Logique temporelle et vérification de modèles

Gilles Bernot

La.M.I. UMR 8042
Université d'Évry & CNRS / Genopole®
Boulevard François Mitterrand
F-91025 Évry CEDEX

bernot@lami.univ-evry.fr

Résumé : On montre ici comment une logique formelle dite temporelle peut être techniquement employée en conjonction avec les réseaux de régulation biologiques définis par René Thomas pour valider les liens entre modélisation et expérimentation en biologie. Après un rappel de notre méthode de travail pluridisciplinaire biologie/informatique, l'objet de ce cours est de décrire rapidement la modélisation discrète des réseaux de régulation biologiques, la logique temporelle CTL et le « Model-Checking ».

1 Expérimenter & observer des formules plutôt que des modèles

1.1 Le contexte

L'atelier *Observabilité* de Genopole®-Évry cherche à développer des méthodes de modélisation à la croisée de l'informatique et de la biologie permettant de faciliter la validation expérimentale des modèles proposés. Compte-tenu des conditions expérimentales en biologie d'une part, et de la complexité des comportements à modéliser d'autre part, presque tous les modèles pour la biologie contiennent des paramètres qui ne sont pas directement mesurables expérimentalement. Pire, compte tenu des capacités expérimentales à une période donnée, il n'est pas possible de déterminer des valeurs uniques de ces paramètres même par des détours et des déductions sophistiquées. Ces paramètres « cachés et incertains » sont pourtant nécessaires de manière intrinsèque à la construction du modèle afin d'obtenir des comportements crédibles. Par conséquent, pour modéliser un phénomène donné, toute une classe de modèles distincts et non équivalents sont indistinguables expérimentalement à l'heure actuelle.

Cette constatation conduit à l'idée qu'on ne valide pas expérimentalement *un* modèle mais une classe de modèles partageant certaines propriétés comportementales. L'atelier *Observabilité* aborde alors cette question sous l'angle des propriétés elles mêmes : on se donne les moyens d'énoncer les propriétés (donc les hypothèses biologiques) partagées par une classe de modèles et l'on cherche à les valider, un modèle n'étant alors plus qu'un représentant parmi d'autres de ces propriétés. Cela permet d'aborder la validation de modèles pour la biologie en se focalisant essentiellement sur les questions (hypothèses biologiques) d'intérêt.

Dans la mesure où l'on souhaite instrumenter cette démarche par des logiciels, cela conduit à adopter un langage formel manipulable par ordinateur pour énoncer ces propriétés, c'est-à-dire essentiellement adopter une *logique formelle*. Les propriétés à valider constituent alors un ensemble de formules Φ.

Poser cet ensemble de propriétés biologiques Φ comme objet principal du travail ne dispense cependant pas de l'activité de construction de modèles. En effet on se retrouve immédiatement confronté à deux questions :

1. *Est-il possible que* Φ *?*
   Compte-tenu de la complexité des phénomènes étudiés il n'est pas rare en effet que l'union des connaissances biologiques acquises et des hypothèses considérées soit incohérente pour des raisons profondément enfouies dans la combinatoire des conséquences issues de Φ. La cohérence de Φ est un préalable incontournable avant toute expérience « humide » coûteuse. Le seul moyen rationnel et rigoureux d'y répondre est de construire un modèle mathématique crédible qui satisfasse les propriétés Φ. L'objet de ce cours est de montrer comment la science informatique peut presque entièrement automatiser cette activité en s'appuyant sur la logique formelle.
2. *Si oui, est-il vrai in vivo que* Φ *?*
   Ce n'est pas parce qu'on a été capable de construire un modèle cohérent avec les connaissances biologiques du moment qu'il reflète la réalité de l'objet biologique étudié. Il faut donc concevoir des plans d'expériences nouvelles aptes à se convaincre raisonnablement de Φ. La logique formelle permet également d'instrumenter cette activité.

1.2 Construire des modèles crédibles

La première question peut être résolue dans le cas particulier des réseaux de régulation biologiques en utilisant une logique appelée CTL (Computation Tree Logic) et c'est l'objet de ce cours. La section 2 indique comment construire un graphe d'états fini à partir d'un réseau de régulation biologique au sens de René Thomas. La section 3 décrit la logique CTL et indique ce que signifie qu'un graphe d'états satisfait une formule dans cette logique. Enfin la section 4 explique comment le *Model Checking* permet de trouver automatiquement les modèles qui satisfont un ensemble de formules Φ exprimées en CTL.

1.3 Suggérer des expériences

La seconde question est plus difficile et pas toujours réductible à un nombre fini d'expériences à un coût envisageable. Il n'empêche que là aussi une approche s'appuyant sur la logique formelle peut instrumenter de manière logicielle le construction de plans d'expériences optimisés selon par exemple des critères de coût. Ce n'est pas l'objet de ce cours mais on peut donner ci-dessous quelques grandes lignes d'une approche qui est en point de mire de l'atelier *Observabilité*, inspirée des techniques de test de logiciel.

Parmi les formules de CTL, certaines expriment des propriétés qui sont directement « expérimentables » par un plan d'expériences donné. Si l'on note *Obs* l'ensemble de ces formules (que l'on qualifie d'observables), la difficulté de validation de Φ résulte du fait qu'il n'est pas inclus dans *Obs*. Cependant d'un point de vue technique, Φ possède un ensemble de conséquences formelles bien défini ($Th(\Phi)$, l'ensemble des théorèmes de Φ), et Φ est *observationnellement* validée expérimentalement si et seulement si toutes les propriétés de $Th(\Phi) \cap Obs$ sont expérimentalement vraies.

Malheureusement $Th(\Phi) \cap Obs$ contient généralement un nombre infini de formules ; ainsi, répondre à la seconde question, c'est essentiellement extraire de $Th(\Phi) \cap Obs$ un sous-ensemble fini de formules donnant lieu à un ensemble d'expériences de coût raisonnable. D'un point de vue théorique, des algorithmes pour assister ces choix existent. Ils sont en particulier utilisés pour tester si de grand logiciels répondent à leurs spécifications, encore faut-il les adapter au domaine de la biologie.

2 Réseaux de régulation et graphes d'états

Dans toute la suite on se consacre à la première question, c'est-à-dire comment, techniquement, trouver automatiquement les modèles sur un graphe de régulation donné qui satisfont un ensemble de propriétés données.

Un graphe de régulation (cf. cours de Janine Guespin) est un graphe dont les nœuds sont des *variables* qui représentent abstraitement des gènes, leur produit, ou des macromolécules en général. On associe à chaque variable x un *niveau de concentration* n_x (appartenant à $\mathbb{N}$ dans le cas discret qui nous occupe ici) variable dans le temps et la connaissance de ces niveaux de concentration pour l'ensemble des variables à un instant donné constitue un *état du système*. Les arcs du graphe représentent les influences entre variables et peuvent être vus comme des *transitions*. Une transition $x \to y$ peut représenter une influence positive si x est un inducteur de y, et négative si x est un répresseur de y. Par ailleurs ces influences n'agissent qu'à partir d'un niveau de concentration seuil de l'inducteur ou du répresseur ; ainsi une transition est étiquetée par un couple (s, ε) où $s \in \mathbb{N}$ est le seuil et $\varepsilon \in \{+, -\}$ est le signe positif ou négatif de la transition.

Supposons que la variable y reçoive des transitions à partir des variables $x_1 \cdots x_k$ dans le graphe de régulation. À un instant donné, la variable y va tendre vers un niveau de concentration qui est entièrement déterminé par l'ensemble des x_i qui passent ou non leur seuil. On note par convention $k_{y,\omega}$ ce niveau de concentration « attracteur » de y où ω est le sous-ensemble de $\{x_1, \cdots, x_k\}$ formé d'une part, des inducteurs de y qui passent leur seuil et d'autre part, des répresseurs de y qui ne passent pas leur seuil. ω est appelé l'ensemble des *ressources* de y à un instant donné.

On appelle *réseau de régulation* un graphe de régulation muni de la connaissance de tous les paramètres $k_{\ldots}$ précédents. En pratique, le graphe de régulation peut être obtenu de manière relativement fiable par les connaissances biologiques mais en revanche les paramètres $k_{\ldots}$ sont généralement non observables expérimentalement, et constituent de fait l'élément déterminant pour ajuster le comportement du modèle avec le comportement *in vivo* ou prédire de nouveaux comportements.

De manière classique en informatique, le comportement du modèle est caractérisé par un *graphe d'états*. Les nœuds de ce graphe sont tous les états possibles du système. Ici, cet ensemble est fini car chaque variable x possède une borne supérieure de ses niveaux de concentration possibles $b_x \in \mathbb{N}$. Par conséquent si les variables du réseau de régulation sont $\{x_1, \cdots, x_p\}$ alors les nœuds du graphe d'états peuvent être rangés dans une grille finie de dimension p : $[0, b_{x_1}] \times [0, b_{x_2}] \times \cdots \times [0, b_{x_p}]$. Le temps est « découpé en instants successifs » et un arc du graphe d'états $(n_{x_1}, \cdots, n_{x_p}) \to (m_{x_1}, \cdots, m_{x_p})$ indique que si à un instant donné chaque variable x_i possède le niveau de concentration n_{x_i}, alors à l'instant d'après ce niveau sera m_{x_i}.

On construit les arcs du graphe d'états en deux phases : on construit d'abord un graphe d'états dit *synchrone* puis on le « désynchronise » pour obtenir le graphe d'états *asynchrone* qui définit la sémantique du réseau de régulation.

Étant donné un état $(n_{x_1}, \cdots, n_{x_p})$, il détermine de manière unique pour chaque variable x_i l'ensemble de ses ressources ω_i (selon les prédécesseurs de x_i qui passent ou non leur seuil). Le graphe d'états synchrone possède un unique arc sortant de chaque état $(n_{x_1}, \cdots, n_{x_p})$ directement vers l'état « attracteur » $(k_{x_1,\omega_1}, \cdots, k_{x_p,\omega_p})$.

Pour comprendre comment est construit le graphe d'états asynchrone, supposons par exemple que $p = 2$. On peut aisément imaginer une configuration où l'état $(0, 0)$ (i.e. $x_1 = 0$ et $x_2 = 0$)

possède pour attracteur l'état $(0,3)$ et le graphe d'état synchrone traduit alors une augmentation brutale de x_2 de plusieurs niveaux de concentration d'un coup. Ceci n'est pas crédible *in vivo* où l'augmentation est nécessairement continue et progressive : pour aller de 0 à 3, le niveau de concentration passe nécessairement une période significative au niveau 1 d'abord, et le comportement du système sera ensuite celui de l'état $(0,1)$. Ainsi, pour construire le graphe d'états asynchrone, on « écourte » tout arc du graphe synchrone à la longueur 1. $(0,0) \to (0,3)$ est donc remplacé par $(0,0) \to (0,1)$ dans le graphe asynchrone. Imaginons maintenant que l'arc $(0,0) \to (1,3)$ soit dans le graphe synchrone : il est en fait improbable *in vivo* que les vitesses de changement de niveau de x_1 et x_2 soient identiques chez un individu et certains choisiront le comportement de $(0,0) \to (1,0)$ alors que d'autres choisiront celui de $(0,0) \to (0,1)$ (dans des proportions sans doute déséquilibrées dans une population d'individus). Ainsi, pour construire le graphe d'états asynchrone, on « désynchronise » les variables et ceci donne donc lieu à un choix non déterministe de l'état suivant ($(0,1)$ ou $(1,0)$ dans notre exemple).

Partant d'un état initial donné dans un réseau de régulation biologique, une succession des états ultérieurs forme un chemin dans le graphe des états asynchrone, appelé une *trace* du réseau de régulation. Une trace est donc une succession d'états telle que d'un état à l'autre, une variable au plus change son niveau de concentration d'une unité et en avançant le long d'une trace on peut éventuellement choisir la variable en question (aux points de désynchronisation de variables). L'ensemble des traces partant d'un état initial donné possède donc une structure arborescente (l'état initial est la racine et les désynchronisations donnent lieu à autant de branches que de variables concernées).

3 Les formules de la logique temporelle CTL

CTL (Computation Tree Logic) offre un langage formel pour exprimer des propriétés à propos de l'ensemble des traces d'un graphe d'états et présente l'avantage que l'on peut entièrement vérifier algorithmiquement (par « *Model Checking* ») si une formule est vraie ou fausse sur un graphe d'états donné.

En CTL, deux « dimensions » sont à prendre en considération sur les ensembles de traces : l'aspect non déterministe des choix des branches en chaque état (issus des désynchronisations), et l'aspect linéaire d'une trace (i.e. les propriétés concernant la succession des états le long d'un chemin).

- Sur la première dimension, à partir d'un état initial donné, on peut demander qu'une propriété soit vraie pour au moins un choix de branche (« *there **E**xists a path such that ...* ») ou bien qu'elle soit vraie pour tous les choix, c'est-à-dire tous les futurs possibles (« *for **A**ll possible pathes ...* »).
- Sur la seconde dimension, partant d'un état initial le long d'un chemin donné, on peut demander qu'une propriété soit vraie : à l'instant juste après l'état initial (« *ne**X**t state* »), au moins à un instant donné sur le chemin (« *some **F**uture state* »), tout le temps (« ***G**lobally* ») ou jusqu'à ce qu'une autre propriété devienne vraie (« ... ***U**ntil ...* »).

On obtient les modalités temporelles de CTL en accolant ces deux dimensions. Par exemple $AX(\varphi)$ signifie que φ est vraie pour tous les états qui suivent immédiatement l'état initial ; $EG(\varphi)$ signifie qu'il existe au moins un chemin partant de l'état initial le long duquel φ reste tout le temps vraie ; $A[\varphi U \psi]$ signifie que pour tous les chemins partant de l'état initial, φ reste vraie jusqu'à ce que ψ devienne vraie à un instant donné ; etc.

Enfin on dispose des connecteurs logiques habituels pour relier entre elles les modalités précédentes : la négation ($\neg\varphi$), la conjonction ($\varphi \wedge \psi$), la disjonction ($\varphi \vee \psi$), l'implication ($\varphi \Longrightarrow \psi$), etc.

Dans notre cas, les formules atomiques qui nous intéressent sont liées aux niveaux de concentration des variables. On peut par exemple écrire :

$$(x = 0) \implies AG(\neg(x = 2))$$

(*si x a pour niveau de concentration 0 à l'état initial, alors il n'atteindra jamais, quel que soit le chemin suivi, un état où son niveau vaut 2*, une propriété liée en fait aux « bassins d'attraction »).

Si tous les états du graphe d'états asynchrone d'un réseau de régulation biologique $\mathcal{R}$ satisfont un ensemble de formules Φ, on dira que $\mathcal{R}$ satisfait Φ.

4 Confronter les modèles aux formules : le *Model Checking*

Étant donnés un graphe d'état et une formule en CTL, un outil de *Model Checking* est capable de fournir l'ensemble des états du graphe d'états qui satisfont la formule, avec des performances algorithmiques remarquables. Intuitivement, les algorithmes de *Model Checking* traitent les formules en partant de leurs atomes ($x = 0$ ou $x = 2$ dans l'exemple précédent) et étiquètent les états du graphe qui les satisfont ; ce n'est pas difficile puisqu'un état est précisément déterminé par la connaissance des niveaux de concentration des variables. Ensuite on « remonte » les connecteurs et modalités apparaissant dans la formule. Pour cela, certaines transformations de la formule peuvent éventuellement être utiles. Par exemple il est *a priori* difficile de « remonter » une modalité AG puisqu'il faudrait parcourir tous les futur possibles d'un état avant de l'étiqueter par $AG(\cdots)$, cependant on remarque que, par exemple, $AG(\neg(x = 2))$ équivaut à $\neg EF(x = 2)$. Dès lors, tout devient plus facile :

- tous les états étiquetés par $x = 2$ peuvent être étiquetés par $EF(x = 2)$ (car en CTL « le futur inclut le présent »)
- si un état est étiqueté par $EF(x = 2)$, alors on peut étiqueter tous ses prédécesseurs par $EF(x = 2)$
- on arrête l'étiquetage lorsque tous les prédécesseurs des états rencontrés lors de ces parcours arrières du graphe d'états ont déjà été visités.

Pour clore l'exemple, tous les états qui ne sont pas étiquetés par $x = 0$ sont étiquetés par $(x = 0) \implies \neg EF(x = 2)$ (car en logique « faux implique n'importe quoi ») et ceux qui sont étiqueté à la fois par $x = 0$ et par $\neg EF(x = 2)$ sont également étiquetés par $(x = 0) \implies \neg EF(x = 2)$.

En fait, les algorithmes performants de Model Checking ne parcourent pas tout le graphe d'états, grâce à des techniques de BDD (Binary Decision Diagrams) qui permettent de représenter l'ensemble des états sous forme d'arbres binaires « de décision » dans lesquelles certaines branches permettent de regrouper d'un coup des sous-ensembles entiers d'états ayant la même véracité en regard d'une formule donnée...

On peut utiliser cette puissance algorithmique pour déterminer une classe de réseaux de régulation biologiques « crédibles » et leurs paramètres « acceptables ». Nous avons vu en effet que pour modéliser un système biologique par un réseau de régulation, après avoir proposé un ou plusieurs graphes de régulation possibles, la partie difficile est de déterminer les ensembles de paramètres $k...$ cohérents avec les connaissances et les hypothèses biologiques considérées. Si l'on exprime ces propriétés en CTL, il suffit d'engendrer automatiquement toute la combinatoire des paramètres sur ces graphes de régulation, de construire tous les graphes d'états engendrés *via* ces $k...$, et enfin de trier, par Model Checking, les modèles dont tous les états satisfont les propriétés.

Grâce à ce couplage de CTL et des réseaux de régulation biologiques, le Model-Checking est un outil de « force brute » pour résoudre la première question mentionnée plus haut.