la sémantique habituelle des formules LTL doit être adaptée. La sémantique des formules LTL sur une trace numérique finie $T = (s_0, s_1, \cdots, s_n)$ est donnée Table 1.2. Cette sémantique correspond à la sémantique standard sur les traces finies complétées par une boucle sur le dernier état. Elle diffère de la sémantique neutre introduite dans [28] sur les traces finies seulement pour l'opérateur X qui dans cette définition est toujours fausse sur le dernier état. En pratique l'opérateur X étant principalement utilisé pour détecter des extremum locaux cette différence n'est pas significative.

Les exemples de description de la trace de la Figure 1.1 peuvent être formalisés en

$T \models \phi$	ssi	$s_0 \models \phi$,
$s_i \models \alpha$	ssi	α est une formule propositionnelle vraie en l'état s_i,
$s_i \models !\psi$	ssi	$s_i \not\models \psi$,
$s_i \models \psi \ \& \ \psi'$	ssi	$s_i \models \psi$ et $s_i \models \psi'$,
$s_i \models \psi \mid \psi'$	ssi	$s_i \models \psi$ ou $s_i \models \psi'$,
$s_i \models \psi \Rightarrow \psi'$	ssi	$s_i \models \psi'$ ou $s_i \not\models \psi$,
$s_i \models X\psi$	ssi	$i < n$ et $s_{i+1} \models \psi$ ou $i = n$ et $s_n \models \psi$,
$s_i \models \psi \mathbf{U} \psi'$	ssi	il existe $j \in [i, n]$ tel que $s_j \models \psi'$ et $s_k \models \psi$ pour tout $k \in [i, j-1]$.
$s_i \models \psi \mathbf{W} \psi'$	ssi	soit pour tout $j \in [i, n]$, $s_j \models \psi$. soit il existe $j \in [i, n]$ tel que $s_j \models \psi \& \psi'$ et pour tout $k \in [i, j-1]$, $s_k \models \psi$.

TABLE 1.2 – Définition inductive de la valeur de vérité des formules LTL ψ et ψ' sur une trace numérique finie $T = (s_0, s_1, \cdots, s_n)$.

logique temporelle :

- [A] dépasse la valeur 60 :
  F([A]>60)

- [A] dépasse la valeur 60 puis reste au dessus de 20 :
  F([A]>60 $\wedge$ **G**([A]> 20))

- [A] augmente puis diminue (et rien d'autre ne se produit) :
  (d[A]/dt>0) **U** (**G** (d[A]/dt<0))

- [A] atteint un maximum local de valeur 70 :
  F([A] < 70 $\wedge$ X([A] = 70 $\wedge$ X([A] < 70)))

Un grand nombre de propriétés peuvent être représentées en LTL($\mathbb{R}$) comme par exemple des propriétés d'oscillations par une succession de changements de signes de la dérivée de M au moins K fois :
`F((d[M]/dt > 0)` $\wedge$ `F((d[M]/dt < 0)`
$\wedge$ `F((d[M]/dt > 0)...)))`
Le chapitre 5 d'application à des problèmes biologiques fournit d'autres exemples de représentations de propriétés temporelles

Il est intéressant de noter que lorsque la trace numérique correspond à une discrétisation d'un processus continu, certains événements de la trace peuvent être perdus. Par exemple la formule **F**(`[A]` ≥ 60) interprétée sur la trace de la Figure 1.1 peut être vraie ou fausse selon le pas d'intégration. On peut se rapporter à [28, 29] pour une discussion sur la vérification de formules temporelles sur des traces discrètes finies.

1.3 Vérification de modèle par model-checking

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($\mathbb{R}$) avec contraintes [7, 8]

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 $X\phi$ au prédécesseur immédiat d'un instant étiqueté par ϕ ;
3. assigner les sous formules de la forme $\phi_1 \mathbf{U} \phi_2$ aux instants précédant ceux étiquetés par ϕ_2 et cela tant que ϕ_1 est vrai.
4. assigner les sous formules de la forme $\phi_1 \ W \ \phi_2$ au dernier instant de la trace s'il est étiqueté par ϕ_1, et aux prédécesseurs des instants étiquetés par $\phi_1 \ W \ \phi_2$ tant que ϕ_1 est vrai et assigner les sous formules de la forme $\phi_1 \ W \ \phi_2$ 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.

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 [7].

Chapitre 2

Résolution de contraintes temporelles

La logique temporelle linéaire avec contraintes LTL($\mathbb{R}$) présentée au chapitre 1 permet d'exprimer des comportements sur des traces linéaires, des traces représentant l'évolution au cours du temps des concentrations des espèces d'un modèle de réactions biochimiques. En biologie des systèmes le problème central est, plutôt qu'un problème de vérification, un problème de *reverse engineering* : il faut inférer un modèle à partir des observations globales d'un système dans différentes conditions. Cela correspond au problème de synthèse de modèle à partir de spécification temporelle [30, 31].

2.1 LTL($\mathbb{R}$) avec variables : QFLTL($\mathbb{R}$)

Nous allons présenter dans cette section la logique *Quantifier Free Linear Time Logic* QFLTL($\mathbb{R}$), construite à partir de la logique temporelle linéaire en introduisant des variables libres, qui permet de généraliser le problème de model-checking en un problème de résolution de contraintes.

Exemple 2.1.1 *Considérons la trace donnée figure 2.1 qui représente l'évolution dans le temps de la concentration de la molécule A. La concentration de A augmente, atteint son maximum, diminue à son minimum, puis se stabilise à une valeur intermédiaire. Ce comportement temporel peut par exemple se représenter par la formule LTL :*

$$F([A] \geq 7 \wedge F([A] \leq 0))$$

Cette formule, par l'imbrication des opérateurs temporels F, indique que [A] atteint la valeur 7 puis la valeur 0. La satisfaction de cette formule sur la trace T est vérifiée par l'algorithme de model-checking présenté au chapitre 1. La formule est fausse sur la trace T car la valeur minimale atteinte par $[A]$ est seulement 2.

La réponse du model-checking classique est binaire, la formule est vraie ou fausse, et ne fournit donc pas d'information plus précise sur la satisfaction ou non de la formule : une partie de la formule est-t-elle vraie ? si une valeur particulière de seuil n'est pas atteinte, pour quelle valeur de seuil la formule serait elle vraie ?

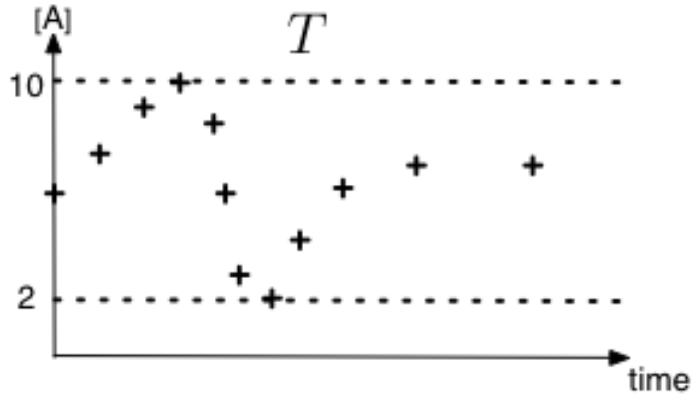


FIGURE 2.1 – Trace de simulation indiquant l'évolution au cours du temps de la concentration de A notée $[A]$.

Pour obtenir ces informations quantitatives on considère le fragment sans quantificateur de l'extension au premier ordre de LTL($\mathbb{R}$) que l'on nomme QFLTL($\mathbb{R}$) (*Quantifier Free Linear Time Logic*). Syntaxiquement le langage QFLTL($\mathbb{R}$) permet la présence de variables libres à valeurs réelles la où des valeurs sont présentes dans les formules LTL($\mathbb{R}$) :

$$\text{Valeur} ::= \text{Variable} \mid \text{flottant} \mid [\text{molécule}] \mid \text{d}[\text{molécule}]/\text{dt} \mid \text{Temps}$$

Le problème de model-checking de formules LTL($\mathbb{R}$) se généralise pour les formules QFLTL($\mathbb{R}$) en un problème de satisfaction de contraintes : pour quelles valeurs $\boldsymbol{y}$ de ses variables, une formule QFLTL($\mathbb{R}$) $\phi(\boldsymbol{y})$ est-elle vraie sur une trace T ?

Définition 2.1.1 *Domaine de validité*
Etant donné une trace T et une formule QFLTL($\mathbb{R}$) $\phi(\boldsymbol{y})$ portant sur un vecteur de variables réelles $\boldsymbol{y}$, le problème de satisfaction de contraintes est de déterminer les valuations des variables $\boldsymbol{y}$ pour lesquelles la formule est vraie. On définit le domaine de validité $\mathcal{D}_{T,\phi(\boldsymbol{y})}$ d'une formule portant sur des variables y par rapport à une trace T :

$$\mathcal{D}_{T,\phi(\boldsymbol{y})} = \{\boldsymbol{y} \in \mathbb{R}^q \mid T \models \phi(\boldsymbol{y})\} \tag{2.1}$$

2.2 Algorithme de résolution de contraintes

Le domaine de validité exact $\mathcal{D}_{T,\phi(\boldsymbol{y})}$ de toute formule QFLTL($\mathbb{R}$) $\phi(\boldsymbol{y})$ est obtenu par l'algorithme 2.

L'algorithme de résolution de contraintes QFLTL($\mathbb{R}$) calcule les domaines de validité des variables de la formule pour chaque instant de la trace T, par double induction sur les sous-formules de ϕ et sur les états de T. Cet algorithme a été publié dans [32, 33] et est généralisé en un algorithme de point fixe pour la résolution de contraintes CTL dans la section 2.6.

Algorithme 2 Résolution de contraintes temporelles (calcul du domaine de validité)

Soit $T = (s_0, s_1, \ldots, s_n)$ une trace et $\phi(\boldsymbol{y})$ une formule QFLTL, le domaine de validité $\mathcal{D}_{T,\phi(\boldsymbol{y})}$ est calculé par induction sur T et les sous formules de $\phi(\boldsymbol{y})$ en utilisant les égalités suivantes :

- $\mathcal{D}_{T,\phi} = \mathcal{D}_{s_0,\phi}$,
- $\mathcal{D}_{s_i,\alpha} = \{\boldsymbol{y} \in \mathbb{R}^v \mid \alpha(\boldsymbol{y})$ est vrai à l'état $s_i\}$,
- $\mathcal{D}_{s_i,\phi\wedge\psi} = \mathcal{D}_{s_i,\phi} \cap \mathcal{D}_{s_i,\psi}$,
- $\mathcal{D}_{s_i,\phi\vee\psi} = \mathcal{D}_{s_i,\phi} \cup \mathcal{D}_{s_i,\psi}$,
- $\mathcal{D}_{s_i,\mathbf{X}\phi} = \mathcal{D}_{s_{i+1},\phi}$,
- $\mathcal{D}_{s_i,\mathbf{F}\phi} = \cup_{j\geq i}\mathcal{D}_{s_j,\phi}$,
- $\mathcal{D}_{s_i,\mathbf{G}\phi} = \cap_{j\geq i}\mathcal{D}_{s_j,\phi}$,
- $\mathcal{D}_{s_i,\phi\mathbf{U}\psi} = \cup_{j\geq i}(\mathcal{D}_{s_j,\psi} \cap \cap_{k\in[i,j-1]}\mathcal{D}_{s_k,\phi})$.

A chaque valuation des variables de la formule appartenant au domaine de validité correspond une formule LTL($\mathbb{R}$). Le domaine de validité d'une formule QFLTL($\mathbb{R}$) est ainsi un ensembles de formules LTL($\mathbb{R}$), ce sont les formules du type défini par $\phi(\boldsymbol{y})$ vraies sur la trace T.

Un domaine de validité vide correspond à une formule fausse pour toute valuation de ses variables tandis que le domaine de validité d'une formule vraie pour toute valuation de ses variables est l'espace entier.

Exemple 2.2.1 *Considérons de nouveau la formule LTL($\mathbb{R}$) $F([A] \geq 7 \wedge F([A] \leq 0))$. En remplaçant les constantes 7 et 0 par les variables x et y on obtient la formule QFLTL($\mathbb{R}$) : $F([A] \geq x \wedge F([A] \leq y))$. Le domaine de validité de cette formule sur la trace 2.1 retourné par l'algorithme 2.3 est $x \leq 10 \wedge y \geq 2$ et est représenté figure 2.2. Ces valeurs correspondent respectivement au maximum de $[A]$ dans la trace puis à son minimum.*

Le calcul du domaine de validité permet donc, selon le type de formule, d'analyser et de récupérer certaines caractéristiques d'une trace. L'analyse de traces par calcul de domaine de satisfaction est abordé chapitre 5 section 5.1.

2.3 Structure du domaine de validité

Les propositions atomiques des formules QFLTL($\mathbb{R}$) considérées dans les sections précédentes incluent des inégalités non linéaires qui seraient trop générales pour permettre le calcul du domaine de validité sans approximation.

On considère le fragment de QFLTL, qu'on nomme QFLTL($\mathbb{R}_{\text{box}}$) dans lequel chaque formule atomique contient au plus une variable :

Atome : := *Terme Op Terme* | *Terme Op variable*
Valeur : := *flottant* | *[molécule]* | *d[molécule]/dt* | *Temps*

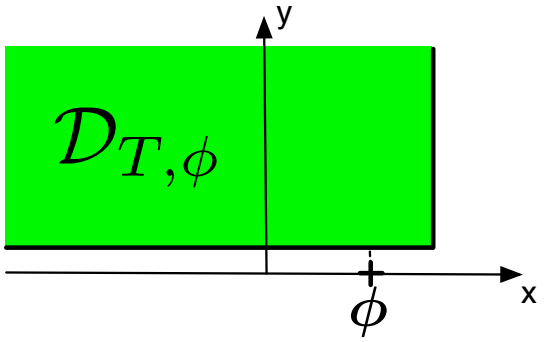


FIGURE 2.2 – Domaine de validité de la formule $F([A] \geq x \wedge F([A] \leq y))$ sur la trace 2.1.

On considère également le fragment linéaire QFLTL($\mathbb{R}_{\text{lin}}$) qui étend QFLTL($\mathbb{R}_{\text{box}}$) en permettant des contraintes linéaires sur les variables dans les propositions atomiques.

On décrit dans cette section la structure générale du domaine de validité généré par l'algorithme de résolution de contrainte pour les fragments QFLTL($\mathbb{R}_{\text{lin}}$) et QFLTL($\mathbb{R}_{\text{box}}$). On définit d'abord les demi-espaces affines, les polyèdres convexes et les orthotopes.

Définition 2.3.1 *Demi-espace affine*
On note $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ *le produit scalaire de deux vecteurs de* $\mathbb{R}^n$. *Pour* $\boldsymbol{a} \in \mathbb{R}^n$ *et un scalaire* $b \in \mathbb{R}$ *et pour chaque symbole de relation* $\bowtie \in \{\geq, > \leq, <\}$ *la contrainte linéaire* $\langle \boldsymbol{a}, x \rangle \bowtie b$ *définit un demi-espace affine. Pour* $\boldsymbol{a} = \boldsymbol{0}$ *la contrainte* $\langle \boldsymbol{0}, x \rangle \bowtie b$ *peut être soit une tautologie soit être inconsistante et définir ainsi soit l'espace entier* $\mathbb{R}^n$ *soit l'ensemble vide* $\emptyset$.

Définition 2.3.2 *Polyèdre convexe*
Un ensemble $\mathcal{P} \subset \mathbb{R}^n$ *est un* polyèdre convexe *si et seulement si il peut s'écrire sous la forme d'une intersection finie de demi-espaces affines de* $\mathbb{R}^n$.

On utilisera dans ce travail uniquement des polyèdres convexes qu'on appellera simplement polyèdres.

Définition 2.3.3 *Orthotope*
On appelle orthotope *de* $\mathbb{R}^n$ *le produit cartésien de n intervalles de* $\mathbb{R}$, *éventuellement ouverts ou non bornés. De manière équivalente on peut définir un orthotope comme un polyèdre particulier qui peut s'écrire sous la forme d'une intersection finie de demi-espaces affines représentés par des contraintes linéaires où* $\boldsymbol{a}$ *est non nul dans une seule dimension (contrainte sur une seule variable).*

La proposition suivante décrit la structure du domaine de validité généré par l'algorithme appliqué aux fragments QFLTL($\mathbb{R}_{\text{lin}}$) et QFLTL($\mathbb{R}_{\text{box}}$).

Proposition 2.3.1 *Le domaine calculé par l'algorithme de résolution de contraintes QFLTL($\mathbb{R}_{box}$) est une union finie d'orthotopes. Le domaine calculé par l'algorithme de résolution de contraintes QFLTL($\mathbb{R}_{lin}$) est une union finie de polyèdres.*

Preuve. Dans le cas QFLTL($\mathbb{R}_{\text{box}}$) : le domaine de validité d'une conjonction de contraintes contenant uniquement une variable est un intervalle sur les réels. Pour des dimensions plus élevées, c'est à dire quand la formule porte sur v variables, le domaine de validité d'une conjonction de propositions atomiques est un orthotope, c'est à dire le produit cartésien de v intervalles de $\mathbb{R}$. Ces orthotopes sont calculés dans le cas de base de la définition inductive du domaine de validité dans l'algorithme 2. Dans les autres cas, l'algorithme de résolution de contraintes calcule des intersections et unions finies de domaines de validité. Une intersection finie d'unions d'orthotopes étant une union finie d'orthotopes, les domaine calculés par l'algorithme sont toujours des unions finies d'orthotopes.

Dans le cas QFLTL($\mathbb{R}_{\text{lin}}$) : les domaines de validité de contraintes linéaires sont des polyèdres. L'intersection de deux polyèdres étant un polyèdre, l'algorithme de résolution de contraintes sur QFLTL($\mathbb{R}_{\text{lin}}$) calcule des unions finies de polyèdres.

□

Pour le problème du calcul de l'ensemble des états atteignables d'un système il a été proposé d'utiliser la classe des zonotopes (sous classe des polyèdres) [34, 35]. Les zonotopes sont stables par somme de Minkowski et permettent des transformations linéaires efficaces. Cette classe est donc un compromis intéressant car elle a une richesse d'expressivité supérieure aux orthotopes et une manipulation plus efficace que les polyèdres. Cependant les zonotopes ne sont pas stables par intersection et ne peuvent donc pas être utilisés dans notre cas pour calculer le domaine de validité exact d'une formule.

2.4 Correction et complétude

Théorème 2.4.1 (Complétude forte) *L'algorithme de résolution de contraintes est correct et complet : une valuation $\vec{v}$ rend une formule QFLTL($\mathbb{R}$) 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($\mathbb{R}$) que pour tout instant t, toute formule QFLTL ϕ et toute 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 $Valeur\ Op\ Variable$ ou $Valeur\ Op\ Valeur$ où $Valeur$ est une expression arithmétique 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 $(\phi_1 \; W \; \phi_2) = (\phi_1 \wedge \phi_2) \vee (\phi_1 \wedge X(\phi_1 \; W \; \phi_2))$.

□

2.5 Complexité

2.5.1 Orthotopes

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

Théorème 2.5.1 (Complexité du domaine de validité) *Le domaine de validité d'une formule QFLTL($\mathbb{R}_{box}$) de taille f contenant k variables, sur une trace de longueur n est une union d'au plus $(nf)^{2k}$ orthotopes.*

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 $Valeur(t_i)$ Op $Variable$. Une telle contrainte peut éventuellement être évaluée sur chaque instant de la trace et ainsi créer au plus n bornes pour x. Le nombre maximum de bornes pour la variable x est donc n fois le nombre d'occurrence de x dans ϕ, ce 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 $\mathcal{B}_v(\phi_1 \wedge \phi_2) \subset \mathcal{B}_v(\phi)$ car 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 $(nf)^{2k}$ orthotopes pour une formule contenant k variables. Ainsi, le domaine calculé par l'algorithme est une union finie d'orthotopes (Prop. 2.3.1) de taille inférieure à $(nf)^{2k}$.

□

2.5.2 Polyèdres

Le fragment linéaire QFLTL($\mathbb{R}_{\text{lin}}$) étend QFLTL($\mathbb{R}_{\text{box}}$) en permettant des contraintes linéaires sur les variables dans les propositions atomiques.

Proposition 2.5.1 *Le domaine de validité d'une formule QFLTL($\mathbb{R}_{lin}$) de taille f sur une trace de longueur n est une union d'au plus 2^{nf} polyèdres.*

Preuve. Un polyèdre est représenté par un ensemble de contraintes linéaires, considérons le nombre possible de contraintes linéaires apparaissant dans le domaine de validité $\mathcal{D}_{T,\phi}$ d'une formule ϕ de taille f sur une trace T de taille n. Les contraintes sont uniquement générées par le cas de base de l'algorithme 2 de calcul du domaine. Chaque contrainte de ϕ peut être évaluée sur chaque point de la trace, générant ainsi au plus n contraintes différentes. Les intersection et les unions ne créent pas de nouvelles contraintes et comme le nombre de contraintes dans ϕ est inférieur à f, le nombre de contraintes dans $\mathcal{D}_{T,\phi}$ est inférieur à nf.

Un polyèdre est défini par un nombre arbitraire de contraintes. Avec au plus nf contraintes on peut créer au plus 2^{nf} polyèdres. Ainsi le domaine de validité $\mathcal{D}_{T,\phi}$ est une union d'au plus 2^{nf} polyèdres.

□

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 $[A_i]$ où les valeurs des $[A_i] + k$ sont toutes différentes pour $1 \leq i \leq v$ et $0 \leq k \leq f$.

La performance de l'algorithme de calcul de domaine évaluée chapitre 5 section 2 indique que cette borne maximale sur la complexité n'est pas atteinte dans les cas usuels, où la plupart des domaines sont des orthotopes.

2.6 Degré de satisfaction continu de formules

Il est intéressant de noter qu'une formule LTL($\mathbb{R}$) peut être vue comme une instance d'une formule QFLTL($\mathbb{R}$) obtenue par abstraction des constantes de la formule par de nouvelles variables $\boldsymbol{y} \in \mathbb{R}^q$. Par exemple, à la formule $\phi_1 = \mathbf{F}(\texttt{[A]} > 7 \wedge \mathbf{F}\ \texttt{[A]} < 3)$, on associe la formule $\phi(\boldsymbol{y}) = \phi(y_1, y_2) = \mathbf{F}(\texttt{[A]} > y_1 \wedge \mathbf{F}\ \texttt{[A]} < y_2)$. On a alors $\phi_1 = \phi(7, 3)$.

De manière plus générale le processus d'abstraction/d'instanciation de variables permet de voir une formule LTL($\mathbb{R}$) comme un point dans l'espace de formules QFLTL($\mathbb{R}$)

$\mathbb{R}^q$, où q est le nombre de constantes apparaissant dans ϕ. (ou le nombre de constantes remplacées par des variables si toutes les constantes ne sont pas abstraites par des variables). Ainsi l'algorithme de résolution de contraintes QFLTL détermine dans cet espace de formules QFLTL($\mathbb{R}$) l'ensemble des formules vraies sur une trace donnée, appelé le domaine de validité de la formule. On peut donc comparer une formule LTL($\mathbb{R}$) à cet ensemble de formules vraies pour évaluer la satisfaction de la formule sur une trace.

Afin d'évaluer de manière numérique l'adéquation d'un modèle vis à vis d'une spécification temporelle, on définit un degré de violation continu reliant la trace numérique d'un modèle à une formule LTL($\mathbb{R}$) avec contraintes. Lorsque le modèle satisfait sa spécification le degré de violation est nul. Plus la trace est différente de sa spécification plus le degré de violation est important.

2.6.1 Satisfaction quantitative de formules QFLTL($\mathbb{R}$)

Soit T une trace et ϕ une formule QFLTL($\mathbb{R}$) que l'on veut évaluer de manière continue sur l'intervalle $[0, 1]$. Pour cela, on introduit le *motif de formule* QFLTL($\mathbb{R}$) $\psi(x_1, ..., x_k)$ obtenu en remplaçant certaines constantes de ϕ par de nouvelles variables $\{x_1, ..., x_k\}$: on a $\phi = \psi(v_1, ..., v_k)$ pour une instanciation des variables par les valeurs réelles $v_1, ..., v_k$ (la section suivante indique comment on peut faire automatiquement cette abstraction).

La résolution de contraintes QFLTL($\mathbb{R}$) calcule le *domaine de validité* de ψ sur T sous la forme du domaine sur ses k variables $D_{T,\psi} \subset \mathbb{R}^k$ Le degré de satisfaction de ϕ sur T est alors défini en utilisant la distance entre le domaine de validité des $x_1, ..., x_k$ sur T et les valeurs objectif $(v_1, ..., v_k)$ dans $\mathbb{R}^k$.

Définition 2.6.1 *Le* degré de violation $vd(T, \phi, \psi)$ *d'une formule QFLTL* ϕ *sur une trace numérique* T *par rapport au motif de formule* $\psi(\boldsymbol{y})$ *tel que* $\phi = \psi(\boldsymbol{v})$ *pour certaines valeurs* $\boldsymbol{v}$ *des variables* $\boldsymbol{y}$*, est la distance Euclidienne entre* $\boldsymbol{v}$ *et la projection sur les variables* $\boldsymbol{y}$ *du domaine* $(D_{T,\psi})$*, ou* $+\infty$ *si* $D_{T,\psi} = \emptyset$ *:*

$$vd(T, \phi, \psi) = min_{\boldsymbol{v'} \in D^{\boldsymbol{y}}_{T,\psi}} d(\boldsymbol{v'}, \boldsymbol{v})$$

Le degré de satisfaction *de* ϕ *sur* T *par rapport à* ψ *est*

$$sd(T, \phi, \psi) = \frac{1}{1 + vd(T, \phi, \psi)}$$

L'abstraction de constantes par des variables dans les formules LTL est ainsi un moyen de définir une *métrique* sur l'ensemble des formules. Intuitivement, ϕ définit une spécification, i.e. un objectif, ψ défini l'espace de recherche, tandis que la trace T permet de définir quelle partie de l'espace de recherche satisfait la spécification.

Le degré de violation est ici défini vis à vis des formules ϕ et ψ, de manière équivalente on peut définir le degré de violation vis à vis d'une formule QFLTL avec variables ψ et de l'instanciation de certaines de ces variables (ce qui revient à définir la formule instanciée ϕ). On appelle l'instanciation de ces variables *l'objectif* (ou spécification objectif) des variables de la formule.

Le degré d'évaluation continu de formules et son application a été publié dans [36, 37].

Exemple 2.6.1 *Considérons de nouveau la formule LTL* $\phi = F([A] \geq 7 \wedge F([A] \leq 0))$ *de l'exemple 2.2.1 indiquant qu'il a été observé dans des expériences qu'après un certain temps la concentration de l'espèce A devient plus grande que 7 puis plus petite que 0. Prenons la formule QFLTL* $\psi = F([A] \geq x \wedge F([A] \leq y))$ *comme motif de formule* $\mathbb{R}$ *indiquant que l'on est intéressé par les valeurs réellement atteintes par* $[A]$*. La valeur objectif est* $\boldsymbol{v} = (7, 0)$*.*

Etant donné la trace 2.1, l'algorithme de résolution de contraintes QFLTL appliqué à ψ *sur la trace T calcule* $D^x_{T,\psi} =]-\infty, 10]$ *et* $D^y_{T,\psi} = [2, \infty[$ *comme domaine pour les variable x et y. Le domaine de validité* $D_{T,\psi}$ *de la formule est le produit cartésien de ces deux intervalles. Comme* $\boldsymbol{v} = (7, 0)$ *on obtient* $vd(T, \phi, \psi) = 2$*, c'est à dire que le degré de violation est 2 car le composé atteint bien le le maximum 7 mais seulement le minimum 2 alors que la formule indique que le seuil 0 est atteint. Le degré de satisfaction de* ϕ *est alors 1/3.*

Exemple 2.6.2 *Considérons la formule QFLTL* $\phi = F([A] \geq x) \wedge F([A] \leq y) \wedge x - y \geq 10$*. Les valeurs possibles de x vont de* $-\infty$ *à la valeur maximum M atteinte par* $[A]$ *dans la trace. De manière similaire, y va du minimum m de* $[A]$ *à* ∞*. Ainsi la quantité* $x - y$ *va de* $-\infty$ *à* $M - m$*, la contrainte* $x - y \geq 10$ *contraint les amplitudes maximales des variations de* $[A]$ *dans la trace.*

La formule $\psi = F([A] \geq x) \wedge F([A] \leq y) \wedge x - y \geq amp$ *permet de raisonner sur l'amplitude de ces variations. Comme amp est la seule variable d'intérêt (la seule instanciée entre* ψ *et* ϕ*) elle est la seule variable utilisée pour définir l'objectif que l'amplitude doit être d'au moins 10.*

Supposons que l'algorithme de résolution de contrainte calcule le domaine de validité suivant pour x et y : $D^x_{T,\psi} =]-\infty, 15]$ *et* $D^y_{T,\psi} = [10, +\infty[$*. Pour la formule* ϕ*, la valeur maximale de* $D^x_{\pi,\psi}$ *représente la valeur maximale de* $[A]$ *et la valeur minimale de* $D^y_{\pi,\psi}$ *sa valeur minimum dans la trace. Le domaine pour la variable amp est* $D^{amp}_{\pi,\psi} =]-\infty, 5]$ *car on sait que* $amp \leq x - y$*. Ainsi, comme la valeur objectif est* 10*, on obtient* $vd(T, \phi, \psi) = 5$*, c'est à dire que l'amplitude de la courbe est 5 alors que l'on voulait qu'elle soit d'au moins 10. Le degré de satisfaction de* ϕ *est 1/6.*

2.7 Abstraction de constantes par des variables : de (QF)LTL à QFLTL

Une formule LTL(ℝ) peut être vue comme une formule QFLTL(ℝ) ne contenant pas de variables. Les définitions de la section précédente s'appliquent à une formule QFLTL sans variable. Cependant, le plus souvent, on définit une spécification sous la forme d'une formule LTL sans ajouter une formule QFLTL ψ.

On présente dans cette section une abstraction α de LTL vers QFLTL fournissant un choix par défaut pour la formule ψ lorsque ϕ est une formule LTL. Ceci permet une définition directe du degré de violation d'une formule LTL sur une trace. On parlera ainsi dans la suite de ce travail de degré de violation (ou de satisfaction) d'une formule ϕ sans toujours préciser le choix de ψ. Dans ce cas on note le degré de violation de ϕ sur une trace T, $vd(T, \phi)$ et le degré de satisfaction $sd(T, \phi)$.

Étant donné une spécification LTL ϕ décrivant le comportement requis d'un système, on génère une formule QFLTL $\alpha(\phi)$ en remplaçant les occurrences de constantes (valeurs réelles correspondant aux seuils de concentration, amplitudes, etc.) $c_1, \ldots, c_j$ apparaissant dans ϕ, par les variables distinctes $x_1, \ldots, x_j$.

Plus formellement, α est défini sur les valeurs terminales et se propage par les opérateurs des formules LTL :

$$\begin{aligned} \alpha(flottant) &= variablefraiche \\ \alpha([molecule]) &= [molecule] \\ \alpha(d[molecule]/dt) &= d[molecule]/dt \\ \alpha(Temps) &= Temps \end{aligned}$$

Il est également intéressant de noter que la substitution inverse σ, qui est une instanciation des variables $x_1, \ldots, x_n$, est telle que $\phi = \psi\sigma$.

Si des restrictions sont imposées sur la formule QFLTL, comme dans la section 2.5 pour des raisons d'efficacité du calcul, α peut ne pas fournir par défaut une formule ψ valide et il faut alors la définir à la main.

Exemple 2.7.1 *Considérons la formule : $\phi = F([A] > 20)$, on obtient $\alpha(\phi) = F([A] > x)$, qui correspond à la formule ψ qui avait été choisie car le seuil 20 est la seule occurrence de constante dans ϕ.*

Comme les expressions atomiques dans les formules QFLTL doivent y être des contraintes linéaires sur les variables libres, il n'est pas toujours possible d'abstraire toutes les constantes apparaissant dans une formule LTL par des variables.

Le résultat de l'abstraction par des variables d'une formule LTL est une formule QFLTL. Le même procédé d'abstraction peut être appliqué à une formule QFLTL, ce qui fournit une formule QLFTL contenant plus de variables.

Exemple 2.7.2 *Considérons la formule de l'exemple 2.6.2 :*

on a $\phi = F([A] \geq v) \wedge F([A] \leq w) \wedge v - w \geq 10$ et $\psi = F([A] \geq v) \wedge F([A] \leq w) \wedge v - w \geq amp$.

ϕ n'est pas une formule LTL mais l'abstraction ci-dessus donne encore $\alpha(\phi) = \psi$.

Cependant cette abstraction n'est pas toujours ce que l'on voudrait. α n'est par exemple pas adapté lorsque l'on veut normaliser les contributions de différentes variables. Par exemple si $\phi = F([A] > 20) \wedge\ F([B] < 0.001)$, αmettra le même poids sur les deux distances, alors qu'une formulation directe avec ψ permettrait une normalisation, par exemple $\psi = F([A]/20 < x) \wedge\ F([B]/0.001 > y)$.

2.8 CTL avec variables

La logique temporelle CTL, *Computation Tree Logic*, est utilisée pour raisonner sur les exécutions de systèmes non déterministes en considérant des opérateurs temporels et des opérateurs de quantification sur les chemins d'exécution.

2.8.1 Syntaxe

Soit $\mathcal{V}$ un ensemble infini de variables, $V \subseteq \mathcal{V}$ l'ensemble des *variables d'état* et Σ les constantes, fonctions et symboles de prédicat sur un domaine $\mathcal{D}$. Une *contrainte atomique* est une formule atomique sur Σ et $\mathcal{V}$, une *contrainte* notée c est une conjonction de contraintes atomiques.

Les formules CTL de premier ordre sans quantificateur (QFCTL*) sont formées à partir des contraintes atomiques, des connecteurs logiques $\neg$, $\vee$, $\wedge$, $\Rightarrow$, des quantificateurs de chemin **E**, **A** et des opérateurs temporels **X**("next"), **U**("until"), **W**("weak until"), **F**("finally", un instant dans le futur) et **G**("globally", tous les instants dans le futur). QFCTL est le fragment de QFCTL* dans lequel les opérateurs temporels sont immédiatement précédés d'un quantificateur sur les chemins.

On nomme $V(\phi)$ l'ensemble des variables d'une formule ϕ. On dit qu'une formule ϕ est *close* si $V(\phi) \subseteq V$, c'est à dire qu'elle ne contient que des variables d'état. L'ensemble des *variables libres* d'une formule ϕ est l'ensemble $V(\phi) \setminus V$.

Exemple 2.8.1 *Considérons une contrainte d'inégalité sur les réels et la formule QFCTL** $\mathbf{EF}(x = v_1 \wedge \mathbf{X}(x = v_2 \wedge v_1 < v_2 \wedge \mathbf{X}(x = v_3 \wedge v_3 < v_2)))$*, où x est une variable d'état et v_1, v_2, v_3 sont des variables libres. Cette formule exprime qu'il existe un chemin où pour un instant la valeur de x est v_1 et qu'à l'instant suivant x prend une valeur plus élevée v_2 puis à l'instant suivant prend à nouveau une valeur plus faible. Ainsi la formule indique que v_2 est un maximum local de x.*

On suppose que les prédicats sont clos par négation, tout prédicat p a un dual $\overline{p}$ tel que $\models_{\mathcal{D}} \neg p(e_1, \ldots, e_n)$ si et seulement si $\models_{\mathcal{D}} \overline{p}(e_1, \ldots, e_n)$ pour tout $e_1, \ldots, e_n \in \mathcal{D}$.

2.8.2 Sémantique

Les formules QFCTL sont interprétées sur des structures de Kripke, et sur un domaine de calcul $\mathcal{D}$. On suppose que le problème de satisfaction de contraintes sur $\mathcal{D}$ est décidable, i.e. que $\models_{\mathcal{D}} \exists X\ c$ où $X = V(c)$ est décidable,

Un *état*, $s : V \to \mathcal{D}$, est l'ensemble des valeurs de variables d'état à un moment donné. Un état est donc représenté par une $\mathcal{D}$-valuation des variables de V. On écrit $s(v)$ pour représenter la valeur de la variable v dans l'état s, et par extension $s(e)$ pour la valeur d'une expression close e dans un état s. L'ensemble des états sur V et $\mathcal{D}$ est $\mathcal{S}(V, \mathcal{D})$, ou plus simplement $\mathcal{S}$ lorsque V et $\mathcal{D}$ sont implicites.

Définition 2.8.1 *Une formule QFCTL$^*(\Sigma, V, \mathcal{D})$ close ϕ est* vraie en un état s d'une structure de Kripke $K(S, R, V, \mathcal{D})$*, si la relation $K, s \models_{\mathcal{D}} \phi$ est vérifiée selon la définition inductive de la Table 2.1.*

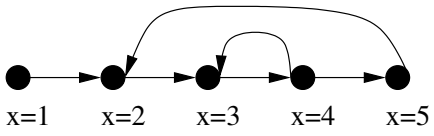
La seule différence entre la définition inductive de la Table 2.1 et la définition usuelle pour les formules propositionnelles CTL* [11] est dans le cas d'un proposition atomique α. Une proposition atomique est supposée close et est évaluée en $\mathcal{D}$ en vérifiant $\models_{\mathcal{D}} s(\alpha)$.

$K, s \models_{\mathcal{D}} \alpha$	si	α est une formule atomique et $\models_{\mathcal{D}} s(\alpha)$,
$K, s \models_{\mathcal{D}} \mathbf{E}\phi$	si	pour un chemin π de départ s, $K, \pi \models_{\mathcal{D}} \phi$,
$K, s \models_{\mathcal{D}} \mathbf{A}\phi$	si	pour tous les chemins π partant de s, $K, \pi \models_{\mathcal{D}} \phi$,
$K, \pi \models_{\mathcal{D}} \phi$	si	$K, s \models_{\mathcal{D}} \phi$ où s est le premier état de π,
$K, \pi \models_{\mathcal{D}} \mathbf{X}\phi$	si	$K, \pi^1 \models_{\mathcal{D}} \phi$,
$K, \pi \models_{\mathcal{D}} \mathbf{F}\phi$	si	il existe $k \geq 0$ s.t. $K, \pi^k \models_{\mathcal{D}} \phi$,
$K, \pi \models_{\mathcal{D}} \mathbf{G}\phi$	si	pour tous $k \geq 0$, $K, \pi^k \models_{\mathcal{D}} \phi$,
$K, \pi \models_{\mathcal{D}} \phi \mathbf{U} \phi'$	si	il existe $k \geq 0$ tel que $K, \pi^k \models_{\mathcal{D}} \phi'$ et $K, \pi^j \models_{\mathcal{D}} \phi$ pour tout $0 \leq j < k$.
$K, \pi \models_{\mathcal{D}} \phi \mathbf{W} \phi'$	si	pour tous $k \geq 0$, $K, \pi^k \models_{\mathcal{D}} \phi$ ou s'il existe $k \geq 0$ tel que $K, \pi^k \models_{\mathcal{D}} \phi \wedge \phi'$ et pour tous $0 \leq j < k$, $K, \pi^j \models_{\mathcal{D}} \phi$.
$K, \pi \models_{\mathcal{D}} \neg\phi$	si	$K, \pi \not\models_{\mathcal{D}} \phi$,
$K, \pi \models_{\mathcal{D}} \phi \ \wedge \ \phi'$	si	$K, \pi \models_{\mathcal{D}} \phi$ et $K, \pi \models_{\mathcal{D}} \phi'$,
$K, \pi \models_{\mathcal{D}} \phi \ \vee \ \phi'$	si	$K, \pi \models_{\mathcal{D}} \phi$ ou $K, \pi \models_{\mathcal{D}} \phi'$,
$K, \pi \models_{\mathcal{D}} \phi \Rightarrow \phi'$	si	$K, \pi \models_{\mathcal{D}} \phi'$ ou $K, \pi \not\models_{\mathcal{D}} \phi$,

TABLE 2.1 – Définition inductive des valeurs d'une formule QFCTL* close sur une structure de Kripke

Définition 2.8.2 *Une formule QFCTL$^*(\Sigma, \mathcal{V}, \mathcal{D})$ est* satisfiable sur une structure de Kripke $K = (S, R, V, \mathcal{D})$*, ce que l'on note* $K \models_{\mathcal{D}} \exists Y\ \phi$ *où* $Y = V(\phi) \setminus V$*, s'il existe un état* $s \in S$ *et une valuation* $\rho : Y \to \mathcal{D}$ *telle que* $K, s \models_{\mathcal{D}} \rho(\phi)$*.*

Exemple 2.8.2 *Soit la strucutre de Kripke suivante sur les réels, composée de cinq états où la variable d'état x prend respectivement les valeurs de* 1 *à* 5 *:*



La formule QFCTL $\mathbf{EG}(x \leq V)$ *a une varibale libre* V*. Cette formule est satisfiable dans tous les états. elle est vraie pour toutes les valuations telles que* $V \geq 5$ *dans le dernier état, et pour toutes les valuations* $V \geq 4$ *dans les autres états. Le problème de résolution de contrainte consiste à calculer les domaines de validité des variables libres de la formule dans chaque état.*

Les formules QFCTL* satisfont les propriétés de dualité suivantes ; $\neg\mathbf{E}\phi = \mathbf{A}\neg\phi$, $\neg\mathbf{X}\phi = \mathbf{X}\phi$, $\neg\mathbf{F}\phi = \mathbf{G}\neg\phi$, $\neg(\phi_1\mathbf{U}\phi_2) = (\neg\phi_2\mathbf{W}\neg\phi_1)$. Aussi, les opérateurs $\mathbf{F}$ et $\mathbf{G}$ peuvent etre définis comme des abbréviations : $\mathbf{F}\phi = (true\mathbf{U}\phi)$, $\mathbf{G}\phi = (\phi\mathbf{W}false)$. De manière similaire, $\mathbf{W}$ peut être défini à partir de $\mathbf{G}$ et $\mathbf{U}$ par $\phi_1\mathbf{W}\phi_2 = \mathbf{G}\phi_1 \vee (\phi_1\mathbf{U}\phi_1 \wedge \phi_2)$. En supposant que le langage de contraintes est clos par négation les négations (et les implications) peuvent être éliminées en propageant vers le bas les négations aux contraintes. Sans perte de généralité on peut donc se restreindre à des formules QFCTL sans négation, formées à partir de $\vee$, $\wedge$, $\mathbf{E}$, $\mathbf{A}$, $\mathbf{X}$, $\mathbf{U}$, et $\mathbf{G}$ seulement.

$[c] = \{(s|c) \ : \ s \in S \text{ et } \models_{\mathcal{D}} \exists(s(c))\}$ pour une contrainte c,

$[\mathbf{EX}\phi] = ex([\phi])$	$[\mathbf{AX}\phi] = ax([\phi])$
$[\mathbf{EF}\phi] = \mu Z.[\phi] \cup ex(Z)$	$[\mathbf{AF}\phi] = \mu Z.[\phi] \cup ax(Z)$
$[\mathbf{EG}\phi] = \nu Z.[\phi] \sqcap ex(Z)$	$[\mathbf{AG}\phi] = \nu Z.[\phi] \sqcap ax(Z)$
$[\mathbf{E}(\phi_1 \mathbf{U} \phi_2)] = \mu Z.[\phi_2] \cup ([\phi_1] \sqcap ex(Z))$	$[\mathbf{A}(\phi_1 \mathbf{U} \phi_2)] = \mu Z.[\phi_2] \cup ([\phi_1] \sqcap ax(Z))$
$[\mathbf{E}(\phi_1 \mathbf{W} \phi_2)] = \nu Z.[\phi_1] \cup ([\phi_2] \sqcap ex(Z))$	$[\mathbf{A}(\phi_1 \mathbf{W} \phi_2)] = \nu Z.[\phi_1] \cup ([\phi_2] \sqcap ax(Z))$

TABLE 2.2 – Caractérisation par point fixe des états satisfaisant une formule QFCTL.

2.8.3 Algorithme de résolution de contraintes CTL

Définition 2.8.3 *Le* problème de satisfiabilité *QFCTL($\Sigma, \mathcal{V}, \mathcal{D}$) est défini par :*
Entrée : *une structure de Kripke finie $K = (S, R, V, \mathcal{D})$, une formule QFCTL($\Sigma$,$\mathcal{D}$) ϕ où $Y = V(\phi) \setminus V$,*
Sortie : *l'ensemble des états s tels que $K, s \models_{\mathcal{D}} \exists Y \ \phi$.*

Le problème de satisfaction de contraintes *se distingue du problème de satisfiabilité en demandant en plus de calculer les domaines de validité des variables.*

Sortie : *ensemble de paires (s, ρ) où s est un état et $\rho : Y \to \mathcal{D}$ est une valuation telle que $K, s \models_{\mathcal{D}} \rho(\phi)$, en supposant une représentation finie d'un ensemble infini de valuations, e.g. par un ensemble fini de contraintes.*

2.8.4 Calcul du domaine de validité par point fixe

Une formule propositionnelle CTL ϕ peut être identifiée avec l'ensemble des états qui la satisfont, i.e. $\{s \in \mathcal{S} \mid K, s \models_{\mathcal{D}} \phi\}$. Pour les structures de Kripke finies, les opérateurs CTL usuels peuvent être caractérisés comme les plus petits ou plus grands points fixes de certains opérateurs monotones de $2^{\mathcal{S}} \to 2^{\mathcal{S}}$, appelés transformateurs de prédicat [11, 38].

Cette caractérisation par point fixe s'étend aux solutions de formules QFCTL avec variables libres en associant, à une formule QFCTL ϕ, un ensemble d'états défini par des contraintes pour les variables libres de ϕ décrivant les solutions du problème de satisfiabilité pour ϕ.

Soit $\mathcal{S}_{\mathcal{C}}$ l'ensemble de paires états-contraintes, notées $s|c$, où s est un état et c est une contraintes $\mathcal{D}$-satisfiable. Considérons les treillis $(2^{\mathcal{S}_{\mathcal{C}}}, \emptyset, \mathcal{S}_{\mathcal{C}}, \cup, \sqcap)$ avec les opérations d'union $\cup$ et d'intersection $\sqcap$, i.e intersection d'états avec une conjonction de contraintes satisfiables :

$$A \sqcap B = \{(s|c \wedge c') \ : \ s|c \in A, \ s|c' \in B, \ \models_{\mathcal{D}} \exists(c \wedge c')\}.$$

Soit $ex, ax : 2^{\mathcal{S}_{\mathcal{C}}} \to 2^{\mathcal{S}_{\mathcal{C}}}$ les deux transformateurs de prédicat (associés aux opérateurs CTL **EX**et **AX**) définis par :

$$ex(Z) = \{(s|c) \in \mathcal{S}_{\mathcal{C}} \ : \ \exists (s,t) \in R \ s.t. \ t|c \in Z\},$$

$$ax(Z) = \{(s|c_1 \wedge \ldots \wedge c_n) \in \mathcal{S}_{\mathcal{C}}) \ : \ \begin{array}{l} \{t \ : \ (s,t) \in R\} = \{t_1, \ldots, t_n\}, \\ \forall i, \ 1 \leq i \leq n, \ \ t_i|c_i \in Z, \ \models_{\mathcal{D}} \exists(c_1 \wedge \ldots \wedge c_n)\}. \end{array}$$

Considérons les équations de point fixe entre formules QFCTL et ensembles de paires

états-contraintes données dans la Table 2.2. Ces équations peuvent être utilisées pour calculer les domaines de validité des variables libres d'une formule QFCTL en chaque état, par une itération finie de points fixes.

Exemple 2.8.3 *Pour la formule* $\mathbf{EG}(x \leq V)$ *et la structure de Kripke de l'exemple 2.8.2, l'itération de point fixe donne le résultat suivant ;*

état	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$
$x \leq V$	$1 \leq V$	$2 \leq V$	$3 \leq V$	$4 \leq V$	$5 \leq V$
$\tau^0_{\mathbf{EG}(x\leq V)}$	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*
$\tau^1_{\mathbf{EG}(x\leq V)}$	$1 \leq V$	$2 \leq V$	$3 \leq V$	$4 \leq V$	$5 \leq V$
$\tau^2_{\mathbf{EG}(x\leq V)}$	$2 \leq V$	$3 \leq V$	$4 \leq V$	$4 \leq V$	$5 \leq V$
$\tau^3_{\mathbf{EG}(x\leq V)}$	$3 \leq V$	$4 \leq V$	$4 \leq V$	$4 \leq V$	$5 \leq V$
$\tau^4_{\mathbf{EG}(x\leq V)}$	$4 \leq V$	$4 \leq V$	$4 \leq V$	$4 \leq V$	$5 \leq V$
$\tau^5_{\mathbf{EG}(x\leq V)} = [\mathbf{EG}(x \leq V)]$	$4 \leq V$	$4 \leq V$	$4 \leq V$	$4 \leq V$	$5 \leq V$

Alors que pour la formule $\mathbf{AG}(x \leq V)$ *mettant en jeu le calcul de plus grand point fixe, en partant de tous les états étiquetés par la contrainte vraie, on obtient le point fixe*

état	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$
$[\mathbf{AG}(x \leq V)]$	$5 \leq V$	$5 \leq V$	$5 \leq V$	$5 \leq V$	$5 \leq V$

On dit qu'un opérateur τ sur un ensemble S est *borné* si $\forall s \in S\ \exists i \in N\ \tau^{i+1}(s) = \tau^i(s)$.

En utilisant la preuve du théoreme de Knaster-Tarski-Kleene on voit que :

Proposition 2.8.1 *Un opérateur monotone borné* τ *sur un treillis* $(L, \bot, \top, \sqcup, \sqcap)$ *admets un plus petit point fixe égal à* $\tau^i(\bot)$ *pour un* $i \geq 0$*, et un plus grand point fixe égal à* $\tau^j(\top)$ *pour un* $j \geq 0$*.*

Lemme 2.8.4 *Les transformateurs de prédicats ex et ax, et les transformateurs de prédicats associés aux opérateurs QFCTL sont monotones et bornés dans les structures de Kripke finies.*

Preuve. Pour la monotonicité, il est clair que si $Z_1 \subset Z_2$ alors $ex(Z_1) \subseteq ex(Z_2)$, $ax(Z_1) \subseteq ax(Z_2)$. La même chose est valable pour les transformateurs de prédicats d'opérateurs QFCTL car ils procèdent par intersection ou union d'opérateurs monotones.

Les transformateurs de prédicats ex et ax sont aussi bornés car sinon on pourrait trouver une chaîne infinie d'états telle que $(s_i, s_{i+1}) \in R$ avec $\forall i, j,\ 1 \leq i < j,\ s_i \neq s_j$, une contradiction dans une structure de Kripke finie. La même chose est valable pour les transformateurs de prédicats d'opérateurs QFCTL car ils procèdent par intersection ou union d'opérateurs bornés.

□

Proposition 2.8.2 *Si* $s|c \in [\phi]$ *alors* $K, s \models_{\mathcal{D}} \rho(\phi)$ *pour chaque valuation* ρ *telle que* $\models_{\mathcal{D}} \rho(c)$*.*

Preuve. Par induction structurelle sur ϕ. □

Proposition 2.8.3 (Complétude) *Si* $K, s \models_{\mathcal{D}} \rho(\phi)$ *alors il existe* $s|c \in [\phi]_K$ *tel que* $\models_{\mathcal{D}} \rho(c)$.

Preuve. Par induction structurelle sur ϕ. □

Ainsi on obtient :

Théorème 2.8.1 *Le problème de satisfiabilité de fomrules QFCTL sur une strucutre de Kripke finie pour un domaine* $\mathcal{D}$ *avec un langage de contraintes décidables est décidable.*

Chapitre 3

Applications

3.1 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 [2] 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 `[B]>p2` ?

Stabilité : `G([A]=<p1` $\wedge$ `[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` $\wedge$ `[A]>p2))`.

Oscillation : `F((d([A])/dt>0` $\wedge$ `[A]>v1)` $\wedge$ `(F((d([A])/dt<0` $\wedge$ `[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.

Maximum local : `F(d([A])/dt > 0` $\wedge$ `X(d([A])/dt < 0)` $\wedge$ `[A]=p )`, quelle sont les valeurs des maximum locaux de la trace ?

Influence : `G(d[A]/dt>p1 -> d2[B]/dt2>=0)`, à partir de quels seuils 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.

3.2 Recherche de paramètres à partir de propriétés temporelles

Trouver des modèles mathématiques satisfaisant une spécification obtenue par la formalisation d'expériences biologiques est une étape cruciale de la modélisation. Nous abordons dans ce chapitre le problème de la recherche de paramètres, cinétiques ou conditions initiales, dans les modèles d'interactions biochimiques à partir de spécifications en logique temporelle des comportements du système.

3.2.1 Degré de violation comme fonction de coût

Le degré de violation continu de formules de logique temporelle fournit une mesure de l'éloignement entre une trace numérique et une spécification temporelle. Il est donc naturel d'utiliser cette mesure pour guider la recherche quand on cherche à satisfaire une formule.

Une évaluation uniquement booléenne des formules permet uniquement une recherche par échantillonnage (ou aléatoire) car l'évaluation de solutions fausses n'indique pas ou chercher de bonnes solutions. Le degré de violation continu permet au contraire de savoir, entre plusieurs solutions fausses, quelles sont les moins mauvaises et alors orienter la recherche dans les directions les plus prometteuses. Ceci est réalisé en utilisant le degré de violation comme fonction de coût pour une méthode d'optimisation continue. Toute méthode d'optimisation continue peut a priori être utilisée. Nous présentons dans ce chapitre une méthode de recherche locale générique dans un premier temps puis la stratégie d'évolution par adaptation d'une matrice de covariance CMA-ES (Covariance Matrix Adaptation Evolution Strategy) [40], la méthode que nous retenons pour la recherche de paramètres.

3.2.2 Méthode générique de recherche

Considérons une formule (QF)LTL ϕ, une formule QFLTL ψ telle que $\phi = \psi\sigma$ pour une substitution σ, un modèle de réactions SBML/BIOCHAM avec des conditions initiales et un ensemble de paramètres connus et un ensemble de paramètres inconnus avec pour chacun d'entre eux un intervalle de recherche. On considère le problème visant à trouver les valeurs des paramètres inconnus tel que le degré de violation de la trace correspondante T obtenue par simulation numérique soit $vd(T, \phi, \psi) = 0$.

On peut rechercher les paramètres inconnus par la méthode de recherche de paramètres générique suivante :

Algorithme 3 Méthode de recherche de paramètres locale

1. Choisir comme point courant dans l'espace de paramètres un point aléatoire contenu dans la zone de recherche, calculer par simulation numérique la trace de simulation T correspondante et le degré de violation $vd(T, \phi, \psi)$;
2. si $vd = 0$ aller à 5.
3. pour chaque point défini dans le voisinage du point courant, calculer une trace et son degré de violation ;
4. en fonction des degrés de violation des voisins, choisir le point de l'itération suivante, en faire le point courant, calculer vd et aller à 2.
5. retourner le point courant dans l'espace de paramètres.

Cette procédure peut être interrompue après un nombre donné d'itérations, en retournant le meilleur jeu de paramètres (celui qui minimise le degré de violation). La procédure peut aussi être redémarrée avec un nouveau point initial (étape 1) plusieurs fois afin de diversifier la recherche.

Une méthode naïve serait de déterminer comme voisinage de l'état courant les ensembles de paramètres obtenus en modifiant un paramètre de $\pm\delta$ et de choisir comme paramètre pour l'itération suivante le meilleur voisin.

3.2.3 Recherche de paramètres avec CMAES

Présentation de CMA-ES

Les stratégies d'évolution sont des métaheuristiques d'optimisation basées sur des idées d'adaptation et d'évolution inspirées de la théorie de l'évolution. Par exemple les algorithmes génétiques [41] font évoluer une population de solutions par des opérateurs de sélection, de croisement et de mutation. Dans ce travail on utilise la stratégie d'évolution CMA-ES [40] (Covariance Matrix Adaptation Evolution Strategy) qui ne requiert pas de choisir de paramètres internes à la méthode. CMA-ES considère la fonction à optimiser comme une boîte noire, c'est à dire que seules les évaluations de la fonction sont considérées. Les valeurs de gradient ne sont pas utilisées. La méthode est ainsi utilisable sur des fonctions non dérivables et également sur des fonctions non continues.

Cette méthode utilise une matrice de covariance pour définir de manière probabiliste un voisinage autour d'une solution. A chaque itération une population de solutions candidates est choisie aléatoirement sur ce voisinage probabiliste et l'évaluation de la fonction objectif sur cette population est utilisée pour mettre à jour la matrice de covariance. Cela remplace l'approximation de la matrice Jacobienne et de la matrice Hessienne d'une méthode de quasi-Newton.

En plus d'une fonction objectif à optimiser, CMA-ES effectue une recherche de paramètres à partir d'une solution initiale, de critères d'arrêt et de redémarrage, et d'un espace de recherche. La recherche s'arrête soit lorsque un certain nombre d'évaluations de la fonction objectif ont été calculées, soit lorsque le degré de satisfaction passe sous un seuil donné. Quand les différences entre les évaluations successives du degré de satisfac-

tion sont sous une certaine valeur, c'est à dire lorsque le paysage du degré de satisfaction est trop plat, la recherche redémarre à partir d'un point aléatoire dans l'espace de recherche. L'algorithme CMA-ES est présenté schématiquement algorithme 4.

Une solution candidate est un ensemble de valeurs de paramètres (paramètres cinétiques, conditions initiales ou lois de contrôle). Etant donné une solution candidate, le calcul de son degré de satisfaction se décompose en deux étapes, d'abord le calcul d'une trace numérique par intégration numérique, puis le calcul du degré de violation continu d'une spécification temporelle sur cette trace. La figure 3.1 donne sur un exemple d'exécution de CMA-ES l'évolution de la fonction de coût et l'évolution au cours de la recherche du centre de la distribution des paramètres recherchés.

Algorithme 4 CMA-ES

Entrées: $f : \mathbb{R}^n \to \mathbb{R}$, paramètres $\boldsymbol{\theta}$ de distribution, taille de population λ
Sorties: minimisation de f
  tant que critère d'arrêt non atteint **faire**
    $\{P$: distribution normale multidimensionnelle $\mathcal{N}(\boldsymbol{m}, \sigma^2\boldsymbol{C})\}$
    Echantilloner $P(\boldsymbol{x}|\boldsymbol{\theta}) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
    Evaluer $f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda)$
    $\{\boldsymbol{\theta} = \{\boldsymbol{m}, \boldsymbol{C}, \sigma\}$, $\mu < \lambda$ et $x_{i:\lambda}$ est le i-eme meilleur des λ points $f(\boldsymbol{x}_j)\}$
    Mettre à jour $\boldsymbol{\theta} \leftarrow F(\theta, \boldsymbol{x}_{1:\lambda}, \ldots, \boldsymbol{x}_{\mu:\lambda})$
  fin tant que

Options

Contrairement à un algorithme génétique ou un algorithme de recuit simulé, où l'utilisateur doit choisir des paramètres internes à la méthode, la méthode CMA-ES adapte au cours de la recherche les paramètres internes. Ainsi, pour démarrer une recherche par CMA-ES, l'utilisateur doit fixer la valeur de départ des paramètres recherchés, le coefficient de variation (ratio de l'écart type et de la moyenne) initialement utilisé pour échantillonner la première population (ce coefficient de variation est ensuite adapté automatiquement au cours de la recherche), la taille de la population et le critère d'arrêt de la recherche. La valeur de départ est usuellement fixée au centre d'une zone de faisabilité des paramètres, le coefficient de variation est habituellement fixé à 0,3 (ou à une valeur plus faible si on veut explorer le voisinage proche d'une solution précédemment trouvée par CMA-ES ou une autre méthode).

On choisit comme critère d'arrêt un nombre maximum d'itération et une valeur minimale de la fonction de coût à minimiser. La taille de population, seul paramètre de la méthode non adapté au cours de la recherche, est quant à elle fixée à une valeur proche du nombre de paramètres recherchés. Lorsque la recherche aboutit à un paysage de la fonction de coûts trop plats (les différences entre les évaluations successives de la fonction de coût sont inférieures à un seuil arbitrairement fixé) la recherche redémarre en partant d'un point choisi aléatoirement dans la zone de faisabilité.

Espace de recherche

On appelle zone de faisabilité l'espace de recherche des paramètres inconnus défini en fixant pour chaque paramètre recherché un intervalle fermé de $\mathbb{R}$. Cette zone de faisabilité est indiquée à CMA-ES mais il convient également pour aider la recherche de fournir le plus d'informations connues sur le problème. Ainsi, pour certains paramètres, on peut choisir une échelle de recherche logarithmique (pour une exploration plus rapide de paramètres dont on sait qu'ils peuvent prendre potentiellement des valeurs sur plusieurs ordres de grandeur). Toujours afin d'indiquer le plus d'informations connues à la méthode de recherche, on peut fixer sous la forme d'une formule QFLTL des contraintes linéaires sur les paramètres de recherche.

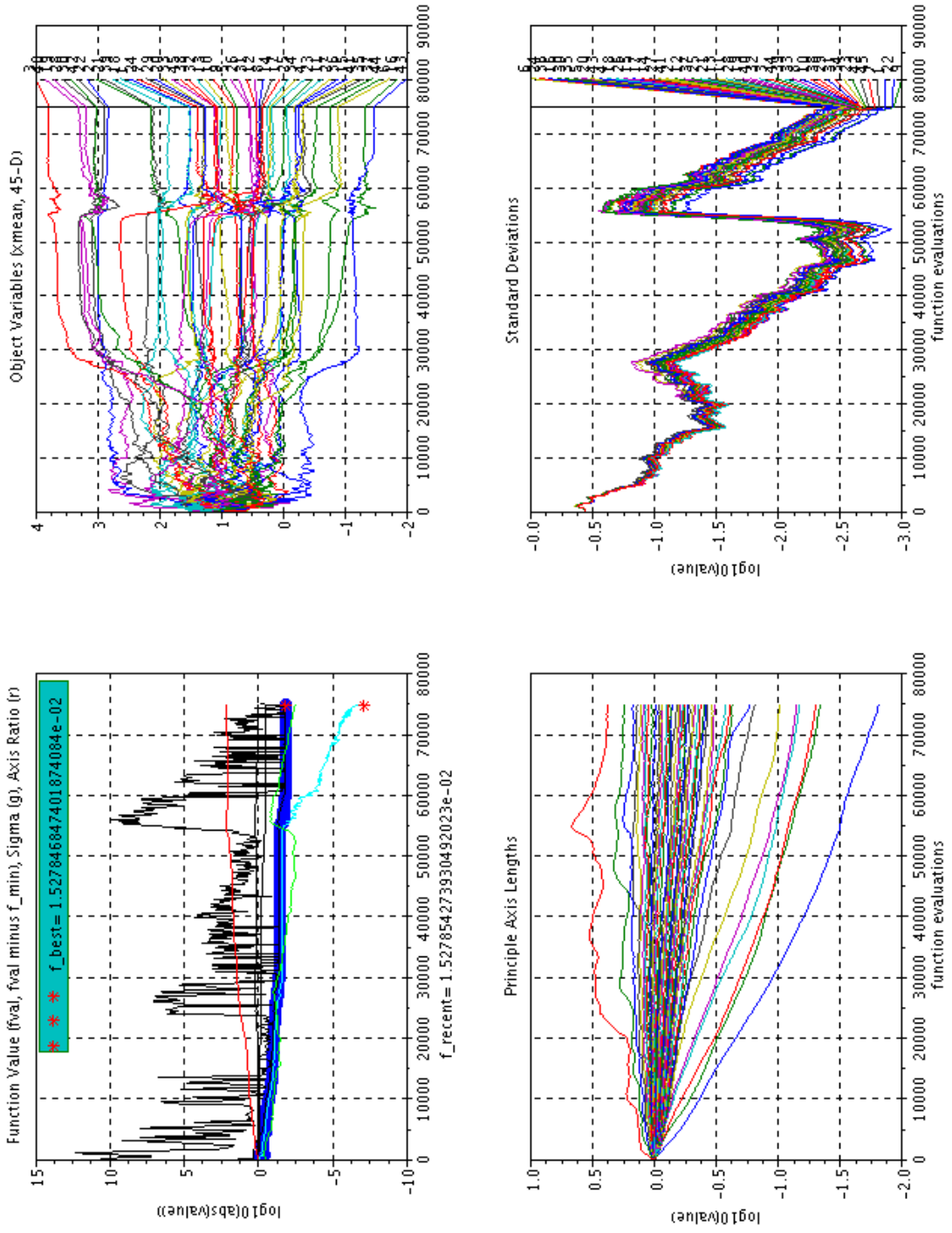


FIGURE 3.1 – Trace d'exécution de CMA-ES. (i) Evolution de la fonction de coût (en bleu sombre épais) (ii) Evolution des valeurs des paramètres (iii) Valeurs propres de la matrice de covariance (longueurs des axes de la distribution) (iv) écarts types des paramètres (taille du pas de recherche sur chaque paramètre)

3.2.4 Recherche de paramètres avec conditions multiples

La recherche de paramètres peut être effectuée vis à vis de spécifications connues dans différents contextes (mutants, conditions extérieures différentes, etc.). On parle alors de recherche de paramètres dans des conditions multiples. Dans ce cas une formule de logique temporelle spécifie le comportement connu dans chaque contexte. Chaque formule est évaluée sur la trace de simulation du modèle dans la condition correspondante. Il y a ainsi plusieurs fonctions à minimiser, c'est une optimisation multi objectif. L'approche retenue est de minimiser par CMA-ES la somme des fonctions de coût dans chaque condition. Cela revient à définir une fonction unique, agrégation des fonctions dans chaque condition. Il peut être utile de pondérer ces fonctions, cela peut se faire dans les formules de logique temporelle (par exemple en remplaçant les occurrences des concentrations $[x]$ de la formule en $[x]\times$ facteur de normalisation) ou en définissant de nouvelles variables normalisées dans le modèle.

3.3 Analyse de robustesse

3.3.1 Contexte

La robustesse peut être définie comme la capacité d'un système à maintenir une fonction en présence de perturbations. Plusieurs études ont montré de manière théorique et expérimentale que la robustesse est une propriété cruciale de nombreux problèmes biologiques et ont proposé des mécanismes qui améliorent la robustesse (*e.g.* [42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]). La robustesse est désormais vue comme une des caractéristiques fondamentales, mais encore mal comprise, des systèmes biologiques car elle permet leur fonctionnement correct malgré la présence de fluctuations environnementales et de non linéarités.

Plusieurs travaux ont évalué le rôle biologique de la robustesse, et considéré ses relations avec la capacité d'évolution des systèmes biologiques, la modularité des réseaux biologiques et le compromis entre la robustesse et la fragilité (*e.g.* [54, 55, 56]). En particulier, dans le domaine de la biologie synthétique, ce sont des problèmes essentiels qu'il faut prendre en compte lors de la conception d'un système.

La notion de robustesse peut se comprendre facilement de manière intuitive. On considère (i) un système particulier, (ii) une fonction particulière et (iii) un ensemble donné de perturbations et on estime de quelle manière les perturbations affectent ou non la fonction donnée. Cependant, avec l'exception notable de Kitano [57], aucune définition formelle de la robustesse n'a été proposée. La définition précise de la robustesse est souvent très dépendante du problème considéré. Il est donc difficile de comparer la robustesse étudiée dans différents contextes, ou même dans des contextes similaires mais calculés avec différentes définitions formelles de la robustesse. Dans [57], Kitano propose les fondements théoriques d'une théorie de la robustesse en biologie avec pour objectif de fournir un cadre général d'étude de la robustesse.

Cependant, même si très intéressante d'un point de vue théorique, la définition de la robustesse de Kitano est trop générale quand on veut l'appliquer à des problèmes particuliers. En effet, la définition repose sur une fonction d'évaluation définie à partir d'une

fonction réelle d'évaluation -non spécifiée- de la performance dépendante du problème considéré.

Nous proposons de définir la fonction d'évaluation à partir de la notion de degré de satisfaction de formule de logique temporelle. Ce degré d'évaluation permet de mesurer l'adéquation entre le comportement d'un système perturbé, donné par une trace numérique, et le comportement requis, exprimé par une formule de logique temporelle.

La richesse d'expressivité de la logique temporelle fait de notre instanciation de la définition de la robustesse de Kitano une méthode générale d'analyse de la robustesse. Nous proposons ainsi une méthode de calcul pour l'estimation automatique de la robustesse qui s'applique à une large variété de propriétés dynamiques et de perturbations, la condition d'application étant que la propriété décrivant le comportement requis peut être exprimé en logique temporelle et que le comportement du système peut être représenté par une trace numérique (obtenue par simulation numérique de modèles déterministes ou stochastiques).

Aussi, une autre contribution de ce travail est de proposer deux notions proches de la robustesse qui sont souvent utilisées indistinctement dans les publications, la *robustesse absolue* d'un système représentant la fonctionnalité moyenne du système soumis à des perturbations et la *robustesse relative* définie par rapport à un comportement de référence qui mesure l'impact des perturbations sur le comportement de référence.

3.3.2 Définition de la robustesse

Robustesse absolue

Nous utilisons la définition générale de la robustesse de Kitano. Dans [57], Kitano définit la robustesse d'une propriété a d'un système s par rapport à un ensemble P de perturbations comme la *moyenne* d'une fonction d'évaluation D_a^s du système sur toutes les perturbations $p \in P$, pondérées par les probabilités des perturbations $prob(p)$:

$$R_{a,P}^s = \int_{p \in P} prob(p)\ D_a^s\ dp \tag{3.1}$$

Cette définition est très générale et peut être utilisée dans de nombreux cas. Cependant, Kitano ne donne pas d'indication sur la manière de définir la *fonction d'évaluation* D_a^s du système. Cette fonction doit mesurer quantitativement si le système maintient encore sa fonction quand il est soumis à des perturbations. Cette fonction d'évaluation doit être définie de manière spécifique à chaque problème et implémentée pour le calcul de la robustesse.

Une contribution centrale de cette section est de montrer qu'utiliser la notion de degré de satisfaction continu de formules permet d'obtenir une méthode de calcul générale basée sur la logique temporelle et la définition de Kitano qui peut être utilisée pour évaluer la robustesse de nombreux types de propriétés dynamiques et de perturbations.

Formellement la robustesse du système est définie de la manière suivante :

$$R_{\phi,P}^s = \int_{p \in P} prob(p)\ sd(T_p, \phi)\ dp, \tag{3.2}$$

où ϕ est la spécification de la fonctionnalité du système en logique temporelle et T_p est la trace représentant le comportement du système soumis à la perturbation p. Cette notion de robustesse correspond à la *fonctionnalité moyenne*, autrement dit décrit le comportement moyen du système soumis aux perturbations. Pour illustrer ces notions considérons les courbes 1 et 2 de la Figure 3.2 qui décrit la performance D^s_a, dans notre cas le degré de satisfaction, de deux systèmes hypothétiques soumis à la perturbation p. Ces ceux courbes ont la même moyenne, la robustesse de ces deux systèmes est donc la même si les perturbations sont uniformément distribuées. Par exemple si la fonction considérée est la quantité produite d'un composé donné exporté par des cellules, ces deux systèmes produisent en moyenne la même quantité de produit.

Robustesse relative

Quand on compare les courbes 1 et 2 de la Figure 3.2, il apparaît que les conséquences des perturbations par rapport au comportement de référence T_0 (correspondant au milieu de l'axe des abscisses) ne sont pas les mêmes. Dans le système 1 les perturbations diminuent de manière plus importante la fonctionnalité que dans le système 2. Avec une autre notion de robustesse on pourrait dire que le système 2 est plus robuste que le système 1 : son comportement est moins affecté par les perturbations. Pour refléter cette seconde interprétation de la robustesse on définit la *robustesse relative* d'un système comme la robustesse absolue divisée par la valeur de fonctionnalité au point de référence - le degré de satisfaction de la formule au comportement de référence.

$$R^{s,p*}_{\phi,P} = R^s_{\phi,P}/sd(T_{p*}, \phi), \tag{3.3}$$

où T_{p^*} est le comportement non perturbé du système (comportement de référence). Dans la Figure 3.2, les performances de référence des systèmes 1 et 2 ne sont pas affectés de la même manière par les perturbations, leurs robustesse relatives sont différentes. La fonctionnalité de référence du système 2 est davantage diminuée que celle du système 1. La fonctionnalité du système 3 est en tout point exactement la moitié de celle du système 1. Ces deux systèmes ont la même robustesse relative tandis que le système 3 a une robustesse absolue moitié moindre que celle du système 1.

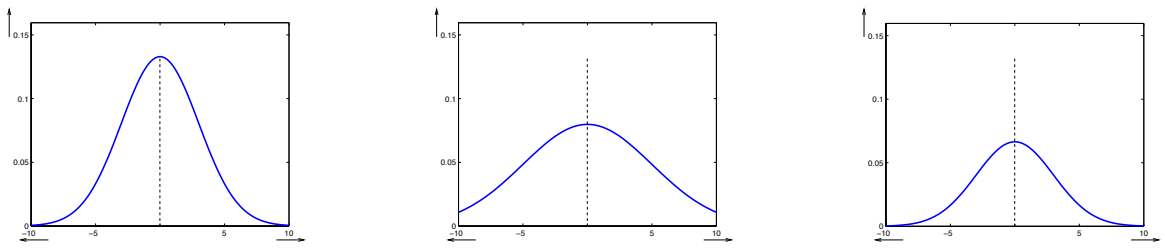


FIGURE 3.2 – Systèmes ayant la même robustesse absolue (1 et 2) ou la même robustesse relative (1 et 3), pour des perturbations distribuées uniformément. Les fonctions d'évaluation des systèmes 1 et 2 ont la même moyenne, celle du système 3 est la moitié de celle du système 1.

Degré de satisfaction robuste

En utilisant la même notion de domaine de satisfaction, on peut aussi définir la *distance de satisfaction robuste* d'une propriété ϕ par rapport à un ensemble de perturbations P comme $dist(\cap_p \mathcal{D}_{T_p}, \phi)$. Cette distance indique le changement minimum dans la formule pour la rendre satisfaite dans toutes les perturbations. On définit le *degré de satisfaction robuste* comme :

$$Rsd^s_{\phi,P} = \frac{1}{1 + dist(\cap_p \mathcal{D}_{T_p}, \phi)} \tag{3.4}$$

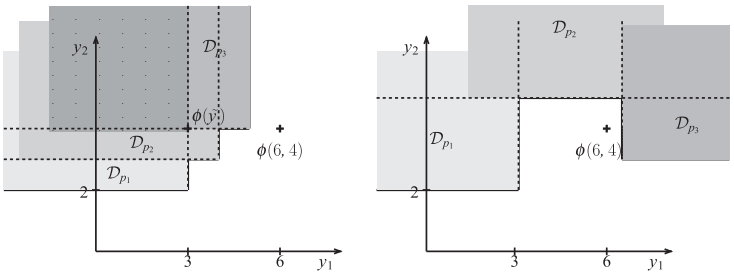


FIGURE 3.3 – Représentation des domaines de satisfaction de trois perturbations p_1, p_2, et $p_3 \in P$. $\mathcal{D}_{p_i}$ représente $\mathcal{D}_{T_{p_i},\phi(\boldsymbol{y})}$. A gauche : l'intersection des domaines (en grisé) n'est pas vide et $Rsd^s_{\phi,P} = 3$. La propriété $\phi(\tilde{\boldsymbol{y}}) = \phi(3,4) = \mathbf{F}(\texttt{[A]} > 3 \wedge \mathbf{F}\ \texttt{[A]} < 4)$ est satisfaite pour toutes les perturbations. A droite : l'intersection des domaines est vide et $Rsd^s_{\phi,P} = \infty$.

Cette notion permet de déterminer si il est possible de relâcher la spécification pour la rendre satisfaite sous toutes les perturbations. Dans le cas de la Figure 3.3, on peut affirmer que le système présente toujours un comportement sous optimal. De plus, la propriété la plus proche $\phi(\tilde{\boldsymbol{y}})$ satisfaite de manière robuste (*i.e.* telle que $\tilde{\boldsymbol{y}} = argmin_{\boldsymbol{y} \in \cap_p \mathcal{D}_{T_p}} dist(\phi(\boldsymbol{y}), \phi)$) fournit un indice intéressant pour la conception du système : comme $\tilde{\boldsymbol{y}} = (3,4)$, cela suggère que seule la première valeur de ϕ (*i.e.* le maximum de `[A]` dans T) a besoin d'être modifiée.

3.3.3 Calcul de la robustesse

Pour le calcul de $R^s_{\phi,P}$, $R^{s,p*}_{\phi,P}$, and $Rsd^s_{\phi,P}$ il est nécessaire de distinguer les cas où l'ensemble de perturbations est fini (*e.g.* gene knockouts) ou infini (*e.g.* variations de paramètres distribuées selon une loi normale). Dans le premier cas les valeurs de robustesse peuvent être calculées de manière exacte, tandis que dans le second cas les valeurs sont estimées en échantillonnant l'ensemble des perturbations.

Etant donné un ensemble d'équation différentielles f, un ensemble de perturbations P des paramètres cinétiques ou des conditions initiales, et une propriété QFLTL ϕ et son instanciation $\phi(\boldsymbol{y})$, l'algorithme 5 calcule la robustesse, la robustesse relative et le degré de satisfaction robuste par rapport aux perturbations données. Le calcul de la trace T_p se fait par intégration numérique.

Algorithme 5 Calcul de la robustesse

Entrées: un modèle d'un système f, les formules (QF)LTL ϕ et $\phi(\boldsymbol{y})$, un ensemble de perturbations P et leurs probabilités, le comportement de référence p^*

Sorties: estimation des robustesses $R_{\phi,P}$, $R^{p*}_{\phi,P}$, and $Rsd_{\phi,P}$

pour chaque perturbation $p \in P \cup \{p^*\}$ **faire**

$T_p :=$ `Compute_trace`(f, p)

$\mathcal{D}_{T_p,\phi(\boldsymbol{y})} :=$ `Compute_sat_domain` $T_p, \phi(\boldsymbol{y})$

fin pour

$R_{\phi,P} := \sum_{p\in P} prob(p)(1 + dist(\mathcal{D}_{T_p,\phi(\boldsymbol{y})}, \phi))^{-1}$

$R^{p*}_{\phi,P} := R_{\phi,P} * (1 + dist(\mathcal{D}_{T_{p^*},\phi(\boldsymbol{y})}, \phi))$

$Rsd_{\phi,P} := (1 + dist(\cap_{p\in P}\mathcal{D}_{T_p,\phi(\boldsymbol{y})}, \phi)))^{-1}$

L'utilisation du degré de satisfaction continu de formules pour l'analyse de robustesse est publiée dans [58].

3.4 Analyse de sensibilité

Cette section présente les méthodes d'analyse de sensibilité que l'on peut utiliser avec le degré de satisfaction continu de formules QFLTL($\mathbb{R}$) pour former des méthodes d'analyse de sensibilité via à vis de spécifications temporelles.

3.4.1 Présentation

L'*analyse de sensibilité* étudie les relations entre les entrées et les sorties d'un modèle. Plus précisément l'analyse de sensibilité examine comment des perturbations sur les entrées du modèle entraînent des perturbations sur ses sorties. Dans notre cas le modèle est un système d'équations différentielles ordinaires (modèle Biocham écrit sous la forme de règles de réactions chimiques), les entrées sont les paramètres cinétiques et conditions initiales du modèle tandis que l'ensemble des variables de sortie est réduit à un unique élément : le degré de satisfaction continu d'une formule QFLTL($\mathbb{R}$).

L'analyse de sensibilité est un outil utile à l'élaboration d'un modèle. D'abord elle peut être utilisée pour valider un modèle en vérifiant que celui ci est bien sensible à des paramètres dont on connait déjà par ailleurs leur influence sur la réponse du processus modélisé. Aussi l'analyse de sensibilité permet de guider les efforts d'estimation des paramètres. Si une petite variation sur un paramètre entraîne une grande variation sur les sorties, ont dit que le modèle est sensible à ce paramètre et celui ci doit donc être évalué précisément. Au contraire si de grandes variations sur un paramètre n'a pas d'influence sur la réponse du modèle il n'est pas nécessaire de l'évaluer précisément.

Ainsi, l'analyse de sensibilité s'utilise préalablement à la recherche de paramètres : on peut simplifier la recherche en en éliminant les paramètres non influents qu'on fixe à une valeur constante. Dans le cas où une évaluation de certains paramètres est possible directement par mesure expérimentale l'analyse de sensibilité permet de déterminer sur quels paramètres il est intéressant d'utiliser les méthodes de mesure les plus coûteuses et les plus précises. Enfin l'analyse de sensibilité, en particulier utilisée avec comme sortie

d'un modèle le degré de satisfaction continu de formule QFLTL($\mathbb{R}$), est un outil précieux pour la compréhension du comportement d'un modèle. Son utilisation avec la logique temporelle permet de relier à un comportement de haut niveau (oscillations, point de passage, etc.) le ou les paramètres (et donc les réactions chimiques qui y sont associés) qui engendrent ce comportement.

Selon [59] l'analyse de sensibilité peut être groupée en trois classes : méthodes locales, méthodes de screening et méthodes globales.

Méthodes locales

Elles se concentrent sur l'analyse autour d'un point particulier de l'espace des paramètres. Par exemple la méthode *local one at a time* est une méthode locale. C'est la méthode d'analyse de sensibilité la plus utilisée. On fixe les paramètres à une valeur donnée, un paramètre à la fois est perturbé (habituellement d'environ 1%) et l'effet sur la sortie du modèle est mesuré.

Méthodes de *screening*

Ces méthodes analysent qualitativement l'importance des variables d'entrée sur la variabilité de la sortie. Elle permettent d'établir une hiérarchie des variables d'entrée en fonction de leur influence sur la variabilité de la sortie du modèle. L'idée principale de ces méthodes est qu'elles constituent un compromis entre l'information obtenue et la complexité de la méthode.

Méthodes globales

Ces méthodes explorent tout le domaine des paramètres. Ce sont les plus informatives et les plus coûteuses en temps de calcul.

Les méthodes locales nécessitent de connaître les valeurs des paramètres et ne fournissent pas d'information sur les interactions entre paramètres. Les méthodes locales et globales, contrairement au *screening* donnent un ordre de grandeur quantitatif des écarts d'influence au sein de la hiérarchie des paramètres. Ici on s'intéresse aux méthodes globales car elles fournissent le plus d'information (interactions entre paramètres) et peuvent être utilisées avant la recherche de paramètres (exploration de tout le domaine de l'espace des paramètres). Une approche classique est d'utiliser d'abord une méthode de screening pour déterminer les paramètres les plus influents et appliquer ensuite, seulement à ces paramètres, une méthode d'analyse de sensibilité globale.

Les sections suivantes présentent la méthode de screening de Morris [60] et la méthode d'analyse de sensibilité globale de Sobol [61].

3.4.2 Méthode de Morris

La méthode de Morris peut être qualifiée de *one at a time*, car elle utilise comme étape élémentaire l'approche locale *one at a time*, et de globale car elle explore tout l'espace des

paramètres. La méthode de Morris, aussi appelée méthode des effets élémentaires, estime l'effet principal d'un paramètre en calculant r mesures locales à des points aléatoires $\vec{x}_1, \cdots, \vec{x}_r$ en en prenant la moyenne.

On discrétise l'espace des paramètres en une grille de p valeurs $\{0, 1/(p-1), 2/(p-1), \cdots, 1\}$ pour chaque paramètre. On suppose ici que le domaine de chaque paramètre est l'intervalle $[0, 1]$, en pratique les valeurs des paramètres sont normalisées pour se ramener à ce cas. Soit Δ un multiple prédéterminé de $1/(p-1)$, Morris définit l'*effet élémentaire* sur le i-eme paramètre au point $\vec{x}$ par :

$$d_i(\vec{x}) = \frac{Y(x_1, \cdots, x_i + \Delta, \cdots, x_k)}{\Delta}$$

où $\vec{x}$ est une valeur du domaine des paramètres choisie telle que $\vec{x} + \Delta$ soit dans le domaine et Y la sortie du modèle considérée. C'est cette sortie que l'on peut remplacer par le degré de satisfaction continu d'une formule QFLTL($\mathbb{R}$).

Après r échantillonnages on obtient une distribution F_i d'effets élémentaires. En pratique, pour réduire le nombre d'évaluations du modèle, les échantillonnages sont faits sous la forme de chemins dans l'espace des paramètres où la valeur de seulement un paramètre est changée à chaque pas. De cette manière chaque évaluation du modèle (sauf le début et la fin du chemin) est utilisée pour le calcul de deux effets élémentaires [60] .

La moyenne μ_i et l'écart type σ_i de cette distribution F_i indiquent l'influence du paramètre i sur la sortie du modèle. Des valeurs basses de μ et de σ correspondent à un paramètre non influent. Une valeur de μ_i élevée indique l'influence globale élevée du paramètre i, une valeur de σ_i élevée indique des effets non linéaires du paramètre i ou des interactions avec les autres paramètres.

3.4.3 Méthode de Sobol

La méthode de Sobol fait partie des méthodes basés sur la variance. Considérons le modèle :

$$Y = f(X_1, \cdots, X_p)$$

Pour évaluer l'effet d'une variable X_i sur la variance de la sortie Y on étudie de combien la variance décroit si on fixe X_i à une valeur x_i^* :

$$V(Y|X_i = x_i^*)$$

Cette quantité est dépendante du choix de x_i^*, pour résoudre cela on en considère l'espérance pour toutes les valeurs possibles de x_i^* :

$$E[V(Y|X_i)]$$

Plus la variable X_i est importante vis à vis de la variance de Y plus cette quantité sera petite. De plus selon le théorème de la variance totale on a :

$$V(Y) = V(E[Y|X_i]) + E[V(Y|X_i)]$$

La quantité $V(E[Y|X_i])$ est donc un indicateur de la sensibilité de Y à X_i. Plus cette quantité est grande, plus la variable X_i est importante vis à vis de la variance de Y. Afin d'obtenir un indicateur normalisé on considère la la grandeur suivante, appelée *indice de sensibilité* :

$$S_i = \frac{V(E[Y|X_i])}{V(Y)}$$

Cette quantité est appelée indice de sensibilité de premier ordre par Sobol. Elle mesure la part de la variance de Y due à la variable X_i. On ne détaillera pas ici les indices de sensibilité d'ordre supérieur. On peut se référer à [62] pour leur description.

L'estimation de ces indices se fait en estimant :

$$V(E[Y|X_i]) = E[E[Y|X_i]^2] - E[E[Y|X_i]]^2 = U_i - E[Y]^2$$

Sobol propose d'estimer la quantité U_i par :

$$U_i = \frac{1}{N}\sum_{k=1}^{N} \quad f(x_{k1}^{(1)}, \cdots, x_{k(i-1)}^{(1)}, x_{ki}^{(1)}, x_{k(i+1)}^{(1)}, \cdots, x_{kp}^{(1)}) \times f(x_{k1}^{(2)}, \cdots, x_{k(i-1)}^{(2)}, x_{ki}^{(1)}, x_{k(i+1)}^{(2)}, \cdots, x_{kp}^{(2)})$$

où les les $x_i^{(1)}$ et $x_i^{(2)}$ sont deux échantillons de réalisations des variables d'entrée.

Les indices de sensibilité sont alors estimés par :

$$\hat{S}_i = \frac{\hat{V}_i}{\hat{V}} = \frac{\hat{U}_i - \hat{f}_0^2}{\hat{V}}$$

avec :

$$\hat{f}_0^2 = \frac{1}{N}\sum_{k=1}^{N} f^2(x_{k1}, \cdots, x_{kp})$$

$$\hat{V} = \frac{1}{N}\sum_{k=1}^{N} f^2(x_{k1}, \cdots, x_{kp}) - \hat{f}_0^2$$

La méthode classique d'estimation de ces grandeurs est la méthode de Monte Carlo (échantillonnage aléatoire). Sobol propose pour évaluer ces grandeurs une méthode dite de Quasi-Monte Carlo qui définit une séquence déterministe d'échantillons qui possède une meilleure répartition uniforme qu'un échantillon aléatoire. Il a été montré que la convergence est plus rapide avec ce type d'échantillonnage. ([63])

Les méthode de Morris et la méthode de Sobol (estimation des indices de sensibilité et échantillonnage de Sobol) ont été implémentées dans le cadre de ce travail à l'environnement de modélisation Biocham en utilisant le degré de satisfaction continu de formule QFLTL($\mathbb{R}$) comme sortie d'un modèle. L'utilisation des indices de sensibilité globale est illustrée section 5.5.

Chapitre 4

Implémentation

Les outils de développement et d'analyse de réseau biochimique présentés dans la section application sont tous implémentés dans l'environnement de développement et modélisation Biocham. Biocham est développé depuis 2003 par l'équipe Contraintes à l'INRIA Paris-Rocquencourt. Il consiste en un langage de description de modèles (sous la forme de règles de réaction chimiques) et un langage de description de propriétés biologiques (sous la forme de formules de logique temporelle). Biocham est disponible en téléchargement sous licence GPL et est écrit en gprolog et en C. Les parties ajoutées dans le cadre de cette thèse représentent 5000 lignes de code en C et 3500 lignes de code en gprolog.

Les sections suivantes décrivent les choix d'implémentation et les bibliothèques utilisées pour une utilisation pratique des applications de la résolution de contraintes temporelles à la biologie des systèmes.

4.1 Représentation du domaine de validité d'une formule QFLTL

Toutes les applications présentées dans la section application reposent en premier lieu sur la résolution de contraintes temporelles de formule QFLTL, c'est-à-dire le calcul du domaine de validité d'une formule sur une trace. L'algorithme de résolution de contraintes temporelles calcule pour chaque instant de la trace et chaque sous formule un domaine de validité et obtient par unions et intersections successives le domaine de la formule complète au premier instant de la trace. Il est ainsi nécessaire d'avoir une représentation des domaines permettant des opérations d'union et d'intersection efficaces.

4.1.1 Représentation d'un polyèdre

Un polyèdre convexe possède deux représentions duales : il peut être vu comme une intersection finie de demi-espaces ou comme combinaison de générateurs (points, droites et demi-droites). La première représentation est nommée représentation implicite tandis que la deuxième est appelée représentation paramétrique. L'opération d'intersection de

polyèdres possède une implémentation naturelle triviale dans le cas de la représentation implicite (concaténation de contraintes) tandis que l'opération d'union a une implémentation naturelle dans le cas de la représentation paramétrique (concaténation des générateurs).

4.1.2 Polyèdres : bibliothèque PPL

Dans le cas général, le domaine de validité est une union fine de polyèdres (voir section 2.3) et l'algorithme effectue des unions et intersections sur ces domaines. On utilise la bibliothèque de gestion de Polyèdres Parma Polyhedra Library [16] et son interface en C pour son utilisation dans Biocham.

Plus précisément, on appelle des fonctions définies par la bibliothèque PPL pour (i) créer les domaines des formules atomiques (ii) calculer les unions et intersections de domaines (iii) représenter le domaine final sous la forme de contraintes pour son affichage dans biocham.

La bibliothèque PPL utilise une double description des polyèdres (implicite et paramétrique) et des algorithmes de changement de représentation entre les représentations implicites et paramétriques permettant des implémentations efficaces des opérations d'unions et d'intersections.

Aussi, au cours de l'algorithme on utilise la fonction de simplification de domaine d'oméga réduction permettant de réduire la taille de la représentation d'un domaine et accélérer les opérations d'unions et d'intersections restant à calculer.

Définition 4.1.1 *Omega réduction*
Soit $\mathcal{D} = \bigcup d_i$ une union fine de polyèdres. On appelle réduction oméga *de $\mathcal{D}$ pour la relation $\subset$ l'ensemble :*

$$\Omega^{\subset}(\mathcal{D}) = \mathcal{D} \setminus \{d_i \in \mathcal{D} | \exists d_j \in \mathcal{D}. d_i \subset d_j \wedge d_i \neq d_j\}$$

4.1.3 Orthotopes : implémentation en C

Lorsque une formule QFLTL($\mathbb{R}$) contient au plus une variable par formule atomique, le domaine de validité est une union d'orthotopes. Pour les orthotopes les changements de représentation entre la représentation implicite et paramétrique sont très simplifiés (chaque contrainte linéaire définissant un demi espace contient exactement une variable et indique immédiatement la coordonnée d'un générateur de l'orthotope). Les orthotopes ont ainsi une représentation particulière qui permet des opérations d'unions et d'intersection plus efficaces que s'ils sont vus uniquement comme des polyèdres.

Pour les orthotopes les opérations d'unions et d'intersection ont été écrites en langage C sans utiliser la bibliothèque PPL. Cette implémentation en C contient l'opération d'oméga réduction ainsi que l'opération de fusion deux à deux pour simplifier la représentation du domaine pendant le déroulement de l'algorithme de résolution de contraintes.

Définition 4.1.2 *Fusion deux à deux*
Lorsque l'union de deux orthotopes peut être représentée par un orthotope on dit qu'on peut les fusionner. La fusion deux à deux *d'une union finie d'orthotopes est l'opération de simplification de la représentation qui consiste à chercher de manière récursive dans le domaine deux orthotopes pouvant être fusionnés. L'opération s'arrête lorsqu'il n'y a plus d'orthotopes pouvant être fusionnés.*

L'opération de fusion deux à deux, présente également dans la bibliotheque PPL, n'est pas utilisée dans le cas d'union de polyèdres car trop coûteuse en pratique par rapport au gain apporté par la simplificaiton du domaine. Les opérations de fusion deux à deux et d'oméga réduction sont illustrées Figure 4.1

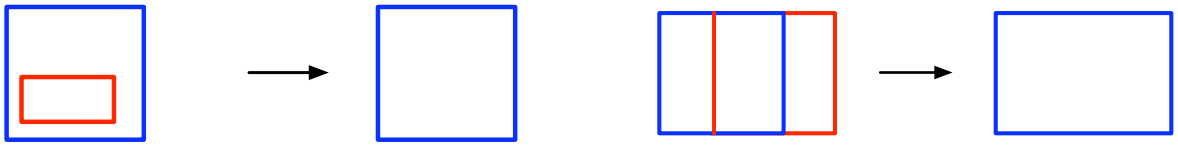


FIGURE 4.1 – Opérations de simplification de la représentation du domaine de validité d'une formule QFLTL : oméga reduction (à gauche) et fusion deux à deux (à droite).

4.1.4 Passage paresseux à une représentation par polyèdre en PPL

Dans le cas général, une formule QFLTL($\mathbb{R}$) peut contenir à la fois des formules atomiques avec une seule variable et des formules atomiques avec plusieurs variables. Autrement dit une formule de QFLTL($\mathbb{R}_{\text{lin}}$) contient des sous formules de QFLTL($\mathbb{R}_{\text{box}}$). L'algorithme de résolution de contraintes calculant pour chaque sous formule son domaine sur chaque instant de la trace, un grand nombre de domaines représentables par union d'orthotopes est calculé même dans ce cas général.

En pratique, dans les exemples abordés chapitre 5, les formules atomiques contenant plusieurs variables sont souvent proches de la racine de la formule (comme dans l'exemple 2.6.2). Dans ce cas, le calcul implémenté en C des opérations sur les orthotopes étant bien plus rapide que le même calcul mais en représentant les orthotopes comme des polyèdres dans la bibliotheque PPL, il est très intéressant d'adopter une approche paresseuse : dans l'implémentation les opérations sur les orthotopes sont effectués en C et la représentation d'un orthotope est convertie en représentation sous forme de polyèdre en PPL uniquement lorsque celui-ci doit être intersecté ou uni avec un polyèdre.

4.2 Recherche de paramètres par CMA-ES

La stratégie d'évolution CMA-ES [40] est disponible en licence GPL dans de nombreux langages de programmation dont le langage C. On utilise son implémentation en C pour son utilisation avec Biocham. L'algorithme de résolution de contraintes calculant le domaine de validité d'une formule QFLTL($\mathbb{R}$) est écrit en gprolog, avec des appels à la bibliothèque PPL en C. Le calcul du degré continu de violation de la formule se fait

également en C avec la bibliothèque PPL et est utilisé par CMA-ES comme fonction de coût à minimiser. CMA-ES fournit en retour des solutions candidates (jeux de valeurs de paramètres) qui sont d'abord utilisées pour calculer une trace de simulation par intégration numérique d'un modèle Biocham. La trace est ensuite utilisée par l'algorithme de résolution de contraintes pour calculer le domaine de validité d'une formule puis son degré de violation.

4.3 Parallélisation de la recherche de paramètres

4.3.1 Introduction

La méthode d'optimisation continue CMA-ES [40] utilisée pour la recherche de paramètres, présentée section 3.2, peut être appliquée à des problèmes allant jusqu'à plusieurs centaines de paramètres inconnus. Notre application à la recherche de paramètres de système de réaction biochimique aborde des problèmes contenants de quelques paramètres à plusieurs centaines de paramètres inconnus.

4.3.2 Parallelisation de la méthode de recherche CMA-ES

La recherche de paramètres est effectuée en évaluant un grand nombre de fois une fonction de coût, et donc en simulant un grand nombre de fois un modèle de réactions chimiques. Selon la complexité du modèle une simulation prendre de quelques secondes à quelques minutes.

De plus, bien que la méthode CMA-ES nécessite un nombre très variable d'évaluations selon les cas, on peut évaluer l'ordre de grandeur du nombre d'itérations requis entre 10 000 et 100 000 pour 100 paramètres inconnus, et estimer le temps de calcul requis pour une recherche de paramètres à 100 000 minutes pour un modèle nécessitant une minute pour sa simulation et le calcul du degré de violation d'une formule QFLTL. Cette durée est multipliée par le nombre de conditions, jusqu'à 100, dans le cas de recherche de paramètres dans des conditions multiples.

Ces temps de calcul nécessitent d'examiner la parallélisation de la recherche de paramètres pour l'utilisation pratique de la méthode sur des modèles complexes, en particulier dans le cadre de modèle à conditions multiples. Chaque itération de la méthode CMA-ES est constituée de l'évaluation indépendante d'une population de solutions candidates. Le calcul se parallélise donc naturellement sur cette population, permettant ainsi l'utilisation simultanée d'un nombre de processeurs égale à la taille de la population de CMA-ES, de 10 à 100 selon les problèmes. Les algorithmes 6 et 7 présentent la parallélisation de CMA-ES en un processus serveur et un nombre quelconque (> 0) de processus clients.

Dans le cas de conditions multiples, l'évaluation d'une solutions candidate est la combinaison des évaluations de la solution dans chaque condition. Ces évaluations dans différentes conditions peuvent se faire de manière indépendante et ajoutent ainsi un niveau de parallélisation possible. A chaque itération, en combinant les parallélisations sur la population et les conditions, le nombre de tâches indépendantes, et donc le nombre

maximum de processeurs sur lequel le calcul peut être parallélisé est nombre de conditions × taille de la population soit 100 * 100 sur les problèmes les complexes [64].

Algorithme 6 Serveur de la parallélisation de CMA-ES

Entrées: $f : \mathbb{R}^n \to \mathbb{R}$, paramètres $\boldsymbol{\theta}$ de distribution, taille de population λ
Sorties: minimisation de f

tant que critère d'arrêt non atteint **faire**
  {P : distribution normale multidimensionnelle $\mathcal{N}(\boldsymbol{m}, \sigma^2\boldsymbol{C})$}
  Echantilloner $P(\boldsymbol{x}|\boldsymbol{\theta}) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
  pour $i = 1$ à λ **faire**
    {lorsque le serveur reçoit un message (évaluation de f, éventuellement non pertinente à la première itération) d'un client k il envoie a ce client une tâche à effectuer :}
    $f(\boldsymbol{x}_j) \leftarrow$ Reçoit(client k)
    {le serveur envoie un travail au client k :}
    Envoie((*Continue*, $\boldsymbol{x}_i$), client k)
  fin pour
  {$\boldsymbol{\theta} = \{\boldsymbol{m}, \boldsymbol{C}, \sigma\}$, $\mu < \lambda$ et $x_{i:\lambda}$ est le i-eme meilleur des λ points $f(\boldsymbol{x}_j)$}
  Mettre à jour $\boldsymbol{\theta} \leftarrow F(\theta, \boldsymbol{x}_{1:\lambda}, \ldots, \boldsymbol{x}_{\mu:\lambda})$
fin tant que
Envoie((*Stop*, _), tous les clients)

Algorithme 7 Client de la parallélisation de CMA-ES

état ← *Continue*
{le client demande un travail au serveur}
Envoie(_, serveur)
tant que état ≠ *Stop* **faire**
  (état, $\boldsymbol{x}_i$) ← Reçoit(serveur)
  Evalue($f(\boldsymbol{x}_i)$)
  {le client envoie son résultat et demande un travail au serveur}
  Envoie($f(\boldsymbol{x}_i)$, serveur)
fin tant que

4.3.3 Communication entre processus

Une tâche élémentaire de calcul effectué sur un processeur est l'évaluation d'une solution dans une condition donnée. Une évaluation (la simulation d'un modèle et le calcul du degré de violation) peut prendre de quelques secondes à quelques minutes. Une tâche est définie par un ensemble de valeurs de paramètres (vecteur de flottants) et une condition (représentée par un indice) tandis que son résultat est une valeur flottante.

Ce type de problème, à grande granularité de parallélisation, (calcul de quelques secondes à quelques minutes) et à communication faible (quelques dizaines de flottants toutes les secondes ou minutes) est adapté à une parallélisation à mémoire non partagée et une communication par message entre les processus, c'est à dire sur des cluster de calcul. Le schéma de parallélisation utilisé consiste en un processus serveur exécutant

la méthode de recherche, créant à chaque itération un ensemble de tâches et les distribuant aux processus clients. La communication entre les processus se fait par passage de messages selon la norme MPI.

4.3.4 Répartition de charge

Lorsque les tâches effectuées sont de longueurs différentes, il faut les répartir au mieux sur les processus client afin de maximiser l'utilisation des processeurs disponibles. Pour cela chaque client, dès qu'il est disponible, demande une tâche à effectuer au processus serveur (Algorithme 7).

4.3.5 Evaluation des performances

La parallélisation des calculs de recherche de paramètres se fait sur le supercalculateur jade du CINES composé de 23 000 processeurs. Chaque processeur exécute exactement un processus et il est possible de choisir d'utiliser tout nombre de processeurs multiples de huit.

La figure 4.2 fournit le temps de calcul de quatre problèmes de recherche de paramètres en fonction du nombre de processeurs utilisés. Les problèmes 1 à 3 sont des problèmes réels. Le problème 4 est un problème artificiel créé pour évaluer la méthode sur des problèmes comportant beaucoup de conditions multiples (comme le problème de recherche de paramètre abordé dans [64]).

Le problème 1 comporte 6 conditions et une population de taille 40 et a donc 240 tâches indépendantes à chaque itération. Le problème 2 a une seule condition et une population de taille 240. Les problèmes 3 et 4 ont respectivement 7 et 97 conditions et des populations de taille 40 et 100. Dans ce cas les tâches indépendantes sont respectivement 280 et 9700.

Ces problèmes sont choisis de sorte que les tâches soient homogènes pour les problèmes 2 et 3 (10 % de variation au plus de temps de calcul) et hétérogènes pour les problèmes 1 et 4 (300 % de variation au plus).

Le problème 1 est le problème de recherche de paramètres présenté section 5.5, les problèmes 2 et 3 correspondent aux recherches de paramètres présentées section 5.6. Le problème 4 est un problème synthétique créer à partir du modèle de la section 5.6 pour évaluer la parallélisation sur un problème comportant un grand nombre de conditions.

Dans le cas de tâches homogènes l'accélération est presque linéaire avec l'augmentation du nombre de processeurs (problèmes 2 et 3). Dans le cas de taches hétérogènes l'accélération diminue lorsque l'on s'approche du nombre de tâches indépendantes disponibles (problème 1). Cela peut s'expliquer par le fait qu'une répartition de charge efficace est alors impossible, des processeurs dont la tâche est courte doivent attendre que les autres aient également terminé.

Cependant pour le problème 4, ou 9700 tâches hétérogènes sont disponibles, on peut utiliser avec très peu de pertes 1024 processeurs. La parallélisation et l'utilisation d'un grand équipement de calcul permet ainsi une accélération efficace sur ce type de

problème.

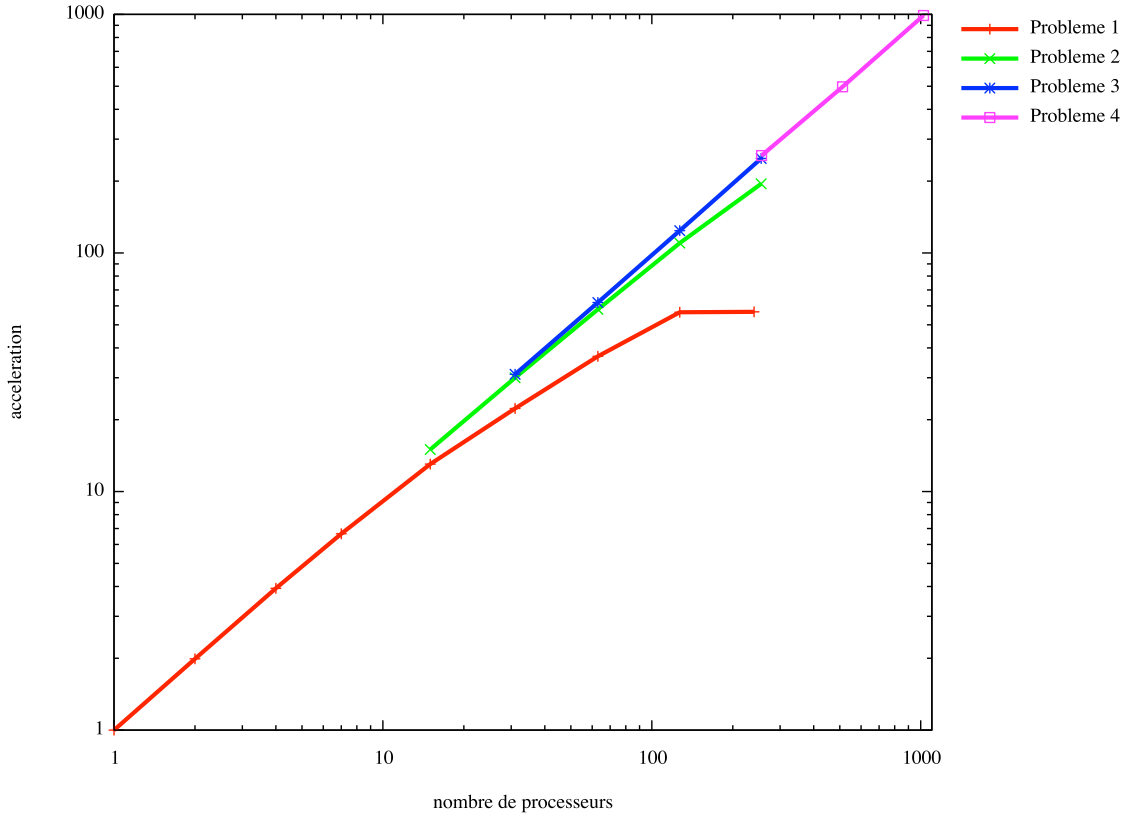


FIGURE 4.2 – Accélération de la parallélisation de la recherche de paramètres sur quatre problèmes de recherche en fonction du nombre de processeurs. Les problèmes 1, 2, 3 et 4 ont respectivement à chaque itération 240, 255, 280 et 9700 tâches indépendantes.
Le problème 1 comporte 6 conditions et une population de taille 40 et a donc 240 tâches indépendantes à chaque itération. Le problème 2 a une seule condition et une population de taille 240. Les problèmes 3 et 4 ont respectivement 7 et 97 conditions et des populations de taille 40 et 100. Dans ce cas les tâches indépendantes sont respectivement 280 et 9700.
Ces problèmes sont choisis de sorte que les tâches soient homogènes pour les problèmes 2 et 3 (10 % de variation au plus de temps de calcul) et hétérogènes pour les problèmes 1 et 4 (300 % de variation au plus).

Chapitre 5

Exemples

Ce chapitre présente l'application à des problèmes biologiques des méthodes d'analyse utilisant la résolution de contraintes temporelles et le degré de violation (ou de satisfaction) continu de formules de logiques temporelles LTL présentées au chapitre 3. La première application est l'utilisation de la résolution de contraintes pour l'analyse de traces et l'inférence de propriétés temporelles section 5.1. Les problèmes biologiques abordés ensuite sont l'analyse des caractéristiques des oscillations d'un modèle du cycle cellulaire de la levure section 5.2, la recherche de paramètres d'un réseau de signalisation vis à vis de données expérimentales dans des conditions multiples section 5.5, l'optimisation d'un traitement anticancéreux section 5.6 (problème de contrôle) et l'optimisation d'un système de biologie synthétique section 5.7 pour rendre un comportement souhaité robuste dans une population de cellules.

5.1 Inférence de propriétés temporelles du cycle cellulaire

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 [65]. 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 :

```
MA(k1)       for                    _ => Cyclin.
MA(k3)       for  Cyclin + Cdc2~{p1} => Cdc2-Cyclin~{p1,p2}
MA(k4p)      for Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}
AUTOCAT(k4)  for Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}
MA(k6)       for Cdc2-Cyclin~{p1}    => Cyclin~{p1} + Cdc2
MA(k7)       for         Cyclin~{p1} => _
MA(k8)       for                Cdc2 => Cdc2~{p1}
MA(k9)       for           Cdc2~{p1} => Cdc2
```

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)

TABLE 5.1 – Résultats d'atteignabilité (valeur maximale atteinte) et de stabilité (valeurs minimales et maximales atteintes)

Les notations `~{p1}` et `{p1,p2}` représentent les formes phosphorylées des molécules. La figure 5.1 donne les traces simulées obtenues pour quatre molécules de ce modèle.

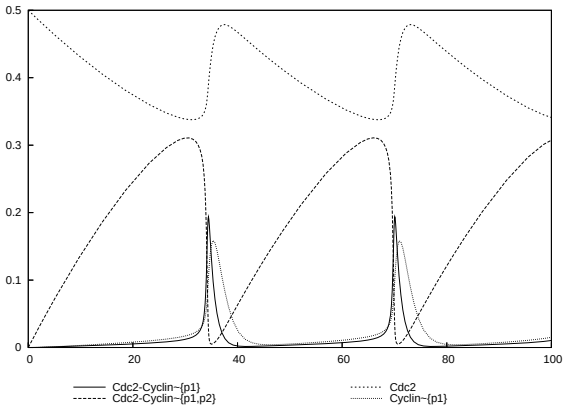


FIGURE 5.1 – Trace de simulation du cycle cellulaire de la levure sur 100 unités de temps. La trace est constituée de 94 points.

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

Atteignabilité

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 retournés 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 5.1 donne les résultats des requêtes d'atteignabilité pour les quatre espèces dont la trace est donnée dans la Figure 5.1.

Molécules	Amplitude d'au moins n oscillations	
	$n = 1$	$n = 2$
`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

TABLE 5.2 – Résultats des requêtes d'amplitude d'au moins n oscillations.

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 5.1.

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 5.2. 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
    )
```

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

TABLE 5.3 – Scores d'influence positives de toutes les 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 $G(d[A]/dt < v2 \Rightarrow d^2[B]/dt^2 < 0)$.

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 $A \rightarrow B$ 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 contraintes 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.

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 5.3 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}]`.

5.2 Recherche de paramètres sur un modèle du cycle cellulaire

Dans cette section nous présentons l'application de la recherche de paramètre au modèle du cycle cellulaire de la levure de Tyson et al. [65]. Ce modèle décrit comment les protéines `cdc2` et `cyclin` interagissent pour former l'hétérodimère `Cdc2-Cyclin~{p1,p2}`, appelé facteur de promotion de la maturation (MPF) et qui joue un rôle crucial dans le contrôle des cycles de la mitose.

Ce modèle est formé des règles de réactions suivantes :

```
MA(k1)       for                    _ => Cyclin.
MA(k3)       for  Cyclin + Cdc2~{p1} => Cdc2-Cyclin~{p1,p2}
MA(k4p)      for Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}
AUTOCAT(k4)  for Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}
MA(k6)       for Cdc2-Cyclin~{p1}    => Cyclin~{p1} + Cdc2
MA(k7)       for         Cyclin~{p1} => _
MA(k8)       for                Cdc2 => Cdc2~{p1}
MA(k9)       for           Cdc2~{p1} => Cdc2
```

`MA(k)` indique une cinétique de loi d'action de masse de paramètre `k`, `~{p1}` et `~{p1,p2}` représentent les formes phosphorylées de molécules. Le taux de la réaction 4 est décrit par :
`AUTOCAT(k4)= k4*[Cdc2-Cyclin~{p1,p2}]*[Cdc2-Cyclin~{p1}]^2` et représente une autocatalye par `Cdc2-Cyclin~{p1}`.

On utilise comme valeur de référence des paramètres les valeurs $\vec{k}_{Tyson}$ définies dans [65]. La simulation du système d'équations différentielles ordinaires extrait des règles de réaction, pour ces valeurs de références $\vec{k}_{Tyson}$, est donnée Figure 5.2. La quantité totale de `cyclin` présente des oscillations de période 35 tandis que la molécule MPF a des pics d'activité de même période.

Nous appliquons la méthode de recherche de paramètres pour déterminer s'il est possible de trouver des valeurs des paramètres cinétiques correspondant a des pics d'activités de MPF plus élevés ou a des oscillations de `cyclin` d'amplitude plus élevée ou de période plus courte.

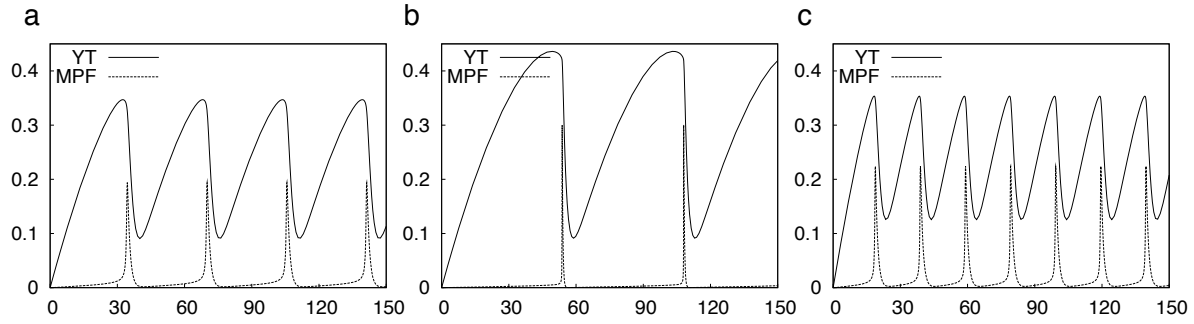


FIGURE 5.2 – Comportement dynamique du modèle du cycle cellulaire. Les courbes représentent les quantités totales de cyclin (YT) et de facteur de promotion de la maturation (MPF). (a) Oscillations obtenues avec les valeurs de paramètres $\vec{k}_{Tyson}$. (b) Pics d'activité élevés de MPF obtenus avec les paramètres $\vec{k}^*_{Tyson}$ (solution du problème S1). (c) Oscillations de période courte obtenues avec $\vec{k}^*_4$ (solution du problème S4).

Problème de recherche S1 : pics d'activité de MPF élevés (2 paramètres inconnus)

Il est indiqué dans [65] que les paramètres k4 et k6 jouent un rôle particulier dans l'existence d'oscillations. En fonction de la valeur de ces deux paramètres le système présente soit un comportement d'état stationnaire soit des oscillations de cycle limite. On cherche à déterminer, par la recherche de paramètres, s'il est possible d'obtenir des pics d'activité de MPF de plus grande amplitude en changeant seulement les deux paramètres k4 et k6, tous les autres paramètres restant fixés a leur valeur $\vec{k}_{Tyson}$ choisie dans [65]. Plus précisément on cherche à obtenir des pics d'activité de MPF atteignant la valeur 0.3 alors que la valeur maximale atteinte par MPF avec $\vec{k}_{Tyson}$ est de 0.19.

On définit la formule LTL suivante correspondant à cette spécification : $\phi_1 = F([MPF] > 0.3)$ et la formule QFLTL correspondante :

$$\psi_1 = F([MPF] > max)$$

L'espace des variables associé à ϕ_1 et ψ_1 est $\mathbb{R}$ et correspond à la seule variable présente dans ψ_1 : max. L'objectif est $obj_{\phi_1} = 0.3$, c'est à dire que le but est d'atteindre des pics d'activité de MPF de valeur 0.3. On appelle $\vec{k}^*_{Tyson}$ les paramètres obtenus en résultat de la recherche de paramètres. Avec ces paramètres l'objectif est entièrement atteint, c'est à dire que $vd(T, \phi_1, \psi_1) = 0$ où T est la trace de simulation obtenue avec les paramètres $\vec{k}^*_{Tyson}$ (voir Figure 1b). Les valeurs de paramètres$\vec{k}^*_{Tyson}$ sont donnés Table 5.5.

On peut remarquer sur cette courbe que des propriétés remarquables sont conservées - pics d'activité de MPF répétés - alors que cela n'est pas imposé par la spécification. Ainsi une quantité constamment croissante de MPF aurait aussi satisfait la formule et correspondrait aussi à un degré de violation nul de la formule.

Tous les calculs sur ce modèle ont été effectués sur un processeur Intel Core 2 Duo à 2Ghz avec 2Go de mémoire vive. Rappelons que la méthode de recherche de paramètres

par CMA-ES utilise un voisinage probabiliste, ainsi deux exécution successives peuvent avoir des résultats et des temps d'exécution différents. Pour cette formule le temps d'exécution est habituellement inférieur à 30s et correspond à environ 150 simulations numériques et calculs du degré de violation.

Problème de recherche S2 : amplitude des oscillations de MPF (2 paramètres inconnus)

Dans cet exemple on affine la requête précédente en imposant également le niveau minimum de MPF. Plus précisément on recherche des valeurs de k4 et de k6 pour lesquelles au moins deux périodes d'oscillations de MPF sont présentes et ont la même amplitude que celles observées avec $\vec{k}_{Tyson}$. De cette manière on peut, en partant de valeurs $\vec{k}_2$ différentes de k4 et k6 essayer de retrouver, avec des valeurs potentiellement différentes, le comportement de $\vec{k}_{Tyson}$

Pour spécifier que les amplitudes sont d'au moins 0.19, on utilise l'espace de variable $\mathbb{R}$ correspondant à une seule variable, *amp* et on fixe comme objectif $obj_{\phi_2} = 0.19$. Cette valeur correspond a l'amplitude obtenue avec $\vec{k}_{Tyson}$.

$$\begin{aligned}
\phi_2 &= F([MPF] > max \wedge F([MPF] < min \\
&\quad \wedge F([MPF] > max \wedge F([MPF] < min)))) \\
&\quad \wedge max - min > 0.19 \\
\psi_2 &= F([MPF] > max \wedge F([MPF] < min \\
&\quad \wedge F([MPF] > max \wedge F([MPF] < min)))) \\
&\quad \wedge max - min > amp
\end{aligned}$$

Les résultats de cette recherche de paramètre sont donnés Table 5.5 et sont nommés $\vec{k}_2^*$. Pour ces paramètres la requête est entièrement satisfaite ($vd(T, \phi_2, \psi_2) = 0$).

Afin d'illustrer le chemin parcouru depuis $\vec{k}_2$ vers $\vec{k}_2^*$ lors de la recherche, on calcule le paysage du degré de violation de la formule ψ_2 dans l'espace de paramètres k4, k6. Le paysage est donné Figure 5.3.

Il est intéressant de remarquer que comme toutes les constantes apparaissant dans la formule ϕ_2 ont été abstraites par des variables dans la formule ψ_2, le degré de violation est nécessairement fini. En particulier, pour une trace ne présentant pas d'oscillations de MPF, *amp* sera égal à zéro et le degré de violation de la propriété sera donc de 0.19. Ainsi on peut facilement repérer dans cet espace de variables les régions correspondant à un régime stationnaire : elles correspondent à un degré de violation de 0.19 tandis que les régions ou le degré de violation est entre 0 et 0.19 sont des régions ou au moins deux oscillations sont présentes.

En approximant le système d'équations différentielles par un système à deux équations Tyson a déterminé les équations déterminant les régions de cet espace pour lesquelles il y a des oscillations. Ces régions sont délimitées par les deux droites dans la

Figure 5.3. Nos résultats sont cohérents avec ces régions, déterminées par une approximation du système de réaction, et fournissent plus d'information sur l'amplitude des oscillations quand on fait varier les paramètres k4 et k6.

Problèmes de recherche S3 et S4 : amplitude et période des oscillations (8 paramètres inconnus)

Pour tester le passage à l'échelle de la méthode, on essaie la méthode de recherche de paramètres sur les 8 paramètres du modèle. Le problème S3 utilise la même spécification (ϕ_2 et ψ_2) que le problème S2 mais porte sur tous les paramètres. Le problème S4 utilise une spécification portant sur la période des oscillations de `Cdc2` afin de déterminer des paramètres pour lesquels la période des oscillations (et donc du cycle cellulaire) est plus courte. Pour spécifier que l'objectif pour la période est de 20, on utilise l'espace de variable $\mathbb{R}$ correspondant à la variable *per* avec comme but $obj_{\phi_3} = 20$.

$$
\begin{aligned}
\phi_3 &= F(d([Cdc2])/dt < 0 \wedge X(d([Cdc2])/dt > 0 \wedge Time > t1 \\
&\quad \wedge X(F(d([Cdc2])/dt > 0 \wedge X(d([Cdc2])/dt < 0 \wedge Time < t2)) \\
&\quad \wedge t2 - t1 < 20 \\
\psi_3 &= F(d([Cdc2])/dt < 0 \wedge X(d([Cdc2])/dt > 0 \wedge Time > t1 \\
&\quad \wedge X(F(d([Cdc2])/dt > 0 \wedge X(d([Cdc2])/dt < 0 \wedge Time < t2)) \\
&\quad \wedge t2 - t1 < per
\end{aligned}
$$

Pour le problème S3 on définit la valeur de départ $\vec{k_3}$ satisfaisant les contraintes sur les ordres de grandeur des paramètres donnés dans [65]. Le problème S4 démarre de $\vec{k_4} = \vec{k_{Tyson}}$. Des valeurs de paramètres satisfaisant les spécifications sont trouvées en environ 350 s pour S3 et 30 s pour S4. Les résultats sont donnés Table 5.5.

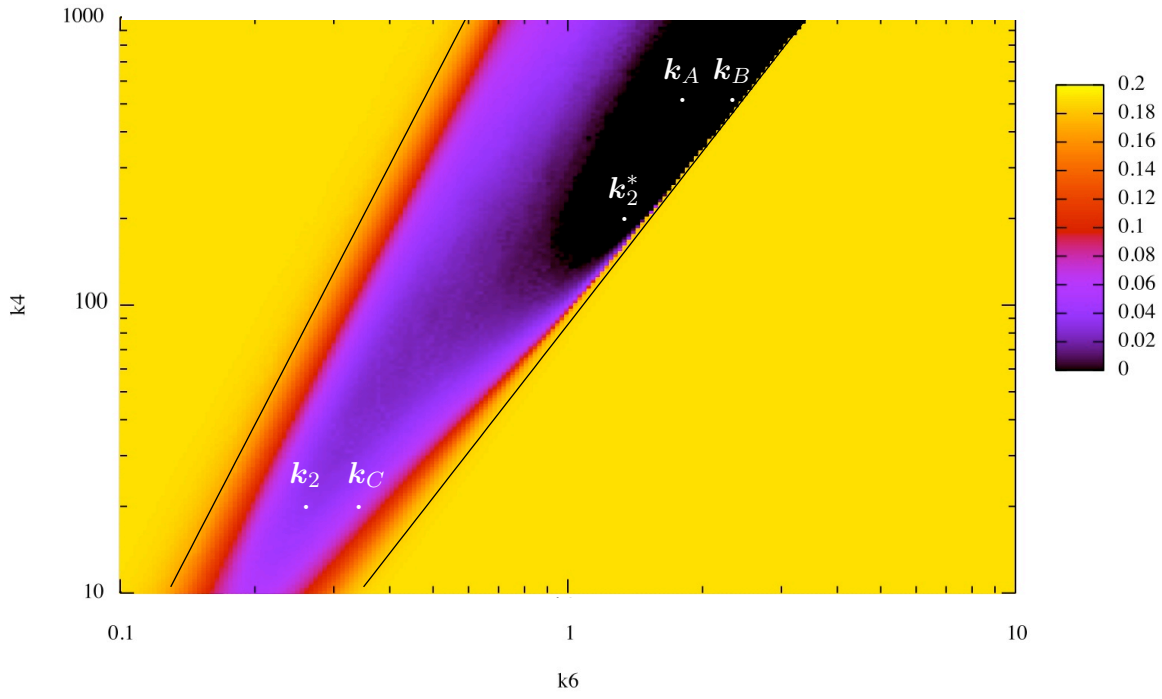


FIGURE 5.3 – Paysage du degré de violation de la spécification du problème S2. Le degré de violation mesure l'amplitude des oscillations. Les régions sans oscillations ont le degré de violation le plus élevé.

	S1		S2		S3		S4	
	Valeurs initiales	Résultats	Valeurs initiales	Résultats	Valeurs initiales	Résultats	Valeurs initiales	Résultats
$vd(T, \phi_i, \psi_i)$	0.11	0	0.04	0	0.19	0	15.1	0
Paramètres	$\vec{k}_{tyson}$	$\vec{k}^*_{tyson}$	$\vec{k}_2$	$\vec{k}^*_2$	$\vec{k}_3$	$\vec{k}^*_3$	$\vec{k}_4$	$\vec{k}^*_4$
k1	1.50e-2	1.50e-2	1.50e-2	1.50e-2	1.00e-2	**1.14e-2**	1.50e-2	**2.41e-2**
k3	2.00e2	2.00e2	2.00e2	2.00e2	1.00e2	**1.13e2**	2.00e2	**2.83e2**
k4p	1.80e-2	1.80e-2	1.80e-2	1.80e-2	1.00e-2	**8.77e-3**	1.80e-2	**2.24e-2**
k4	1.80e2	**8.99e2**	2.00e1	**1.94e2**	1.00e2	**1.82e2**	1.80e2	**2.28e2**
k6	1.00	**3.23**	0.25	**1.41**	1.00	**4.17e-1**	1	**1.13**
k7	0.60	0.60	0.60	0.60	1.00	**1.37**	0.60	**5.99e-1**
k8	1.00e2	1.00e2	1.00e2	1.00e2	1.00e3	**8.99e2**	1.00e2	**1.42e2**
k9	1.00e2	1.00e2	1.00e2	1.00e2	1.00e2	**8.44e1**	1.00e2	**6.94e1**
Temps CPU total (s)	29.4		72.3		347.2		31.4	
Temps de simulation (s)	26.1		17.3		56.3		12.5	
Temps de calcul de vd (s)	2.7		49.7		271.2		16.8	
Nombre d'évaluations de vd	136		128		480		50	
Taille moyenne d'une trace	150		150		150		310	

TABLE 5.4 – Résultats des recherche de paramètres des problèmes S1, S2, S3 et S4. Tous ces problèmes de recherche s'arrêtent lorsque la spécification est totalement satisfaite, c'est à dire quand le degré de violation atteint 0. Les temps de calculs sont les moyennes sur trois exécutions. Le temps de simulation est le temps total passé à produire des traces de simulation par intégration numérique du système d'ODE dérivant des règles de réaction. Le temps de calcul de vd est le temps total passé à calculer le degré de violation de la spécification.

5.3 Recherche de paramètres sur un modèle de transduction du signal MAPK

Le modèle de transduction du signal MAPK [66] est utilisé pour tester davantage le passage à l'échelle de la méthode sur un modèle plus complexe.

Ce modèle représente des réactions de phosphorylation en cascade et est composé des règles de réaction suivantes :

```
(MA(k1), MA(k2)) for RAF + RAFK <=> RAF-RAFK.
(MA(k3),MA(k4)) for RAF~{p1} + RAFPH <=> RAF~{p1}-RAFPH.
(MA(k5),MA(k6)) for MEK~$P + RAF~{p1} <=> MEK~$P-RAF~{p1}
   where p2 not in $P.
(MA(k7),MA(k8)) for MEKPH + MEK~{p1}~$P <=> MEK~{p1}~$P-MEKPH.
(MA(k9),MA(k10)) for MAPK~$P + MEK~{p1,p2} <=> MAPK~$P-MEK~{p1,p2}
   where p2 not in $P.
(MA(k11),MA(k12)) for MAPKPH + MAPK~{p1}~$P <=> MAPK~{p1}~$P-MAPKPH.
MA(k13) for RAF-RAFK => RAFK + RAF~{p1}.
MA(k14) for RAF~{p1}-RAFPH => RAF + RAFPH.
MA(k15) for MEK~{p1}-RAF~{p1} => MEK~{p1,p2} + RAF~{p1}.
MA(k16) for MEK-RAF~{p1} => MEK~{p1} + RAF~{p1}.
MA(k17) for MEK~{p1}-MEKPH => MEK + MEKPH.
MA(k18) for MEK~{p1,p2}-MEKPH => MEK~{p1} + MEKPH.
MA(k19) for MAPK-MEK~{p1,p2} => MAPK~{p1} + MEK~{p1,p2}.
MA(k20) for MAPK~{p1}-MEK~{p1,p2} => MAPK~{p1,p2} + MEK~{p1,p2}.
MA(k21) for MAPK~{p1}-MAPKPH => MAPK + MAPKPH.
MA(k22) for MAPK~{p1,p2}-MAPKPH => MAPK~{p1} + MAPKPH.
```

On appelle k_{MAPK} l'ensemble des valeurs de paramètres cinétiques utilisé comme référence pour ce modèle et indiqués dans [66].

Problème de recherche S5 : curve fitting à des points de temps spécifiques (22 paramètres inconnus)

Dans cet exemple on utilise la méthode de recherche de paramètre comme outil de curve fitting. On recherche les valeurs de 22 paramètres afin que la somme des écarts à des instants spécifiques entre la courbe simulée et des données spécifiés soit la plus faible possible.

Pour exprimer la distance classique entre deux courbes aux points 30 et 60 on utilise la formule :

$$\begin{aligned}\psi_4 = G(\quad & Time = 30 \rightarrow [MEK\text{-}RAF\tilde{}\{p1\}] = u \\ & \wedge\, Time = 60 \rightarrow [MEK\text{-}RAF\tilde{}\{p1\}] = v)\end{aligned}$$

L'espace de paramètre de cette formule est $\mathbb{R}^2$ et est défini par les deux variable u et v. On fixe l'objectif obj_{ϕ_4} aux valeurs objectives de $[MEK\text{-}RAF\tilde{}\{p1\}]$ aux instants 30 et 60, ce qui définit complètement le problème S5.

Ce type de formule indique une spécification à des instants précis. Si la trace numérique simulée ne contient pas ces instants, la formule ψ_4 sera toujours vraie, et le degré de violation sera nul. Afin de pouvoir utiliser la méthode pour du curve fitting, on doit imposer lors de la simulation numérique de calculer les valeurs des concentrations aux instants spécifiques contenus dans la formule. On réalise cela par interpolation linéaire des valeurs de concentrations adjacentes. La formule peut être étendue à un nombre d'instants et de molécules quelconque si nécessaire.

On utilise ce type de formule pour rechercher les valeurs des 22 paramètres de manière à ajuster la concentration de $[MEK\text{-}RAF\tilde{}\{p1\}]$ à 6 instants donnés. Les valeurs objectifs à ces instants sont les valeurs obtenues avec k_{MAPK}. Les valeurs de départ des paramètres pour la recherche sont des valeurs aléatoires des 22 paramètres appelées $k_{MAPK_{alt}}$.

Les simulations numériques correspondant à k_{MAPK} et $k_{MAPK_{alt}}$ sont données Figure 5.4. Le temps de calcul est de 290 s.

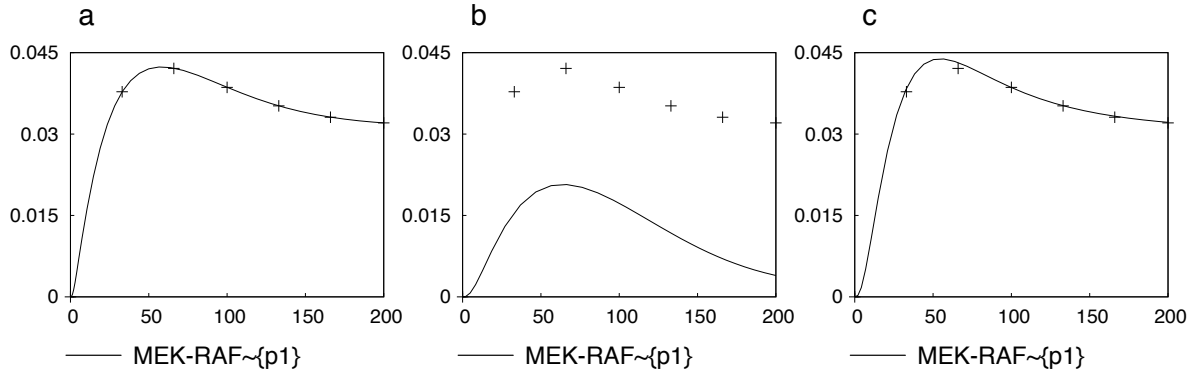


FIGURE 5.4 – Comportement dynamique du modèle MAPK. La courbe représente la concentration de $MEK\text{-}RAF\tilde{}\{p1\}$. (a) Courbe de référence obtenue avec k_{MAPK} (b) Courbe obtenue avec les valeurs de paramètres altérés $k_{MAPK_{alt}}$. Les points sont les valeurs de référence tirées de la courbe (a). (c) Courbe obtenue après curve fitting (solution du problème S5).

Problème de recherche S6 : trouver des oscillations (30 paramètres cinétiques et 7 conditions initiales inconnues)

Dans [67], des oscillations ont été trouvées dans le modèle de cascade MAPK de [66] bien que ce modèle ne contienne pas de réaction de rétroaction négative. Cela ne contredit pas les conditions nécessaires de Thomas sur les oscillations soutenues. Bien que les réactions de ce modèle soient purement unidirectionnelles, son graphe d'influence contient bien une boucle de rétroaction négative comme montré dans [68] et analysé dans [69].

Cependant, pour montrer que les circuits négatifs du graphe d'influence peuvent être

fonctionnels, il faut rechercher les valeurs des paramètres cinétiques et les conditions initiales pour lesquelles le système présente des oscillations soutenues.

En définissant la formule :

$$\phi_5 = F([MAPK\tilde{}\{p1,p2\}] > max \wedge F([MAPK\tilde{}\{p1,p2\}] < min)) \wedge \ max - min > 0.5$$
$$\psi_5 = F([MAPK\tilde{}\{p1,p2\}] > max \wedge F([MAPK\tilde{}\{p1,p2\}] < min)) \wedge \ max - min > amp$$

qui utilise l'espace de variables $\mathbb{R}$ pour la variable $amp < max - min$, et en demandant un objectif d'amplitude de 0.5 en fixant $obj_{\phi_5} = 0.5$, des valeurs de paramètres donnant des oscillations soutenues (même si la formule impose seulement une oscillation), sont trouvées en quelques minutes.(voir Figure 5.5)

Les temps de calcul de cette recherche sont données table 5.5. Comparée à une méthode de recherche (pour le MC) exhaustive de complexité exponentielle par rapport au nombre de paramètre, notre méthode passe bien à l'échelle. La comparaison des temps de calcul entre les problèmes S3 et S5 indique que le facteur déterminant serait plutôt le type de formule, la difficulté du problème (liée aux caractéristiques du paysage du degré de violation) que le nombre de paramètres recherchés.

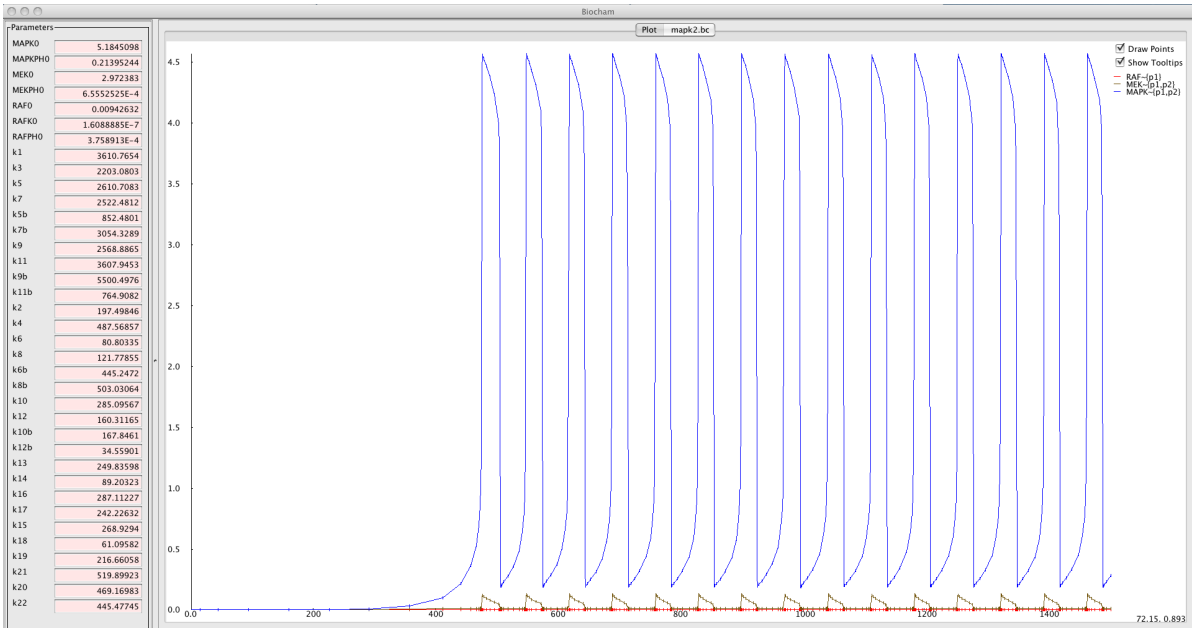


FIGURE 5.5 – Oscillations de MAPK trouvées avec CMA-ES dans Biocham.

5.4 Analyse de la robustesse du modèle du cycle cellulaire

On utilise le même modèle du cycle cellulaire que dans la section 5.2. On compare la robustesse de propriétés d'oscillation par rapport à des perturbations des valeurs des paramètres k4 et k6 à différents points dans l'espace des paramètres.

	S5		S6	
	Valeurs initiales	Résultats	Valeurs initiales	Résultats
$vd(\pi, \phi_i, \psi_i)$	0.06	0.0025	0.5	0
Temps CPU total (s)	290		372.4	
Temps de simulation (s)	263.9		229.6	
Temps de calcul de vd (s)	8.1		124.9	
Number of vd evaluations	906		1092	
Taille moyenne d'une trace	36		300	

TABLE 5.5 – Temps CPU total, temps de simulations et de calcul du degré de violation des problèmes S5 et S6. La recherche du problème S5 est arrêtée lorsque le degré de violation est inférieur au seuil arbitraire 0.005. S6 s'arrête lorsque le degré de violation est nul.

On considère pour les perturbations une distribution normale autour de leur valeur de référence et de coefficient de variation égal à 0.2. On impose que $k4 \geq 0 \wedge k6 \geq 0$. On examine la robustesse des propriétés exprimées par ϕ_2 et ψ_2 dans la section 5.2, c'est à dire que les oscillations de MPF ont une amplitude d'au moins 0.19.

Le degré de robustesse de cette propriété est comparé pour trois différentes valeurs de k4 et k6. Ces trois points $\boldsymbol{k}_A$, $\boldsymbol{k}_B$ et $\boldsymbol{k}_C$ sont donnés Figure 5.3. La robustesse est estimée par 500 tirages de la distribution des perturbations (loi normale).

Les degrés de robustesse des paramètres $\boldsymbol{k}_A$, $\boldsymbol{k}_B$ et $\boldsymbol{k}_C$ sont respectivement 0.991, 0.917 et 0.932. Cela est cohérent avec l'emplacement des points $\boldsymbol{k}_A$, $\boldsymbol{k}_B$ et $\boldsymbol{k}_C$. Les perturbations autour du point $\boldsymbol{k}_A$ ont de fortes probabilités de rester dans la région satisfaisant la spécification tandis que les perturbations autour du point $\boldsymbol{k}_B$ ont de fortes probabilités de déplacer le système dans la région dépourvue d'oscillations. $\boldsymbol{k}_C$ est plus robuste que $\boldsymbol{k}_B$ alors que son degré de violation est non nul, contrairement à celui de $\boldsymbol{k}_B$. Cela s'explique par le fait que la transition est très abrupte entre les régions oscillantes et non oscillantes autour de $\boldsymbol{k}_B$ et plus douce autour de $\boldsymbol{k}_C$.

La robustesse peut être calculée pour des perturbations sur un nombre quelconque de paramètres. Par exemple, en calculant une estimation de la robustesse pour des perturbations sur tous les paramètres avec un coefficient de variation de 0.2, pour la spécification ϕ_2, ψ_2 et pour les paramètres $\boldsymbol{k}_{Tyson}$ et $\boldsymbol{k}_3^*$ on trouve que les robustesses sont de 0.962 et 0.970 respectivement pour $\boldsymbol{k}_{Tyson}$ et $\boldsymbol{k}_3^*$. Cela signifie que les oscillations sont plus robustes aux variations des valeurs de paramètres pour $\boldsymbol{k}_3^*$ que pour les paramètres $\boldsymbol{k}_{Tyson}$ du modèle originel de Tyson.

```
Vm9*GRK23*[R-H]/(Km10+[R-H]) for R-H => R~{P1}-H.
Vm18*GRK56*[R-H]/(Km19+[R-H]) for R-H => R~{P2}-H.
MA(k1) for G =[R~{P1}-H]=> G_a.
MA(k2) for G =[R-H]=> G_a.
MA(k0) for G_a => G.
MA(k26) for G => G_a.
MA(k3) for PIP2 =[G_a]=> DAG.
MA(k4) for DAG => PIP2.
MA(k5) for PKC =[DAG]=> PKC_a.
MA(k6) for PKC_a => PKC.
MA(k7) for ERK =[PKC_a]=> GpERK.
MA(k8) for GpERK => ERK.
(MA(k12),MA(k13)) for R~{P1}-H + barr1 <=> R~{P1}-H-barr1.
MA(k14) for R~{P1}-H-barr1 => R-H + barr1.
(MA(k15),MA(k16)) for R~{P1}-H + barr2 <=> R~{P1}-H-barr2.
MA(k17) for R~{P1}-H-barr2 => R-H + barr2.
(MA(k11),MA(k20)) for barr2 + R-H <=> barr2-R-H.
(MA(k21),MA(k22)) for barr2 + R~{P2}-H <=> barr2-H-R~{P2}.
MA(k23) for ERK =[barr2-R-H]=> bpERK.
MA(k24) for ERK =[barr2-H-R~{P2}]=> bpERK.
MA(k27) for bpERK =>ERK.
MA(k25) for R~{P2}-H => R-H.
```

FIGURE 5.6 – Modèle Biocham du réseau de signalisation induit par l'angiotensine.

5.5 Recherche de paramètres dans des conditions multiples du réseau de signalisation induit par l'angiotensine

La recherche de paramètres est utilisée dans cette partie pour rechercher les paramètres inconnus d'un modèle de réseau de signalisation induit par l'angiotensine développé par Domitille Heitzler dans l'équipe BIOS de l'INRA Tours.

5.5.1 Modèle

Le modèle considéré comporte 26 réactions cinétiques et 18 variables. Des cinétiques de lois d'action de masse sont utilisées dans tout le modèle à l'exception des activités des enzymes GRK qui sont contrôlées par des cinétiques de Michaelis-Menten. Le modèle contient 30 paramètres cinétiques, il est donné sous la forme de règles de réactions Biocham Figure 5.6.

5.5.2 Spécification

On utilise pour trouver les paramètres cinétiques du modèle les données temporelles sur 90 minutes de trois variables lorsque le système est soumis à un signal : une quantité donnée d'angiotensine se fixant au récepteur, c'est à dire une quantité donnée du composant $H-R$. Les trois variables mesurées sont l'activité de PKC, la quantité de DAG et les données de phosphorylation de ERK. Plus précisément les quantités observées sont :

$$pERKtotal\_observée = \frac{[GpERK] + [bpERK]}{erknorm}$$

$$DAG\_observée = \frac{[DAG] - D0\_a}{DAGnorm} + 1$$

$$PKC\_observée = \frac{[PKC\_a] - P0\_a}{PKCnorm} + 1$$

où $D0\_a$ et $P0\_a$ sont les quantités initiales en DAG et PKC. Ces observations sont liées aux concentrations des espèces par les facteurs de normalisations $erknorm$, $DAGnorm$ et $PKCnorm$.

Les données sur ERK sont mesurées dans une condition contrôle ainsi que dans 4 conditions perturbées : inactivation de $barr2$, de $GRK2$, $GRK3$, $GRK5$ et $GRK6$ par des siRNA (*small interferings RNAs*) et inactivation de PKC par un inhibiteur de PKC. Les données obtenues par les inactivations de GRK2 et 3 ou GRK5 et 6 sont moyennées pour correspondre aux paramètres agrégés du modèle. Les données sur DAG et PKC sont disponibles dans la condition contrôle uniquement. On appelle ce cas de figure une recherche de paramètres dans des conditions multiples : des spécifications sont connues dans différentes conditions et on cherche des valeurs de paramètres minimisant la somme des erreurs entre la spécification et le comportement simulé dans chaque condition. Ici on a choisi de pondérer les conditions de sorte que les spécifications sur $pERKtotal\_observée$, $DAG\_observée$ et $PKC\_observée$ aient les mêmes contributions à l'erreur totale.

On considère 5 paramètres de perturbations inconnus qui représentent les perturbations des différentes conditions. Par exemple le paramètre `GRK23` indique l'activité (combinée) des enzymes GRK2/3 dans la condition contrôle et on cherche aussi la valeur de ce paramètre dans la condition où $GRK2/3$ sont inactivées pour déterminer leur degré d'inactivation, éventuellement incomplet.

Aux 30 paramètres cinétiques et 5 paramètres de perturbations s'ajoutent également les 18 conditions initiales des 18 variables du modèle et les 3 paramètres de normalisation de données. Il y a ainsi en tout 56 paramètres inconnus dont 16 sont déterminés par des mesures directes, 40 restent à estimer par la méthode de recherche de paramètres.

Les données temporelles de DAG et PKC sur les 5 premières minutes et de $pERKtotal$ sur 90 minutes sont représentées par une spécification de curve-fitting par une formule QFLTL de la forme :

$$\psi = G(\quad Time = 5 \rightarrow DAG\_observée = v1 \\ \wedge Time = 10 \rightarrow DAG\_observée = v2 \cdots)$$

où $v1$ et $v2$ ont pour valeurs objectif les valeurs des données expérimentales aux instants 5 et 10.

Pour DAG entre 5 et 35 minutes et pour PKC entre 20 et 80 minutes on impose des contraintes de seuil correspondant aux valeurs minimales et maximales définies par l'écart type sur les mesures expérimentales par les formules suivantes :

```
G( (5=<Time & Time=<35) -> (d1=<DAG_observee & DAG_observee=<d2))
```

```
G( (20=<Time & Time=<80) -> (p1=<PKC_observee & PKC_observee=<p2))
```

où $d1$, $d2$, $p1$ et $p2$ ont pour valeurs objectifs les valeurs minimales et maximales requises.

On impose également des contraintes entre les paramètres avec les formules :

```
k21/k11 > cc1 &
k12/k15 > cc2 &
GRK23/GRK23_si > cc3 &
GRK56/GRK56_si > cc4 &
P0/P0_Ro > cc5 &
P0_a/P0_a_Ro > cc6 &
```

où les variables $cc1$, $cc2$, $cc3$, $cc4$, $cc5$, $cc6$ ont pour valeur objectif la valeur 5 ce qui revient à imposer que `k21 > 5* k11` pour `k21` et `k11` par exemple.

Enfin on impose que le niveau basal de $pERKtotal$ soit faible an ajoutant une condition imposant que lorsqu'il n'y a pas de signal (concentration de $R-H$ nulle) $pERKtotal$ est compris entre 1 et 6% de sa valeur maximale mesurée. La formule spécifiant dans cette condition, qu'à l'état stable, $pERKtotal$ soit faible est :

```
F(G(pERKtotal_observee> 3.5-v & pERKtotal_observee< 3.5 + v))
```

où v a pour valeur objectif 1. Sans cette dernière spécification des valeurs de paramètres satisfaisant les spécifications dans les autres conditions pourraient être trouvées mais en partie en raison d'un niveau basal élevé et pas en raison d'une réponse correcte du modèle au signal.

5.5.3 Résultats

Les spécifications données ci-dessus sont utilisées avec la méthode de recherche de paramètres présentée section 3.2 dans le cadre de conditions multiples. Elles sont utilisées pour trouver des valeurs de paramètres satisfaisant les spécifications.

Lorsqu'il n'est pas possible de trouver de telles valeurs de paramètres cela peut être en raison d'erreurs dans la structure du modèle. Dans ce cas on peut essayer la recherche de paramètres avec une autre structure. Des éléments de la structure du modèle sont ainsi en réalité recherchés par le biais de la recherche de paramètres : on choisit la structure pour laquelle la recherche de paramètres aboutit à l'erreur la plus faible. Le modèle présenté ci-dessus est le modèle final qui est le résultat de ce processus.

En particulier, ce processus d'essai de structures a permis de montrer que pour satisfaire la spécification il est nécessaire que les réactions de paramètres k12 à k21 soient réversibles et que les réactions de paramètres k23 et k24 soient des amplifications enzymatiques et non une réaction de stœchiométrie 1 : 1.

De manière intéressante la recherche de paramètre trouve pour les paramètres de perturbation une diminution de plus de 99% indiquant une forte inhibition par les siRNA et l'inhibiteur de PKC dans tous les cas sauf pour GRK23. La valeur trouvée par la recherche de paramètres pour GRK23 dans les conditions où GRK2 ou GRK3 sont inactivées est de 48% de sa valeur normale. Cela suggère que contrairement à GRK5 et GRK6 (les deux doivent être actives pour avoir un effet), GRK2 et GRK3 ont une action additive (chacune a un effet séparément de l'autre).

La figure 5.7 donne le résultat de la simulation dans la condition contrôle pour les paramètres trouvés qui minimisent l'erreur sur la spécification dans toutes les conditions. Les simulations des autres conditions ont des erreurs similaires.

La définition de la spécification par les formules de logique temporelle permet dans cet exemple de combiner facilement une spécification de curve-fitting classique à des contraintes sur les paramètres et des contraintes de seuil sur les concentrations.

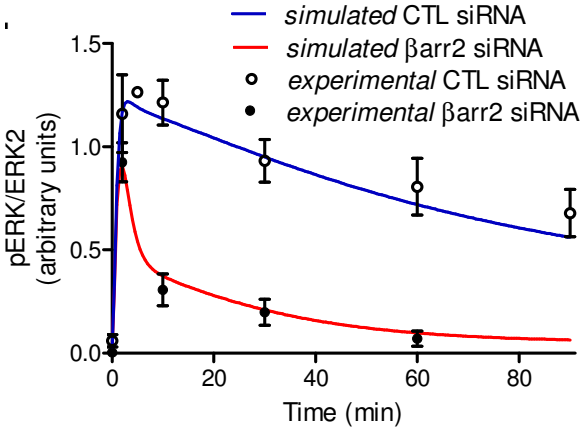


Figure 5.7 – Résultat de la recherche de paramètre dans la condition contrôle du modèle de réseau de signalisation

5.6 Couplage de modèles et recherche de traitement optimal pour la chronothérapie du cancer

Le problème de couplage de modèles et de recherche de traitement optimal est un travail réalisé au sein de l'équipe contraintes. Le travail de modélisation a été réalisé

par Elisabetta de Maria tandis que la méthode de recherche de paramètres présentée chapitre 3 a été utilisée pour trouver des valeurs de paramètres de règles de réaction couplant différents modèles et pour optimiser le planning d'injection d'un traitement anticancéreux.

On présentera ici brièvement le modèle avant de décrire les spécifications utilisées pour la recherche de paramètres et les résultats obtenus. Une description plus détaillée du modèle et de ce travail est publiée dans [72]. On présente ici principalement les formules de logique temporelle utilisées car elles illustrent les possibilités d'application de la méthode de recherche de paramètres.

5.6.1 Présentation

Le nombre de modèle disponibles en biologie des systèmes augmente rapidement. Cependant leur réutilisation dans différents contextes ou pour différentes questions reste un problème ambitieux. Ici on étudie le couplage de différents modèles jouant un rôle dans le cycle cellulaire des mammifères et dans les traitements du cancer. On montre comment on peut formaliser des observations expérimentales en logique temporelle et utiliser la recherche de paramètres pour trouver les valeurs inconnues des paramètres cinétiques de réaction reliant les modèles entre eux. Plus précisément la recherche de paramètres est utilisée pour former un modèle complexe obtenu par le couplage de modèles du cycle cellulaire, de l'horloge circadienne, du système de réparation de l'ADN p53/Mdm2 et du métabolisme de l'agent anticancéreux irinotecan. Le modèle obtenu est ensuite utilisé pour rechercher un traitement optimal par l'irinotecan.

5.6.2 Modèles du cycle cellulaire, de l'horloge circadienne, du système de réparation de l'ADN p53/Mdm2 et du métabolisme de l'irinotecan

Cycle cellulaire

Le cycle cellulaire, pendant lequel le matériel génétique est dupliqué et réparti entre les deux cellules filles, est habituellement divisé en 4 phases [73] : les phases G1 (pause entre la mitose et le début de la synthèse de l'ADN), S (réplication de l'ADN), G2 (pause entre la synthèse de l'ADN et le début de la mitose) et M (séparation des molécules d'ADN entre les deux cellules filles). Ce cycle dure environ 24 heures chez les mammifères et est régulé par différents points de vérification où l'avancement dans le cycle est stoppé pour vérifier l'état de la cellule.

On utilise le modèle du cycle cellulaire chez les mammifères proposé par Novák et Tyson dans [74].

Horloge circadienne

Dans de nombreux organismes, l'activité de certains gènes et protéines présente spontanément des oscillation soutenues de période proche de 24 heures. Une horloge

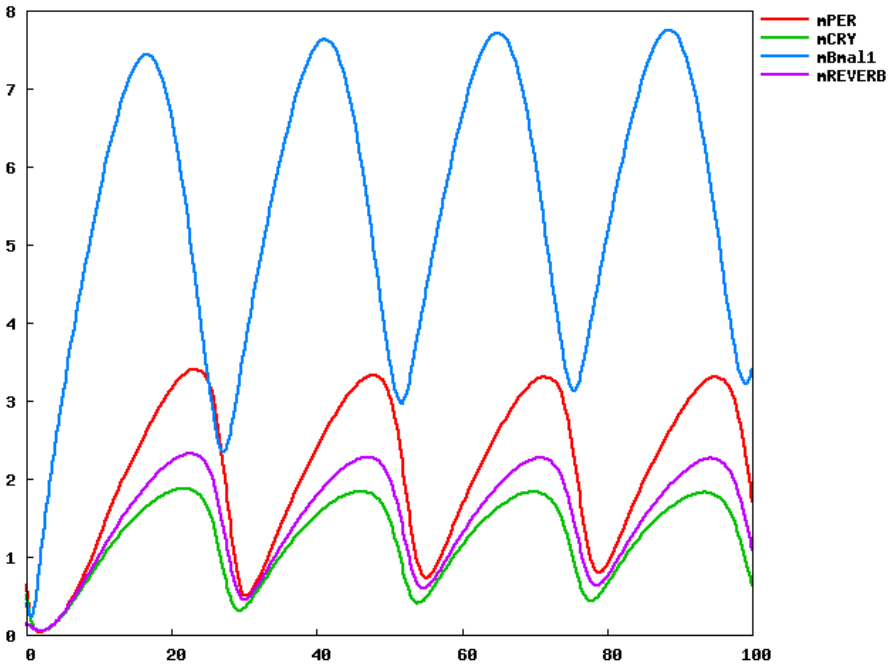


FIGURE 5.8 – Simulation du cycle circadien.

biochimique présente dans chaque cellule est responsable de maintenir ces oscillations. On considère ici le modèle composé de 19 équations différentielles proposé par Leloup et Goldbeter dans [75]. Sa simulation est donnée figure 5.8.

Système de réparation de l'ADN P53/Mdm2

Le troisième modèle décrit comment la protéine p53, dite protéine de suppression de tumeur car elle a la capacité d'arrêter le cycle cellulaire et mener à la mort de la cellule (apoptose), est activée en réponse a des dégâts sur l'ADN. Plus précisément on utilise le modèle de Ciliberto et al. [76] qui décrit l'interaction de p53 avec une autre protéine, Mdm2. Sa simulation est donnée figure 5.9.

Métabolisme de l'irinotecan

L'irinotecan est un médicament anticancéreux. Plus précisément l'irinotecan est transformé en sa forme active, SN38, par une carboxylesterase, une enzyme principalement située dans le foie, les intestins et les tissus tumoraux. SN38 est un inhibiteur de la topoisomerase-1, une enzyme essentielle à la synthèse de l'ADN. SN38 a alors pour effet d'endommager les cellules, saines ou cancéreuses qui sont en phase de réplication de l'ADN. Ce travail utilise le modèle d'action de l'irinotecan élaboré par Dimitrio [77] et actuellement développé par Ballesta [78]. Sa simulation est donnée figure 5.10.

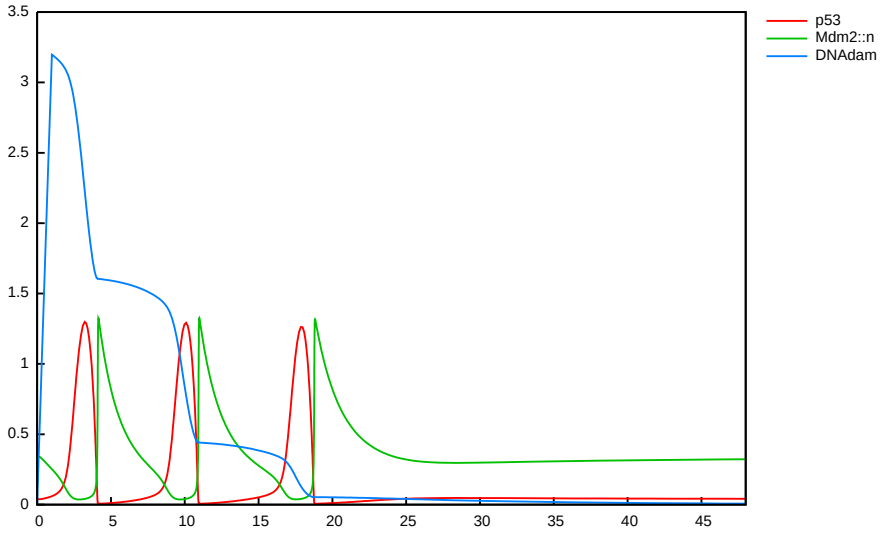


FIGURE 5.9 – Simulation du système de réparation de l'ADN P53/Mdm2.

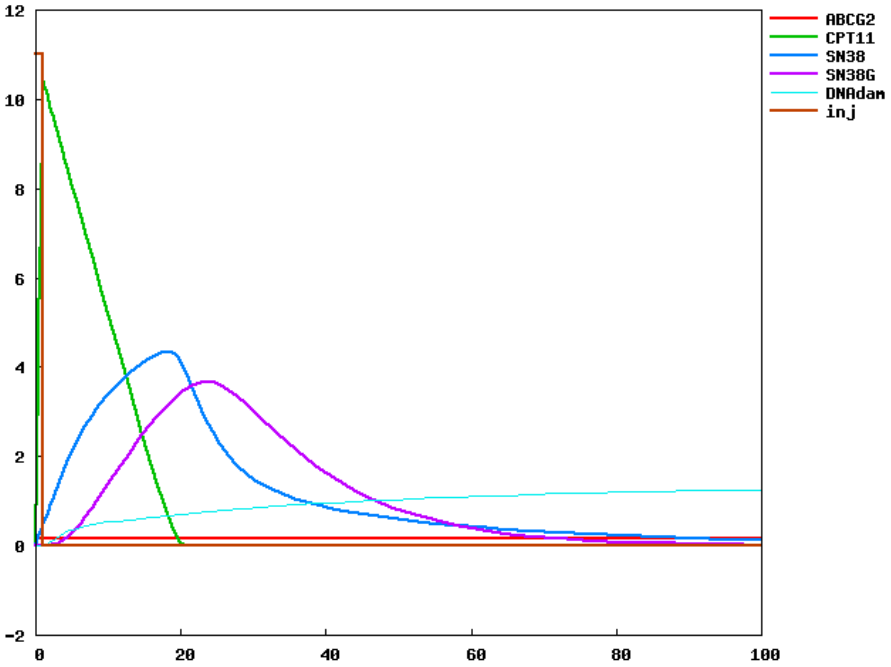


FIGURE 5.10 – Simulation du métabolisme de l'irinotecan.

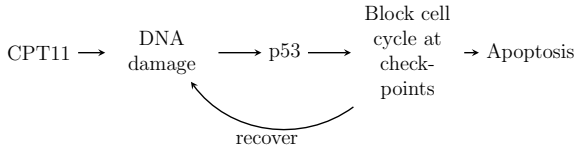


Figure 5.11 – Comportement schématique du modèle couplé.

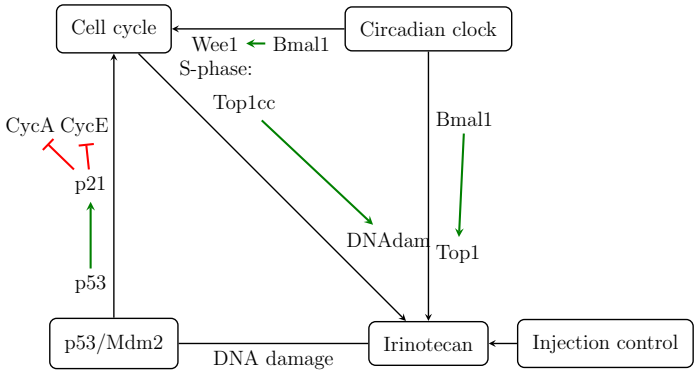


Figure 5.12 – Schéma global du modèle couplé.

5.6.3 Couplage des modèles

Comportement du modèle global

La littérature fournit des informations sur les liens connus entre les modèles présentés ci-dessus. Tout d'abord décrivons le comportement attendu de la cellule présenté dans la Figure 5.11. Les injections d'irinotecan (CPT11) endommagent l'ADN, la cellule répond en activant la protéine p53 qui bloque le cycle cellulaire et permet la réparation de l'ADN. Si la réparation est possible le cycle cellulaire est redémarré, dans le cas contraire l'apoptose est déclenchée (suicide de la cellule).

Spécification LTL($\mathbb{R}$) du couplage

Dans cette section, on montre comment on utilise les contraintes de logique temporelle et la méthode de recherche de paramètres pour obtenir les paramètres cinétiques du modèle couplé.

La simulation numérique du modèle est effectuée sur un horizon de temps de 100 heures. On considère les valeurs des variables du modèle ainsi que de leurs dérivées.

On considère directement le modèle composé des ses cinq parties et de toutes les règles de liaison, comme illustré Figure 5.12 pour effectuer la recherche de paramètres. Ces règles de liaison sont utilisées ensemble dans Biocham par la méthode de recherche pour les conditions multiples. On présente ici les spécifications des différents couplages séparément. Ces spécifications sont utilisées ensemble pour trouver des paramètres les satisfaisant toutes.

Le lien entre le cycle cellulaire et le modèle de l'irinotecan est encodé par la règle de

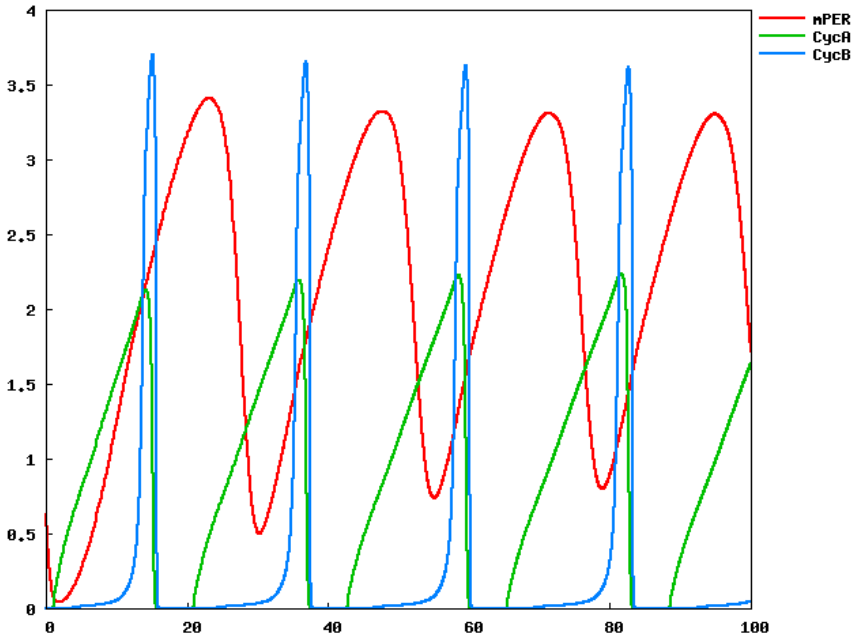


FIGURE 5.13 – Simulation du cycle cellulaire (CycA, CycB) et de l'horloge circadienne (mPER) sans couplage. Le cycle cellulaire a une période libre de 23 heures.

réaction : `MA(top1bis) for _=[CycA]=>TOP1.` On utilise la recherche de paramètres pour trouver la valeur minimale du paramètre `top1bis` telle que la propriété F1, qui impose une corrélation entre les concentrations de Top1 et CycA soit satisfaite :

F1 : Lorsque CycA dépasse la valeur 1, il existe un instant dans le futur ou Top1 est plus grand que 1.
LTL : $\mathbf{G}(([CycA] > 1) \rightarrow \mathbf{F}([TOP1] > 1))$.
Résultat : La valeur minimale de `top1bis` satisfaisant F1 est 0.45.

La règle Biocham suivante représente le lien entre l'horloge circadienne et le modèle du cycle cellulaire :
`(ksweemp+ksweem*[Bmal1_nucl])/(kweem+kwpcn*[PER_nucl-CRY_nucl])`
`for _=[Bmal1_nucl]=>Wee1.`
Alors que le cycle cellulaire présente des oscillations de période de 23 heures, le cycle circadien présente une période proche de 24 heures. On dit que le modèle du cycle cellulaire est correctement entrainé par l'horloge circadienne lorsque sa période est aussi d'environ 24 heures (voir Figures 5.13 et 5.14).
F2 : la période de CycA et de CycB est 24h.
LTL($\mathbb{R}$) : $period(CycA, 24) \wedge period(CycB, 24)$.
Résultat : les valeurs trouvées sont `ksweemp`=0.521, `ksweem`=0.5, `kweem`=1, et `kwpcn`=2.

Les règles de réaction introduites pour lier les modèles p53/Mdm2 et du cycle cellulaire sont les suivantes :

```
MA(k5321) for _=[p53]=>p21.
MA(kA21) for CycA=[p21]=>_.
MA(kA21) for CycE=[p21]=>_.
```

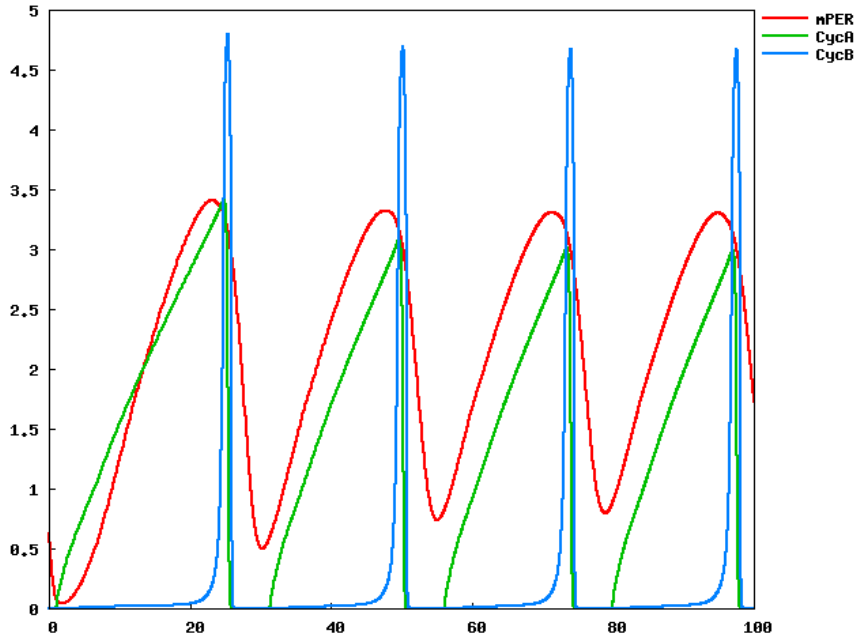


Figure 5.14 – Simulation du cycle cellulaire entrainé par l'horloge circadienne par le couplage sur Wee1. La période du cycle cellulaire est de 24 heures.

Des valeurs de paramètres pour `k5321` et `kA21` (`kA53` dans le second cas) ont été cherchées de sorte que la propriété F3, qui représente les oscillations de CycA du cycle cellulaire entraîné par le cycle circadien, soit conservée lorsque le module p53/Mdm2 est ajouté mais qu'il n'y a pas d'exposition à l'irinotecan.

F3 : Avant 100 heures, CycA est plus grand que 2.7 dans au moins 4 oscillations.
LTL($\mathbb{R}$) : $oscil([CycA], 4, 2.7)$.
Résultat : la propriété est satisfaite pour les valeurs de paramètres `k5321=0.487`, `kA21=0.00507`, et `kA53=0.283`.

Enfin, pour relier le cycle cellulaire et le modèle pour l'irinotecan on considère la règle : `MA(kdam) for TOP1cc=>DNAdam.`
On fait dépendre le paramètre `kdam` de la phase du cycle cellulaire : il prend la valeur $v1$ pendant la réplication et la valeur $v2$ en dehors avec $v2 > v1$. On recherche des valeurs de $v1$ et $v2$ telles que la propriété F4 soit satisfaite.

F4 : Lorsque Top1cc dépasse 0.2, il existe un état futur pour lequel la dérivée première de DNAdam dépasse 0.15.
LTL($\mathbb{R}$) : $\mathbf{G}([TOP1cc] > 0.2 -> \mathbf{F}(d([DNAdam])/dt > 0.15))$.
Résultat : la propriété est vérifiée pour $v1$ égal à 1.42 et $v2$ égal à 1.89.

Les valeurs des paramètres de couplage trouvées par la méthode de recherche de paramètres, sont récapitulées dans la Table 5.6, ainsi que les spécifications utilisées pour les trouver.

La figure 5.13 montre les simulations du cycle cellulaire et de l'horloge circadienne sans couplage. La figure 5.14 montre la simulation avec couplage.

Parameter	Value	Formula
kbmaltop	0.207	F1
top1	0.212	F1
ksweemp	0.521	F2
ksweem	0.5	F2
kweem	1.12	F2
kwpcn	5	F2
k5321	0.486	F3
kA21	0.00507	F3
kA53	0.283	F3
v1	1.42	F4
v2	1.89	F4

TABLE 5.6 – Valeurs de paramètres trouvées par la méthode de recherche de paramètre.

5.6.4 Recherche d'exposition optimale au médicament

Les propriétés LTL de cette section portent sur les lois de contrôle de l'irinotecan. Afin d'optimiser l'efficacité du traitement anticancéreux par l'irinotecan sur les cellules tumorales tout en minimisant la toxicité sur les cellules saines, on cherche le planning d'exposition des cellules à l'irinotecan qui permet la dose la plus importante d'irinotecan tout en maintenant la toxicité sous un seuil donné dans les cellules saines.

5.6.5 Evaluation d'exposition pulsatile

Dans la Figure 5.15 on montre le comportement du module de réparation de l'ADN p53/Mdm2 lorsque l'exposition à l'irinotecan est répétée toutes les 24 heures. Cette figure met en évidence comment les dégâts à l'ADN augmentent après chaque période d'exposition.

5.6.6 Optimization de la loi d'exposition à l'anticancéreux

Afin de trouver la loi d'exposition la plus efficace, on cherche le planning et la quantité maximum d'irinotecan tels que les dégâts causés à l'ADN restent sous un seuil donné.

F6 : DNAdam est toujours inférieur à 1 et la dose totale d'irinotecan est supérieure à 50.
LTL($\mathbb{R}$) :G$([DNAdam] < 1) \wedge totalinjection > 50$.
On cherche les paramètres qui vont satisfaire F6 avec la plus petite erreur possible, c'est à dire les valeurs qui maintiennent les dégâts à l'ADN inférieurs à 1 et pour lesquels la dose totale d'irinotecan est la plus proche possible de 50.

Résultat : le motif d'exposition occasionnant le moins de dégâts a l'ADN est formé de boites rectangulaires de largeur 1h et de hauteur 7. La première injection doit avoir lieu 23h30 après l'état initial de la simulation (correspondant à la phase G1 du cycle cellulaire). Les injections suivantes doivent avoir lieu à chaque oscillation du cycle cellulaire.

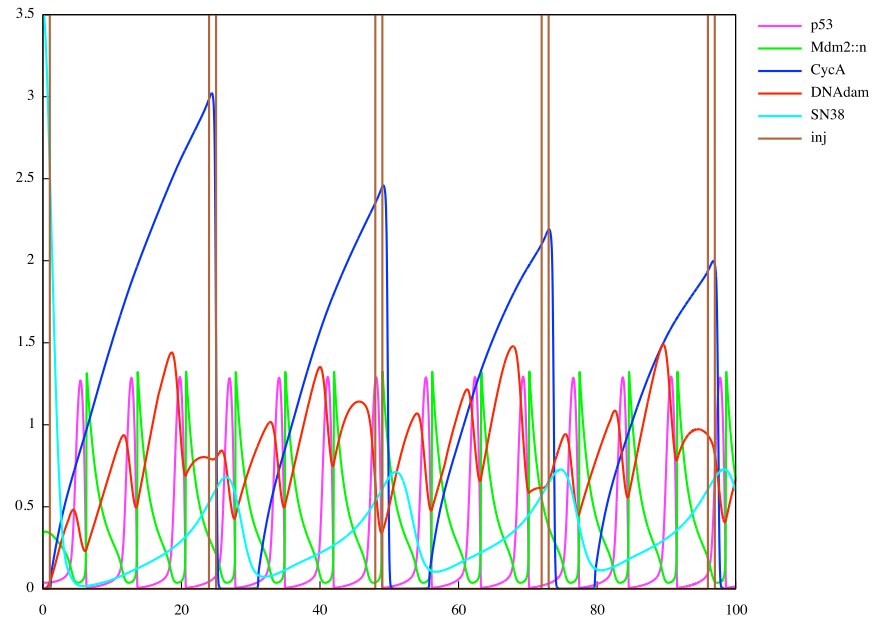


Figure 5.15 – Simulation des dégâts à l'ADN (en rouge) sous une exposition pulsatile d'irinotecan (en marron) de période 24 heures avec le module de réparation p53/Mdm2.

(voir Figure 5.16, où KDAM délimite la phase de synthèse du cycle cellulaire)

Le dégât à l'ADN n'est pas causé directement par l'irinotecan mais par un complexe (nommé Top1cc) qui se forme quelques heures après l'exposition à l'irinotecan. Ainsi on peut remarquer que le planning d'exposition décrit ci dessus correspond à une concentration maximale du complexe Top1cc en dehors de la phase S.

Au contraire, pendant la phase de synthèse de l'ADN, la toxicité de Top1cc est la plus élevée. La Figure 5.17 montre les effets de la même exposition que pour la Figure 5.16 mais avec un décalage de phase de 12h. Avec ce décalage, qui peut être atteint pour des cellules tumorales non synchronisées, le dégât à l'ADN est 70% plus élevé que pour des cellules synchronisées.

La dernière spécification porte sur la capacité de réparation de l'ADN par la cellule.
F7 : Apres une exposition à l'irinotecan, les dommages à l'ADN peuvent redescendre sous le seuil 0.1 avant l'exposition suivante .
LTL($\mathbb{R}$) : $\mathbf{G}(([CPT11] > d) \vee (([CPT11] \leq d)\mathbf{U}([DNAdam] < 0.1)))$, où d dépend de la dose d'irinotecan.

On cherche l'intervalle de temps entre des expositions successives d'irinotecan de dose 10 rendant la formule vraie.

Résultat : l'intervalle de temps minimum multiple de 12 pour lequel F7 est satisfait est 36. Ainsi au plus une exposition toutes les 36 heures doit être effectuée pour permettre aux dégâts à l'ADN d'être réparés avant l'exposition suivante.

Pour des doses d'expositions de 20 il faut au plus une injection toutes les 48 heures.

La définition de la spécification par les formules temporelles permet dans cet exemple

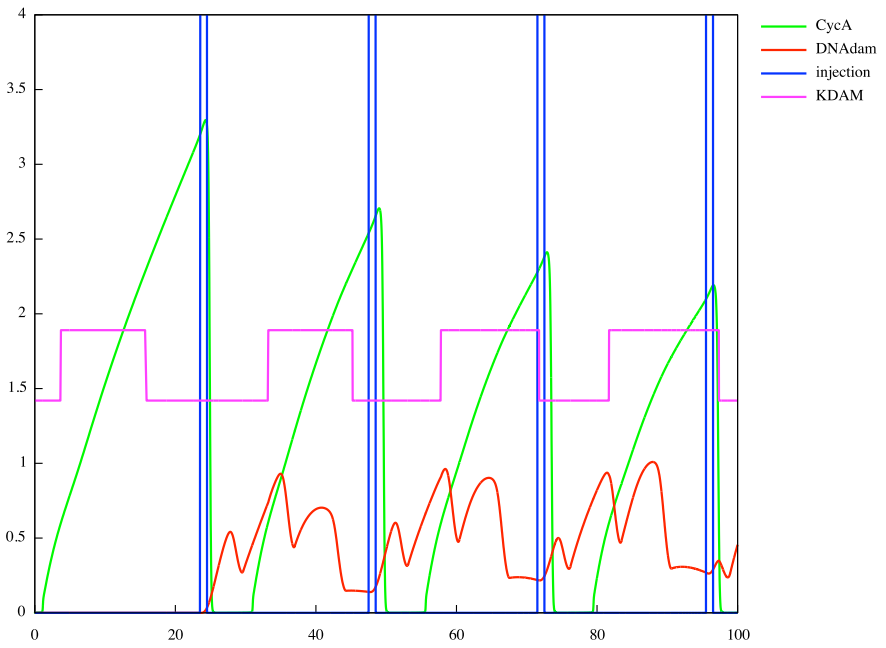


FIGURE 5.16 – Exposition maximum d'anticancéreux (en bleu) maintenant les dégâts à l'ADN (en rouge) sous le seuil 1.

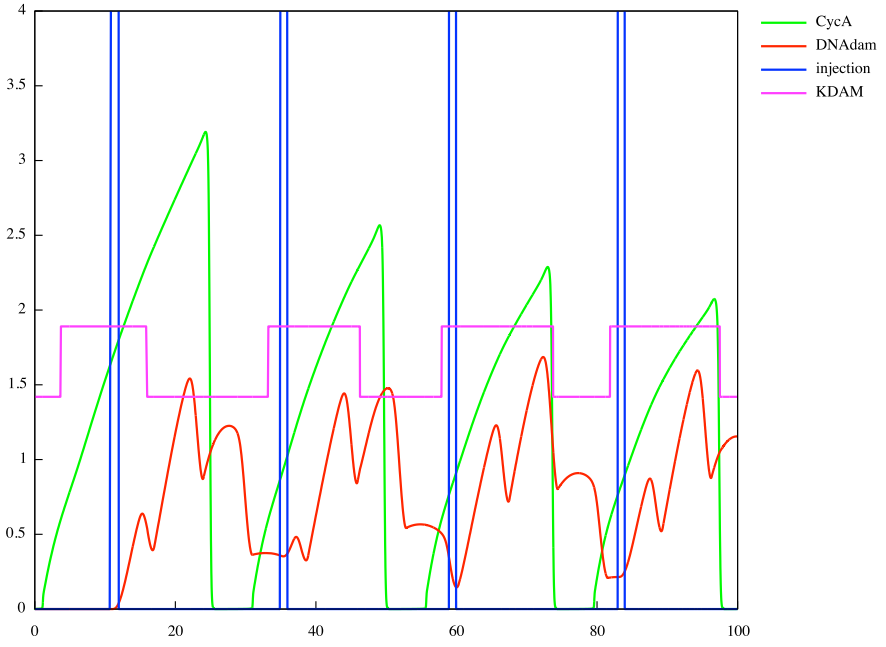


FIGURE 5.17 – Dommages à l'ADN (en rouge) provoqués par la même loi d'exposition (en bleu) que dans la Figure 5.16. mais sur des cellules en opposition de phase.

d'exprimer facilement des propriétés de période d'oscillations pour le couplage et des propriétés de seuil et d'implication entre plusieurs événements concernant les dégâts à l'ADN pour l'optimisation du traitement.

5.7 Analyse de robustesse d'un système de biologie synthétique : cascade d'inhibition génique

On considère dans cette partie le problème de l'optimisation d'un système synthétique constitué d'une succession de réactions de transcriptions qui pourrait être utilisé pour le contrôle temporel d'événements successifs dans des applications de biologie synthétique. Cette cascade a été construite par Hooshanghi et.al [70], nous considérons ici la robustesse d'un comportement souhaité et les possibilités d'améliorer sa robustesse. Pour cela, après avoir introduit le système, nous formalisons le comportement requis, développons un modèle du système prenant en compte la variabilité observée et appliquons la méthode d'analyse de robustesse de la section 3.3 pour évaluer la robustesse de la propriété souhaitée.

5.7.1 Description du système

On considère la cascade d'inhibitions transcriptionnelles successives construite dans E.coli [70]. Le réseau est représenté Figure 5.18. Il est constitué de quatre genes : *tetR*, *lacI*, *cI* et *eyfp* qui codent respectivement pour trois répresseurs TetR, LacI, and CI, et la protéine fluorescente EYFP. Le système peut être contrôlé par l'ajout ou l'élimination d'une petite molécule diffusible, aTc, dans le milieu de culture. Plus précisément, aTc se lie à TetR et empêche la répression de *lacI*. La concentration en aTc est l'entrée, contrôlable, du système.

La fluorescence du système est provoquée par la protéine EYFP, fabriquée au dernier niveau de la succession de réactions du systèmes. La fluorescence est la sortie, mesurable, du système. Intuitivement, la sortie du système a l'état stationnaire est basse pour un niveau bas de l'entrée (concentration en aTc) et élevée pour un niveau élevé de l'entrée.

Il a été montré que le temps de réponse du système à un changement de niveau de l'entrée est caractérisé par une montée rapide de la fluorescence, précédée d'une phase de délai. Aussi une variabilité importante entre les cellules a été observée et cette hétérogénéité des réponses entre les cellules rend difficile l'utilisation de ce système comme horloge biologique, par exemple dans des programmes de développement comme suggéré dans [70].

Dans ce contexte, comme dans beaucoup d'applications à la biologie synthétique, même si seulement une faible proportion de cellules envoie un signal trop tôt ou trop tard cela peut compromette le fonctionnement correct de tout le système. Notre but est de déterminer si et comment on peut obtenir un système possédant un comportement temporel robuste, c'est à dire que toutes les cellules changent d'état dans un intervalle de temps donné.

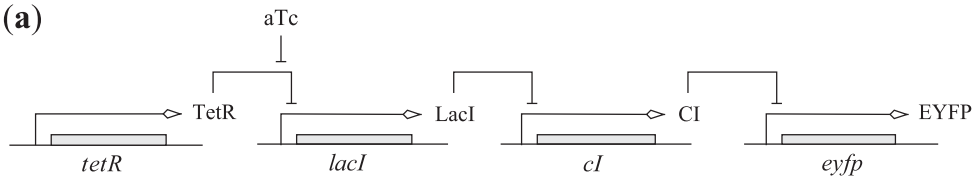


FIGURE 5.18 – Cascade synthétique de transcriptions. TetR réprime *lacI*, LacI réprime *cI*, et CI réprime *eyfp*. aTc contrôle la répression de *lacI* par TetR. La fluorescence de la protéine EYFP est la sortie.

5.7.2 Spécification du comportement souhaité

On dit que le système a un comportement temporel correct si la fluorescence reste sous le seul 10^3 pendant au moins 150 minutes, puis dépasse le seuil 10^5 après au plus 450 minutes et que la transition entre ces nivaux bas et hauts est rapide,i.e se fait en moins de 150 minutes. Ces spécifications sont cohérentes avec le comportement expérimentalement observé de la population de cellules. Ces spécifications sont représentes graphiquement Fig. 5.19 et peuvent être formalisées en logique temporelle de la manière suivante :

$$\begin{aligned}\phi(t_1,t_2) = \quad & \mathbf{G}(time < t_1 \rightarrow [\texttt{EYFP}] < 10^3) \\ \wedge \quad & \mathbf{G}(time > t_2 \rightarrow [\texttt{EYFP}] > 10^5) \\ \wedge \quad & t_1 > 150 \wedge t_2 < 450 \wedge t_2 - t_1 < 150\end{aligned}$$

et

$$\begin{aligned}\psi(t_1,t_2,b_1,b_2,b_3) = \quad & \mathbf{G}(time < t_1 \rightarrow [\texttt{EYFP}] < 10^3) \\ \wedge \quad & \mathbf{G}(time > t_2 \rightarrow [\texttt{EYFP}] > 10^5) \\ \wedge \quad & t_1 > b1 \wedge t_2 < b_2 \wedge t_2 - t_1 < b_3\end{aligned}$$

pour le calcul des domaines de validité et du degré de satisfaction dans une trace donnée.

Modélisation de la variabilité du système

Il y a de nombreuses façons de modéliser la variabilité des cellules (voir par exemple [71])). Notre but est de construire un modèle simple de la variabilité qui soit en accord avec les données expérimentales disponibles.

D'abord on développe un modèle d'équations différentielles similaire à celui de [45] mais en utilisant des fonctions de Hill et avec des paramètres cinétiques ajustés de manière à avoir la meilleure correspondance possible entre les simulations et les données expérimentales (Fig 5.20(a) et (b)). Ces valeurs de référence des paramètres sont appelés $\boldsymbol{p}^*$ dans la suite.

Ensuite on examine différentes manières de modéliser la variabilité entre les cellules, les équations différentielles stochastiques avec un bruit additif ou multiplicatif ou des variations aléatoires des paramètres cinétiques d'un modèle d'ODE selon des distributions normales ou log-normales. On obtient un bon accord quantitatif et qualitatif entre les moyennes et coefficients de variation prédits et observés pour une distribution log-normale des paramètres. Cette correspondance est montrée Fig 5.21(a) et (b). Ainsi on

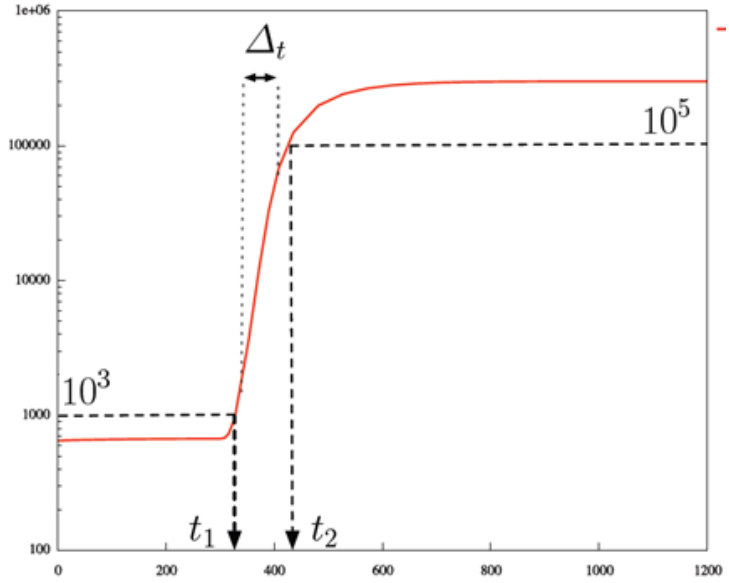


FIGURE 5.19 – Représentation graphique d'un comportement temporel correct : la fluorescence reste sous la valeur 10^3 jusqu'à l'instant t_1, dépasse 10^5 après l'instant t_2 et passe entre ces niveaux bas et hauts en un temps Δ_t. On veut que $t_1 > 150$, $t_2 < 450$ et $\Delta_t < 150$. Les croix représentent les données expérimentales de [70].

choisit ces paramètres distribués selon une loi log-normale comme modèle de perturbation. En utilisant des équations différentielles stochastiques ou des paramètres distribués selon une loi normale nous n'avons pas pu obtenir une bonne correspondance entre les prédictions du modèle et les observations expérimentales. Cela peut être en partie expliqué par la très forte variabilité entre les cellules. En particulier le coefficient de variation observé atteint 1.4 pour un instant donné, ce qui signifie que l'écart type est supérieur à la moyenne.

5.7.3 Amélioration de la robustesse du comportement requis

Une fois le système ainsi que la variabilité modélisés et le comportement temporel requis formalisé on peut déterminer si le système a un comportement temporel requis robuste en calculant la robustesse. En considérant 5000 valeurs des paramètres cinétiques distribués selon une loi log-normale on estime la robustesse du système a $\hat{R}_{\phi,P} = 0.9$: la spécification du système n'est pas satisfaite de manière robuste. Comme attendu la propriété est satisfaite pour les valeurs de référence des paramètres $\boldsymbol{p}^*$ (*i.e.* $sd(T_{\boldsymbol{p}^*}, \phi) = 1$) et ainsi la robustesse et la robustesse absolue sont égales ($\hat{R}^{\boldsymbol{p}^*}_{\phi,P} = 0.9$). La distribution des degrés de satisfaction est représentée Fig 5.21(d), montrant que bien que la majorité des traces satisfasse la spécification, cela n'est pas toujours le cas.

Pour une application en biologie synthétique une horloge plus robuste est requise. Peut on trouver d'autres paramètres du système de manière à améliorer la robustesse du système pour les mêmes perturbations sur ces paramètres ? Pour cela on utilise la méthode de recherche de paramètres définie section 3.2 mais en utilisant la robustesse comme critère d'optimisation, i.e en utilisant comme fonction objectif de la méthode d'optimisation la valeur de la robustesse plutôt que la valeur du degré de satisfaction.

$$
\begin{aligned}
\dot{x}_{lacI} &= \kappa^0_{lacI} + \kappa_{lacI}\frac{u_{aTc}^{\eta_{aTc}}}{\theta_{aTc}^{\eta_{aTc}} + u_{aTc}^{\eta_{aTc}}} - \gamma_{lacI}\, x_{lacI},\\
\dot{x}_{cI} &= \kappa^0_{cI} + \kappa_{cI}\,\frac{\theta_{lacI}^{\eta_{lacI}}}{\theta_{lacI}^{\eta_{lacI}} + x_{lacI}^{\eta_{lacI}}} - \gamma_{cI}\, x_{cI},\\
\dot{x}_{eyfp} &= \kappa^0_{eyfp} + \kappa_{eyfp}\,\frac{\theta_{cI}^{\eta_{cI}}}{\theta_{cI}^{\eta_{cI}} + x_{cI}^{\eta_{cI}}} - \gamma_{eyfp}\, x_{eyfp},\\
\dot{u}_{aTc} &= 0.
\end{aligned}
$$

(a)

κ^0_{lacI}	3.0	θ_{aTc}	1100
κ^0_{cI}	5.1	θ_{lacI}	380
κ^0_{eyfp}	3.2	θ_{cI}	1000
κ_{lacI}	1150	η_{aTc}	3.0
κ_{cI}	538	η_{lacI}	2.1
κ_{eyfp}	3617	η_{cI}	4.0
γ_{lacI}	0.017		
γ_{cI}	0.015		
γ_{eyfp}	0.012		

(b)

$$
\begin{aligned}
\boldsymbol{P} &\hookrightarrow LogN(ln(\boldsymbol{p}^*) - ln(1+\boldsymbol{\sigma}^2)/2, \sqrt{ln(1+\boldsymbol{\sigma}^2)}),\\
\text{avec } \boldsymbol{\sigma} &= \begin{cases} 0.064 & \text{pour } \boldsymbol{\gamma}, \boldsymbol{\theta}, \boldsymbol{\eta},\\ 0.128 & \text{pour } \boldsymbol{\kappa}^0,\\ 0.256 & \text{pour } \boldsymbol{\kappa} \end{cases}\\
\text{et avec } E(\boldsymbol{P}) &= \boldsymbol{p^*} \text{ et } \sigma(\boldsymbol{P}) = \boldsymbol{\sigma p^*}
\end{aligned}
$$

(c)

FIGURE 5.20 – (a) modèle d'ODE de la cascade de transcription. Les concentrations des protéines LacI, CI, EYFP et de aTc sont représentées par x_{lacI}, x_{cI}, x_{eyfp}, et u_{aTc} respectivement. La concentration de la protéine TetR, exprimée de manière constitutive est supposée constante. (b) Valeurs de référence des paramètres $\boldsymbol{p}^*$ et (c) distribution des paramètres modélisant la variabilité du système. $\boldsymbol{\sigma}$ est un vecteur représentant l'intensité du bruit sur chaque paramètre.

Nous avons trouvé les valeurs de paramètres suivants :

$$\tilde{\boldsymbol{p}} = (\boldsymbol{\kappa}^0, \boldsymbol{\kappa}, \boldsymbol{\gamma}, \boldsymbol{\theta}, \boldsymbol{\eta}) = ((2.30, 4.20, 3.78), (1234.5, 514.5, 5174.3), (0.024, 0.015, 0.012), (1647.2, 662.8, 936.4), (4.8, 3.7, 8.4))$$

En comparant les paramètres originaux $\boldsymbol{p}^*$ à ces paramètres optimisés $\tilde{\boldsymbol{p}}$ on s'aperçoit que le taux de production de EYFP et les coefficients de Hill ont été significativement augmentés. Etant donné que la spécification indique que la transition doit se faire rapidement il n'est pas étonnant qu'il faille pour cela augmenter ces paramètres. Expérimentalement, il est difficile de modifier ces coefficients de Hill donc nous avons recherché et trouvé des paramètres cinétiques qui garantissent un comportement temporel robuste sans changer les valeurs des coefficients de Hill.

Les simulations numériques montrent que le comportement souhaité est en effet plus robuste. (comparer Fig 5.21(c) et Fig 5.22(a)). Aussi le coefficient de variation suggère que la variabilité d'une cellule a une autre est significativement diminuée lorsque les contraintes temporelles de la spécifications sont satisfaites (pour $time < 150$) et significativement augmentée en dehors (pour $150 < time < 450$, voir Fig 5.22(b)).

Il pourrait être intéressant de vérifier si ceci se produit systématiquement pour des valeurs de paramètres améliorant la robustesse du comportement souhaité. Cela pourrait révéler une loi de conservation globale de la variabilité dans ce système ainsi qu'un compromis entre robustesse et fragilité [54].

5.7.4 Influence des paramètres sur la robustesse du comportement

Afin d'estimer l'influence des paramètres sur la robustesse du comportement on évalue les indices de sensibilité (présentés section 3.4.3) du comportement requis de huit paramètres ($(\boldsymbol{\kappa}^0$,$\boldsymbol{\kappa}$, γ, et u_{aTc}). On définit pour cette analyse de sensibilité globale une zone de valeurs de paramètres centrée sur $p*$, avec des valeurs maximales correspondant à $10 \times p*$ et des valeurs minimales correspondant à $0.1 \times p*$. Les indices de sensibilité du premier ordre de 8 paramètres sont donnés table 5.7.

S_{γ}	20.2 %, %
$S_{\kappa_{eyfp}}$	7.4 %%
$S_{\kappa_{cI}}$	6.1 %%
$S_{\kappa^0_{lacI}}$	3.3 %%
$S_{\kappa^0_{cI}}$	2.0 %%
$S_{\kappa_{lacI}}$	1.5 %%
$S_{\kappa^0_{eyfp}}$	0.9 %%
$S_{u_{aTc}}$	0.4 %%
total	40.7 %

TABLE 5.7 – Indices de sensibilité globaux du premier ordre.

L'analyse des indices de sensibilité montre que les variations de γ ont un impact très fort sur la robustesse de la cascade. Les variations de ce paramètre seul sont responsables de 20 % des variations de la robustesse. Ce paramètre correspond à la dégradation des espèces du système (et correspond donc à la dilution causée par la croissance des cellules). Au contraire les variations de aTc ont un impact faible sur le comportement de la cascade. Bien que cela semble être en contradiction avec l'ultrasensibilité entre l'entrée et la sortie de la cascade [70], cela montre simplement que les niveaux de concentra-

tions d'aTc utilisés sont assez élevés pour rendre la cascade insensible même à de larges variations d'aTc.

Un résultat surprenant de cette analyse est la différence d'importance des variations des niveaux de base et régulés de production d'EYFP : κ^0_{eyfp} et κ_{eyfp}. Etant donné que l'on impose des contraintes similaires aux niveaux bas et élevés d'EYFP et que ces niveaux sont respectivement fortement reliés aux rapports $\kappa^0_{eyfp}/\gamma_{eyfp}$ et $\kappa_{eyfp}/\gamma_{eyfp}$, on pourrait s'attendre à avoir des indices de sensibilité similaires pour κ^0_{eyfp} et κ_{eyfp}.

En réalité le niveau bas d'EYFP dépend aussi de la valeur à l'état stable de Cl, proportionnel à κ_{cI}/γ_{cI}. Comme les variations de κ_{cI} ont un fort impact sur la robustesse nos résultats suggèrent que lorsque la cascade n'est pas induite (aTc faible) le niveau basal de production d'EYFP est dû à une répression incomplète par le promoteur Cl (expliquant la valeur élevée de l'indice de sensibilité pour κ_{cI}) plutôt que par une expression constitutive faible du promoteur.

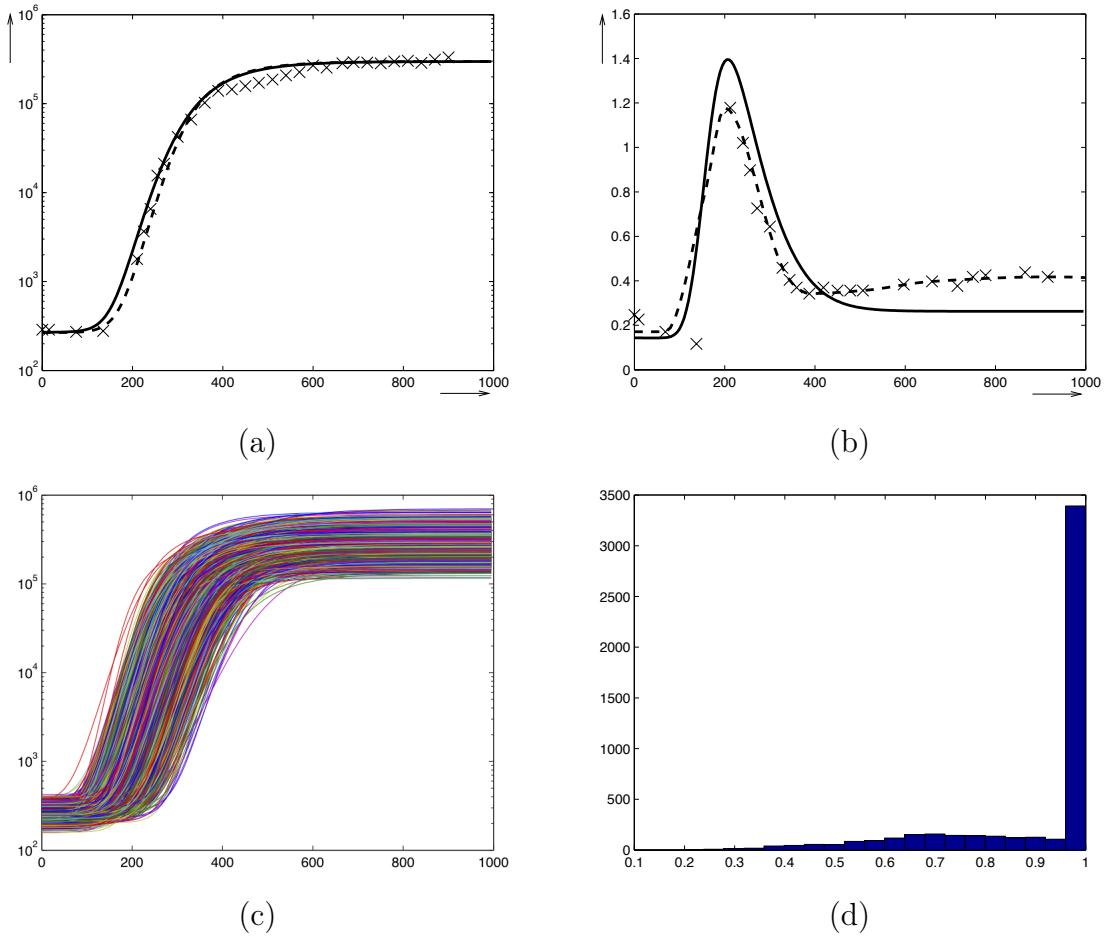
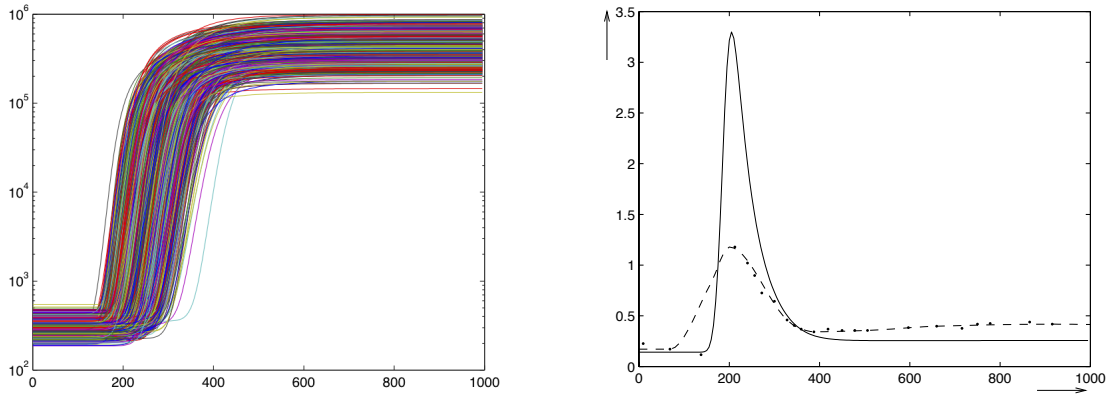


FIGURE 5.21 – (a) Évolution temporelle de la fluorescence après l'ajout d'aTc. Les croix représentent les données expérimentales de [70], la ligne en pointillée les prédictions du modèle d'ODE simulé avec les valeurs de référence des paramètres $\boldsymbol{p}^*$ et la ligne continue la moyenne de 5000 simulations du modèle pour des distributions log-normales des paramètres. (b) Évolution temporelle du coefficient de variation de la fluorescence après l'ajout d'aTc. Les croix représentent les coefficients de variation obtenus à partir des données expérimentales de [70], la ligne continue ceux obtenus à partir de 5000 simulations du modèle pour des paramètres aléatoirement distribués selon une loi log-normale de moyenne $\boldsymbol{p}^*$. (c) Simulation numérique du système d'ODE pour des paramètres distribués selon une loi log-normale de moyenne $\boldsymbol{p}^*$. (d) Distribution des degrés de satisfaction de 5000 traces numériques du modèle perturbé. La robustesse correspondante est $\hat{R}_{\phi,P} = 0.9$



(a) (b)

FIGURE 5.22 – (a) Simulation numérique du système d'ODE pour des paramètres distribués selon une loi log-normale de moyenne $\tilde{\boldsymbol{p}}$. (b) Évolution temporelle du coefficient de variation de la fluorescence apres l'ajout d'aTc. Les croix représentent les coefficients de variation obtenus à partir des données expérimentales de [70], la ligne continue ceux obtenus à partir de 5000 simulations du modèle pour des parametres aléatoirement distribués selon une loi log-normale de moyenne $\tilde{\boldsymbol{p}}$.

Chapitre 6

Conclusion

6.1 Contributions

Nous avons défini un langage de logique temporelle avec variables libres et étendu le model-checking classique de formules instanciées à un problème de résolution de contraintes. La résolution de contraintes temporelles et le calcul du domaine de validité d'une formule de logique temporelle permet la définition d'un degré de satisfaction continu dans l'intervalle [0,1] de formules de logique temporelle. Cette extension est d'abord effectuée pour la logique temporelle linéaire LTL et ensuite étendue à la logique avec branchement CTL.

Ce travail donne de nouvelles possibilités d'utilisation de la logique temporelle pour la biologie des systèmes. Plus généralement le degré de satisfaction continu de formules de logique temporelle ouvre la voie à l'utilisation de spécifications en logique temporelle pour des problèmes d'optimisation. La spécification requise est formalisée en logique temporelle et le degré d'évaluation continu de la formule est utilisé comme fonction objectif à optimiser par une méthode d'optimisation continue.

Ce travail présente l'utilisation du degré de violation continu de formules de logique temporelle avec la stratégie d'évolution CMA-ES pour fournir une méthode de recherche de paramètres de modèles continus de réseaux d'interaction biochimiques. Cette méthode est intégrée à l'environnement de modélisation Biocham. Son utilisation est illustrée sur plusieurs modèles biologiques, (modèles du cycle cellulaire, de la cascade de signalisation MAPK, du réseau de signalisation induit par l'angiotensine, et du couplage du cycle cellulaire à l'horloge circadienne). La méthode de recherche de paramètres a été parallélisée et peut être utilisée sur plusieurs milliers de processeurs pour aborder des problèmes plus complexes.

Le degré de satisfaction continu de formules est aussi utilisé pour fournir une méthode de calcul générale pour l'analyse de robustesse de systèmes biologiques en se basant sur la définition de la robustesse de Kitano [57]. Enfin on a présenté les méthodes d'analyse de sensibilité globale pouvant être utilisées avec ce degré de satisfaction continu. Ces méthodes sont également intégrées à l'environnement de modélisation Biocham.

6.2 Discussion

Le degré de satisfaction continu de formules de logique temporelle fournit plus d'informations qu'une interprétation booléenne et peut être utilisé dans de nombreuses situations en biologie des systèmes pour raisonner sur les traces numériques. Il permet l'utilisation efficace de la logique temporelle pour des problèmes de recherche de paramètres en étant utilisé comme fonction objectif pour des méthodes d'optimisation continues. Une évaluation booléenne permet uniquement un échantillonnage régulier ou aléatoire de l'espace de recherche. La complexité de la recherche est alors exponentielle par rapport au nombre de paramètres recherchés et limitée à 2 ou 3 paramètres.

La recherche de paramètres est un problème crucial de la biologie des systèmes. La méthode présentée dans ce travail utilise la stratégie d'évolution CMA-ES et est appliquée pour rechercher plusieurs dizaines de paramètres inconnus vis à vis de spécifications définies dans une ou plusieurs conditions expérimentales. On a montré sur un problème synthétique que la parallélisation est efficace sur plusieurs milliers de processeurs, et permettrait donc d'aborder des problèmes plus complexes.

Ce formalisme est adapté aux données biologiques, souvent incomplètes et bruitées, en permettant la définition de spécification de haut niveau (existence d'oscillations, atteignabilité, point de passage ...). On a illustré l'application de ces méthodes au travers de plusieurs problèmes biologiques.

On a montré que la complexité de la résolution de contraintes d'une formule de taille f comprenant k variables sur une trace de taille n est dans le pire cas $\mathcal{O}(nf)^{2k}$) (pour des formules contenant une variable par contrainte atomique) et $\mathcal{O}(2^{nf})$ (dans le cas général de contraintes linéaires sur les variables). Cependant l'évaluation des performances de la résolution de contraintes, présentée tableaux 5.4 et 5.5 indique que dans le cas général ces bornes ne sont pas atteintes. En pratique le temps de résolution des contraintes temporelles est équivalent au temps requis pour la génération de la trace par simulation d'un modèle d'ODE. Dans les exemples abordés le temps de calcul pour la résolution de contraintes peut aller au plus jusqu'à 3 fois le temps requis pour générer une trace par simulation d'un modèle. Le degré de satisfaction continu peut donc être utilisé efficacement comme fonction objectif pour la recherche de paramètres.

Les logiques temporelles probabilistes et le model-checking probabiliste ont été utilisées pour l'analyse d'un modèle probabiliste de la cascade de signalisation MAPK [79]. Cependant ces méthodes calculent la probabilité qu'une propriété soit exactement satisfaite. Cette probabilité se calcule en effectuant plusieurs simulations d'un modèle probabiliste tandis que le degré de satisfaction continu introduit dans ce travail est calculé par rapport à une seule simulation d'un modèle. Ces méthodes ne fournissent pas d'information quantitatives sur les formules non satisfaites et ne peuvent donc pas être comparées au degré de satisfaction continu présenté dans ce travail.

Fainekos et Pappas ont proposé [80] un degré de satisfaction pour les spécifications temporelles. Leur degré de satisfaction correspond à une distance entre une trace et l'ensemble des traces satisfaisant une formule tandis que dans notre cas le degré d'évaluation correspond à une distance entre une formule et l'ensemble des formules satisfaites sur une trace. Un avantage de notre approche est que la dimension de l'espace des formules (nombre de variables libres dans la formule) est en général bien plus faible que la di-

mension de l'espace des traces (nombre de points dans la trace). Il est ainsi plus facile d'avoir une implémentation efficace sur cet espace de plus faible dimension.

6.3 Perspectives

La méthode de résolution de contraintes temporelles se fait par rapport à une trace de simulation numérique. Dans la plupart des cas, le temps de calcul pour la résolution de contraintes est similaire au temps de calcul pour obtenir la trace à partir d'un système d'équations différentielles ordinaires. Cependant la méthode de résolution utilisée dans ce travail étant la méthode implicite à pas adaptatif de Rosenbrock, le nombre de points dans une trace est fortement dépendant de la forme de la solution. Lors de la recherche de paramètres de nombreuses solutions (valeurs de paramètres) sont essayées, dont certaines correspondent à des comportements pour lesquels la trace est de très grande taille. Le calcul du domaine de validité se faisant en calculant sur chaque point de la trace le domaine de validité de chaque sous formule, il pourrait être intéressant pour accélérer le calcul de simplifier ces grandes traces avant le calcul. Plusieurs pistes sont envisageables : échantillonnage régulier plus grossier de la trace, conservation des points ou au moins un signe de dérivée change ou simplification spécifique à un type de formule. Par exemple pour une formule d'oscillations, conserver uniquement les points ou la dérivée change ne modifie pas le domaine de validité de la formule. Pour une formule portant sur des seuils il conviendrait de conserver tous les points ou un seuil est franchi pour ne pas modifier le domaine de validité de la formule. Ce problème correspond au problème de la vérification de formules temporelles sur des traces discrètes [28, 29].

L'utilisation de la logique temporelle est freinée par la difficulté, pour les non experts, de formaliser des comportements temporels en logique temporelle. Cette difficulté est accentuée dans le cas de formules avec variables pour la définition d'un degré de satisfaction continu : il faut écrire une formule, choisir les constantes que l'on va abstraire par des variables et choisir l'objectif de ces variables. Ce sont ces choix qui définissent un comportement et la manière dont il sera évalué sur une trace. Il a été proposé pour aborder la difficulté pour les non experts de tirer profit de la logique temporelle d'utiliser des motifs en langage naturel pouvant être automatiquement traduits en logique temporelle [81]. Il serait intéressant d'utiliser cette approche pour des formules de logique temporelle avec variables en définissant des motifs correspondant à des comportements et la manière dont on veut les évaluer.

Trouver des paramètres tels que le comportement d'un modèle soit valide par rapport aux connaissances expérimentales est un problème qui permet de valider un modèle et donc éventuellement de valider des hypothèses de modélisation. Cela permet de faire des prédictions et des expériences non réalisables en pratique et d'améliorer la compréhension d'un système biologique. Souvent la recherche de paramètres se fait en même temps que la recherche de la structure d'un modèle. Certaines parties de la structure peuvent être connues et déjà validées dans la littérature. Les parties inconnues peuvent être testées lors de la recherche de paramètre : on retient les modèles pour lesquels on trouve des paramètres satisfaisants. Cette approche permet uniquement de tester un faible nombre de structures possibles. Une alternative serait de combiner la recherche de paramètres à la recherche de la structure : on ajoute au modèle les réactions supposées et la recherche de paramètre sur les paramètres cinétiques associés à ces réactions re-

vient à rechercher la validité de ces réactions. Par exemple en dessous d'un certain seuil d'activité des réactions on peut décider de les supprimer du modèle.

L'analyse de sensibilité globale apporte une autre piste à ce problème. Lorsqu'une recherche de paramètre échoue, on peut utiliser le degré de satisfaction de la spécification (ou mieux de la partie de la spécification posant problème, identifiable en examinant le domaine de validité) pour une analyse de sensibilité globale. On obtiendrait alors, indépendamment d'une valeur particulière des paramètres la contribution de chaque paramètre sur la variation du degré de satisfaction. Cela pourrait permettre d'identifier les réactions, associées à ces paramètres, qui influent sur un comportement donné et qu'il conviendrait de modifier. Plus généralement la combinaison de spécifications de haut niveau en logique temporelle avec l'analyse de sensibilité est intéressante car elle pourrait permettre d'identifier les modules d'un système responsable d'un comportement de haut niveau.

Bibliographie

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

[2] Chabrier, N., Fages, F. : Symbolic model checking of biochemical networks. In Priami, C., ed. : CMSB'03 : Proceedings of the first workshop on Computational Methods in Systems Biology. Volume 2602 of Lecture Notes in Computer Science., Rovereto, Italy, Springer-Verlag (2003) 149–162

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

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

[5] Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R. : Analysis of signalling pathways using the continuous time markov chains. In Plotkin, G., ed. : Transactions on Computational Systems Biology VI. Volume 4220 of Lecture Notes in BioInformatics. Springer-Verlag (2006) 44–67 CMSB'05 Special Issue.

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

[7] Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S. : Machine learning biochemical networks from temporal logic properties. In Plotkin, G., ed. : Transactions on Computational Systems Biology VI. Volume 4220 of Lecture Notes in BioInformatics. Springer-Verlag (2006) 68–94 CMSB'05 Special Issue.

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

[9] Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M., Schneider, D. : Validation of qualitative models of genetic regulatory networks by model checking : Analysis of the nutritional stress response in *Escherichia coli*. Bioinformatics **21** (2005) i19–i28

[10] Emerson, A.E. : Temporal and modal logic. In van Leeuwen, J., ed. : Handbook of Theoretical Computer Science. Volume B : Formal Models and Sematics. MIT Press (1990) 995–1072

[11] Clarke, E.M., Grumberg, O., Peled, D.A. : Model Checking. MIT Press (1999)

[12] Fages, F. : Temporal logic constraints in the biochemical abstract machine BIOCHAM (invited talk). In Springer-Verlag, ed. : Proceedings of Logic Based Program Synthesis and Transformation, LOPSTR'05. Number 3901 in Lecture Notes in Computer Science, London, UK (2005)

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

[14] Kohn, K.W. : Molecular interaction map of the mammalian cell cycle control and DNA repair systems. Molecular Biology of the Cell **10** (1999) 2703–2734

[15] Batt, G., Yordanov, B., Weiss, R., Belta, C. : Robustness analysis and tuning of synthetic gene networks. Bioinformatics **23** (2007) 2415–2422

[16] Bagnara, R., Hill, P.M., Zaffanella, E. : The Parma Polyhedra Library : Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Science of Computer Programming **72** (2008) 3–21

[17] Rosu, G., Sokolsky, O. : Runtime Verification : First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings. Springer-Verlag New York Inc (2010)

[18] Hucka, M., et al. : The systems biology markup language (SBML) : A medium for representation and exchange of biochemical network models. Bioinformatics **19** (2003) 524–531

[19] Fages, F., Soliman, S. : Formal cell biology in BIOCHAM. In Bernardo, M., Degano, P., Zavattaro, G., eds. : 8th Int. School on Formal Methods for the Design of Computer, Communication and Software Systems : Computational Systems Biology SFM'08. Volume 5016 of Lecture Notes in Computer Science., Bertinoro, Italy, Springer-Verlag (2008) 54–80

[20] Segel, L.A. : Modeling dynamic phenomena in molecular and cellular biology. Cambridge University Press (1984)

[21] Szallasi, Z., Stelling, J., Periwal, V., eds. : System Modeling in Cellular Biology : From Concepts to Nuts and Bolts. MIT Press (2006)

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

[23] Calzone, L., Fages, F., Soliman, S. : BIOCHAM : An environment for modeling biological systems and formalizing experimental knowledge. Bioinformatics **22** (2006) 1805–1807

[24] Gilbert, D., Heiner, M., Lehrack, S. : 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)

[25] Matsuno, H., Doi, A., Nagasaki, M., Miyano, S. : Hybrid petri net representation of gene regulatory network. In : Proceedings of the 5th Pacific Symposium on Biocomputing. (2000) 338–349

[26] Priami, C., Regev, A., Silverman, W., Shapiro, E. : Application of a stochastic name passing calculus to representation and simulation of molecular processes. Information Processing Letters **80** (2001) 25–31

[27] Phillips, A., Cardelli, L. : A correct abstract machine for the stochastic pi-calculus. Transactions on Computational Systems Biology (2005) Special issue of BioConcur 2004.

[28] Eisner, C., Fisman, D., Havlicek, J., Lustig, Y., McIsaac, A., van Campenhout, D. : Reasoning with temporal logic on truncated paths. In : Proceedings of Fifteenth Computer-Aided Verification conference (CAV'O3). Volume 2725 of Lecture Notes in Computer Science., Springer (2003) 27–39

[29] Maler, O., Nickovic, D., Pnueli, A. : Checking temporal properties of discrete, timed and continuous behaviors. In Avron, A., Dershowitz, N., Rabinovich, A., eds. : Pillars of Computer Science, Essays Dedicated to B. Trakhtenbrot. Volume 4800 of Lecture Notes in Computer Science., Springer (2008) 475–505

[30] Abadi, M., Lamport, L., Wolper, P. : Realizable and unrealizable specifications of reactive systems. Automata, languages and programming (1989) 1–17

[31] Jobstmann, B., Bloem, R. : Optimizations for LTL synthesis. In : Conference on Formal Methods in Computer Aided Design, Citeseer (2006) 117–124

[32] Fages, F., Rizk, A. : On the analysis of numerical data time series in temporal logic. 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) 48–63

[33] Fages, F., Rizk, A. : On temporal logic constraint solving for the analysis of numerical data time series. Theoretical Computer Science **408** (2008) 55–65

[34] Dang, T., Le Guernic, C., Maler, O. : Computing reachable states for nonlinear biological models. In : Computational Methods in Systems Biology, Springer (2009) 126–141

[35] Girard, A. : Reachability of uncertain linear systems using zonotopes. Hybrid Systems : Computation and Control (2005) 291–305

[36] Rizk, A., Batt, G., Fages, F., Soliman, S. : On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In Heiner, M., Uhrmacher, A., eds. : CMSB'08 : Proceedings of the fourth international conference on Computational Methods in Systems Biology. Volume 5307 of Lecture Notes in Computer Science., Springer-Verlag (2008) 251–268

[37] Rizk, A., Batt, G., Fages, F., Soliman, S. : Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures. Theoretical Computer Science (2011) In press.

[38] Emerson, A.E., Clarke, E.M. : Characterizing correctness properties of parallel programs using fixpoints. In : Proceedings of the 7th Colloquium on Automata, Languages and Programming, London, UK, Springer-Verlag (1980) 169–181

[39] Fages, F., Rizk, A. : From model-checking to temporal logic constraint solving. In : Proceedings of CP'2009, 15th International Conference on Principles and Practice of Constraint Programming. Number 5732 in Lecture Notes in Computer Science, Springer-Verlag (2009) 319–334

[40] Hansen, N., Ostermeier, A. : Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9** (2001) 159–195

[41] Goldberg, D. : Genetic algorithms in search, optimization, and machine learning. Addison-wesley (1989)

[42] Barkal, N., Leibler, S. : Robustness in simple biochemical networks. Nature **387** (1997) 913–916

[43] Davidson, E., Levine, M. : Properties of developmental gene regulatory networks. Proceedings of the National Academy of Sciences **105** (2008) 20063–20066

[44] Shen, X., Collier, J., Dill, D., Shapiro, L., Horowitz, M., McAdams, H.H. : Architecture and inherent robustness of a bacterial cell-cycle control system. Proceedings of the National Academy of Sciences **105** (2008) 11340–11345

[45] Batt, G., Yordanov, B., Weiss, R., Belta, C. : Robustness analysis and tuning of synthetic gene networks. Bioinformatics **23** (2007) 2415–2422

[46] Shinar, G., Milo, R., Martinez, M., Alon, U. : Input-output robustness in simple bacterial signaling systems. Proceedings of the National Academy of Sciences **104** (2007) 19931–19935

[47] Gonze, D., Halloy, J., Goldbeter, A. : Robustness of circadian rhythms with respect to molecular noise. Proceedings of the National Academy of Sciences of the USA **99** (2002) 673–678

[48] Stelling, J., Gilles, E., III, F.D. : Robustness properties of circadian clock architectures. Proceedings of the National Academy of Sciences of the USA **101** (2004) 13210–13215

[49] El-Samad, H., Kurata, H., Doyle, J., Gross, C., Khammash, M. : Surviving heat shock : Control strategies for robustness and performance. Proceedings of the National Academy of Sciences of the USA **102** (2005) 2736–2741

[50] Ciliberti, S., Martin, O.C., Wagner, A. : Innovation and robustness in complex regulatory gene networks. Proceedings of the National Academy of Sciences of the USA **104** (2007) 13591–13596

[51] von Dassow, G., Meir, E., Munro, E., Odell, G. : The segment polarity network is a robust developmental module. Nature **406** (2000) 188–192

[52] Ingolia, N.T. : Topology and robustness in the drosophila segment polarity network. PLoS Biology **2** (2004) e123

[53] Chaves, M., Sengupta, A., Sontag, E. : Geometry and topology of parameter space : investigating measures of robustness in regulatory networks. Journal of Mathematical Biology **104** (2007) 13591–13596

[54] Kitano, H. : Biological robustness. Nature Review Genetics **11** (2004) 826–837

[55] Yi, T.M., Huang, Y., Simon, M.I., Doyle, J. : Robust perfect adaptation in bacterial chemotaxis through integral feedback control. Proceedings of the National Academy of Sciences of the USA **97** (2000) 4649–4653

[56] Stelling, J., Sauer, U., Szallasi, Z., III, F.D., Doyle, J. : Robustness of cellular functions. Cell **118** (2004) 675–685

[57] Kitano, H. : Towards a theory of biological robustness. Molecular Systems Biology **3** (2007) 137

[58] Rizk, A., Batt, G., Fages, F., Soliman, S. : A general computational method for robustness analysis with applications to synthetic gene networks. Bioinformatics **12** (2009) il69–il78

[59] Saltelli, A., Tarantola, S., Campolongo, F., Ratto, M. : Sensitivity analysis in practice : A guide to assessing scientific models. Wiley Press (2004)

[60] Morris, M. : Factorial sampling plans for preliminary computational experiments. Technometrics **33** (1991) 161–174

[61] Sobol, I. : Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. Mathematics and Computers in Simulation **55** (2001) 271–280

[62] Jacques, J. : Contributions à l'analyse de sensibilité et à l'analyse discriminante généralisée. PhD thesis, Université Joseph Fourier - Grenoble I (2005)

[63] Homma, T., Saltelli, A. : Use of Sobol's quasirandom sequence generator for integration of modified uncertainty importance measure. Journal of Nuclear Science and Technology **32** (1995) 1164–1173

[64] Chen, K.C., Calzone, L., Csikász-Nagy, A., Cross, F.R., Györffy, B., Val, J., Novàk, B., Tyson, J.J. : Integrative analysis of cell cycle control in budding yeast. Molecular Biology of the Cell **15** (2004) 3841–3862

[65] Chen, K.C., Csikász-Nagy, A., Györffy, B., Val, J., Novàk, B., Tyson, J.J. : Kinetic analysis of a molecular model of the budding yeast cell cycle. Molecular Biology of the Cell **11** (2000) 396–391

[66] Levchenko, A., Bruck, J., Sternberg, P.W. : Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. PNAS **97** (2000) 5818–5823

[67] Qiao, L., Nachbar, R.B., Kevrekidis, I.G., Shvartsman, S.Y. : Bistability and oscillations in the huang-ferrell model of mapk signaling. PLoS Computational Biology **3** (2007) 1819–1826

[68] Fages, F., Soliman, S. : From reaction models to influence graphs and back : a theorem. In : Proceedings of Formal Methods in Systems Biology FMSB'08. Number 5054 in Lecture Notes in Computer Science, Springer-Verlag (2008)

[69] Ventura, A.C., Sepulchre, J.A., Merajver, S.D. : A hidden feedback in signaling cascades is revealed. PLoS Computational Biology **4** (2008) e1000041

[70] Hooshangi, S., Thiberge, S., Weiss, R. : Ultrasensitivity and noise propagation in a synthetic transcriptional cascade. Proceedings of the National Academy of Sciences of the USA **102** (2005) 3581–3586

[71] Manninen, T., Linne, M.L., Ruohonen, K. : Developing itô stochastic differential equation models for neuronal signal transduction pathways. Computational Biology and Chemistry **30** (2006) 280–291

[72] De Maria, E., Fages, F., Rizk, A., Soliman, S. : Design, optimization, and predictions of a coupled model of the cell cycle, circadian clock, dna repair system, irinotecan metabolism and exposure control under temporal logic constraints. Theoretical Computer Science (2011) In press.

[73] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P. : Molecular Biology of the Cell, fourth edition. Garland Science (2008)

[74] Novák, B., Tyson, J.J. : A model for restriction point control of the mammalian cell cycle. Journal of Theoretical Biology **230** (2004) 1383–1388

[75] Leloup, J.C., Goldbeter, A. : Toward a detailed computational model for the mammalian circadian clock. Proceedings of the National Academy of Sciences **100** (2003) 7051–7056

[76] Ciliberto, A., Novák, B., Tyson, J.J. : Steady states and oscillations in the p53/mdm2 network. Cell Cycle **4** (2005) 488–493

[77] Dimitrio, L. : Irinotecan : Modelling intracellular pharmacokinetics and pharmacodynamics. m2 master thesis (in french, english summary). Technical report, University Pierre-et-Marie-Curie and INRIA internal report (2007)

[78] Ballesta, A., Dulong, S., Abbara, C., Cohen, B., Okyar, A., Clairambault, J., Levi, F. : A combined experimental and mathematical approach for molecular-based optimization of irinotecan circadian delivery. (Submitted to PLOS Computational Biology)

[79] Kwiatkowska, M., Norman, G., Parker, D. : Using probabilistic model checking in systems biology. SIGMETRICS Performance Evaluation Review **35** (2008) 14–21

[80] Fainekos, G., Pappas, G. : Robustness of temporal logic specifications. In Havelund, K., Núñez, M., Rosu, G., Wolff, B., eds. : Proceedings of the International Workshop on Formal Approaches to Software Testing and Runtime Verification, FATES/RV'06. Volume 4262 of Lecture Notes in Computer Science., Springer-Verlag (2006) 178–192

[81] Monteiro, P., Ropers, D., Mateescu, R., Freitas, A., De Jong, H. : Temporal logic patterns for querying dynamic models of cellular interaction networks. Bioinformatics **24** (2008) i227

Liste des figures

Liste des tableaux

Liste des Algorithmes

Résumé

La logique temporelle et les algorithmes de model-checking ont été utilisés avec succès pour formaliser des propriétés biologiques de systèmes biochimiques complexes et vérifier automatiquement leur satisfaction sur des modèles quantitatifs ou qualitatifs. Cette approche repose sur un paradigme logique pour la biologie des systèmes qui consiste à faire les identifications suivantes [12] :

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. 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 [13].

Cependant une évaluation booléenne classique de formules de logique temporelle n'est pas adaptée à de nombreux problèmes (recherche de paramètres, analyse quantitative de robustesse). En particulier, trouver des modèles mathématiques satisfaisant une spécification obtenue par la formalisation d'expériences biologiques est une étape cruciale de la modélisation pour laquelle une évaluation booléenne est inefficace. Une évaluation booléenne ne fait pas de distinction entre deux modèles faux. Ainsi il n'est pas possible de savoir dans quelle direction orienter la recherche, comment modifier le modèle pour améliorer la satisfaction de la formule. La recherche n'est pas guidée. La résolution de contraintes de logique temporelle permet la définition d'un degré de satisfaction continu de formules de logique temporelle adapté à ces problèmes. Définir un degré de satisfaction continu [36, 39] de formules de logique temporelle avec contraintes et utiliser cette mesure de satisfaction comme fonction objectif ouvre la voie à l'utilisation de méthodes d'optimisation continues pour chercher des paramètres cinétiques satisfaisant des propriétés biologiques formalisées en logique temporelle.

Ainsi après avoir étendu le model checking en problème de résolution de contraintes de logique temporelles et permis la définition d'un degré de satisfaction continu de formules nous présenterons les applications à la biologie des systèmes au travers de la recherche de paramètres, l'analyse de robustesse et l'analyse de sensibilité. Ces méthodes seront enfin évaluées sur plusieurs exemples biologiques.

Mots-clefs : model checking, logique temporelle, résolution de contraintes, biologie des systèmes.