Analyse de données de concentrations protéiques en logique temporelle

François Fages
cours M2-19 bioinformatique formelle
MPRI Examen partiel

November 26, 2007

On s'intéresse à formaliser en logique temporelle les données de concentration de protéines provenant d'expériences biologiques, et à calculer automatiquement les formules LTL avec contraintes numériques d'un certain schéma qui sont vraies sur ces données.

En particulier, on s'intéresse aux deux schémas suivants de formule LTL avec contraintes :

- *Accessibilité* : `F([A]>=v)`, indiquant les seuils de concentration v atteints par la protéine A au cours du temps;
- *Stabilité* : `G([A]>=v1 & [A]=<v2)`, indiquant l'intervalle $[v1, v2]$ des valeurs de concentration prises par la protéine A au cours du temps;

1 Données

Les courbes de la figure 1 indiquent l'évolution temporelle des concentrations de quatre protéines sur un horizon de 100 unités de temps.

Pour chaque protéine, indiquer (approximativement) les domaines de valeurs des variables v, $v1$, $v2$ pour lesquelles les formules d'accessibilité et de stabilité sont vraies.

Molécules	*Accessibilité* v	*Stabilité* $(v1,v2)$
`Cdc2`	*[0, 0.500]*	*([0, 0.338],[0.500, ∞])*
`Cdc2-Cyclin~{p1,p2}`	*[0, 0.311]*	*([0, 0.005],[0.311, ∞])*
`Cdc2-Cyclin~{p1}`	*[0, 0.194]*	*([0, 0.002],[0.194, ∞])*
`Cyclin~{p1}`	*[0, 0.159]*	*([0, 0.004],[0.159, ∞])*

2 Algorithme

Soit une série temporelle de données de concentrations pour un vecteur de protéines $\vec{A}$

$$(t_1, [\vec{A}]_1), ..., (t_n, [\vec{A}]_n)$$

Soit ϕ une formule LTL quelconque, formée avec les opérateurs temporels X, F, U, G, avec les connecteurs logiques $\wedge$ et $\vee$, et contenant des contraintes uniquement de la forme $[A] >= x$ ou $[A] =< y$, où $x, y, ...$ sont des variables réelles positives et A une protéine.

2.1

Donner un exemple de formule ϕ faisant intervenir deux protéines et une seule variable

$G([A] =< v \& [B] =< v)$

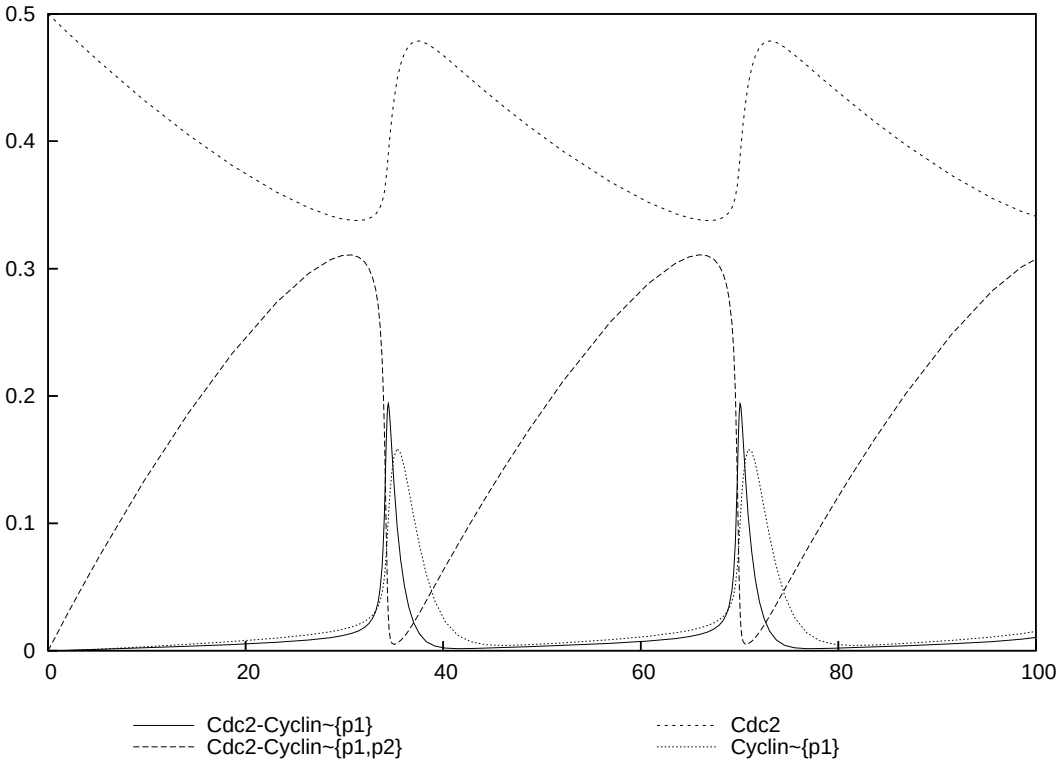

Figure 1: Relevé de concentrations de quatre protéines sur 100 unités de temps.

2.2

Donner un algorithme de calcul des domaines des variables apparaissant dans ϕ, pour lesquels la formule ϕ est vraie.

1. *étiqueter chaque point de la trace par les sous-formules atomiques de ϕ avec leur domaine de validité comme suit:*
   - *pour une formule ψ sans variable, mettre l'étiquette $(\psi, \mathcal{D}_\psi(t_i) = \mathbb{R}^n)$ si ψ est vraie au temps t_i, et $(\psi, \mathcal{D}_\psi(t_i) = \varnothing)$ si elle est fausse;*
   - *pour une contrainte $[A] \geq p$ (soit de la forme valeur $\geq$ variable) mettre l'étiquette $([A] \geq p, \mathcal{D}_{[A]\geq p}(t_i))$ où $\mathcal{D}_{[A]\geq p}(t_i)$ est le demi-espace de $\mathbb{R}^n$ défini par $p \leq [A](t_i)$;*
   - *procéder de façon similaire pour une contrainte $[A] \leq p$.*
2. *en partant de la fin de la trace, étiqueter les points t_i par les sous-formules de la forme :*
   - *$F\psi$ avec pour domaine de validité $\mathcal{D}_{F\psi}(t_i) = \mathcal{D}_{F\psi}(t_{i+1}) \cup \mathcal{D}_\psi(t_i)$;*
   - *$G\psi$ avec pour domaine de validité $\mathcal{D}_{G\psi}(t_i) = \mathcal{D}_{G\psi}(t_{i+1}) \cap \mathcal{D}_\psi(t_i)$;*
   - *$\psi_1 U \psi_2$ avec pour domaine de validité $\mathcal{D}_{\psi_1 U \psi_2}(t_i) = \mathcal{D}_{\psi_2}(t_i) \cup (\mathcal{D}_{\psi_1 U \psi_2}(t_{i+1}) \cap \mathcal{D}_{\psi_1}(t_i))$;*

- *$X\psi$ avec pour domaine de validité $\mathcal{D}_{X\psi}(t_i) = \mathcal{D}_{\psi}(t_{i+1})$;*
- *$\psi_1 \vee \psi_2$ avec pour domaine de validité $\mathcal{D}_{\psi_1 \vee \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cup \mathcal{D}_{\psi_2}(t_i)$;*
- *$\psi_1 \wedge \psi_2$ avec pour domaine de validité $\mathcal{D}_{\psi_1 \wedge \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cap \mathcal{D}_{\psi_2}(t_i)$;*

3. *retourner en résultat le domaine $\mathcal{D}_{\phi}(t_1)$*

3 Correction

Enoncer et prouver la correction de l'algorithme.

On montre que l'algorithme est correct et complet dans le sens qu'une valuation $\vec{v}$ rend ϕ vraie au temps t_i $T, t_i \models_{LTL} (\phi(\vec{v}))$, si et seulement si $\vec{v}$ est dans le domaine calculé de ϕ en t_i, $\vec{v} \in \mathcal{D}_{\phi}(t_i)$.

On montre par induction sur la structure de la formule LTL avec contraintes que pour tout temps t_i, pour toute formule LTL ϕ et pour toute instanciation $\vec{v}$ des variables

si $\phi(\vec{v}, t_i)$ est vraie 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 vraie.

- *les contraintes atomiques considérées sont de la forme $Value\ \mathcal{R}\ Variable$ ou $Value\ \mathcal{R}\ Value$ où $Value$ est une valeur réelle et $\mathcal{R}$ est un opérateur de comparaison. Pour ces contraintes, l'algorithme retourne le domaine exact de validité de la formule.*
- *$\phi_1 \wedge \phi_2$. On a $\mathcal{D}_{\phi_1 \wedge \phi_2}(t_i) = \mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i)$ donc : $\vec{v} \in \mathcal{D}_{\phi_1 \wedge \phi_2}(t_i) \Leftrightarrow \vec{v} \in \mathcal{D}_{\phi_1}(t_i) \wedge \vec{v} \in \mathcal{D}_{\phi_2}(t_i) \Leftrightarrow \phi_1(\vec{v}, t_i) \wedge \phi_2(\vec{v}, t_i) \Leftrightarrow (\phi_1 \wedge \phi_2)(\vec{v}, t_i)$;*
- *$\phi_1 \vee \phi_2$. On a $\mathcal{D}_{\phi_1 \vee \phi_2}(t_i) = \mathcal{D}_{\phi_1}(t_i) \cup \mathcal{D}_{\phi_2}(t_i)$ donc : $\vec{v} \in \mathcal{D}_{\phi_1 \vee \phi_2}(t_i) \Leftrightarrow \vec{v} \in \mathcal{D}_{\phi_1}(t_i) \vee \vec{v} \in \mathcal{D}_{\phi_2}(t_i) \Leftrightarrow \phi_1(\vec{v}, t_i) \vee \phi_2(\vec{v}, t_i) \Leftrightarrow (\phi_1 \vee \phi_2)(\vec{v}, t_i)$;*
- *$F(\phi)$. On a $\mathcal{D}_{F(\phi)}(t_i) = \bigcup_{j \geq i}(\mathcal{D}_{\phi}(t_j))$ donc : $\vec{v} \in \mathcal{D}_{F(\phi)}(t_i) \Leftrightarrow \exists j \geq i, \vec{v} \in \mathcal{D}_{\phi}(t_j) \Leftrightarrow F(\phi)(\vec{v}, t_i)$;*
- *$G(\phi)$. On a $\mathcal{D}_{F(\phi)}(t_i) = \bigcap_{j \geq i}(\mathcal{D}_{\phi}(t_j))$ donc : $\vec{v} \in \mathcal{D}_{G(\phi)}(t_i) \Leftrightarrow \forall j \geq i, \vec{v} \in \mathcal{D}_{\phi}(t_j) \Leftrightarrow G(\phi)(\vec{v}, t_i)$;*
- *$X(\phi)$. On a $\mathcal{D}_{X(\phi)}(t_i) = \mathcal{D}_{\phi}(t_{i+1})$ donc :*
  $\vec{v} \in \mathcal{D}_{X(\phi)}(t_i) \Leftrightarrow \vec{v} \in \mathcal{D}_{\phi}(t_{i+1}) \Leftrightarrow X(\phi)(\vec{v}, t_i)$;
- *$\phi_1 U \phi_2$. $\phi_1 U \phi_2(t_i)$ est vrai si et seulement si $\phi_2(t_i)$ est vrai ou bien $\phi_1(t_i)$ est vrai et $(\phi_1 U \phi_2)(t_{i+1})$ est vrai. On a par construction $\mathcal{D}_{(\phi_1 U \phi_2)}(t_i) = \mathcal{D}_{\phi_2}(t_i) \cup (\mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i))$ donc : $\vec{v} \in \mathcal{D}_{(\phi_1 U \phi_2)}(t_i) \Leftrightarrow \vec{v} \in \mathcal{D}_{\phi_2}(t_i) \cup (\mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i)) \Leftrightarrow (\phi_1 U \phi_2)(\vec{v}, t_i)$.*

4 Complexité

Evaluer la complexité de l'algorithme.

We first evaluate the size of the representation of the variable domains of the output. Let us define a box of $\mathbb{R}^n$ as a finite intersection of half-spaces and the size $S(\mathcal{D})$ of a domain as the minimum number of boxes the domain is made of. Note that for a one dimension domain $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ (resp. $\mathcal{D} = \mathcal{D}_1 \cap \mathcal{D}_2$), the maximum size is $S(\mathcal{D}_1) + S(\mathcal{D}_2)$ (resp. $\max(S(\mathcal{D}_1), S(\mathcal{D}_2))$).

In the worst case, the size of the validity domain of a LTL formula of size k on a trace of length n is n^{k^2}.

Let us prove inductively that the size of the projection on one variable of the validity domain (i.e., the validity domain of a single variable) of a LTL formula of size k, is at most n^k. The size of the validity

domain of an atomic formula is at most 1. The maximum size of a one dimension domain of a formula of size k is:

$$\begin{aligned}
\max(S(\mathcal{D}_{\phi_1}(t_i)), S(\mathcal{D}_{\phi_2}(t_i))) \; &for \; \mathcal{D}_{\phi_1 \; and \; \phi_2}(t_i) \\
S(\mathcal{D}_{\phi_1}(t_i)) + S(\mathcal{D}_{\phi_2}(t_i)) \; &for \; \mathcal{D}_{\phi_1 \; or \; \phi_2}(t_i) \\
\sum_{j \geq i} S(\mathcal{D}_{\phi}(t_j)) \; &for \; \mathcal{D}_{F(\phi)(t_i)} \\
\max_{j \geq i} S(\mathcal{D}_{\phi}(t_j)) \; &for \; \mathcal{D}_{G(\phi)(t_i)} \\
S(\mathcal{D}_{\phi}(t_{i+1})) \; &for \; D_{X(\phi)}(t_i) \\
\max(S(\mathcal{D}_{\phi_1 U \phi_2}(t_{i+1})), S(\mathcal{D}_{\phi_1}(t_i))) + S(\mathcal{D}_{\phi_2}(t_i)) \; &for \; D_{\phi_1 U \phi_2}(t_i)
\end{aligned}$$

In all these cases except operators F an U, the size of the domain is less than the sum of the domains' size of the subformulae at one time point, which entails a size smaller than $2n^{k-1}$. The U and F operators make a sum on all time points which entails a size of at most $n \times n^{k-1} = n^k$. Each projection's size of the validity domain is thus at most n^k. The size of the validity domain of a formula containing v variables is at most $(n^k)^v \leq (n^k)^k = n^{k^2}$.

The algorithm thus computes for each subformula and each time point a validity domain of size at most n^{k^2}.