Université d'Évry-Val d'Essonne
U.F.R. Sciences Fondamentales et Appliquées

THÈSE

présentée pour obtenir

le grade de Docteur en Sciences
de l'Université d'Évry-Val d'Essonne

Spécialité
bioinformatique

Sylvie Troncale

Modélisation et vérification des réseaux de Petri hybrides temporisés.

- Application à la métamorphose amphibienne -

Soutenue le *12 septembre 2008* devant le jury composé de

M. Gilles Bernot	*Directeur de thèse*
M. Jean-Paul Comet	*Directeur de thèse*
M. Alexander Bockmayr	*Rapporteur*
M. François Fages	*Rapporteur*
M. Bruno Bost	*Examinateur*
M. Nicolas Pollet	*Examinateur*
M. Philippe Tracqui	*Examinateur*

Thèse préparée au sein du laboratoire IBISC
et du Programme d'Epigénomique
CNRS / Génopole /Université d'Évry-Val d'Essonne

Ce qui ne te tue pas
te rend plus fort.

Remerciements

Je tiens à remercier les membres du jury. Mr Philippe Tracqui qui m'a fait l'honneur de présider ce jury. Mr François Fages et Mr Alexander Bockmayr d'avoir rapporté ma thèse avec autant de minutie, merci pour vos questions et vos remarques pertinentes. Merci Mr Bruno Bost, qui a su examiner ma thèse avec son œil critique de biologiste.

Je remercie Gilles Bernot pour avoir dirigé cette thèse. Merci à Jean-Paul Comet pour son encadrement, pour sa rapidité et sa vivacité à corriger mon manuscrit et surtout pour son soutien et son appui auprès du CEA.

Un grand merci à Nicolas Pollet dont le soutien a été primordial dans ma thèse. Merci pour notre riche collaboration, pour les retours et remarques pertinentes que vous m'avez apportées. Merci d'avoir toujours été présent et motivé pour que notre travail en commun avance.

Un merci tout particulier à Christian Laforest, qui a pris son rôle de tuteur de thèse très au sérieux. Bien que nos thématiques de recherche soient à l'opposé de l'informatique, tu as toujours été présent, à l'écoute et à la recherche de solutions.

Merci à IBISC de m'avoir accueillie durant ces trois ans. Merci aux thésards d'IBISC pour l'ambiance et le soutien que vous m'avez apportés : merci Stéphane pour nos matinées café, merci François pour avoir enrichi ma culture cinématographique, merci Laurent, Matthieu, Delphine, David, et tous les autres que je ne cite pas mais qui ont compté pour moi. Merci à Chafika, une co-bureau tip-top, merci pour nos longues discussions et pour ton soutien quotidien et merci à Stéfan grâce à qui je me suis lancé dans cette grande aventure qu'est la thèse.

Merci à tous mes amis sur qui j'ai toujours pu compter. Vous avez égayé et j espère que vous égayerez encore mes soirées et mes week-ends : Delphine, Grégory, Fanny, Yohan, Myriam, Mathieu, Gisèle, Jean-Marc, Séverine, François, Nicolas, Franck, Claudia et tous les autres...

Merci à mes parents pour m'avoir soutenue et je sais que ça n'a pas toujours été facile. Merci à ma sœur, qui m'a suivie et soutenue, merci pour ton enthousiasme et ta présence. Merci Médéric pour ton humour et ta bonne humeur. Merci à mon petit Adrien pour apporter autant de bonheur. Merci à ma belle-famille pour son soutien et son aide : merci Mme et Mr Job, merci Edouard, Adeline, Louis, Amélie et la petite Gabrielle.

Et enfin, je voudrais remercier la personne sans qui cette thèse n'aurait eu aucun goût. Merci Henri pour avoir toujours cru en moi, pour avoir été autant à l'écoute, pour m'avoir autant soutenue et je sais que ce n'était pas toujours facile mais tu as toujours été là pour moi et pour ça je te remercie.

Table des matières

Introduction **13**

1 Avantages et limites des HFPN **19**
1.1 Les Réseaux de Petri pour la biologie 19
1.2 Les Réseaux de Petri Hybrides Fonctionnels 20
1.2.1 Définition 21
1.2.2 Illustration biologique 23
1.3 Restriction des Réseaux de Petri Hybrides Fonctionnels pour la vérification 24
1.4 Les Réseaux de Petri Hybrides Temporisés (THPN) 26
1.4.1 Définitions 26
1.4.2 Graphe d'évolution 31
1.4.3 Illustration biologique 38
1.4.4 Résolutions de conflits par priorité en biologie 40
1.5 Démarche adoptée pour la vérification 40

2 Une logique pour les modèles temps réel **43**
2.1 Modèles en temps continu 43
2.2 Traces Discrètes 46
2.3 Application aux Réseaux de Petri Hybrides Temporisés 50

3 Model-checking pour THPN **55**
3.1 Démarche générale 55
3.2 Des graphes d'évolution aux automates «Event-Clock» 56
3.2.1 Automates «Event-Clock» 56
3.2.2 Algorithme de conversion 58
3.2.3 Illustration biologique 60
3.2.4 Propriétés 61
3.3 De la propriété aux automates «Event-Clock» 64
3.3.1 Algorithme de conversion 64
3.3.2 Définition et propriétés 66
3.3.3 Illustration biologique 68
3.4 Un produit avec transformations des étiquettes 68
3.4.1 Produit-LT 69
3.4.2 Illustration biologique 71
3.4.3 Détermination du vide 72
3.5 Correction et complétude de la procédure 79

3.5.1 Théorèmes finaux . . . 79
3.5.2 Illustration biologique . . . 80
3.6 Résumé . . . 80

4 La Plate-forme logicielle : BioPetri 85
4.1 Description du logiciel BioPetri . . . 85
4.2 Implémentation du logiciel BioPetri . . . 85
4.2.1 Construction du graphe d'évolution . . . 86
4.2.2 Model-Checker pour automates Event-Clock . . . 86
4.3 Développement sur un exemple . . . 87
4.3.1 Construction du graphe d'évolution . . . 87
4.3.2 Model-Checker pour automates Event-Clock . . . 88

5 Application à la métamorphose amphibienne 91
5.1 Contexte biologique . . . 91
5.1.1 Le Xénope : modèle étudié . . . 91
5.1.2 Métamorphose . . . 92
5.1.3 Les hormones thyroïdiennes . . . 92
5.1.4 Problématique . . . 96
5.2 Compétition enzymatique . . . 97
5.2.1 Rappels . . . 97
5.2.2 Discussion . . . 98
5.3 Modèles apoptose *versus* prolifération . . . 99
5.3.1 Construction des modèles . . . 100
5.3.2 Propriétés . . . 102
5.3.3 Etude *in silico* du rôle des déiodinases . . . 104
5.3.4 Etude *in silico* de la nature des hormones thyroïdiennes . . . 106
5.3.5 Vers un Xénope partiel *in silico* . . . 107
5.4 Confrontation de modèles à des données transcriptionnelles . . . 108
5.4.1 Contexte biologique . . . 108
5.4.2 Construction des modèles . . . 109
5.4.3 Analyse biologique des modèles . . . 112

6 Extension paramétrique 117
6.1 Motivation . . . 117
6.2 Cadre paramétrique . . . 118
6.3 Graphe d'évolution symbolique . . . 120
6.3.1 Obtention des IB-states . . . 121
6.3.2 Obtention des transitions . . . 124
6.3.3 Algorithme . . . 126
6.3.4 Traitement des contraintes . . . 127
6.3.5 Exemples . . . 128
6.3.6 Propriétés du graphe d'évolution symbolique . . . 132
6.4 Model-checking paramétrique . . . 134
6.4.1 Automates Event-Clock paramétriques . . . 135
6.4.2 Conversion du graphe d'évolution symbolique en automate . . . 137

6.4.3 Transformation en un sp-ECA 139
6.4.4 Construction de l'automate associé à la propriété 141
6.4.5 Produit des deux automates 141
6.4.6 Détermination du vide 142
6.5 Propriétés 148
6.6 Bilan 153

Conclusion **155**

Introduction

La biologie des systèmes au niveau cellulaire vise à comprendre les systèmes biologiques prenant en compte les processus moléculaires, les transductions de signal, les interactions entre compartiments intra-cellulaires,... L'avancée des méthodes de génomique et de post-génomique a engendré, ces dernières années, une avalanche d'informations concernant les interactions moléculaires. Une exploitation possible de ces données consistent à abstraire les détails pour faire un modèle qui intègre un maximum de connaissances. Cette abstraction peut se faire par l'élaboration de modèles mathématiques dont le but est de *comprendre* et de *prédire* les comportements globaux [1, 2, 3, 4].

A cause des cycles de rétroaction dans les chaînes d'interactions, l'intuition ou les simples déductions de bon sens fondées sur des connaissances, nécessairement partielles, sont très souvent de mauvaises lignes directrices. Entre autres, la connaissance d'une double interaction, positive et négative, sur une même cible ne permet pas de déduire intuitivement le devenir de la cible en question et ceci se complexifie avec les cycles qui peuvent engendrer soit la multistationarité soit l'homéostasie [4, 5]. Or, avant de faire des prédictions sur un modèle, il faut s'assurer que le modèle abstrait suit bien les comportements connus. Des simulations automatisées, exécutées à partir de ce modèle sont alors indispensables pour comprendre de façon précise le fonctionnement global. Néanmoins, celles-ci peuvent s'avérer insuffisantes pour établir formellement des prédictions pertinentes. Ceci amène à distinguer deux catégories de formalismes de modélisation :

- ceux qui sont assez puissants pour encoder précisément les connaissances biologiques, au détriment de toutes applications de preuves et de vérification, c'est-à-dire qu'il n'est pas possible de vérifier formellement qu'un modèle satisfait une ou des propriétés biologiques données. Ces formalismes engendrent même parfois des simulations difficiles à mettre en œuvre, c'est le cas, par exemple des équations différentielles [6, 7, 8] ;
- ceux qui permettent d'exécuter à la fois des simulations et des preuves, mais ceux-ci obligent à faire d'importantes "simplifications", plus exactement des abstractions compatibles avec les propriétés étudiées. C'est le cas dans la modélisation des réseaux métaboliques avec des modes élémentaires [9, 10, 11], ou dans la modélisation de réseaux de gènes avec l'application des méthodes de model-checking, *i.e* de vérification de propriétés sur un modèle [12, 4, 13].

Les réseaux de Petri hybrides fonctionnels, appelés HFPN pour Hybrid Functional Petri Nets [14, 15, 16, 17] constituent actuellement un des formalismes les plus puissants pour la biologie des systèmes, parmi ceux fournissant des simulations totalement automatisées. Ils offrent un maximum de flexibilité : modélisation de la production et de la

consommation de ressources, modélisation de processus discrets et continus et définition de la quantité consommée ou produite sous forme de fonctions de l'état du système. Cette classe étendue des réseaux de Petri [18] est appropriée pour la simulation dans le contexte de la biologie des systèmes. De nombreux systèmes complexes ont déjà été modélisés en utilisant ce formalisme [17, 19, 20, 21].

Lors de l'étude de modèles biologiques, nous avons fréquemment besoin de valider des propriétés spécifiques. Ces propriétés peuvent être des hypothèses que les biologistes souhaitent établir afin d'approfondir leur recherche, il peut également s'agir de comportements récurrents observés *via* de nombreuses simulations et que nous souhaitons prouver comme invariants de notre système. Dans les deux cas, les chaînes de causalité sont si complexes que l'accumulation de simulations ne permettent pas d'obtenir une preuve. Des preuves formelles établies ou vérifiées par un ordinateur deviennent donc nécessaires.

Les logiques temporelles utilisant des automates discrets ou hybrides jouent un rôle central dans les formalismes de modélisation des systèmes dynamiques où les preuves sont automatisables. Les réseaux de régulation géniques avec des niveaux d'expression discrets correspondent au formalisme de modélisation le plus avancé dans l'automatisation des preuves [4, 22, 23, 24, 25, 26, 27], et l'école de René Thomas a été le point clé de ce succès [28, 29, 30, 31, 32, 33].

La motivation à long terme de la recherche décrite dans ce manuscrit est d'étendre au formalisme des HFPN la méthodologie de vérification formelle associée au formalisme de René Thomas. Cette extension est en fait quasi-impossible sur un formalisme aussi expressif que les HFPN : l'expression quantitative du temps ainsi que les fonctions offrent un tel pouvoir d'expression qu'ils sont les principaux obstacles à l'exécution de preuves formelles.

Dans ce manuscrit, nous définissons une procédure de preuves automatisée pour une sous-classe des HFPN exprimant quantitativement le temps, mais sans fonctions. Cette sous-classe est appelée réseau de Petri hybrides temporisés [34, 35] et notée THPN pour Timed Hybrid Petri Nets.

Etant donné que les processus modélisés dans les modèles HFPN ou THPN sont souvent continus et quantitatifs, les procédures classiques de model-checking, comme celle proposée par l'approche René Thomas [4, 12], ne semblent pas adaptées. Nous nous sommes donc orientés vers des procédures de model-checking temps-réel, et plus spécifiquement sur celle développée par Raskin et Schobbens dans [36]. Cette procédure est basée sur la logique temps-réel Event-Clock ainsi que sur le produit d'automates temps-réel Event-Clock. Le travail décrit dans ce manuscrit introduit une extension de la logique Event-Clock ainsi qu'une conversion compatible de nos modèles THPN en automate Event-Clock. Cette conversion en automate permet alors d'exécuter des raisonnements automatisés sur ces modèles.

Organisation du document

Ce mémoire s'organise en six chapitres :

- Le Chapitre 1 présente le formalisme des HFPN, il détaille la sémantique et met en avant les avantages et inconvénients d'un formalisme aussi expressif. Ce chapitre

présente également la sous-classe des HFPN, les réseaux de Petri hybrides temporisés, notés THPN et détaille l'algorithme de construction d'un graphe d'évolution à partir d'un modèle. Ce graphe d'évolution récapitule les traces obtenues lors de l'exécution du réseau de Petri. Ce chapitre rappelle également toutes les notions indispensables attachées aux THPN.

- Le Chapitre 2 définit la syntaxe de notre logique temporelle permettant d'exprimer des propriétés biologiques (*i.e.* l'ensemble des formules bien formées), ainsi que la sémantique associée (*i.e.* l'ensemble des modèles associés à la signature et les règles de satisfaction d'une formule bien formée). Cette logique, que nous avons appelée CTEL pour Continuous-Time Evolution Logic en anglais, utilise des nombres réels mesurant le temps. Nous verrons que CTEL correspond à un cadre logique suffisamment expressif dans de nombreux cas biologiques.

- Le Chapitre 3 présente la procédure complète de Model-Checking permettant de prouver formellement qu'une formule CTEL bien formée est satisfaite par un modèle THPN donné. Notre procédure se divise en quatre grandes étapes.

  1. La première étape de notre procédure consiste à convertir les traces de notre modèle THPN, synthétisées sous forme de graphes d'évolution, en un automate temps-réel : un automate Event-Clock [37]. Nous présentons donc un algorithme de conversion d'un graphe d'évolution en un automate Event-Clock. Le langage de l'automate ainsi obtenu correspond à l'ensemble des traces des graphes d'évolution.
  2. La deuxième étape consiste à construire l'automate Event-Clock associé à la négation de la propriété étudiée, exprimée sous forme de formule CTEL. Le langage de cet automate correspond alors à l'ensemble des traces satisfaisant la négation de la propriété. La méthode permettant la construction d'un tel automate à partir d'une formule logique est rappelée dans ce chapitre.
  3. Nous définissons alors un produit d'automates permettant de transformer les étiquettes des arcs. On l'appelle le produit-LT, pour Label Transformation. L'automate résultant du produit-LT de l'automate associé au THPN avec celui associé à la négation de la propriété reconnaît le langage à l'intersection des langages des deux automates de départ.
  4. La dernière étape consiste alors à déterminer si le langage de l'automate produit est vide ou non. En effet, étant donnée la prise en compte de temps réel dans nos modèles, la détermination du vide d'un automate n'est pas immédiate. La procédure de détermination du vide est fondée sur la construction de l'automate de région défini, pour la première fois, par Alur pour les automates temporisés dans [38]. La définition d'automate de région a alors été étendue aux automates Event-Clock par Rakin et Schobbens dans [36]. L'automate de région construit suivant cette procédure, reconnaît exactement le langage atemporisé de l'automate produit.

     Ainsi, dans le cas où le langage de l'automate produit n'est pas vide, on conclut que le modèle ne satisfait pas la propriété étudiée étant donné qu'il partage

au moins une trace commune avec la négation de cette propriété. Par contre, on conclut à la satisfaction de la propriété par notre modèle si le langage de l'automate produit est vide.

- Etant donné l'objectif d'automatisation annoncé ci-dessus, la procédure de model-checking définie dans le chapitre précédent est entièrement algorithmique et peut être automatisée. Le Chapitre 4 décrit notre prototype de plate-forme logicielle que nous avons appelée *BioPetri*. Le manuel d'utilisation est présenté en détail dans ce chapitre.

- Dans le Chapitre 5, nous appliquons notre procédure de model-checking sur une problématique biologique. En collaboration avec le laboratoire de Développement et Evolution de Paris Sud, nous nous sommes intéressés aux mécanismes moléculaires contrôlant la métamorphose chez le têtard *Xenopus tropicalis*. Cette étude s'est déroulée en trois parties. Dans un premier temps, nous nous sommes concentrés sur la compétition existant entre deux enzymes de rôle opposé. Suite à ces résultats, nous avons modélisé, en parallèle, les réseaux de régulation entraînant deux mécanismes inverses lors de la métamorphose : la pousse des pattes arrières et la résorption de la queue. Enfin, dans la dernière partie, nous exploitons les données transcriptionnelles obtenues dans le laboratoire de Développement et Evolution afin de délimiter le réseau de régulation responsable du passage de l'état larvaire à l'état adulte.

- Le développement de notre application biologique a mis en exergue une limite classique à toute approche par model-checking. Les paramètres d'un modèle biologique sont soit connus dans la littérature soit estimés de façon plus ou moins arbitraire. Pour aider à la détermination de certains paramètres, il serait utile de développer un outil qui donne des contraintes sur un ou plusieurs paramètres inconnus du modèle pour que celui-ci satisfasse une propriété connue supposée vraie. Il s'agit d'un model-checking paramétrique, puisque la réponse n'est pas binaire mais fournit les contraintes sur les paramètres pour que la propriété soit satisfaite. Le Chapitre 6 présente une extension paramétrique de notre procédure de model-checking décrite dans le Chapitre 3. Cette extension consiste à construire des graphes d'évolution symboliques qui sont alors convertis en automate Event-Clock paramétriques.

Chapitre 1

Avantages et limites des Réseaux de Petri Hybrides Fonctionnels

1.1 Les Réseaux de Petri pour la biologie

Les réseaux de Petri [39] permettent la description et la modélisation de systèmes concurrents. Ils sont principalement utilisés pour spécifier, modéliser et comprendre les systèmes dans lesquels plusieurs processus sont interdépendants. Leur utilisation couvre un large spectre d'applications, comme les chaînes de production [40] ou les protocoles de communication [41]. La première utilisation des réseaux de Petri pour la modélisation de systèmes biologiques a été proposée par Reddy *et al.* [42]. Celui-ci développa une méthode de représentation des voies métaboliques qui fut par la suite étendue par Hofestädt [43]. Ainsi, au cours des années, différents types de réseaux de Petri étendus ont été utilisés pour modéliser des phénomènes biologiques. Genrich *et al.* [44] modélisèrent des voies métaboliques par réseaux de Petri colorés dans le but d'analyses quantitatives, ce type de réseau fut ensuite utilisé par Voss *et al.* [45] dans un but qualitatif. L'aléatoire des systèmes biologiques fut pris en compte grâce aux réseaux de Petri stochastiques [46, 47]. Tous ces formalismes dérivés des réseaux de Petri sont néanmoins inappropriés pour représenter, au sein d'un même modèle, des dynamiques discrètes et continues. Or, de nombreuses voies biologiques se définissent par une partie discrète telle que le contrôle d'un switch épigénétique et d'une partie continue représentant les réactions métaboliques, par exemple. Matsuno *et al.* [48] utilisèrent le formalisme des réseaux de Petri Hybride (HPN) développé par David et Alla [49] pour modéliser les dynamiques discrètes et continues du réseau de régulation génétique du phage lambda. L'apport d'une modélisation hybride a motivé Matsuno *et al.* à étendre le formalisme des HPN. Ils ont alors introduit les réseaux de Petri hybrides fonctionnels, notés HFPN pour Hybrid Functional Petri Nets en anglais [16, 17, 14, 15]. Ces réseaux permettent une modélisation intuitive et naturelle des voies biologiques. Le formalisme des HFPN permet de modéliser la consommation et la production de ressources d'entités discrètes ou continues. L'aspect fonctionnel des HFPN autorise la définition des quantités consommées ou produites sous forme de fonctions dépendant du marquage dans les places. Grâce à ces fonctions, il est facile de modéliser par HFPN un système d'équations différentielles ordinaires (ODE), traditionnellement utilisées pour modéliser des réactions biochimiques [50]. La représentation par HFPN permet,

au delà des ODE, une visualisation intuitive du système complet, essentiellement lors du traitement de grandes cascades de réactions.

Les HFPN constituent un formalisme très puissant capable de modéliser des processus biologiques divers à différentes échelles (par exemple modèle de l'évolution d'une population cellulaire tout en considérant les mécanismes intra-cellulaires, voir notre application à l'hématopoïèse dans [51]). Une telle puissance limite cependant les méthodes d'analyse et d'exploitation de ces modèles. En effet, un modèle HFPN ne peut être étudié qu'au travers de ses simulations obtenues grâce au logiciel *Cell Illustrator* [16] implémenté à cet effet. La validation (ou la réfutation) d'un modèle ne se fait donc que par l'observation d'un ensemble de traces sur un temps déterminé.

Contrairement aux simulations numériques, les méthodes formelles utilisent des techniques rigoureuses de raisonnements, basées sur la logique mathématique. Appliquées en génie logiciel, ces méthodes permettent d'obtenir une très forte assurance de l'absence de bogues. Appliquées à la modélisation, elles permettent la validation ou la réfutation formelle de modèles [4, 52]. Etant donnée une propriété comportementale connue du système biologique, l'application de telles méthodes mathématiques permet de ne sélectionner que les modèles satisfaisant la propriété. Par contre, ces méthodes ne sont facilement applicables que sur des formalismes dont le pouvoir d'expression est limité : c'est le cas par exemple du formalisme introduit par René Thomas [53] ou encore celui des réseaux de Petri usuels [18].

Nous voici confronté à la balance opposant pouvoir d'expression et applicabilité de méthodes formelles. Notre démarche a consisté à restreindre le formalisme des HFPN jusqu'à l'obtention d'un formalisme sur lequel il était possible de développer une procédure de model-checking. Dans la suite de ce chapitre, nous présentons le formalisme des HFPN, de même que les restrictions choisies, nous détaillons alors le formalisme des Réseaux de Petri Hybrides Temporisés ainsi obtenu.

1.2 Les Réseaux de Petri Hybrides Fonctionnels

Comme les réseaux de Petri usuels, les réseaux de Petri Hybrides fonctionnels (HFPN) [16, 17] sont composés de places, transitions et arcs, modélisant respectivement les entités biologiques, les processus biologiques et les relations entre ces deux derniers. L'aspect hybride du formalisme permet à chaque composant d'être discret ou continu. Les places discrètes (Figure 1.1) représentent les entités entières comme un nombre de cellules par exemple, alors que les places continues représentent des entités réelles comme une concentration moléculaire. La quantité de jetons (entière ou réelle) d'une place P est appelée le *marquage* de P et est notée $m(P)$.

Les deux types de transitions sont temporisés : une transition discrète est tirée après un délai de temps dt (Figure 1.1) alors qu'une transition continue tire continuellement à une vitesse V pouvant être définie par une fonction des valeurs du marquage des places du réseau. Par exemple, si T_1 est une transition continue et P_1, P_2 sont des places (continues ou discrètes) la vitesse de tir de T_1 peut se définir par la fonction suivante : $V(T_1) = 3.m(P_1) + \frac{1}{2}m(P_2)$ ou encore par la fonction $V(T_1) = e^{m(P_1)} + m(P_2)$. Les transitions discrètes modélisent des processus biologiques discrets comme la mitose ou un switch

épigénétique. A l'inverse, les transitions continues modélisent des processus continus dont un exemple est l'évolution de la concentration moléculaire en réponse à des réactions enzymatiques.

Trois types d'arcs peuvent lier les places aux transitions ou les transitions aux places (Figure 1.1).

- Un *arc activateur* avec un poids r active une transition en consommant les ressources uniquement si le marquage de la place à la source de l'arc est supérieur au poids r.
- Un *arc inhibiteur* avec un poids r permet à une transition de tirer uniquement si le marquage de la place à la source de l'arc est inférieur à r.
- Enfin, un *arc test* avec le poids r active une transition dans les mêmes conditions que l'arc activateur sans consommation de ressources.

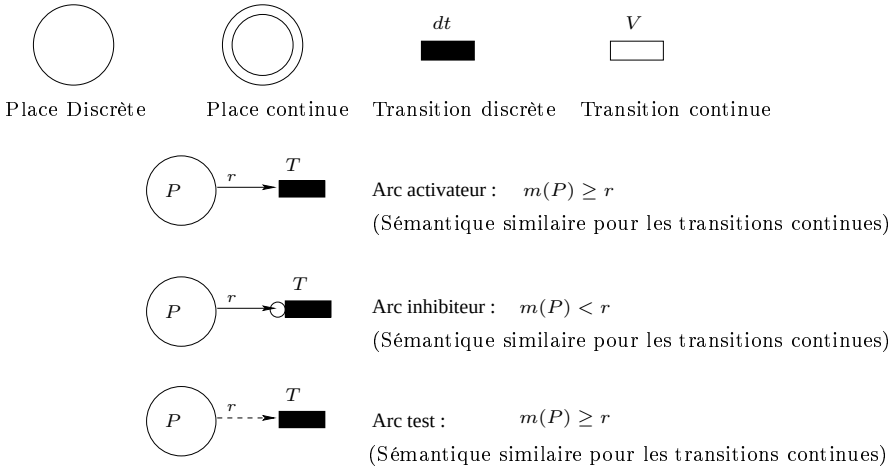

FIG. 1.1 – Notations utilisées pour les HFPN.

1.2.1 Définition

De manière formelle, les HFPN se définissent comme suit (Figure 1.2) [16].

Définition 1.1 (HFPN). *Un Réseau de Petri Hybride Fonctionnel est un sextuplet,* $HFPN =< P, T, h_N, E, h_E, \mathcal{F} >$ *où :*

- $\mathcal{P}$ *et* $\mathcal{T}$ *sont des ensembles disjoints de places et transitions,*
- $h_N : \mathcal{P} \cup \mathcal{T} \rightarrow \{D, C\}$ *appelée* fonction hybride *indique pour chaque nœud s'il est discret* (D) *ou continu* (C)*,*
- E *est un ensemble fini, non vide d'arcs* $(E \subseteq (P \times T) \cup (T \times P))$*. Chaque arc* e *est de la forme* (T, P) *(resp.* (P, T)*) si l'arc sort d'une transition* T *(resp. d'une place* P*) et rentre dans une place* P *(resp. dans une transition* T*),*
- $h_E : E \rightarrow \{A, I, Te\}$ *indique pour chaque arc s'il est activateur, inhibiteur ou test,*
- $\mathcal{F}$ *est un ensemble fini de fonctions associées à chaque transition.*

On distingue deux ensembles de fonctions, un pour les transitions continues (Définition. 1.2) et l'autre pour les transitions discrètes (Définition 1.3).

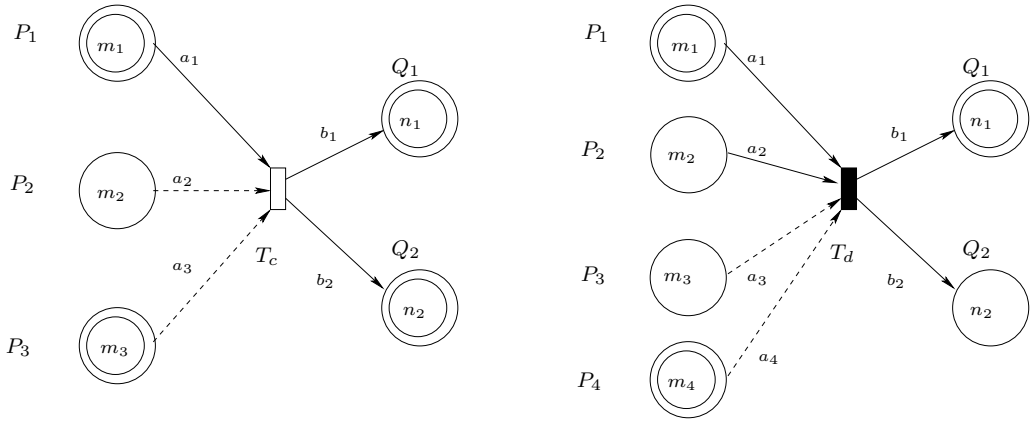

FIG. 1.2 – Représentation de configurations possibles des places et transitions dans les réseaux de Petri hybrides fonctionnels.

Définition 1.2 (Sémantique du tir d'une transition continue)**.** *Soit T une* transition continue *dont les places prédécesseurs sont $P_1, ..., P_p$ et les places successeurs $Q_1, ..., Q_q$ (voir la Figure 1.2 pour les notations). On note $a_1, ..., a_p$ le poids des arcs entrants (continus ou tests) et $b_1, ..., b_q$ celui des arcs sortants (continus). Supposons que $m_1(t), ..., m_p(t)$ et $n_1(t), ..., n_q(t)$ soient les marquages des places $P_1, ..., P_p$ et $Q_1, ..., Q_q$ au temps t. On définit comme suit les fonctions f et g comme les quantités consommées et produites durant le tir de la transition T.*

- La condition de tir *est donnée par un prédicat $c(m_1(t), ..., m_p(t))$. Tant que cette condition est vraie, T tire continuellement.*
- *Pour chaque arc entrant a_i, T est étiquetée par une fonction $f_i(m_1(t), ..., m_p(t)) > 0$ qui définit la vitesse de consommation de la place P_i quand T est tirée. Si a_i est un arc test, alors $f_i \equiv 0$ et aucune quantité n'est retirée de la place P_i. Nous avons alors $\frac{d[a_i](t)}{dt} = f_i(m_1(t), ..., m_p(t))$ où $[a_i](t)$ dénote la quantité retirée de la place P_i au temps t au travers des arcs continus a_i durant la période de tir de la transition T.*
- *Pour chaque arc sortant b_j, T est étiquetée par une fonction $g_j(m_1(t), ..., m_p(t)) > 0$ qui définit la vitesse de production de ressources des places Q_j au temps t au travers des arcs sortants continus b_j quand T est tirée. Nous avons alors $\frac{d[b_j](t)}{dt} = g_j(m_1(t), ..., m_p(t))$, où $d[b_j](t)$ dénote la quantité de ressources ajoutées aux places Q_j au temps t à travers l'arc continu b_j durant la période de tir de T.*

Définition 1.3 (Sémantique du tir d'une transition discrète)**.** *Soit T une* transition discrète *dont les places prédécesseurs sont $P_1, ..., P_p$ et les places successeurs $Q_1, ..., Q_q$ (voir la Figure 1.2 pour les notations). On note $a_1, ..., a_p$ le poids des arcs entrants (discrets ou tests) et $b_1, ..., b_q$ celui des arcs sortants (discrets). Supposons que $m_1(t), ..., m_p(t)$ et $n_1(t), ..., n_q(t)$ soient les marquages des places $P_1, ..., P_p$ et $Q_1, ..., Q_q$ au temps t. On définit comme suit les fonctions f et g comme les quantités consommées et produites durant le tir de la transition T.*

- La condition de tir *est donnée par un prédicat $c(m_1(t), ..., m_p(t))$. Si cette condition est vraie, T est permise pour tirer.*

- La fonction délai *est donnée par une fonction à valeur dans les rationnels positifs ou nuls* $d(m_1(t), ..., m_p(t))$. *Si la condition de tir est satisfaite au temps t,* T *tire dans un délai* $d(m_1(t), ..., m_p(t))$. *Cependant, si la condition de tir est changée durant ce délai, la transition* T *n'est plus permise pour tirer et la condition de tir est remise à zéro.*
- *Pour chaque arc entrant* a_i, T *est étiquetée par une fonction évaluée sur les entiers positifs ou nuls* $f_i(m_1(t), ..., m_p(t)) > 0$ *qui définit le nombre de jetons retirés de la place* p_i *à travers l'arc* a_i *par le tir de* T. *Si* a_i *est un arc test, alors* $f_i \equiv 0$ *et aucun jeton n'est consommé.*
- *Pour chaque arc sortant* b_j, T *est étiquetée par une fonction évaluée sur les entiers positifs ou nuls* $g_j(m_1(t), ..., m_p(t)) > 0$ *qui définit le nombre de jetons produits dans la place* Q_j *à travers l'arc* b_j *par le tir de* T.

1.2.2 Illustration biologique

Contexte biologique Tout au long de ce rapport, nous allons illustrer les différents formalismes et méthodologies sur un petit exemple biologique extrait de notre application biologique (qui sera plus détaillée dans le Chapitre 5).

La plupart des amphibiens subissent de nombreuses modifications morphologiques pour passer de l'état larvaire à l'état adulte, ce processus biologique s'appelle la métamorphose amphibienne [54]. Tous ces changements morphologiques (pousse des pattes arrières, disparition de la queue,...) sont contrôlés par les hormones thyroïdiennes (HT) [55]. On distingue deux types d'hormone : la thyroxine T4 est la forme inactive des hormones thyroïdiennes contrairement à la triiodothyronine T3 qui est la forme biologiquement active de l'hormone mais qui est synthétisée en quantité beaucoup plus faible [56]. Lorsque T3 est associée à son récepteur RHT [57], elle forme un facteur de transcription responsable de l'induction de nombreux gènes. Parmi ces gènes, nous allons spécifiquement nous intéresser à deux enzymes : la déiodinase de type 2, notée D2 [58, 59] et la déiodinase de type 3, notée D3 [60, 61]. Ces deux enzymes sont responsables du métabolisme des hormones thyroïdiennes. Alors que D2 agit en tant qu'activateur, en produisant la forme active de l'hormone (T3) à partir de sa forme inactive T4 ($D2 + T4 \rightarrow D2 + T3$), D3 inhibe aussi bien T3 que T4 en produisant deux nouvelles formes inactives d'hormones rT3 et T2 ($D3 + T4 \rightarrow D3 + rT3$ et $D3 + T3 \rightarrow D3 + T2$). Les deux enzymes D2 et D3 sont régulées différemment par T3. Le gène associé à D3 est un gène de réponse directe aux hormones thyroïdiennes alors que le gène associé à D2 est activé indirectement, le temps de réponse à T3 est donc plus long (quelques jours) [59].

La métamorphose requiert une très forte concentration de T3. Il est communément reconnu que cette forte concentration est obtenue grâce à l'action de l'enzyme D2 [58, 59]. Notre objectif est de vérifier cette hypothèse en modélisant la compétition entre ces deux enzymes grâce aux HFPN.

Modèle HFPN Quatre places continues sont utilisées pour modéliser les quatre entités biologiques (D2, D3, T3 et T4)(Figure 1.3). D3 est induit directement par T3, la transition continue t_2 modélise cette induction. La vitesse de réaction peut être obtenue dans la littérature grâce aux cinétiques de l'enzyme [62]. La vitesse de synthèse de D3 par T3

est proportionnelle à la concentration de T3, une simplification pertinente considère une relation linéaire entre T3 et D3. Celle-ci est modélisée comme suit : $V(t_2) = k_1.m(T3)$. Le facteur k_1 correspond au taux de synthèse de D3, obtenu par la pente des courbes cinétiques de D3.

D2 est un gène de réponse indirecte au hormones thyroïdiennes, on observe donc un délai entre l'induction par T3 et la synthèse de D2, ce délai est modélisé par la transition discrète t_1, la quantité produite de D2 est alors indiquée sur l'arc sortant de t_1, elle est modélisée comme pour D3.

Les transitions continues t_3 et t_4 modélisent les actions respectives de D3 et de D2. Tout comme les vitesses de synthèse de D2 et D3, nous avons considéré une relation linéaire pour les vitesses de réactions des enzymes. La vitesse de réaction de D3 est ainsi donnée comme suit : $V(t_3) = k_3.m(D3)$ où k_3 est associée aux constantes cinétiques (constante de Michaelis Km et vitesse maximale Vmax) de D3. On procède de même pour D2. Tous les paramètres sont rappelés sur la Figure 1.3.

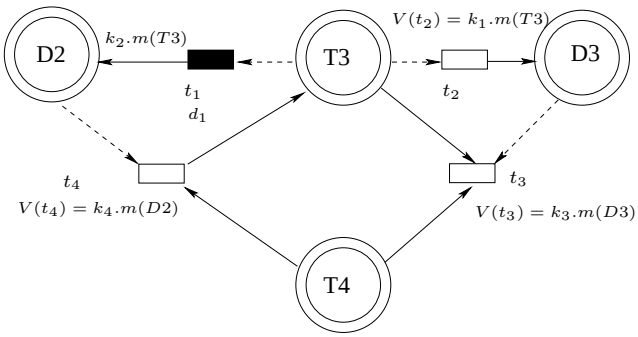

FIG. 1.3 – Modèle HFPN de la compétition enzymatique D2/D3.

Simulation Le logiciel *Cell Illustrator* a été implémenté par Matsuno *et al.* [16, 17] dans le but de simuler spécifiquement des modèles HFPN. La simulation de notre modèle nous donne les courbes présentées sur la Figure 1.4.

La simulation de notre modèle montre la diminution de T3 et de T4 malgré l'augmentation de la concentration des enzymes D2 et D3. Ceci traduit la prédominance *in silico* de D3 sur D2. L'exploitation de ces résultats sera faite dans le Chapitre 5 consacré à l'application biologique.

1.3 Restriction des Réseaux de Petri Hybrides Fonctionnels pour la vérification

Pouvoir d'expression des Réseaux de Petri hybrides fonctionnels. Les définitions précédentes montrent que le formalisme des Réseaux de Petri hybrides fonctionnels a un pouvoir d'expression très fort, il permet une modélisation fine et précise. En effet, l'aspect hybride permet d'associer au sein d'un même modèle des processus discrets et des processus continus. Une modélisation multi-échelle de systèmes biologiques est alors

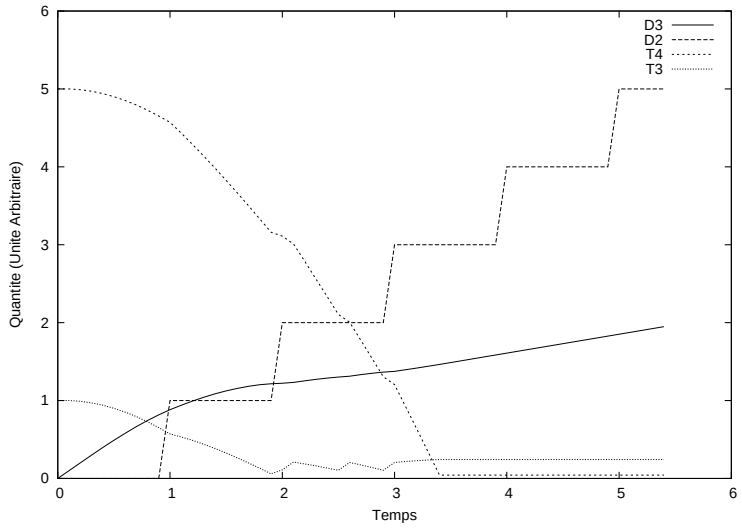

FIG. 1.4 – Simulation *in silico* du modèle de compétition enzymatique (logiciel *Cell Illustrator*).

possible en faisant coexister, par exemple, l'évolution d'une population cellulaire (processus discret) sous l'influence de réactions biochimiques (processus continus) [51]. L'aspect temporisé des HFPN permet de considérer le temps d'une manière quantitative. De nombreuses données biologiques comme les constantes cinétiques, les délais de pulses,... peuvent donc être directement intégrées dans les modèles. Enfin, l'aspect fonctionnel des HFPN permet la modélisation de fonctions biologiques très variées par les fonctions mathématiques qui leur sont associées. Dans le cas d'un modèle de la contraction des fibres musculaires, par exemple, nous utiliserons les fonctions mathématiques associées à la résistance des ressorts... Un tel pouvoir d'expression permet l'élaboration de modèles *in silico* beaucoup plus proches des modèles *in vivo*. La contrepartie d'un tel pouvoir d'expression est l'impossibilité de résoudre un certain nombre d'équations, rendant impossible l'étude et l'exploitation des modèles autrement que par simulation. Le paragraphe ci-dessous discute l'apport de méthodes de vérification par rapport à la simulation.

Simulation *versus* vérification. Reprenons notre exemple biologique et mettons en évidence quelques limites. La place T3 est à l'origine du tir de trois transitions (t_1, t_2 et t_3). La quantité de ressources dans la place T3 peut ne pas être suffisante pour le tir de ces trois transitions, nous sommes alors dans un cas de conflit. Les simulations ne traitent pas explicitement les conflits, pourtant très présents dans les systèmes biologiques. Par conséquent, les courbes observées sont le résultat d'une stratégie de gestion de conflit qui est *ad hoc* à la simulation. Une première question vient tout de suite à l'esprit : un traitement différent des conflits produirait-il les mêmes simulations ? La vérification oblige un traitement rigoureux des conflits, elle a donc la capacité de distinguer les différents comportements d'un même modèle selon le mode de résolutions de conflits.

Un second point essentiel concerne le traitement de traces finies ou infinies. Dans le cas des HFPN, la simulation ne peut être effectuée que sur une période plus ou moins

longue. Néanmoins, beaucoup de processus biologiques sont à l'origine d'oscillations infinies (homéostasie) ou au contraire sont à l'origine de l'obtention d'états stables. Dans le cas de simulations, nous ne pourrons observer ces oscillations ou ces états stables que sur une période donnée, en supposant que ceux-ci perdureront. La vérification permet, au contraire des simulations, le traitement de traces infinies permettant ainsi de vérifier mathématiquement que les oscillations observées sont bien infinies ou que l'état atteint est bien un état stable.

Enfin, l'argument majeur en faveur de la vérification est lié au traitement d'un nombre infini de traces par des manipulations symboliques. Ceci permet un inventaire exhaustif des états stables (même ceux rarement atteints).

Restriction aux Réseaux de Petri Hybrides Temporisés. Dans le but de vérifier la satisfaction de propriétés sur des modèles HFPN, nous devons préalablement restreindre le formalisme. Etant donné que les fonctions des HFPN engendrent des comportements difficilement prévisibles et contrôlables, nous avons dans un premier temps retiré l'aspect fonctionnel. En effet, dans le cas de fonctions simples comme les vitesses proportionnelles au marquage, on observe une croissance ou décroissance exponentielle du marquage des places. De plus, les vitesses de tir changent continuellement, ce qui rend difficile la manipulation symbolique de traces.

Le formalisme des HFPN privé de l'aspect fonctionnel correspond au formalisme des réseaux de Petri hybrides temporisés, notés THPN pour Timed Hybrid Petri Nets [34, 35]. Contrairement aux HFPN, les THPN permettent un traitement rigoureux des conflits. La section suivante présente ce formalisme ainsi que certains algorithmes capables d'extraire des traces infinies.

1.4 Les Réseaux de Petri Hybrides Temporisés (THPN)

Dans cette section, nous présentons le formalisme des THPN. Après avoir donné une définition statique de ces réseaux, nous donnerons les définitions permettant de comprendre la sémantique du formalisme. Nous verrons alors qu'il est possible d'étudier l'évolution du THPN en construisant son graphe d'évolution [34, 35]. Le graphe d'évolution permet de découper l'exécution continue du THPN en des tops d'horloge spécifiques, malgré la présence de processus continus.

1.4.1 Définitions

Avant de définir les THPN, rappelons que le symbole $\amalg$ représente l'union disjointe, $\mathbb{Q}^+$ et $\mathbb{R}^+$ représentent respectivement l'ensemble des rationnels et des réels positifs ou nuls, et $\mathbb{N}$ représente l'ensemble des entiers naturels.

Définition 1.4 (THPN). *Un Réseau de Petri Hybride Temporisé est un 8-uplet* $THPN = < \mathcal{P}, \mathcal{T}, \zeta, Pre, Post, m_0, delai, V >$ *où :*

- $\mathcal{P}$ *et* $\mathcal{T}$ *sont des ensembles disjoints de places et transitions,*

- $\zeta : \mathcal{P} \cup \mathcal{T} \rightarrow \{D, C\}$ *appelée fonction hybride indique pour chaque nœud s'il est discret ou continu.*
  On appellera T^D *(resp.* P^D*) et* T^C *(resp.* P^C*) l'ensemble des transitions (resp. places) discrètes et continues,*
- $Pre : \mathcal{P} \times \mathcal{T} \rightarrow \mathbb{Q}^+ \amalg \mathbb{N}$ *est la matrice d'incidence d'entrée. Si* $T \in T^D$ *alors* $Pre(P,T) \in \mathbb{N}$ *sinon* $Pre(P,T) \in \mathbb{Q}^+$*,*
- $Post : \mathcal{T} \times \mathcal{P} \rightarrow \mathbb{Q}^+ \amalg \mathbb{N}$ *est la matrice d'incidence de sortie. Si* $T \in T^D$ *alors* $Post(T,P) \in \mathbb{N}$ *sinon* $Post(T,P) \in \mathbb{Q}^+$*,*
- $m_0 : \mathcal{P} \rightarrow \mathbb{R}^+ \amalg \mathbb{N}$ *est le marquage initial. Si* $P \in P^D$ *alors* $m_0(P) \in \mathbb{N}$ *sinon* $m_0(P) \in \mathbb{R}^+$*,*
- $delai : T^D \rightarrow \mathbb{Q}^+$ *est le temps associé à* T*,*
- $U : T^C \rightarrow \mathbb{Q}^+$ *représente le flux d'une transition continue.*

Soit $^\circ T$ (resp. $^\circ P$) l'ensemble des places (resp. transitions) précédant la transition T (resp. la place P) et soit T° (resp. P°) l'ensemble des places (resp. transitions) sortant de la transition T (resp. de la place P). Prenons l'exemple de la Figure 1.5, $^\circ T_1 = \{P_1, P_2\}$ et $T_1^\circ = \{P_1\}$.

Les THPN permettent à des réseaux de Petri discrets et continus de coexister au sein d'un même réseau de Petri hybride. Néanmoins, une restriction est imposée. De façon à assurer un marquage discret, une place discrète ne peut être connectée à une transition continue que par deux arcs de même poids : un entrant et un autre sortant de la transition continue (Figure 1.5). Ainsi, même si le tir de la transition continue consomme 0.5 jeton de la place discrète, l'arc de même poids sortant de la transition assure une production simultanée de 0.5 jeton, maintenant le marquage discret entier .

La Définition 1.4 donne une définition statique des THPN. Afin de comprendre la sémantique de ce formalisme, nous définissons des notions importantes comme celles de vitesse instantanée, de balance ou encore de marquage non standard.

1.4.1.1 Vitesse

Les transitions continues des THPN sont définies par un flux maximal U. La Définition 1.5 calcule la vitesse de tir maximal V à partir du flux U.

Définition 1.5 (Vitesse maximale)**.** *La vitesse maximale de tir d'une transition continue* T *se définit comme suit :*

$$V(T) = U(T). \min_{P \in {}^\circ T \cap P^D} m(P)$$

Ainsi, la vitesse maximale d'une transition continue T dépend du marquage de ses places discrètes entrantes. Prenons l'exemple de la Figure 1.5, il est clair que si la place discrète P_1 est vide, la transition continue t_1 ne peut tirer, on a donc $V(T_1) = 0$, si P_1 possède un jeton (resp. 2 jetons), la vitesse maximale vaudra $V(T_1) = 1$ (resp. $V(T_1) = 2$).

Les jetons de la place discrète représentent le nombre de serveurs disponibles pour le processus continu. Si T_1 a deux serveurs (c'est-à-dire si P_1 a deux jetons), le flux de

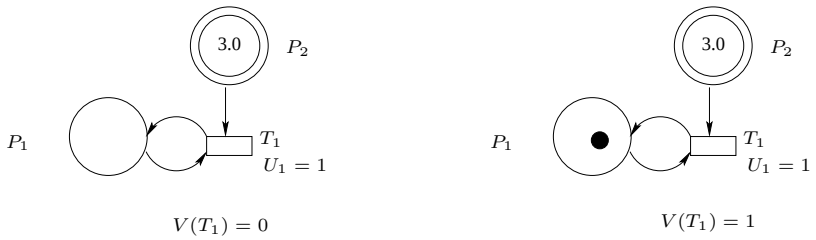

FIG. 1.5 – Illustration de la définition de la vitesse V d'une transition continue à partir de son flux U.

jetons peut aller deux fois plus vite ($V(T_1) = 2.U(T_1)$). Dans un cadre biologique, il est peu cohérent de parler de plusieurs serveurs pour un même processus continu, nous nous situons en réalité dans un cas d'un unique serveur, dans lequel $V(T) = U(T)$ pour toute transition continue T. Par la suite, nous parlerons donc uniquement de vitesse maximale V et non de flux U.

Lorsque le marquage des places entrant dans une transition continue T le permet, T tire à sa vitesse maximale $V(T)$. Néanmoins, le marquage des places entrant dans T peut être insuffisant pour permettre à la transition de tirer à sa vitesse maximale. On définit alors la vitesse instantanée, notée $v(T)$ correspondant à la vitesse maximale à laquelle une transition peut tirer étant données les ressources présentes dans le THPN à un instant donné. On a alors $0 \leq v(T) \leq V(T)$.

***Notation* 1.** *Par la suite, nous noterons souvent $m_i = m(P_i)$ et $v_j = v(T_j)$.*

1.4.1.2 Balance

Une place continue reçoit une certaine quantité de ressources grâce au tir de ses transitions entrantes. A l'inverse, la quantité de ressources d'une place continue diminue par le tir des ses transitions sortantes. Soit P_i une place continue, on note I_i la quantité de ressources entrant dans P_i et O_i la quantité de ressources sortant de cette place. On peut alors calculer la balance B_i de la place P_i correspondant à la différence entre la quantité de ressources entrantes et sortantes. Dans la Figure 1.6, $I_1 = v_3$, $I_2 = 0$ et $I_3 = v_1 + v_2$, $O_1 = v_1$, $O_2 = v_2$ et $O_3 = v_3$, on a donc $B_1 = v_3 - v_1$, $B_2 = -v_2$ et $B_3 = v_1 + v_2 - v_3$. Si $B_i > 0$ la place se remplit au cours du temps, si $B_i < 0$ le marquage de la place P_i diminue et si $B_i = 0$, le marquage reste constant. On remarque que si une place est vide, sa balance ne peut pas être strictement négative, étant donné que le tir des transitions engendrerait un marquage strictement négatif.

Formellement, ces notions se définissent comme suit :

Définition 1.6 (I_i)**.** *La quantité de ressources I_i entrant dans la place continue P_i est donnée par :*

$$I_i = \sum_{T_j \in {}^\circ P_i} Post(T_j, P_i).v_j$$

Définition 1.7 (O_i)**.** *La quantité de ressources O_i sortant de la place continue P_i est donnée par :*

$$O_i = \sum_{T_j \in P^\circ{}_i} Pre(P_i, T_j).v_j$$

Définition 1.8 (B_i)**.** *La balance B_i de la place continue P_i est donnée par :*

$$B_i = I_i - O_i$$

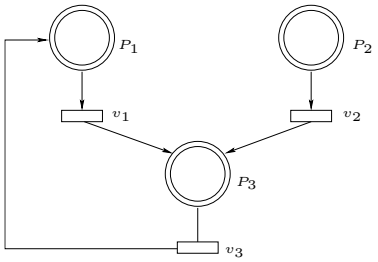

FIG. 1.6 – Illustration de la balance d'une place continue. La place P_1 est remplie à la vitesse v_3 et se vide à la vitesse v_1, on a donc $B_1 = v_3 - v_1$.

1.4.1.3 Marquage non-standard

La Figure 1.7 présente un exemple de THPN dont le marquage initial est $m_0(3.0, 0.0)$. La balance de la place P_2 est nulle, par conséquent $m(P_2) = 0$ pour tout temps $t \in \mathbb{R}^+$. Néanmoins, la place n'est pas complètement vide car un flux de jetons passe continuellement. On notera alors $\bar{m}(P_2) = 0^+$ et on appellera $\bar{m}$ *le marquage non-standard*. Le marquage non-standard est donc défini sur l'ensemble $\mathbb{R}^+ \cup \{0^+\}$. Celui-ci est utilisé dans le calcul des vitesses instantanées. Par convention, $0^+ > 0$ et $\forall r \in \mathbb{R}^+, r < 0^+ \Rightarrow r = 0$.

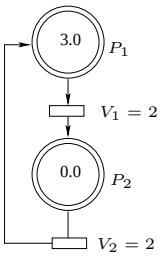

FIG. 1.7 – Illustration du marquage non-standard. Bien que $m(P_2) = 0$ au cours du temps, un flux de jetons ne cesse de passer dans cette place, on a donc $\bar{m}(P_2) = 0^+$.

1.4.1.4 Permission

Le marquage non-standard permet à un instant donné de déterminer quelles transitions ont la permission de tirer. La notion de permission diffère selon le type de transition étudiée (discrète ou continue).

Définition 1.9 (Permission d'une transition discrète). *Une transition discrète T est* permise *si chaque place $P \in {}^\circ T$ satisfait $m(P) \geq Pre(P,T)$, où $m(P)$ est le marquage de la place P.*

Si la transition T reste permise durant le délai $delai(T)$, elle tirera à la fin de ce délai. $Pre(P,T)$ jetons sont alors retirés de chaque place $P \in {}^\circ T$ et $Post(T,P)$ jetons sont ajoutés dans chaque place $P \in T^\circ$. Le marquage peut être suffisant pour permettre plusieurs tirs simultanés de T. Pour une transition discrète T, le nombre de tirs successifs autorisés étant donné un marquage correspond au *degré de permission* de la transition.

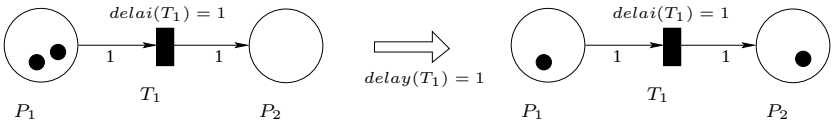

FIG. 1.8 – Sémantique d'une transition discrète.

La place P_1 de la Figure 1.8 possède 2 jetons et chaque tir de la transition discrète T_1 ne retire qu'un jeton à P_1. On en déduit que le degré de permission de T_1 est égal à 2. Une fois le délai $delai(T_1) = 1$ écoulé, $Pre(P_1, T_1) = 1$ jeton est consommé et $Post(T_1, P_2) = 1$ jeton est produit dans la place P_2.

Définition 1.10 (Permission d'une transition continue). *Une transition continue T est* permise *:*

- *si chaque place $P \in {}^\circ T \cap P^D$ satisfait $m(P) \geq Pre(P,T)$ et*
- *si chaque place $P \in {}^\circ T \cap P^C$ satisfait $\bar{m}(P) > 0$.*

De plus, si chaque place continue $P \in {}^\circ T \cap P^C$ satisfait $m(P) > 0$, on dit que T est fortement permise.

Une transition continue T tire à sa *vitesse instantanée* $v(T)$. Le marquage de chaque place $P \in {}^\circ T$ est diminué de la quantité $Pre(P,T) \times v(T)$ et symétriquement, le marquage de chaque place sortante $P \in T^\circ$ est augmenté de la quantité $Post(T,P) \times v(T)$.

La Figure 1.9 illustre la définition de permission d'une transition continue. Dans le THPN (a), les transitions T_1 et T_2 sont fortement permises car $m(P_1)$ et $m(P_2)$ sont strictement positifs, elles peuvent donc tirer à leur vitesse maximale $v_1 = V_1 = 2$ et $v_2 = V_2 = 3$. Dans le THPN (b), $m(P_2) = 0$ et $\bar{m}(P_2) = 0^+$, par conséquent la transition T_2 est faiblement permise. La transition T_2 ne peut tirer à sa vitesse maximale car elle engendrerait un marquage négatif. On a donc nécessairement $v_2 = v_1 = 2 < V_2$. Nous reviendrons, dans la section suivante, sur le détail du calcul des vitesses instantanées v_i.

La définition de la sémantique d'un THPN permet de comprendre comment notre modèle THPN va évoluer au cours du temps. L'étude de l'évolution d'un THPN peut encore être détaillée, dans un cadre cependant plus restrictif, grâce à la construction d'un graphe d'évolution [34].

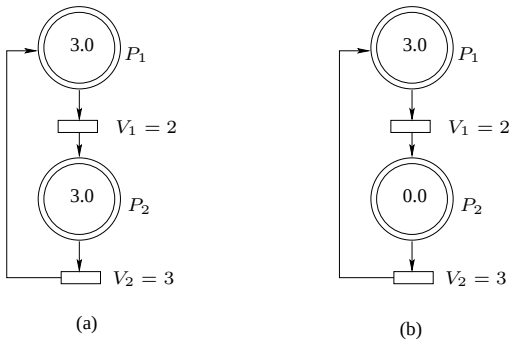

FIG. 1.9 – Illustration de la permission de transitions continues. Les transitions T_2 des THPN (a) et (b) sont permises car $\bar{m}(P_2) > 0$. La transition T_2 du THPN (a) est fortement permise car $m(P_2) > 0$ et la transition T_2 du THPN (b) est faiblement permise car $m(P_2) = 0$.

1.4.2 Graphe d'évolution

Le graphe d'évolution donne une trace de l'exécution du modèle THPN. Il est lui-même représenté par un réseau de Petri standard [34, 35] dans lequel chaque place correspond à un IB-state (pour Invariant-Behavior state) et chaque transition est associée à un événement dont l'occurrence produit un changement d'IB-state (voir Figure 1.10 pour les notations). Par définition, pendant toute la durée d'un IB-state, le THPN vérifie les propriétés suivantes :

- Le marquage des places discrètes est constant,
- Le degré de permission des transitions discrètes est constant,
- Les vitesses instantanées des transitions continues sont constantes,
- A chaque IB-state est associé un marquage continu. Ainsi, lorsqu'un IB-state est atteint le marquage continu a toujours la même valeur.

Le passage d'un IB-state à un autre ne peut avoir lieu que si un des événements suivants a lieu :

- *C1-event* : le marquage d'une place continue devient égal à zéro,
- *D1-event* : une transition discrète est tirée,
- *D2-event* : le degré de permission d'une transition discrète est modifié à cause de la modification du marquage d'une place continue.

Les transitions du graphe d'évolution sont étiquetées par les événements ayant eu lieu, par le temps d'occurrence de ces événements et par le marquage des places continues (Figure 1.10). Un IB-state est représenté par une "case" divisée en deux, chaque partie de la case contient une information pertinente : le marquage des places discrètes étiquette la partie gauche et les vitesses instantanées des transitions continues étiquettent la partie droite.

Selon que l'exécution du THPN se termine ou pas, le graphe d'évolution se termine par *deadlock* ou par *bouclage*. Afin de distinguer dans nos notations les transitions du graphe d'évolution des transitions du THPN, nous noterons T_i^{GE} la i^{ieme} transition du

graphe d'évolution et IB_i l'IB-state qui en résulte. Enfin, on appelle nIB le numéro du dernier IB-state du graphe d'évolution et en cas de bouclage, *nloop* indique le numéro de l'IB-state sur lequel on observe le bouclage (Figure 1.10).

Par la suite, nous ne traitons pas les THPN non bornés, c'est-à-dire que l'algorithme de construction du graphe d'évolution n'est appliqué qu'aux THPN dont le marquage des places ne croît pas indéfiniment. De plus, comme nous l'avons précisé ci-dessus, le graphe d'évolution fournit une trace déterministe de l'exécution du THPN. Ceci n'est possible qu'après avoir résolu tous les conflits potentiels du THPN. Ainsi, avant de présenter la méthodologie pour construire le graphe d'évolution, nous allons définir les différents types de conflit ainsi que leur mode de résolution.

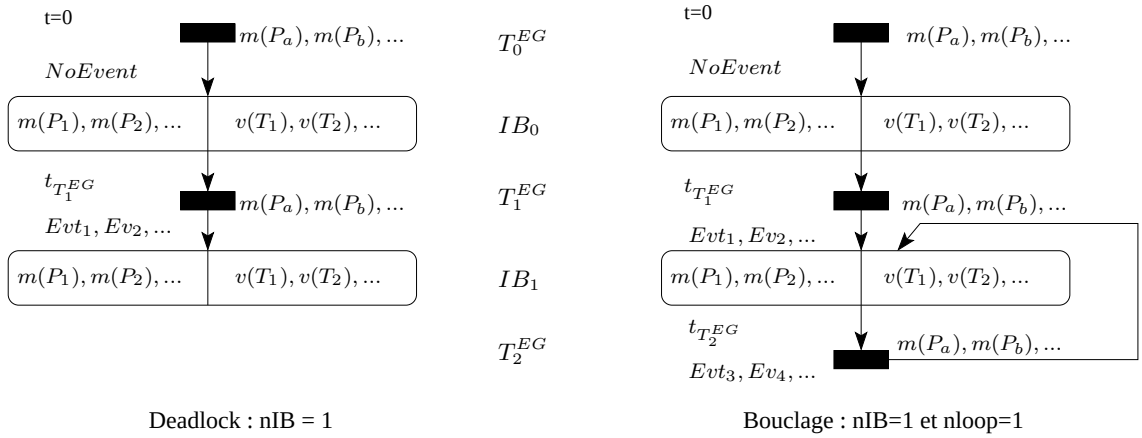

FIG. 1.10 – Schéma d'un graphe d'évolution se terminant soit par deadlock soit par bouclage. Les places $P_a, P_b, ...$ sont de type continu alors que les places $P_1, P_2, ...$ sont de type discret.

1.4.2.1 Gestion des conflits

Une structure de conflit existe quand une même place P est une place entrante de deux transitions ou plus. Si les ressources de P ne sont pas suffisantes pour permettre à toutes ces transitions de tirer simultanément, nous nous retrouvons dans un cas de *conflit réel*. Le calcul des vitesses instantanées requiert alors la résolution des conflits. Soit $K_c =< P_c, \{T_a, T_b\} >$ un conflit impliquant la place P_c et les transitions T_a et T_b. Quatre types de conflit sont considérés :

- Cas 1 : $T_a, T_b \in T^D$
- Cas 2 : $T_a, T_b \in T^C$ et $P_c \in P^C$
- Cas 3 : $T_a \in T^D$ et $T_b \in T^C$ ou *vice-versa*
- Cas 4 : $T_a, T_b \in T^C$ et $P_c \in P^D$

Il existe une infinité de règles de résolution de conflits, on propose deux manières crédibles de résoudre les conflits : *Le partage* propose de partager les ressources entre les

différentes transitions selon un schéma donné et *la priorité* ordonne les transitions suivant un ordre de priorité.

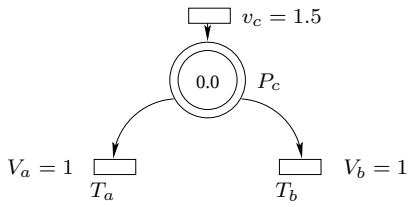

FIG. 1.11 – Exemple d'un conflit de type 2.

La Figure 1.11 montre un exemple simple de conflit de type 2. Les ressources entrent dans la place continue P_c à la vitesse de $v_c = 1.5$, ce qui n'est pas suffisant pour permettre aux transitions T_a et T_b de tirer à leur vitesse maximale ($V_a = V_b = 1$). Par conséquent, nous avons les contraintes suivantes :

$$\begin{array}{rcl} v_a & \leq & 1 \\ v_b & \leq & 1 \\ v_a + v_b & = & 1.5 \end{array}$$

Résolution par priorité. Si la transition T_a est prioritaire sur T_b (noté $T_a < T_b$), la résolution de conflit permet à T_a de tirer à sa vitesse maximale ($v_a = V_a = 1$). Nous avons alors $v_b = 0.5$.

Résolution par partage. Le conflit peut être résolu par partage. Le partage du flux est possible étant donné que le flux est continu. Si nous voulons partager le flux tel que $v_a = v_b$ alors les vitesses de tir instantanées vaudront $v_a = v_b = 0.75$.

Par la suite, nous nous concentrerons uniquement sur des résolutions de conflits par priorité. Les raisons sont d'ordre pédagogique : les algorithmes de traitement de conflits par partage sont plus complexes et moins intuitifs. Néanmoins, ceux-ci existent [34, 35] et peuvent facilement être utilisés dans notre procédure. Un autre avantage de résolution par priorité réside dans la possibilité pour un THPN donné de traiter *toutes* les résolutions possibles pour l'ensemble de conflits coexistant dans le modèle.

Sur l'exemple traité ci-dessus, le calcul des vitesses instantanées se fait très facilement. La méthodologie reste la même pour des réseaux plus compliqués. Elle est généralisée dans la section suivante.

1.4.2.2 Calcul des vitesses instantanées

Réseaux de Petri continus. Dans un premier temps, plaçons-nous dans le cas d'un réseau de Petri purement continu. Il s'agit d'un cas particulier de THPN dans lequel il n'y a ni places ni transitions discrètes. Toutes les places et transitions mentionnées ci-dessous sont donc continues. Le calcul des vitesses instantanées diffère selon la permission de la transition. On distingue 3 cas :

1. Soit la transition continue T_j est fortement permise au temps t (*i.e.*, $m_i(t) > 0$ pour tout $P_i \in {}^{\circ}T_j$). Dans ce cas $v_i(t) = V_i$.

2. Soit la transition continue T_j est faiblement permise à cause d'un ensemble de places $Q_j(t)$ de marquage nul. Si T_j n'est pas impliquée dans un conflit ou que T_j a le premier niveau de priorité, alors $v_j(t)$ se calcule de la façon suivante :

$$v_j(t) = min\left(\underset{P_i \in Q_j(t)}{min}\left(\frac{I_i}{Pre(P_i, T_j)}\right)\ ,\ V_j\right)$$

3. Soit enfin le conflit réel $K_m =< P_m, \{T_1, T_2, ..., T_n\} >$ est tel que $T_1 < T_2 < ... < T_n$ (les transitions sont classées par ordre croissant de priorité, T_1 est prioritaire sur T_2,...). La transition T_1 a le premier niveau de priorité, v_1 est donc calculée comme précédemment. La transition T_2 a le second niveau de priorité. Dans le cas où T_2 n'a pas d'autres places entrantes que P_m, $v_2(t)$ a la forme suivante :

$$v_2(t) = min\left(\begin{array}{l} I_m - Pre(P_m, T_1).v_1(t), \\ V_2 \end{array}\right)$$

La fonction *min* prend le minimum entre le reste des ressources présentes dans P_m après le tir de T_1 ($I_m - Pre(P_m, T_1).v_1(t)$) et la vitesse maximale V_2 à laquelle la transition T_2 peut tirer.

Dans le cas plus général où il existe un autre ensemble de places $Q_k(t)$ de marquage nul entrant dans T_2, on ajoute le cas où une des places de $Q_k(t)$ est limitante :

$$v_2(t) = min\left(\begin{array}{l} I_m - Pre(P_m, T_1).v_1(t), \\ V_2, \\ \underset{P_i \in Q_k(t)}{min}\ \dfrac{I_i}{Pre(P_i, T_2)} \end{array}\right)$$

De la même façon, on obtient la vitesse instantanée de la transition T_3 au temps t notée $v_3(t)$ comme suit :

$$v_3(t) = min\left(\begin{array}{l} I_m - Pre(P_m, T_1).v_1(t) - Pre(P_m, T_2).v_2(t), \\ V_3, \\ \underset{P_i \in Q_k(t)}{min}\ \dfrac{I_i}{Pre(P_i, T_3)} \end{array}\right)$$

La démarche reste la même pour toutes les transitions impliquées dans le conflit.

Dans l'exemple de la Figure 1.11, il n'y a pas de transitions fortement permises. le premier niveau de priorité est attribué à la transition T_a, on a alors $v_a = min(1.5, 1) = 1$. On peut à présent calculer la vitesse de la transition T_b, $v_b = min(1.5 - 1, 1) = 0.5$.

THPN. Etant donné que les valeurs des vitesses maximales d'un THPN dépendent du marquage discret (Définition 1.5), nous devons donc préalablement calculer le marquage des places discrètes. Une fois ce marquage calculé, celui-ci n'intervient plus dans le calcul des vitesses instantanées. Une stratégie permettant d'utiliser l'algorithme précédent consiste donc à retirer toute les places et transitions discrètes du THPN une fois les vitesses maximales calculées. On obtient alors un ensemble de réseaux de Petri continus temporisés comme présenté dans la Figure 1.12.

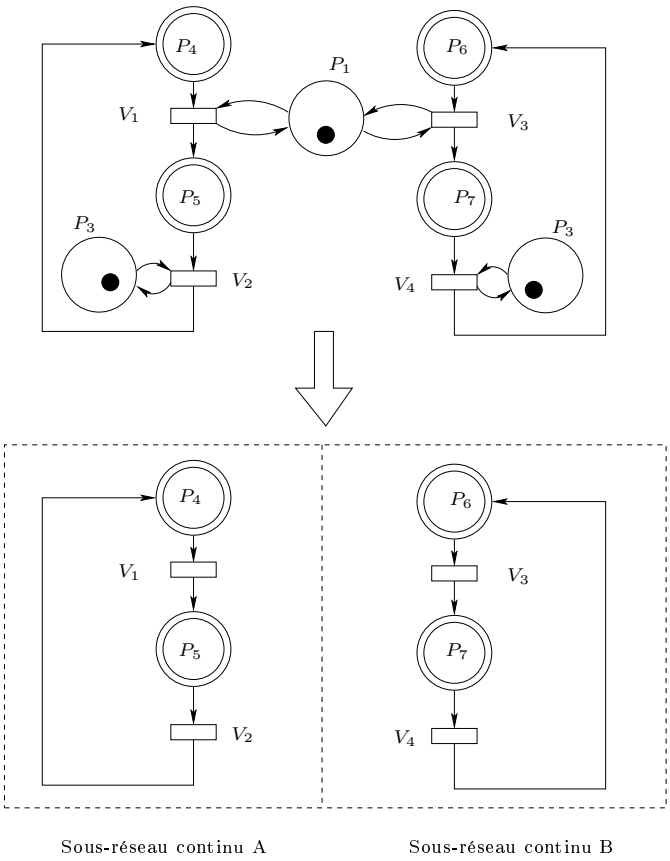

FIG. 1.12 – Construction d'un ensemble de réseaux de Petri continus à partir d'un THPN.

Pour pouvoir appliquer l'algorithme précédent de calcul de vitesses, nous devons ordonnancer les sous-réseaux continus suivant un ordre de priorité. L'algorithme est alors appliqué sous-réseau par sous-réseau. Reprenons l'exemple de la Figure 1.12, nous avons deux choix de résolution par priorité. Soit le sous-réseau A est prioritaire sur le sous-réseau B (A<B), soit le sous-réseau B est prioritaire sur le sous-réseau A (B<A). Dans le cas où A est prioritaire, l'algorithme de calcul des vitesses instantanées est appliqué au sous-réseau A avant d'être appliqué au sous-réseau B.

La méthodologie de calcul de vitesses peut à présent être utilisée pour tout THPN. Néanmoins, pour que la méthode soit automatisable, nous devons considérer les deux hypothèses suivantes.

Hypothèse 1. *Soient T_a et T_b deux transitions continues impliquées dans un conflit*

$< P_i, \{T_a, T_b, ..\} >$ *tel que* P_i *est une place discrète. Les transitions* T_a *et* T_b *n'appartiennent pas au même sous-réseau continu.*

Hypothèse 2. *Soit un quadruplet* $(T_{a,X}, T_{b,X}, T_{a,Y}, T_{b,Y})$ *tel que* $T_{a,X}$ *et* $T_{b,X}$ *appartiennent au sous-réseau continu* X*, et* $T_{a,Y}$ *et* $T_{b,Y}$ *appartiennent au sous-réseau* Y*.* $T_{a,X}$ *et* $T_{a,Y}$ *sont impliqués dans un conflit structurel de type 4, de même que* $T_{b,X}$ *et* $T_{b,Y}$*. Les règles de résolution locale de conflit ne peuvent prendre que les formes suivantes :*

- $T_{a,X} < T_{a,Y}$ *et* $T_{b,X} < T_{b,Y}$*, ou*
- $T_{a,X} > T_{a,Y}$ *et* $T_{b,X} > T_{b,Y}$

Les hypothèses 1 et 2 concernent les conflits de type 4, c'est-à-dire les confits impliquant une place discrète P et n transitions continues $T_1, ..., T_n$ (voir le conflit $< P_1, \{T_1, T_3\} >$ de la Figure 1.12). L'hypothèse 1 assure que chaque transition continue impliquée dans un même conflit de type 4 appartient à un sous-réseau continu différent. Ceci permet de calculer la vitesse instantanée de chaque transition $T_1, ..., T_n$ indépendamment les unes des autres. L'hypothèse 2 assure la cohérence dans l'ordre du calcul des vitesses instantanées. Ainsi, toutes les transitions de même niveau de priorité impliquées dans des conflits de type 4 différents doivent appartenir au même sous-réseau de façon à être calculées simultanément.

Nous insistons sur le fait que ces hypothèses ne sont nécessaires que pour l'implémentation, elles ne seront donc plus mentionnées par la suite.

1.4.2.3 Algorithme de construction du graphe d'évolution

L'algorithme de construction du graphe d'évolution détermine les prochains événements pouvant avoir lieu. Les futurs événements sont listés dans une séquence ordonnée par le temps, appelée TOS pour Timed-Ordered Sequence. TOS est une suite finie $(t_a, X_a)(t_b, X_b)...(t_u, X_u)$ où les t_i sont des temps tels que $t_a < t_b < ... < t_u$ et où X_s représente les événements connus et attendus au temps t_s. Chaque X_s est un triplet $X_s = (C1(t_s), D1(t_s), D2(t_s))$ où $C1(t_s)$ est un ensemble de C1-events, $D1(t_s)$ est un ensemble de D1-events et $D2(t_s)$ est un ensemble de D2-events ayant lieu au temps t_s.

Description de l'Algorithme [34] :

1. Initialisation
   (a) Entrer les données relatives au THPN comprenant le marquage initial, la résolution locale de conflits et l'ensemble de résolutions pour les sous-réseaux continus,
   (b) Initialisation du marquage continu pour toutes les places continues, noté $\bar{m}^c$. Initialisation de TOS par les D1-event possibles. Soit $t_s = t_0 = 0$.

2. Si l'ensemble des événements D2 n'est pas vide au temps t_s, noté $D2(t_s) \neq \emptyset$, alors ajouter le D1-event correspondant dans TOS,

3. Si $D1(t_s) = \emptyset$ alors **aller à l'étape 4** sinon tirer les transitions correspondantes (ou un sous-ensemble s'il existe des conflits de type 1), mettre à jour TOS en retirant les événements ayant eu lieu et **aller à l'étape 2**,

4. Construire les réseaux de Petri continus sous-jacents,

5. Si tous les niveaux de priorité des sous-réseaux continus ont été traités alors **aller à l'étape 6** sinon calculer les vitesses instantanées de chaque transition du sous-réseau continu en cours en utilisant l'algorithme présenté en section 1.4.2.2, augmenter le niveau de priorité et **retourner à l'étape 5**,

6. Mise à jour de TOS en déterminant les temps des prochains C1-events, D2-events et D1-events.

7. Déléter le premier temps t_1 de TOS et soit $t_s = t_1$. Si t_s n'existe pas alors **Fin**, sinon calculer le nouveau marquage au temps t_s.

8. Si les paramètres de l'IB-state s sont déjà présents dans un précédent IB-state k avec $k < s$ alors **Fin** sinon **aller à l'étape 2**.

La preuve de terminaison de cet algorithme est donnée ci-dessous. Celle-ci nous a été donnée par René David et Hassane Alla, auteurs de [34].

Lemme 1.1. *L'algorithme de construction d'un graphe d'évolution se termine.*

Preuve :

Pour faire le raisonnement qui suit, on ajoute l'hypothèse que les marquages initiaux des places continues sont des nombres rationnels.

- ***Cas de THPN bornés :***
  Un IB-state est défini par 4 caractéristiques : le vecteur de vitesses, le marquage de chaque place discrète, le degré de permission de chaque transition discrète, et le marquage des places continues.
  Pour chacune des trois premières (marquage discret, degré de permission et vitesses instantanées) le nombre de cas est borné.
  En ce qui concerne la quatrième composante, le marquage de chaque place continue est nécessairement borné, par hypothèse de construction du graphe d'évolution. Comme toutes les temporisations (temporisations des transitions discrètes et durées conduisant à l'annulation des marquages des transitions continues) sont des nombres rationnels, ces valeurs ont un plus petit commun multiple, ce qui implique que tôt ou tard, on retrouvera le même IB-state.

- ***Cas de THPN non bornés :***

Il est possible, dans ce cas, que l'algorithme de construction du graphe d'évolution ne se termine pas. La Figure 1.13 donne un exemple dans lequel une place discrète n'est pas bornée. L'algorithme ne se termine alors jamais.

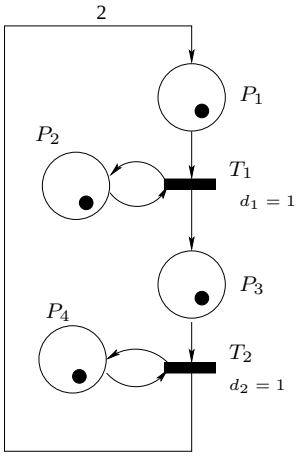

FIG. 1.13 – Exemple de THPN non borné pour lequel l'algorithme de construction du graphe d'évolution ne se termine pas.

On retient néanmoins qu'un système réel est toujours borné. De plus, il est possible de trouver "à la main", un représentation finie du graphe d'évolution de la Figure 1.13, en paramétrant par k (nombre d'itérations) certaines composantes des IB-states (voir [34] pour plus de détails).

□

La section suivante déroule l'algorithme de construction du graphe d'évolution sur un exemple.

1.4.3 Illustration biologique

Modèle THPN. Nous reprenons notre exemple biologique détaillé dans la section HFPN. La structure de notre réseau de Petri reste la même que celle de notre modèle HFPN (Figure 1.14), à l'exception de la disparition des arcs tests. En effet, la notion d'arcs tests n'est pas définie dans les THPN, ils sont néanmoins facilement modélisés grâce à deux arcs de même poids, un entrant et l'autre sortant. La différence réside dans la définition des vitesses maximales. Prenons le cas de la transition t_2, alors que sa vitesse dans le HFPN s'écrit $V(t_2) = k_1.m(T3)$, la restriction aux THPN nous contraint à définir $V(t_2)$ comme suit : $V(t_2) = k_1$ où k_1 correspond au même taux de synthèse de D3 par T3, obtenu dans la littérature biologique grâce aux courbes cinétiques [62]. Par la modélisation THPN, nous perdons donc l'aspect proportionnel des vitesses. Cette restriction peut sembler forte, néanmoins rappelons-nous que grâce à elle, nous allons pouvoir vérifier formellement la satisfaction d'un ensemble de propriétés biologiques connues.

Graphe d'évolution. La place T3 doit tirer trois transitions t_1, t_2 et t_3 (Figure 1.14). Si le marquage de T3 n'est pas suffisant pour permettre le tir simultané de ces transitions, il existe un conflit réel. Différentes résolutions peuvent être proposées : $t_1 < t_2 < t_3$,

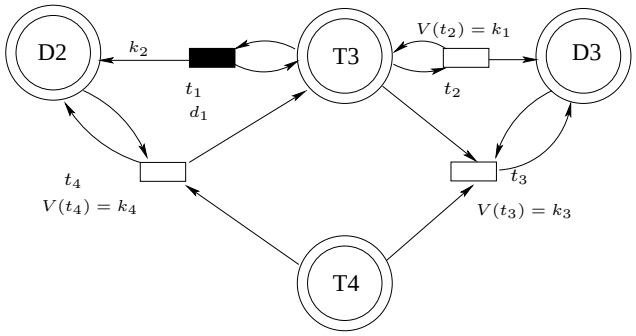

FIG. 1.14 – Modèle THPN associé à la compétition D2/D3.

$t_2 < t_3 < t_1$,...La Figure 1.15 donne le graphe d'évolution du modèle THPN lorsque la priorité est donnée à la transition t_3 : $t_3 < t_1 < t_2$. De même, la place T4 est impliquée dans un conflit avec les transitions t_4 et t_3, nous choisissons $t_3 < t_4$.

Nous rappelons que notre procédure considère *toutes* les résolutions de conflits par priorité et fournit donc un ensemble de graphes d'évolution. Néanmoins, pour l'illustration, nous ne nous concentrerons que sur cette résolution de conflit. L'absence de places discrètes dans notre modèle THPN nous permet de représenter les IB-state (Figure 1.15) à l'aide d'une seule case correspondant aux valeurs des vitesses instantanées.

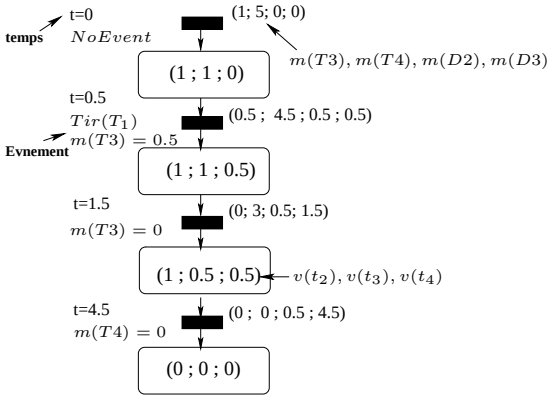

FIG. 1.15 – Graphe d'évolution associé au modèle THPN.

Déroulement de l'algorithme : L'algorithme est déroulé avec les paramètres suivant : $k_1 = k_3 = 1$ et $d_1 = k_4 = 0.5$.

1. Il existe 2 conflits : $< T3, \{t_1, t_2, t_3\} >$ où $t_3 < t_1 < t_2$ et $< T4, \{t_3, t_4\} >$ où $t_3 < t_4$. Le marquage $\bar{m}^c$ est le suivant : $\bar{m} = (m(T3), m(T4), m(D2), m(D3)) = (1; 5; 0; 0)$. Initialement la transition discrète t_1 est permise, on a donc $TOS = (D1(0.5))$.
2. $D2(0) = \emptyset$
3. $D1(0) = \emptyset$
4. Il n'y a qu'un seul sous-réseau continu.

5. Initialement, seules les places T3 et T4 sont remplies. Ainsi, la transition t_4 ne peut pas tirer alors que les transitions t_2 et t_3 tirent à leur vitesse maximale. La transition t_3 peut tirer car la transition t_2 remplie continuellement la place D3. Nous avons alors $v(t_2) = V(t_2) = 1$ et $v(t_3) = V(t_3) = 1$.
6. $TOS = (D1(0.5), C1(1), D2(0.5))$. T3 étant dégradé à la vitesse de 1, sa place se videra donc au temps $t = 1$
7. Les prochains événements ont lieu au temps 0.5 . Il s'agit du tir de t_1 et de l'événement D2 associé à la place T3 qui va passer sous le seuil 0.5 permettant à la transition discrète t_1 de tirer.
2. $D2(0.5) = \emptyset$
3. $D1(0.5) \neq \emptyset$, Tir de t_1, on obtient le marquage $(0.5; 4.5; 0.5; 0.5)$
5. Toutes les transitions sont permises, on donc $v(t_2) = v(t_3) = 1$ et $v(t_4) = V(t_4) = 0.5$.
6. La place T3 se vide à la vitesse 0.5, elle atteindra donc 0 au temps 1.5 (C1(1.5)), c'est le prochain événement...

1.4.4 Résolutions de conflits par priorité en biologie

Au niveau biologique, la résolution de conflits par priorité est tout à fait pertinente. En effet, si nous prenons l'exemple d'une entité biologique responsable de l'induction de plusieurs gènes (comme T3 avec RHT, D2 et D3), la connaissance des constantes d'affinité et/ou des constantes de liaison maximale (notées MBC dans la littérature) suffit à ordonner les différentes réactions par priorité. Dans notre exemple, T3 a plus d'affinité pour son récepteur que pour les enzymes. L'induction de RHT par T3 aurait donc le premier niveau de priorité. On compare ensuite les constantes associées aux deux enzymes pour terminer l'ordonnancement. Lorsque ces constantes sont inconnues, on teste toutes les possibilités : $RHT < D2 < D3$ ou $RHT < D3 < D2$.

De même, dans le cas de transports de molécules dans une cellule, les transports actifs sont prioritaires sur les transports passifs. Dans le cas de réactions enzymatiques, l'ordre de priorité est déterminé suivant les valeurs des activités des différentes enzymes impliquées.

La résolution de conflits par priorité est donc appropriée dans de nombreux cas biologiques.

1.5 Démarche adoptée pour la vérification

Les THPN permettent de construire des graphes d'évolution qui sont des représentations formelles des exécutions des THPN (avec les restrictions mentionnées et les règles de résolution de conflits). Ces graphes d'évolution font intervenir de manière explicite des tops d'horloge où le comportement change. Dans les chapitres suivants, on utilisera cette représentation formelle des traces pour confronter un THPN à une propriété temporelle exprimée elle aussi dans un langage formel (logique temporelle).

Chapitre 2

Une syntaxe et une sémantique pour les modèles en temps continu

Ce chapitre propose un cadre formel aux modèles en temps continu pour THPN. Pour ces modèles, on considère les *traces en temps continu* qui associent à chaque variable du modèle une valeur, mais on considère aussi les *traces temporisées discrètes* qui correspondent intuitivement à des observations sur le modèle. Ainsi, ces traces associent des valeurs aux variables du modèle mais simplement à des tops d'horloge bien spécifiques.

Nous proposons donc une logique temps réel, que nous avons appelée logique d'évolution en temps continu et notée CTEL pour Continuous-Time Evolution logic.

Après avoir donner un cadre de formalisation des modèles en temps continu et avoir défini la syntaxe et la sémantique, nous montrons dans la dernière section que les THPN et les graphes d'évolution peuvent être plongés dans ce cadre formel.

2.1 Modèles en temps continu

Les modèles en temps continu ainsi que les traces temporisées discrètes sont définis sur une signature Σ récapitulant l'ensemble des symboles que nous pouvons manipuler : les "variables" qui représentent des quantités réelles variant au cours du temps et les "propositions" qui peuvent aussi varier au cours du temps.

Définition 2.1 (Signature)**.** *Une signature pour CTEL est un couple* $\Sigma = (V, Pr)$ *où* V *et* Pr *sont respectivement un ensemble de variables et un ensemble de propositions.*

Etant donné que le formalisme des THPN a été détaillé dans le chapitre précédent, nous illustrons nos définitions sur les modèles THPN. Néanmoins, il faut garder en tête que le cadre logique des modèles inclut le cadre des THPN et nous montrons cette inclusion dans la dernière section.

Intuitivement, l'ensemble de variables utilisées par le THPN est composé de $m(P)$, le marquage de chaque place et de $v(T)$ la vitesse de chaque transition et l'ensemble des propositions est composé des événements ayant lieu $(Tir(T_1), NoEvent, ...)$.

Définition 2.2 (Modèle en temps continu)**.** *Etant donnée une signature CTEL* $\Sigma = (V, Pr)$*, un modèle en temps continu* M *est défini par un ensemble* $\pi \subset Pr \times \mathbb{R}^+$ *et une*

fonction $\mu : (V \amalg \mathbb{R}) \times \mathbb{R}^+ \to \mathbb{R}$ *(où* $\amalg$ *représente l'union disjointe) telle que si* r *est une constante* $r \in \mathbb{R}$*, alors pour tout temps* $t \in \mathbb{R}^+$*,* $\mu(r,t) = r$*.*
L'ensemble des modèles défini sur la signature Σ *est noté* $Mod(\Sigma)$*.*

Un modèle en temps continu M associe, grâce à sa fonction μ, une valeur à chaque variable V de la signature Σ pour chaque temps $t \in \mathbb{R}^+$. De plus, l'ensemble π donne un ensemble de couples (p,t) de propositions p vraies au temps t. Par exemple, le THPN de la Figure 1.14 est tel que $\mu(m(T3),\ 0) = 1$ et $(Tir(T_1),\ 0.5) \in \pi$.

Nous distinguons deux types d'atomes : les atomes instantanés, notés α (Définition 2.3) et les atomes généraux (Définition 2.6). Les atomes instantanés ne correspondent pas uniquement à un ensemble de propositions, ils intègrent également un aspect quantitatif en associant une valeur à chaque variable. Ces atomes représentent une sous-classe des atomes généraux qui intègrent également des notions de temps.

Définition 2.3 (Atomes instantanés)**.** *Etant donnée une signature CTEL* $\Sigma = (V, Pr)$*, un atome instantané* α *est une expression de la forme* $v \geq v'$*,* p *ou leur négation, où* $v, v' \in (V \amalg \mathbb{R})$ *et* $p \in Pr$*.*

Les atomes instantanés permettent d'exprimer des propriétés quantitatives des variables du modèle, comme par exemple $m(P_1) \geq 4$ et des propriétés qualitatives grâce aux propositions comme par exemple $Tir(T_1)$. On s'autorise toutes les combinaisons $(\leq, <, >, =, \neq)$: par exemple, $m(P_1) < 4$ signifiera $\neg(m(P_1) \geq 4)$.

Nous définissons tout d'abord les règles de satisfaction d'un atome instantané pour un modèle en temps continu M.

Définition 2.4 ($\models_{t_i}$)**.** *La relation de satisfaction* $\models_{t_i}$ *au temps* $t_i \in \mathbb{R}^+$ *est défini comme suit sur les atomes instantanés :*

- $M \models_{t_i} p$*, ssi* $(p, t_i) \in \pi$
- $M \models_{t_i} v \geq v'$*, ssi* $\mu(v, t_i) \geq \mu(v', t_i)$
- $M \models_{t_i} \neg\alpha$*, ssi* $M \not\models_{t_i} \alpha$

où M *est un modèle en temps continu,* α *est un atome instantané de la forme* p *ou* $v \geq v'$*,* p *appartient à* Pr*,* v *et* v' *appartiennent à* $V \amalg \mathbb{R}$*.*

Remarque 2.1. *A un temps* t_i*, chaque* $v \in (V \amalg \mathbb{R})$ *a une unique valeur réelle, ce qui rend raisonnable le dernier item de la définition 2.4 (pas de substitution).*

Pour chaque atome instantané α, nous introduisons deux horloges, l'horloge historique x_α et l'horloge de prédiction y_α [37]. La valeur d'une horloge historique x_α est le temps écoulé depuis la dernière occurrence de α. La valeur de l'horloge de prédiction y_α est le temps à attendre jusqu'à la prochaine occurrence de α. Par exemple, la *contrainte d'horloge* $y_p \leq 3$ signifie que p sera vrai dans au plus 3 unités de temps et $x_q = 4$ signifie que q a été observé il y a 4 unités de temps. L'introduction des horloges x_α et y_α nous permet à présent de définir l'ensemble des termes sur la signature Σ, noté T_Σ.

Définition 2.5 (Termes)**.** *Un terme sur une signature* Σ *est une variable* $v \in V$ *ou une constante* $c \in \mathbb{R}$ *ou une expression de la forme* x_α *ou une expression de la forme* y_α *où* α *est un atome instantané. Nous notons* T_Σ *l'ensemble des* Σ*-termes.*

Définition 2.6 (Atome). *Etant donnée une signature $\Sigma = (V, Pr)$, un atome est une expression de la forme $r \geq r'$, p ou leur négation, où r, r' sont des termes ($r, r' \in T_\Sigma$) et p est une proposition ($p \in Pr$), telle que si r (resp. r') est de la forme x_α ou y_α, alors l'autre terme r' (resp. r) est nécessairement une constante rationnelle $n \in \mathbb{Q} \subset \mathbb{R}$.*

On remarque que les atomes instantanés sont des atomes particuliers.

Nous avons, à présent, tous les éléments en main pour présenter la logique d'évolution en temps continu, CTEL, ainsi que la relation de satisfaction liant un modèle en temps continu M et une formule CTEL. CTEL est basée sur la syntaxe et la sémantique de la logique Event-Clock définie par Raskin et Schobbens [36]. La différence réside dans l'extension des atomes usuels (propositions) aux atomes généraux.

Définition 2.7 ($For(\Sigma)$). *L'ensemble des formules CTEL bien formées sur la signature Σ est composée d'atomes, des connecteurs usuels $\neg$, $\vee$, $\wedge$, $\Rightarrow$, des opérateurs temporaux qualitatifs Next ($\bigcirc$), Previous ($\ominus$), Until (U) et Since (S) et des opérateurs temps réel : l'opérateur de prédiction $\rhd$ et l'opérateur historique $\lhd$:*

$$f ::= a | \neg f | \bigcirc f | \ominus f | f_1 \wedge f_2 | f_1 \vee f_2 | f_1 \Rightarrow f_2 | f_1 U f_2 | f_1 S f_2 | \lhd_{\sim n} \alpha | \rhd_{\sim n} \alpha$$

où a est un atome, α est un atome instantané, $\sim$ appartient à $\{=, <, >, \leq, \geq\}$, f, f_1 et f_2 sont des formules bien formées et n est un rationnel.

Par exemple, supposons que nous étudiions le cycle cellulaire composé de 4 phases : G_1, G_2, S et mitose. Nous introduisons alors dans la signature les propositions G_1, G_2, S et $Mitose$ qui sont vraies si le système est dans la phase correspondante. Ainsi, la formule "$G_2 \Rightarrow (\lhd_{\leq 10} S \wedge \rhd_{\geq 2} Mitose)$" est un exemple de formule CTEL bien formée. Elle signifie que *"si la cellule est en phase G_2, alors la phase S a eu lieu nécessairement au cours des dernières 10 heures et G_2 est suivi de la mitose qui aura lieu dans au moins 2 heures"*.

Bien que les THPN assignent une valeur à chaque variable pour tout temps $t \in \mathbb{R}^+$, le graphe d'évolution choisit des tops d'horloge spécifiques définissant une suite temporelle. Etant donné que l'étude d'un modèle en temps continu se fait au travers de ses traces temporisées discrètes, la relation de satisfaction entre un modèle en temps continu et une formule CTEL utilise la notion de suite temporelle.

Définition 2.8 (Suite temporelle). *Une suite temporelle h est une suite infinie de temps t_i strictement croissants et divergents, c'est-à-dire pour tout temps $t \in \mathbb{R}^+$, il existe un entier $i \in \mathbb{N}$ tel que $t_i > t$.*

Etant donnée une suite temporelle h, nous définissons une fonction $eval_M^h$ qui évalue un terme dans un modèle continu M sur une suite de temps h. Cette fonction correspond à la restriction à h de la fonction μ du modèle en temps continu (Définition 2.2).

Définition 2.9 (eval). *Etant donné un Σ-modèle M et une suite de temps h, la fonction qui évalue un terme r dans un modèle M sur une suite de temps h, $eval_M^h : T_\Sigma \times |h| \rightarrow \mathbb{R} \cup \{\perp\}$, où $|h|$ est l'ensemble des temps de h, est définie comme suit :*

- $eval_M^h(v, t_i) = \mu(v, t_i)$ *où* $v \in (V \amalg \mathbb{R})$ *et* $t_i \in \mathbb{R}^+$

- $eval^h_M(x_\alpha, t_i) = \begin{pmatrix} t_i - t_j & si\ \exists t_j,\ 0 \leq t_j < t_i |\ M \models_{t_j} \alpha \\ & et\ \forall t_k,\ t_j < t_k < t_i,\ M \not\models_{t_k} \alpha \\ \bot & sinon \end{pmatrix}$
- $eval^h_M(y_\alpha, t_i) = \begin{pmatrix} t_j - t_i & si\ \exists t_j,\ t_j > t_i |\ M \models_{t_j} \alpha \\ & et\ \forall t_k,\ t_i < t_k < t_j,\ M \not\models_{t_k} \alpha \\ \bot & sinon \end{pmatrix}$

Ainsi, un modèle satisfait au temps t_i d'une suite temporelle h un atome de la forme $r \geq r'$, avec $r, r' \in T_\Sigma$ si l'évaluation de ces deux termes par la fonction $eval^h_M$ respecte l'inégalité et il satisfait une proposition $p \in Pr$ si p est vraie au temps t_i. De ces formes de base, nous définissons inductivement la relation de satisfaction pour toute formule CTEL.

Définition 2.10 ($\models^h_{t_i}$). *La relation de satisfaction $\models^h_{t_i} \subset Mod(\Sigma) \times For(\Sigma)$ où $t_i \in |h|$, est définie inductivement comme suit :*

- $M \models^h_{t_i} p$, *où* $p \in Pr$, *ssi* $(p, t_i) \in \pi$
- $M \models^h_{t_i} r \geq r'$ *ssi* $eval^h_M(r, t_i) \geq eval^h_M(r', t_i)$
- $M \models^h_{t_i} \neg f$ *ssi* $M \not\models^h_{t_i} f$
- $M \models^h_{t_i} f_1 \wedge f_2$ *ssi* $M \models^h_{t_i} f_1$ *et* $M \models^h_{t_i} f_2$
- $M \models^h_{t_i} f_1 \vee f_2$ *ssi* $M \models^h_{t_i} f_1$ *ou* $M \models^h_{t_i} f_2$
- $M \models^h_{t_i} \bigcirc f$ *ssi* $M \models^h_{t_{i+1}} f$
- $M \models^h_{t_i} \ominus f$ *ssi* $i > 0$ *et* $M \models^h_{t_{i-1}} f$
- $M \models^h_{t_i} f_1 U f_2$ *ssi* $\exists t_j \geq t_i \mid M \models^h_{t_j} f_2$ *et* $\forall t_k \in [t_i, t_j[,\ M \models^h_{t_k} f_1$
- $M \models^h_{t_i} f_1 S f_2$ *ssi* $\exists t_j \in [0, t_i] \mid M \models^h_{t_j} f_2$ *et* $\forall t_k \in]t_j, t_i],\ M \models^h_{t_k} f_1$
- $M \models^h_{t_i} \rhd_{\sim n} f$ *ssi* $\exists t_j > t_i\ |M \models^h_{t_j} f$ *et* $\forall t_k \in]t_i, t_j[, M \not\models^h_{t_k} f$ *et* $t_j - t_i \sim n$, *avec* $\sim \in \{<, >, =, \geq, \leq\}$ *et* $n \in \mathbb{Q}^+$
- $M \models^h_{t_i} \lhd_{\sim n} f$ *ssi* $\exists t_j \in [0, t_i[\ |M \models^h_{t_j} f$ *et* $\forall t_k \in]t_j, t_i[, M \not\models^h_{t_k} f$ *et* $t_i - t_j \sim n$, *avec* $\sim \in \{<, >, =, \geq, \leq\}$ *et* $n \in \mathbb{Q}^+$

Notre but est de vérifier que notre modèle M satisfait une certaine propriété exprimée en CTEL, c'est-à-dire on veut savoir si $M \models f$. Lorsqu'aucune suite temporelle et qu'aucun temps t_i n'est précisé, ceci est équivalent à regarder toutes les suites temporelles au temps t_0. Sans perte de généralité, on pose $t_0 = 0$.

Définition 2.11. *Etant donné un modèle en temps continu M et une formule CTEL f, $M \models f$ si et seulement si pour toute suite de temps h, $M \models^h_{t_0} f$, avec $t_0 = 0$.*

2.2 Traces Discrètes

Quand nous étudions un système biologique, nous n'observons pas un ensemble de formules aussi large que $For(\Sigma)$. Nous distinguons un sous-ensemble de $For(\Sigma)$, dont les éléments sont appelés des observations.

Définition 2.12 ($Obs(\Sigma)$). *L'ensemble des observations bien formées sur la signature Σ est le sous-ensemble $Obs(\Sigma)$ de $For(\Sigma)$ défini par : $f ::= a | \neg f | f_1 \wedge f_2 | f_1 \vee f_2$ où a est un atome, f, f_1 et f_2 sont des observations bien formées.*

Une observation correspond donc à un état du système à un temps donné. Par exemple, $m(P_1) \geq 4 \wedge Tir(T_1) \wedge \neg NulMark(P_3) \wedge x_{NulMark(P_3)} = 2$ est une observation bien formée.

Nous introduisons à présent une notation importante que nous utiliserons par la suite, elle permet la preuve de la correction et de la complétude de notre procédure de Model-Checking (voir Chapitre 3).

Notation 2. *Trois types d'atomes se distinguent : les propositions $p \in Pr$ ou leur négation, notées α_p ; les autres atomes instantanés $v \geq v'$ ou leur négation, notés α_v et les contraintes d'horloges, notées a_c de la forme $z_\alpha \geq c$, $c \geq z_\alpha$ ou leur négation, où c est une constante rationnelle et z_α est une horloge (historique ou de prédiction).*
La définition des observations (Définition 2.12) se réfère à la logique propositionnelle. Par conséquent, toute observation bien formée peut s'écrire sous forme normale conjonctive. Grâce à la propriété de commutativité et d'associativité du connecteur $\wedge$, toute observation bien formée f peut s'écrire sous forme de 3 sous-observations f_1, f_2 et f_3, telles que $f \equiv f_1 \wedge f_2 \wedge f_3$ où :

- $f_1 \equiv \bigwedge(\vee\alpha_p)$
- $f_2 \equiv \bigwedge(\vee\alpha_v)$
- $f_3 \equiv \bigwedge(\vee a_c)$

Par la suite, $(\vee\alpha_p)$, $(\vee\alpha_v)$ et $(\vee a_c)$ seront nommés des up-littéraux. *Un up-littéral correspond à une disjonction d'atomes. On notera $UpL(f)$ l'ensemble des up-littéraux de l'observation f.*

Reprenons l'observation précédente :

$$f \equiv m(P_1) \geq 4 \wedge Tir(T_1) \wedge \neg NulMark(P_3) \wedge x_{NulMark(P_3)} = 2$$

La notation ci-dessus nous permet d'écrire :

- $f_1 \equiv Tir(T_1) \wedge \neg NulMark(P_3)$
- $f_2 \equiv m(P_1) \geq 4$
- $f_3 \equiv x_{NulMark(P_3)} = 2$

et $f \equiv f_1 \wedge f_2 \wedge f_3$

Intuitivement, une observation correspond à une conjonction d'observations atomiques. Par conséquent, l'utilisation de la disjonction $\vee$ ne semble pas nécessaire pour les propositions α_p. En effet, soit on observe la proposition p_1, soit on observe $\neg p_1$, mais on n'observe pas $p_1 \vee \neg p_1$.

La disjonction n'est ajoutée que dans le but de décrire tous les signes comparatifs $(\leq, \geq, <, >, =)$ intervenant dans les atomes de type α_v et a_c. En effet, pour décrire $v \neq v'$ uniquement avec le comparateur $\geq$, la disjonction $\vee$ est requise : $\neg(v \geq v') \vee \neg(v' \geq v)$. Néanmoins, par souci de généralité la disjonction est également maintenue pour les propositions.

Formellement une trace temporisée discrète bien formée se définit comme suit :

Définition 2.13 (Trace temporisée discrète bien formée)**.** *Une trace temporisée discrète bien formée τ sur un modèle en temps continu M défini sur la signature $\Sigma = (V, Pr)$ est de la forme $\tau = \{(\varphi_i, t_i)\}_{i\in\mathbb{N}}$, où les φ_i sont des observations bien formées et où $h_\tau = (t_i)_{i\in\mathbb{N}}$ est une suite temporelle telle que :*

- $\forall v \in V,\ \forall i \in \mathbb{N},\ \mu(v,t)$ *est continu, et constant ou strictement monotone pour t variant dans l'intervalle $[t_i, t_{i+1}[$,*
- $\forall p \in Pr,\ \forall i \in \mathbb{N},\ \pi(p,t)$ *est constant pour t variant dans l'intervalle $[t_i, t_{i+1}[$.*

L'observation φ_i au temps t_i est imposée "complète" pour les propositions, c'est-à-dire au temps t_i, soit p soit $\neg p$ est observée, pour tout proposition $p \in Pr$.

Par exemple, $\tau = (\neg Fire(T_1) \wedge (m(T4) \geq 2.0),\ 0)(Fire(T_1) \wedge (m(T4) \geq 2.0),\ 2.5)...$
est une trace discrète bien formée.

Un modèle en temps continu est étudié au travers de ses traces discrètes τ. Nous devons donc d'abord définir les conditions de satisfaction d'une trace discrète par un modèle en temps continu.

Définition 2.14. *La relation de satisfaction entre un Σ-modèle M et une trace temporisée discrète bien formée $\tau = \{(\varphi_i, t_i)\}_{i\in\mathbb{N}}$ est définie par : $M \models \tau$ ssi $\forall i \in \mathbb{N},\ M \models^{h_\tau}_{t_i} \varphi_i$.*

Pour vérifier si un modèle en temps continu satisfait une propriété exprimée en CTEL, nous vérifions si les traces discrètes obtenues du modèle en temps continu satisfont la propriété. Une relation de satisfaction entre une trace discrète τ et une formule CTEL est donc requise. Soit $\wr$ cette relation de satisfaction.

Définition 2.15. *La relation de satisfaction $(\tau, i)\wr f$ à une position $i \in \mathbb{N}$ où $\tau = \{(\varphi_i, t_i)\}_{i\in\mathbb{N}}$ est une trace discrète bien formée et f, f_1 et f_2 sont des formules CTEL, est définie inductivement comme suit :*

- $(\tau, i)\wr p$, *où $p \in Pr$, ssi $UpL(p) = \{p\} \subseteq UpL(\varphi_i)$*
- $(\tau, i)\wr \neg p$, *où $p \in Pr$, ssi $(\tau, i) \not\wr p$ ssi $UpL(p) \nsubseteq UpL(\varphi_i)$ ssi $UpL(\neg p) \subseteq UpL(\varphi_i)$*
- $(\tau, i)\wr r \geq r'$ *ssi $\exists M | M \models \tau$ et $M \models^{h_\tau}_{t_i} r \geq r'$*
- $(\tau, i)\wr \neg(r \geq r')$ *ssi $\exists M | M \models \tau$ et $M \models^{h_\tau}_{t_i} \neg(r \geq r')$*
- $(\tau, i)\wr \neg f$ *avec f non atomique, ssi $(\tau, i) \not\wr f$*
- $(\tau, i)\wr f_1 \wedge f_2$ *ssi $(\tau, i)\wr f_1$ et $(\tau, i)\wr f_2$*
- $(\tau, i)\wr f_1 \vee f_2$ *ssi $(\tau, i)\wr f_1$ ou $(\tau, i)\wr f_2$*
- $(\tau, i)\wr \bigcirc f$ *ssi $(\tau, i+1)\wr f$*
- $(\tau, i)\wr \ominus f$ *ssi $i > 0$ et $(\tau, i-1)\wr f$*
- $(\tau, i)\wr f_1 U f_2$ *ssi $\exists j \geq i \mid (\tau, j)\wr f_2$ et $\forall k \in [i, j[,\ (\tau, k)\wr f_1$*
- $(\tau, i)\wr f_1 S f_2$ *ssi $\exists j \in [0, i] \mid (\tau, j)\wr f_2$ et $\forall k \in]j, i],\ (\tau, k)\wr f_1$*
- $(\tau, i)\wr \rhd_{\sim n} f$ *ssi $\exists j > i \mid (\tau, j)\wr f$ et $\forall k \in]i, j[, (\tau, k)\wr \neg f$ et $t_j - t_i \sim n$, avec $\sim \in \{<, >, =, \geq, \leq\}$ et $n \in \mathbb{Q}^+$*
- $(\tau, i)\wr \lhd_{\sim n} f$ *ssi $\exists j \in [0, i[\mid (\tau, j)\wr f$ et $\forall k \in]j, i[, (\tau, k)\wr \neg f$ et $t_i - t_j \sim n$, avec $\sim \in \{<, >, =, \geq, \leq\}$ et $n \in \mathbb{Q}^+$*

L'utilisation de traces discrètes comme abstraction des modèles en temps continu requiert que les traces discrètes d'un modèle en temps continu M satisfont une propriété si

M la satisfait aussi. Le lemme suivant prouve que si un modèle satisfait une trace discrète et une formule CTEL au temps t_i alors la trace discrète satisfait la formule CTEL au même temps.

Lemme 2.1. *Si $M \models \tau$ avec $\tau = \{(\varphi_i, t_i)\}_{i\in\mathbb{N}}$ et $M \models^{h_\tau}_{t_i} f$ où f est une formule CTEL, alors $(\tau, i) \sim f$.*

Preuve. La preuve est faite inductivement sur la formule f.

- Atomes :
  - si f est de la forme p avec $p \in Pr$
    Supposons que $M \models \tau$, $M \models^{h_\tau}_{t_i} p$ et que $(\tau, i) \not\sim p$. Nous avons $(\tau, i) \not\sim p$ alors $UpL(\neg p) \subseteq UpL(\varphi_i)$ où φ_i est une observation sous forme conjonctive. $M \models \tau$ alors $M \models^{h_\tau}_{t_i} \varphi_i$ et en particulier $M \models^{h_\tau}_{t_i} \neg p$. Contradiction.
  - si f est de la forme $\neg p$ avec $p \in Pr$
    Supposons que $M \models \tau$, $M \models^{h_\tau}_{t_i} \neg p$ et que $(\tau, i) \sim p$. Comme on a $(\tau, i) \sim p$ alors on peut en déduire $UpL(p) \subseteq UpL(\varphi_i)$ où φ_i est une observation sous forme conjonctive. $M \models \tau$ alors $M \models^{h_\tau}_{t_i} \varphi_i$ et en particulier $M \models^{h_\tau}_{t_i} p$. Contradiction.
  - si f est de la forme $r \geq r'$ avec $r, r' \in T_\Sigma$
    $M \models \tau$ et $M \models^{h_\tau}_{t_i} r \geq r'$. Alors, par définition 2.15, $(\tau, i) \sim r \geq r'$.
  - si f est de la forme $\neg(r \geq r')$ avec $r, r' \in T_\Sigma$
    $M \models \tau$ et $M \models^{h_\tau}_{t_i} \neg(r \geq r')$. Alors, par définition 2.15 $(\tau, i) \sim \neg(r' \geq r)$.
- Supposons que la propriété soit vraie pour toute sous-formule de f.
  - si f est de la forme $\bigcirc\psi$
    $M \models \tau$ et $M \models^{h_\tau}_{t_i} \bigcirc\psi$. Par définition 2.10, $M \models^{h_\tau}_{t_{i+1}} \psi$. Par hypothèse d'induction, $(\tau, i+1) \sim \psi$. Par définition 2.15, $(\tau, i) \sim \bigcirc\psi$.
    (similaire si f est de la forme $\ominus\psi$)
  - si f est de la forme $\neg f_1$
    On suppose que $M \models \tau$ et que $(\tau, i) \not\sim f_1$. Par contraposée de l'hypothèse d'induction, on obtient la propriété voulue.
  - si f est de la forme $\psi_1 \wedge \psi_2$ (resp. $\psi_1 \vee \psi_2$)
    $M \models \tau$ et $M \models^{h_\tau}_{t_i} \psi_1 \wedge \psi_2$ (resp. $M \models^{h_\tau}_{t_i} \psi_1 \vee \psi_2$). Par définition 2.10, $M \models^{h_\tau}_{t_i} \psi_1$ et (resp. ou) $M \models^{h_\tau}_{t_i} \psi_2$. Par hypothèse d'induction, $(\tau, i) \sim \psi_1$ et (resp. ou) $(\tau, i) \sim \psi_2$. Par définition 2.15, $(\tau, i) \sim \psi_1 \wedge \psi_2$ (resp. $(\tau, i) \sim \psi_1 \vee \psi_2$).
  - si f est de la forme $\psi_1 U \psi_2$
    $M \models \tau$ et $M \models^{h_\tau}_{t_i} \psi_1 U \psi_2$. Par définition 2.10, $\exists t_j \geq t_i \mid M \models^{h_\tau}_{t_j} f_2$ et $\forall t_k \in [t_i, t_j[$, $M \models^{h_\tau}_{t_k} f_1$. Par hypothèse d'induction, $\exists j \geq i \mid (\tau, j) \sim f_2$ et $\forall k \in [i, j[$, $(\tau, k) \sim f_1$. Par définition 2.15, $(\tau, i) \sim \psi_1 U \psi_2$.
    (similaire si f est de la forme $\psi_1 S \psi_2$)
  - si f est de la forme $\rhd_{\sim n}\psi$
    $M \models \tau$ et $M \models^{h_\tau}_{t_i} \rhd_{\sim n}\psi$. Par définition 2.10, $\exists t_j > t_i \mid M \models^{h_\tau}_{t_j} \psi$ et $\forall t_k \in]t_i, t_j[, M \models^{h_\tau}_{t_k} \neg\psi$ et $t_j - t_i \sim n$. Par hypothèse d'induction, $\exists j > i \mid (\tau, j) \sim \psi$ et $\forall k \in]i, j[, (\tau, k) \sim \neg\psi$ et $t_j - t_i \sim n$. Par définition 2.15, $(\tau, i) \sim \rhd_{\sim n}\psi$.
    (similaire si f est de la forme $\lhd_{\sim n}\psi$)

□

2.3 Application aux Réseaux de Petri Hybrides Temporisés

Les sections précédentes présentent les modèles en temps continu et les traces temporisées discrètes dans un cadre général. Dans cette section, nous montrons que les THPN et les traces des graphes d'évolution rentrent bien dans le cadre formel que nous venons de définir.

Les modèles en temps continu sont définis sur une signature Σ. La Définition 2.16 détaille la signature que nous utilisons pour les modèles THPN.

Définition 2.16 (Signature pour les modèles THPN)**.** *La signature $\Sigma = (V, Pr)$ est composée d'un ensemble de variables et d'un ensemble de propositions.*
L'ensemble de variables V contient :

- *une variable $m(P)$, appelée marquage de P, pour chaque place $P \in \mathcal{P}$,*
- *une variable $v(T)$ pour chaque transition continue $T \in T^C$, elle est appelée vitesse instantanée de T.*

L'ensemble des propositions Pr est uniquement composé de :

- *$Tir(T)$: proposition associée respectivement à un événement D1 d'une transition discrète T,*
- *$NulMark(P)$: proposition associée à un événement C1 d'une place continue P dont le marquage devient nul,*
- *$Th(P, x)$: proposition associée à un événement D2, où x représente le seuil atteint par la place continue P modifiant ainsi le degré de permission d'une transition discrète,*
- *$NoEvt$: proposition associée à la première transition du graphe d'évolution quand aucun événement n'a encore eu lieu.*

Remarque 2.2. *Noter que malgré leur notation, il s'agit de variables et pas de fonctions. De même, il s'agit de propositions et pas de prédicats. Celles-ci sont en nombre fini car le nombre de places et de seuils est fini.*

L'exécution d'un modèle THPN est étudiée par l'intermédiaire des traces finies ou infinies de ses graphes d'évolution. Dans ce qui suit, nous montrons que chaque trace d'un graphe d'évolution est une trace bien formée selon la Définition 2.13.

Lemme 2.2. *Chaque trace d'un graphe d'évolution est une trace discrète temporisée bien formée.*

Preuve.

- **Le marquage discret est constant dans un IB-state.**
  Le marquage d'une place discrète est modifiée par le tir de ses transitions entrantes ou sortantes. Si les transitions sont continues, leur tir ne peut modifier le marquage discret étant donné qu'une place discrète ne peut être connectée à une transition continue que par deux arcs de même poids : un entrant et l'autre sortant (voir page 27). Si la transition est discrète, leur tir correspondrait à un événement D1, ce qui mènerait à un changement d'IB-state.

- **Les vitesses de tir instantanées restent constantes dans un IB-state.**
  Les vitesses de tir instantanées des transitions continues ne peuvent être modifiées que si une des places prédécesseur se vide ou si une de ces places initialement vide se remplit. Le premier cas correspond à un événement C1 et mène à un changement d'IB-state. Le second cas est appelé dans la littérature un événement C2. Il a été montré [34] qu'un tel événement ne peut avoir lieu sans être précédé d'un événement C1 ou D1.
- **Le marquage continu évolue continuellement et linéairement dans un IB-state.**
  Dans un IB-state, le marquage continu ne peut être modifié que par le tir de transitions continues (le tir de transitions discrètes engendrant un changement d'IB-state). Par conséquent, seules les transitions continues entrantes et sortantes peuvent influencer le marquage continu. Nous pouvons donc nous restreindre à la partie continue du THPN. On rappelle que la quantité de ressources produites I_i dans la place continue P_i et la quantité de ressources consommées O_i de cette place sont données par :

$$\begin{aligned} I_i &= \sum_{T_j \in {}^\circ P_i} Post(T_j, P_i).v_j \text{ et} \\ O_i &= \sum_{T_k \in P^\circ{}_i} Pre(P_i, T_k).v_k \end{aligned}$$

  où v_j et v_k sont des vitesses instantanées de T_j et T_k.
  Nous déduisons la balance de la place P_i, notée B_i ($B_i = I_i - O_i$) qui correspond à la dérivée par rapport au temps du marquage de la place P_i, *i.e.* :

$$B_i(t) = \frac{dm_i(t)}{dt}.$$

  Par conséquent, $m_i(t + dt) = m_i(t) + B_i(t).dt$. Etant donné que les vitesses de tir instantanées sont constantes dans un IB-state, on en déduit que la balance $B_i(t)$ est aussi constante tant qu'on ne sort pas de l'IB-state. Par conséquent, le marquage évolue linéairement et continuellement dans un IB-state.

□

Les THPN correspondent donc à des modèles en temps continu et l'ensemble de leurs traces, obtenu par construction des graphes d'évolution, sont bien des traces temporisées bien formées.

Chapitre 3

Model-checking pour Réseaux de Petri Hybrides Temporisés

3.1 Démarche générale

Si l'on considère que le comportement d'un système est représenté comme une séquence d'états ou d'événements, l'ensemble des comportements possibles d'un système est un langage formel. Ce système peut alors être modélisé par un automate qui génère ce langage. Par ailleurs, une propriété biologique d'intérêt définit également un ensemble de comportements admissibles. Cette propriété peut également être modélisée par un automate. Le problème de vérification de la satisfaction d'une propriété par un système se réduit donc à tester l'inclusion des langages des deux automates.

La modélisation de systèmes temps-réel utilise souvent la classe des automates temporisés [38]. Ces automates fournissent un moyen facile et puissant d'annoter des automates par des contraintes temporelles, grâce à l'utilisation d'horloges. De nombreux outils modélisant ces automates ont été développés au cours de ces dernières années : Kronos [63], UppAAL [64],... Cependant, le problème de vérification général (*i.e.* l'inclusion de langage) n'est pas décidable pour les automates temporisés [38]. Nous nous sommes donc orientés vers une sous-classe de ces automates pour laquelle le problème d'inclusion de langage est décidable, il s'agit des automates Event-Clock développés par Alur *et al.* dans [37].

Notre procédure de model-checking repose alors sur la conversion en automates Event-Clock du modèle THPN et de la propriété f. Quatre étapes se distinguent (Figure 3.1) :

1. Le graphe d'évolution est converti en automate Event-Clock, qui peut générer n'importe quelle trace du THPN,
2. La négation de la propriété ($\neg f$) est également convertie en automate Event-Clock qui lui génère les traces ne satisfaisant pas la propriété f,
3. Les traces communes aux deux automates sont obtenues en effectuant leur produit,
4. On détermine enfin si cet ensemble de traces communes est vide ou non. S'il est vide on conclut que le modèle THPN satisfait la propriété, dans le cas contraire il ne la satisfait pas.

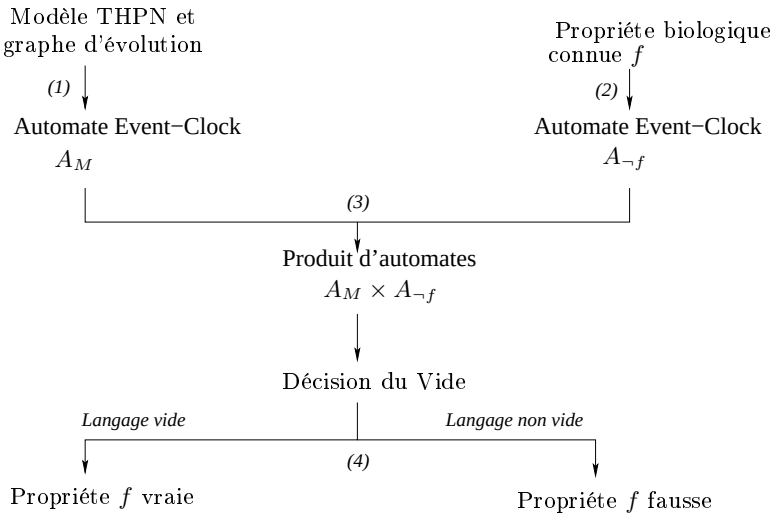

FIG. 3.1 – Procédure générale de model-checking pour les THPN

Le cadre formel présenté dans le chapitre précédent nous permet de prouver la correction et la complétude de chaque étape. Résumons les principales preuves à effectuer.

1. Nous proposons un algorithme de conversion d'un graphe d'évolution en automate Event-Clock. Nous devons donc montrer que l'ensemble des traces de l'automate résultant est exactement l'ensemble des traces du graphe d'évolution.
2. Un automate Event-Clock est obtenu à partir de la négation de la propriété $\neg f$. Il est nécessaire de montrer que l'ensemble des traces associées à $\neg f$ est correct et complet, c'est-à-dire que toute trace de l'automate satisfait $\neg f$ et que toute trace satisfaisant $\neg f$ est reconnue par l'automate.
3. Le produit de deux automates n'est exploitable que si celui-ci est correct et complet, c'est-à-dire qu'une trace est reconnue par l'automate produit si et seulement si elle est reconnue par les deux automates initiaux.
4. Enfin, nous montrerons que le langage de l'automate produit est vide si et seulement si le modèle THPN satisfait la propriété f.

Chaque section présente et détaille une étape de notre procédure.

3.2 Des graphes d'évolution aux automates «Event-Clock»

La première étape de la procédure de model-checking consiste à convertir les traces d'un graphe d'évolution en automate Event-Clock (*(1)* dans la Figure 3.1). Après avoir défini les automates Event-Clock, nous présenterons notre algorithme de conversion. Enfin, nous détaillerons l'ensemble de propriétés associées à cette étape.

3.2.1 Automates «Event-Clock»

Les automates Event-Clock reconnaissent des mots temporisés infinis. Ils font donc parti de la classe des automates de Büchi [65] reconnaissant des ω-langages [66].

Définition 3.1 (Automate Event-Clock)**.** *Un automate Event-Clock sur la signature* $\Sigma = (V, Pr)$ *est un 6-uplet* $A = (L, L_0, \Sigma, \mathcal{C}, E, \mathcal{F})$ *où :*

- L *est un ensemble fini de localisations et* $L_0 \subseteq L$ *est un sous-ensemble de localisations initiales,*
- $\Sigma = (V, Pr)$ *est la signature,*
- $\mathcal{C}$ *est un ensemble d'horloges historiques et de prédiction,*
- E *est un ensemble fini d'arcs. Un arc est un triplet* (l_1, ψ, l_2) *où* $l_1 \in L$ *est la localisation source,* $l_2 \in L$ *est la localisation cible, et* $\psi \in Obs(\Sigma)$ *est une observation décrivant la localisation* l_1*,*
- $\mathcal{F} = \{F_1, ..., F_n\}$ *où* $F_i \subseteq L$*, est un ensemble d'ensembles de localisations acceptantes (condition généralisée de Büchi)*

La Figure 3.2 montre un automate Event-Clock dont la localisation initiale est l_0 (flèche entrante) et l'ensemble d'ensembles de localisations finales n'est composé que d'un ensemble constitué de la localisation l_0 (double cercle). Cet automate reconnaît un ensemble de traces infinies, temporisées grâce à l'utilisation des contraintes d'horloges ($x_a < 1$ et $y_b > 2$). L'ensemble des traces temporisées discrètes forme le langage de l'automate Event-Clock (Définition 3.2).

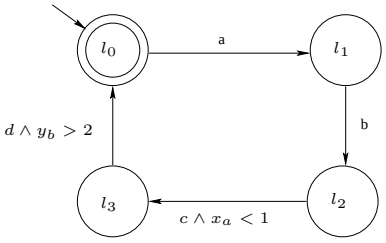

FIG. 3.2 – Exemple d'automate Event-Clock

Définition 3.2 (Trace et langage temporisés)**.**

- *Une trace temporisée* $\tau = \{(\varphi_i, ti)\}_{i \in \mathbb{N}}$ *est reconnue par un automate Event-Clock* $A = (L, L_0, \Sigma, \mathcal{C}, E, \mathcal{F})$ *s'il existe un chemin acceptant infini* $\gamma = l_0 \overset{\psi_0}{\rightarrow} l_1 \overset{\psi_1}{\rightarrow} ... l_n \overset{\psi_n}{\rightarrow} ...$ *où :*
  - *chaque* $l_i \in L$ *et* $l_0 \in L_0$*,*
  - $(l_i, \psi_i, l_{i+1}) \in E$ *et* $(\tau, i) \backsim \psi_i$ *et* $UpL(\psi_i) \subseteq UpL(\varphi_i)$*,*
  - *pour tout* $F_i \in \mathcal{F}$*, il existe une infinité de positions* j *telles que* $l_j \in F_i$*.*
- *Le langage temporisé d'un automate Event-Clock A, noté* $\mathcal{L}(A)$*, est l'ensemble des traces temporisées reconnues par A.*

Reprenons l'exemple de la Figure 3.2,

$$\tau = (a, 0)(b, 0.5)(c \wedge x_a < 1, 0.9)(d \wedge y_b > 2, 1.0)(a, 1.5)(b, 4.0)...$$

est un préfixe d'une trace temporisée discrète reconnue par l'automate Event-Clock. L'atome a est lu au temps 0, début de l'exécution. Il n'y a pas de contraintes de temps pour franchir le deuxième arc, l'atome b est lu au temps 0.5 dans notre exemple. Le troisième

arc possède la contrainte d'horloge $x_a < 1$ indiquant que le dernier a a été lu il y a moins d'une unité de temps. Ainsi, le franchissement du troisième arc au temps 0.9 assure le respect de cette contrainte, etc.

La contrainte sur les up-littéraux de la Définition 3.2 nous empêche d'écrire la trace τ sous sa forme la plus intuitive, notée τ' :

$$\tau' = (a, 0)(b, 0.5)(c, 0.9)(d, 1.0)(a, 1.5)(b, 4.0)...$$

Néanmoins, on remarque que le passage de τ' à τ ne pose pas de difficulté.

Remarque 3.1. *Les contraintes concernant les up-littéraux des arcs et des formules de τ ne sont pas restrictives étant donné qu'il est toujours possible de construire des traces satisfaisant de telles contraintes à partir d'un automate Event-Clock, comme nous l'avons montré sur l'exemple ci-dessus. Néanmoins, ces contraintes sont indispensables pour assurer les preuves de la procédure globale de model-checking.*

3.2.2 Algorithme de conversion

L'algorithme de conversion d'un graphe d'évolution en automate Event-Clock peut à présent être détaillé. On appellera A_M l'automate Event-Clock résultant de cette conversion. L'algorithme se décompose en quatre étapes, la première et la deuxième construisent l'ensemble des localisations de l'automate, la troisième détermine les localisations initiales et finales, enfin, la quatrième construit les arcs de l'automate.

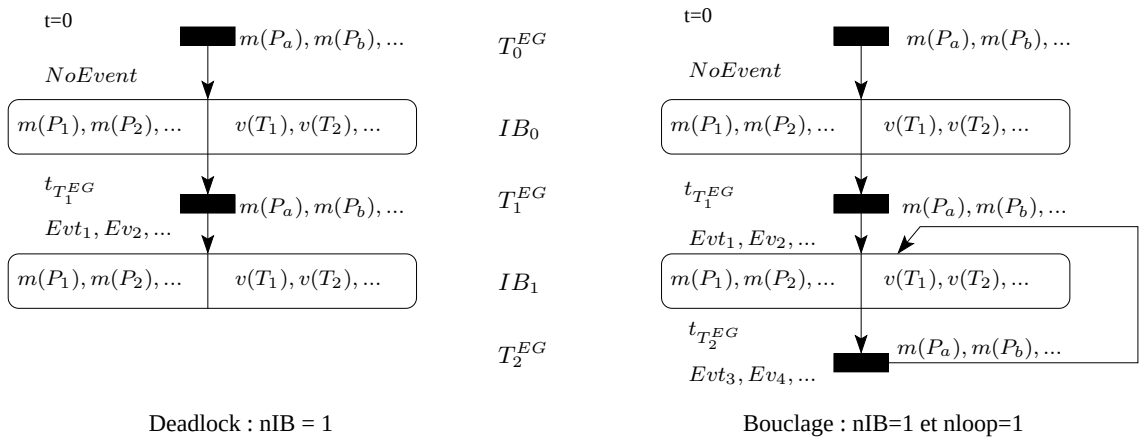

FIG. 3.3 – Rappel des notations utilisées dans les graphes d'évolution.

1. **Des IB-states aux localisations :** Chaque IB-state du graphe d'évolution donne une localisation dans l'automate Event-Clock A_M. A chacune de ces localisations nous associons une observation bien formée $\phi_1(IB_i)$ qui décrit l'état du THPN durant le temps du i^{ieme} IB-state (Voir Figure 3.3 pour notation). $\phi_1(IB_i)$ a la forme suivante où *val* associe à une variable sa valeur courante et où $I_P(T^{EG})_i^{i+1}$ correspond à l'intervalle borné d'une part par la valeur du marquage de la place continue P indiquée sur la transition T_i^{EG}, et d'autre part par la valeur du marquage de la

place continue P indiquée sur la transition T_{i+1}^{EG}. Dans le graphe d'évolution de la Figure 1.15, le marquage continu de la place T3 dans le premier IB-state est compris dans l'intervalle $]0.5, 1[$ où 1 est le marquage de la première transition et 0.5 celui de la deuxième transition du graphe d'évolution.

$$\phi_1(IB_i) \equiv \wedge \left(\begin{array}{l} \bigwedge_{P \in P^D} (m(P) = val(m(P))) \\ \bigwedge_{T \in T^C} (v(T) = val(v(T))) \\ \bigwedge_{P \in P^C} (m(P) \in I_P(T^{EG})_i^{i+1}) \end{array} \right)$$

2. **Des transitions aux localisations :** Chaque transition du graphe d'évolution donne aussi une localisation dans l'automate Event-Clock A_M. A chacune de ces localisations nous associons une observation bien formée $\phi_2(T_i^{EG})$ décrivant l'état du THPN quand l'exécution entre dans la i^{ieme} transition. $\phi_2(T_i^{EG})$ a la forme suivante, où val associe à chaque variable une valeur courante. Notons que x_{de} représente le temps écoulé depuis que le dernier événement de a eu lieu. Ce dernier événement est l'un des événements suivants : $NoEvt, Tir(T), NulMark(P), Th(P, x)$. Le délai $\Delta t_{T_i^{EG}} = t_{T_i^{EG}} - t_{T_{i-1}^{EG}}$ correspond au délai entre les deux transitions T_i^{EG} et T_{i-1}^{EG}, ce délai correspond au temps pendant lequel le système est dans l'IB-state IB_{i-1}.

$$\phi_2(T_i^{EG}) \equiv \wedge \left(\begin{array}{l} \bigwedge_{e \in Evt(T_i^{EG})} e \\ \bigwedge_{P \in P^C} (m(P) = val(m(P))) \\ \bigwedge_{T \in T^C} (v(T) = val(v(T))) \\ \bigwedge_{de \in Evt({}^{\circ\circ}T_i^{EG})} (x_{de} = t_{T_i^{EG}} - t_{T_{i+1}^{EG}}) \end{array} \right)$$

   Notons qu'en cas de bouclage plusieurs transitions peuvent être dans $Evt({}^{\circ\circ}T_i^{EG})$. Dans la Figure 3.3 la transition T_2^{EG} possède T_0^{EG} et T_2^{EG} comme prédécesseurs. Néanmoins, il n'y a pas d'ambiguïté étant donné que seule la première exécution passe par T_0^{EG} et que toutes les autres passent par T_2^{EG}.

3. **Localisations initiales et finales :** La localisation initiale est la localisation correspondant à la première transition du graphe d'évolution T_0^{EG}. Les localisations finales sont déterminées différemment selon que le graphe d'évolution se termine ou pas (deadlock ou bouclage). En cas de deadlock, l'ensemble d'ensembles de localisations finales se réduit à un singleton contenant uniquement la localisation correspondant au dernier IB-state. En cas de bouclage, l'ensemble des localisations impliquées dans la boucle forment un ensemble de localisations finales.

4. **Arcs :** Il y a un arc entre deux localisations s'il y a un arc entre l'IB-state et la transition associés aux localisations dans le graphe d'évolution. De plus, un arc boucle sur chaque localisation obtenue d'un IB-state. Ceci modélise le fait que l'IB-state est vrai durant un laps de temps. Enfin, un arc sortant d'une localisation l_i est étiqueté par la formule de la localisation l_i ($\phi_1(IB)$ ou $\phi_2(T^{GE})$).

Remarque 3.2. *L'algorithme présenté ci-dessus ne se réfère qu'à un unique graphe d'évolution représentant une unique trace. Cet algorithme est généralisable à un ensemble de graphes d'évolution obtenu par un ensemble exhaustif de résolutions de conflits. Lorsque l'on traite plusieurs gestions de conflits, nous obtenons plusieurs ensembles de localisations.*

3.2.3 Illustration biologique

La Figure 3.4 illustre l'algorithme de conversion d'un graphe d'évolution en automate Event-Clock A_M. L'algorithme se déroule comme suit :

Etape 2. **Des transitions aux localisations :** La première transition du graphe d'évolution donne la localisation l_0 dans l'automate A_M (Figure 3.4). On lui associe l'observation $\phi_2(T_0^{EG})$ suivante :

$$\phi_2(T_0^{EG}) \equiv (m(T3) = 1) \wedge (m(T4) = 5) \wedge (m(D2) = 0) \wedge (m(D3) = 0) \wedge \\ NoEvt \wedge (v(t_2) = 1) \wedge (v(t_3) = 1) \wedge (v(t_4) = 0)$$

La deuxième transition T_1^{EG} donne la localisation l_2 dans l'automate A_M, on lui associe l'observation $\phi_2(T_1^{EG})$ suivante :

$$\phi_2(T_1^{EG}) \equiv (m(T3) = 0.5) \wedge (m(T4) = 4.5) \wedge (m(D2) = 0.5) \wedge (m(D3) = 0.5) \wedge \\ Tir(T_1) \wedge (x_{NoEvt} = 0.5) \wedge (v(t_2) = 1) \wedge (v(t_3) = 1) \wedge (v(t_4) = 0.5)$$

La contrainte d'horloge $x_{NoEvt} = 0.5$ permet de temporiser l'automate en respectant les délais donnés par le graphe d'évolution.

Etape 1. **Des IB-states aux localisations :**
Le premier IB-state donne la localisation l_1 dans l'automate A_M. On associe à cette localisation l'observation $\phi_1(IB_0)$ suivante :

$$\phi_1(IB_0) \equiv m(T3) \in]0.5, 1[\wedge m(T4) \in]4.5, 5[\wedge m(D2) \in]0, 0.5[\wedge m(D3) \in]0, 0.5[\wedge \\ (v(t_2) = 1) \wedge (v(t_3) = 1) \wedge (v(t_4) = 0.5)$$

Le marquage de la place continue $T4$ vaut 5 au niveau de la transition T_0^{EG} et 4.5 au niveau de la transition T_1^{EG}. Etant donné la dynamique linéaire des places continues, on en déduit que durant l'IB-state IB_0, le marquage de $T4$ appartient à l'intervalle $]4.5, 5[$.

Etape 3. **Localisations initiales et finales :** L'unique localisation initiale correspond à la première transition du graphe d'évolution T_0^{EG}, il s'agit donc de la localisation l_0 (flèche entrante). Le graphe d'évolution étudié se termine par deadlock, par conséquent l'ensemble d'ensembles de localisations finales se restreint à un singleton formé de la localisation associée au dernier IB-state, c'est-à-dire l_8 (double cercle).

Etape 4. **Arcs :** Il existe un arc entre deux localisations l_i et l_{i+1} s'il y a un arc équivalent dans le graphe d'évolution. Nous utilisons donc des arcs pour relier les localisations l_0 à l_1, de l_1 à l_2, de l_2 à l_3,... De plus, les localisations provenant des IB-states possèdent des arcs bouclant sur ces localisations. Un arc bouclant est donc ajouté aux localisation l_1, l_3, l_5...

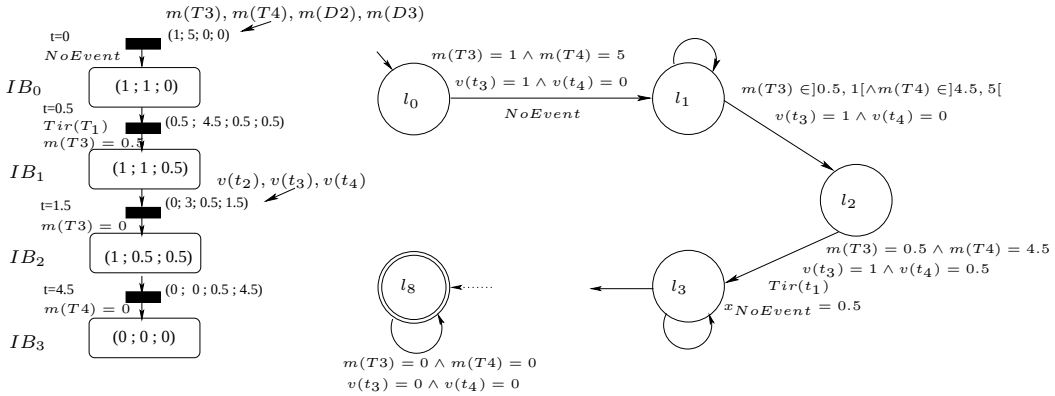

FIG. 3.4 – Conversion d'un graphe d'évolution en un automate Event-Clock A_M.

L'automate Event-Clock du modèle THPN A_M est alors obtenu. Pour pouvoir l'utiliser dans la suite de notre procédure, nous devons montrer que l'ensemble des traces de A_M est exactement celui du graphe d'évolution, c'est-à-dire que notre algorithme n'a pas modifié les traces discrètes du modèle THPN.

3.2.4 Propriétés

L'algorithme proposé dans la section 3.2.2 fournit un automate Event-Clock dont la sémantique est la même que celle décrite dans les THPN. Les preuves de ces égalités sont développées dans cette section (Lemme 3.3).

3.2.4.1 Traces discrètes temporisées du graphe d'évolution

Le graphe d'évolution fournit un ensemble de traces temporisées, noté par la suite TEG. TEG est différent selon que le graphe d'évolution termine par deadlock ou par bouclage (Figure 3.3).

Dans un premier temps, nous définissons formellement ces traces. On rappelle que nIB correspond au numéro du dernier IB-state.

Définition 3.3 (Sch_d)**.** *Chaque trace temporisée discrète d'un graphe d'évolution se terminant par deadlock peut s'écrire comme suit :*

$$\left(\prod_{i=0}^{nIB-1} (\phi_2(T_i^{EG}), t_{T_i^{EG}}) \Big[\ (\phi_1(IB_i), t'_i) | t'_i \in]t_{T_i^{EG}}, t_{T_{i+1}^{EG}}[\Big]^+ \right) \times$$
$$(\phi_2(T_{nIB}^{EG}), t_{T_{nIB}^{EG}}) \Big[(\phi_1(IB_{nIB}), t'_{nIB}) | t'_{nIB} > t_{T_{nIB}^{EG}} \ \Big]^\omega$$

avec $\forall i \in [0, nIB-1], t_{T_{i+1}^{EG}} - t_{T_i^{EG}} = \Delta t_{T_{i+1}^{EG}}$ *et avec* $t_{T_{nIB}^{EG}} = \sum_{i=0}^{i=nIB} \Delta t_{T_i^{EG}}$.

L'ensemble de toutes les traces temporisées possibles suivant ce schéma est noté Sch_d.

La Définition 3.3 peut sembler compliquée, mais l'expression donnée se décompose facilement. En effet, la première partie de la définition (celle entre parenthèse) concerne tous les couples transition/IB-state excepté le dernier indicé par nIB, qui est traité dans la deuxième partie. Par exemple, une trace de Sch_d du graphe d'évolution de la Figure 3.4 peut être :

$$\Big(m(T3) = 1 \wedge m(T4) = 5..., \ 0\Big)\Big(m(T3) \in]0.5, 1[\wedge ..., \ 0.25\Big)\Big(m(T3) \in]0.5, 1[\wedge ..., \ 0.45\Big)...$$

$$...(m(T3) = 0 \wedge m(T4) = 0..., \ 5.25\Big)\Big(m(T3) = 0 \wedge m(T4) = 0..., \ 5.30\Big)...$$

Définition 3.4 (Sch_b)**.** *Tout trace d'un graphe d'évolution qui boucle peut s'écrire comme suit :*

$$\left(\prod_{i=0}^{nloop-1} (\phi_2(T_i^{EG}), t_{T_i^{EG}})\Big[(\phi_1(IB_i), t_i')|t_i' \in]t_{T_i^{EG}}, t_{T_{i+1}^{EG}}[\Big]^+\right) \times (\phi_2(T_{nloop}^{EG}), t_{nloop})$$
$$\times \left[\prod_{i=nloop}^{nIB} \Big[(\phi_1(IB_i), t_i')|t_i' \in]t_{T_i^{EG}}, t_{T_{i+1}^{EG}}[\Big]^+ (\phi_2(T_{i+1}^{EG}), t_{T_{i+1}^{EG}})\right]^{\omega}$$

avec $\forall i \in [0, nloop], t_{T_{i+1}^{EG}} - t_{T_i^{EG}} = \Delta t_{T_{i+1}^{EG}}$.

L'ensemble de toutes les traces temporisées possibles suivant ce schéma est noté Sch_b.

L'expression de la trace temporisée donnée dans la Définition 3.4 se subdivise en trois parties :

- La première concerne les couples transition/IB-state avant bouclage. Cette première partie est traitée comme dans la Définition 3.3,
- La deuxième partie ne concerne que la transition indicée *nloop*, cette transition correspond à la dernière transition avant d'entrer dans la boucle du graphe d'évolution,
- La troisième partie concerne la boucle du graphe d'évolution.

Remarque 3.3. $\tau \in TEG \Leftrightarrow \tau \in (Sch_d \cup Sch_b)$ *selon que le graphe d'évolution termine par deadlock ou bouclage.*

3.2.4.2 Préservation des traces discrètes temporisées

Nous prouvons maintenant l'égalité entre les deux ensembles de traces déduites respectivement du graphe d'évolution et de l'automate Event-Clock A_M. Cette égalité (Lemme 3.3) est déduite des deux prochains lemmes.

Lemme 3.1. $\tau \in \mathcal{L}(A_M) \Rightarrow \tau \in (Sch_d \cup Sch_b)$

Preuve. Remarquons tout d'abord que les formules dans les traces sont les mêmes que l'on prenne l'automate ou le graphe d'évolution. Soit $\tau \in \mathcal{L}(A_M)$ et soient l_i, l_{i+1} l_{i+2} trois localisations successives de l'automate A_M telles que l_i provient de la transition T_j^{EG}. De l_i sort l'arc étiqueté par l'observation $\phi_2(T_j^{EG})$. Les arcs sortant de la localisation l_{i+1} entrent dans l_{i+1} et dans l_{i+2} et sont étiquetés par l'observation $\phi_1(IB_j)$. Finalement, de la localisation l_{i+2} sort un arc étiqueté par $\phi_2(T_{j+1}^{EG})$.

L'automate est temporisé grâce aux horloges historiques, notées dans l'algorithme x_{de}. Ces horloges imposent aux deux localisations l_i et l_{i+2} d'être exactement séparées par le délai suivant : $t_{T_{j+1}^{EG}} - t_{T_j^{EG}} = x_{de} = \Delta t_{T_{j+1}^{EG}}$. Par conséquent, la localisation l_{i+1} peut être visitée durant un intervalle strictement plus petit que ce délai.

- En cas de deadlock, les traces bouclent indéfiniment sur la dernière localisation associée au dernier IB-state (IB_{nIB}) à partir du temps $t_{T_{nIB}^{EG}} = \sum x_{de} = \sum_{i=0}^{i=nIB} \Delta t_{T_i^{EG}}$. *L'ensemble des traces temporisées discrètes τ de l'automate A_M finissant par deadlock est donc inclus dans Sch_d.*
- En cas de bouclage, les traces bouclent au niveau de la localisation associée à l'IB-state numéro *nbloop*. Elles bouclent donc indéfiniment sur les localisations associées aux IB-states et transitions numérotés de *nloop* à $nIB + 1$ dans le graphe d'évolution (Figure 3.3). *L'ensemble des traces discrètes temporisées τ de l'automate A_M finissant par bouclage est donc inclus dans Sch_b.*

□

Lemme 3.2. $\tau \in (Sch_d \cup Sch_b) \Rightarrow \tau \in \mathcal{L}(A_M)$

Preuve. Soit $\tau \in (Sch_d \cup Sch_b)$ et plus précisément considérons la sous-trace $\{(\phi_2(T_i^{EG}), t_{T_i^{EG}})\}$ contenant seulement des observations de la forme ϕ_2. A_M étiquette une localisation sur deux (notée ici l_i) avec une observation $\phi_2(T_j^{EG})$, $j \in [0; nIB]$. De plus, par construction de A_M, nous avons $t_{l_{i+2}} - t_{l_i} = x_{de} = \Delta t_{T_{i+1}^{EG}}$ et par définition du graphe d'évolution, nous avons $t_{T_{i+1}^{EG}} - t_{T_i^{EG}} = \Delta t_{T_{i+1}^{EG}}$. On en déduit que toute sous-séquence $\{(\phi_2(T_i^{EG}), t_i)\}$ du TEG est nécessairement reconnue par une trace de l'automate A_M.

Considérons à présent les sous-traces de TEG $\{(\phi_1(IB_i), t'_i)\}$ contenant seulement des observations de la forme ϕ_1. D'après l'algorithme précédent, un arc étiqueté par une observation $\phi_1(IB_j)$, $j \in [0; nIB]$ boucle sur ses localisations (notées ici l_{i+1}). Comme décrit précédemment, on a $t_{T_{i+1}^{EG}} - t_{T_i^{EG}} = \Delta t_{T_{i+1}^{EG}}$, donc pour tout temps $t' \in]t_{T_i^{EG}}, t_{T_{i+1}^{EG}}[$, l'exécution peut boucler sur la localisation l_{i+1}. Par conséquent, toute sous-séquence de TEG composée des couples $\{(\phi_1(IB_i), t'_i) | t'_i \in]t_{T_i^{EG}}, t_{T_{i+1}^{EG}}[^+\}$ est aussi nécessairement reconnue par une trace discrète temporisée de l'automate A_M.

- En cas de deadlock, les traces incluses dans Sch_d assurent que $\phi_1(IB_{nIB})$ restent vraies pour tout temps $t > t_{T_{nIB}^{EG}}$ avec $t_{T_{nIB}^{EG}} = \sum_{i=0}^{i=nIB} \Delta t_{T_i^{EG}}$. Les traces bouclent sur la localisation finale de l'automate A_M après le temps $t = \sum x_{le} = \sum_{i=0}^{i=nIB} \Delta t_{T_i^{EG}}$.
- En cas de bouclage, les traces incluses dans Sch_b bouclent indéfiniment de la localisation associée à l'IB-state IB_{nloop} à celle associée à T_{nIB+1}^{EG}. De plus, dans cette boucle, l'intervalle de temps séparant deux observations $\phi_2(T_i^{EG})$ restent le même et correspond exactement aux x_{de} de l'automate A_M.

□

Des deux lemmes précédents, on déduit directement le lemme 3.3.

Lemme 3.3. *Les langages de l'automate Event-Clock A_M et du graphe d'évolution sont exactement les mêmes :* $\{\tau|\tau \in \mathcal{L}(A_M)\} = \{\tau|\tau \in TEG\}$.

De la définition du graphe d'évolution et du lemme 6.2, nous déduisons le lemme suivant.

Lemme 3.4. *Soient M un modèle en temps continu et τ une trace temporisée reconnue par le graphe d'évolution alors M satisfait τ. Autrement dit,* $\tau \in TEG \Rightarrow M \models \tau$.

Lemme 3.5. $\tau \in \mathcal{L}(A_M) \Rightarrow M \models \tau$

Preuve. Conséquence directe des lemmes 3.3 et 3.4. □

3.3 De la propriété aux automates «Event-Clock»

Rappelons qu'une propriété temporelle traduit un ensemble de comportements admissibles du système. Par conséquent, une propriété exprimée en CTEL traduit un ensemble de traces temporisées discrètes qui peuvent également être converties en un automate Event-Clock. Si nous étudions la propriété f, l'automate Event-Clock $A_{\neg f}$ de la négation de f est construit. $A_{\neg f}$ représente l'ensemble de *toutes* les traces temporisées discrètes de $\neg f$. Nous cherchons alors s'il existe une trace commune entre A_M et $A_{\neg f}$. Si une telle trace existe, nous concluons que notre modèle THPN ne satisfait pas la propriété étant donné qu'il existe au moins une trace de A_M satisfaisant $\neg f$.

Raskin et Schobbens ont proposé un algorithme pour construire un automate Event-Clock à partir d'une propriété [36]. Cet algorithme est *complet*, une trace temporisée discrète satisfaisant $\neg f$ est donc nécessairement reconnue par l'automate $A_{\neg f}$.

Rappelons dans un premier temps la procédure de construction d'un automate Event-Clock A_f à partir d'une formule CTEL f ainsi que les propriétés principales associées à A_f. Cette procédure correspond à une extension de la procédure de décision pour la logique temporelle linéaire (LTL) [67].

3.3.1 Algorithme de conversion

La définition d'un langage temporisé d'une formule CTEL (Définition 3.5) est la base de la procédure de construction de l'automate A_f. Cette procédure utilise l'ensemble des sous-formules de f (Définition 3.6).

Définition 3.5 (Langage temporisé d'une formule CTEL)**.** *Une formule CTEL f définit un langage temporisé $L(f)$: l'ensemble des traces temporisées τ est tel que* $(\tau, 0) \backsim f$.

Définition 3.6 (Fermeture)**.** *L'ensemble de fermeture d'une formule CTEL f, noté $Cl(f)$ (pour closure en anglais), est le plus petit ensemble de formules défini inductivement sur la forme de f, comme suit :*

- $a \in Cl(a)$
- $\{f_1 \vee f_2\} \cup Cl(f_1) \cup Cl(f_2) \subseteq Cl(f_1 \vee f_2)$
- $\{f_1 \wedge f_2\} \cup Cl(f_1) \cup Cl(f_2) \subseteq Cl(f_1 \wedge f_2)$
- $Cl(f) \subseteq Cl(\neg f)$

- $\{\bigcirc f\} \cup Cl(f) \subseteq Cl(\bigcirc f)$
- $\{\ominus f\} \cup Cl(f) \subseteq Cl(\ominus f)$
- $\{f_1 U f_2\} \cup \{\bigcirc(f_1 U f_2)\} \cup Cl(f_1) \cup Cl(f_2) \subseteq Cl(f_1 U f_2)$
- $\{f_1 S f_2\} \cup \{\ominus(f_1 S f_2)\} \cup Cl(f_1) \cup Cl(f_2) \subseteq Cl(f_1 S f_2)$
- $\{\lhd_{\sim c} f\} \cup Cl(f) \subseteq Cl(\lhd_{\sim c} f)$
- $\{\rhd_{\sim c} f\} \cup Cl(f) \subseteq Cl(\rhd_{\sim c} f)$
- $f' \in Cl(f) \Rightarrow \neg f' \in Cl(f)$ *où* $\neg\neg f''$ *est identifié à* f''

où a *est un atome,* f_1, f_2 *sont des formules CTEL.*

Définition 3.7 (Etat). *Un état d'une formule CTEL* f *est un sous-ensemble* $\Theta \subseteq Cl(f)$ *satisfaisant les points suivants :*

- Θ *est propositionnellement cohérent et complet. Plus formellement :*
  - *Pour tout* $\rho_1 \in Cl(f)$, $\rho_1 \in \Theta$ *ssi* $\neg\rho_1 \notin \Theta$,
  - *Pour tout* $\rho_1 \vee \rho_2 \in Cl(f)$, $\rho_1 \vee \rho_2 \in \Theta$ *ssi* $\rho_1 \in \Theta$ *ou* $\rho_2 \in \Theta$
  - *Pour tout* $\rho_1 \wedge \rho_2 \in Cl(f)$, $\rho_1 \wedge \rho_2 \in \Theta$ *ssi* $\rho_1 \in \Theta$ *et* $\rho_2 \in \Theta$
- Θ *respecte les contraintes locales des opérateurs Until (U) et Since (S) :*
  - *Pour tout* $\rho_1 U \rho_2 \in Cl(f)$, $\rho_1 U \rho_2 \in \Theta$ *ssi :*
    - *soit* $\rho_2 \in \Theta$,
    - *soit* $\rho_1 \in \Theta$ *et* $\bigcirc(\rho_1 U \rho_2) \in \Theta$
  - *Pour tout* $\rho_1 S \rho_2 \in Cl(f)$, $\rho_1 S \rho_2 \in \Theta$ *ssi :*
    - *soit* $\rho_2 \in \Theta$,
    - *soit* $\rho_1 \in \Theta$ *et* $\ominus(\rho_1 S \rho_2) \in \Theta$

Intuitivement, si $\rho_1 U \rho_2$ est vrai, soit ρ_2 vient d'être atteint, soit ρ_2 n'est pas encore atteint, ρ_1 est donc vrai et $\rho_1 U \rho_2$ reste vrai jusqu'à la prochaine étape. Un raisonnement miroir est effectué pour $\rho_1 S \rho_2$.

Définition 3.8 (A_f). *Etant donnée une formule CTEL* f, *l'automate Event-Clock associé* $A_f = (L, L_0, \Sigma, \mathcal{C}, E, \mathcal{F})$ *est défini comme suit :*

- *Les localisations* L *de* A_f *sont les états de* $Cl(f)$,
- *Les localisations initiales* L_0 *sont les états* Θ *tels que :* $f \in \Theta$ *et pour toute formule* $\ominus\rho \in Cl(f)$, $\neg \ominus \rho \in \Theta$,
- $\mathcal{C} = \{x_\alpha | \lhd_{\sim c} \alpha \in Cl(f)\} \cup \{y_\alpha | \rhd_{\sim c} \alpha \in Cl(f)\}$
- *Il y a un arc* $(l_1, \psi, l_2) \in E$ *(où* $\psi \equiv \psi_1 \wedge \psi_2 \wedge \psi_3$ *selon la notation 2 page 47) ssi :*
  - *Pour tout* $\bigcirc\rho \in Cl(f)$ *:* $\bigcirc\rho \in l_1$ *ssi* $\rho \in l_2$
  - *Pour tout* $\ominus\rho \in Cl(f)$ *:* $\rho \in l_1$ *ssi* $\ominus\rho \in l_2$
  - $\psi_1 \equiv \bigwedge\{\alpha_p | \alpha_p \in l_1\}$
  - $\psi_2 \equiv \bigwedge\{\alpha_v | \alpha_v \in l_1\}$
  - $\psi_3 \equiv \bigwedge\{x_\alpha \sim c | \lhd_{\sim c} \alpha \in l_1\} \bigwedge\{y_\alpha \sim c | \rhd_{\sim c} \alpha \in l_1\} \bigwedge\{\neg(x_\alpha \sim c) | \neg(\lhd_{\sim c}\alpha) \in l_1\} \bigwedge\{\neg(y_\alpha \sim c) | \neg(\rhd_{\sim c}\alpha) \in l_1\}$
- $\mathcal{F} = \{\{l | \neg(\rho_1 U \rho_2) \in l \text{ ou } \rho_2 \in l\} | \rho_1 U \rho_2 \in Cl(f)\}$

De façon à illustrer la procédure de construction d'un automate Event-Clock à partir d'une formule CTEL, nous détaillons un exemple simple de formule $f \equiv aUb$, dont la fermeture est $Cl(aUb) = \{aUb, \bigcirc(aUb), a, b, \neg(aUb), \neg\bigcirc(aUb), \neg a, \neg b\}$. $2^4 = 16$ états sont obtenus de la fermeture.

Etats.

$aUb, \bigcirc(aUb), a, b$	$aUb, \bigcirc(aUb), a, \neg b$	$aUb, \bigcirc(aUb), \neg a, b$
$aUb, \neg\bigcirc(aUb), a, b$	$\neg(aUb), \bigcirc(aUb), a, b$	$(aUb), \bigcirc(aUb), \neg a, \neg b$
$(aUb), \neg\bigcirc(aUb), a, \neg b$	$\neg(aUb), \bigcirc(aUb), a, \neg b$	$(aUb), \neg\bigcirc(aUb), \neg a, b$
$\neg(aUb), \bigcirc(aUb), \neg a, b$	$\neg(aUb), \neg\bigcirc(aUb), a, b$	$(aUb), \neg\bigcirc(aUb), \neg a, \neg b$
$\neg(aUb), \bigcirc(aUb), \neg a, \neg b$	$\neg(aUb), \neg\bigcirc(aUb), a, \neg b$	$\neg(aUb), \neg\bigcirc(aUb), \neg a, b$
$\neg(aUb), \neg\bigcirc(aUb), \neg a, \neg b$		

Incohérences. D'après la Définition 3.7, $(aUb) \Leftrightarrow b \vee (a \wedge \bigcirc(aUb))$. Par conséquent, 4 familles d'états sont incohérents :

$aUb, \_, \neg a, \neg b$	$aUb, \neg\bigcirc(aUb), \_, \neg b$
$\neg(aUb), \bigcirc(aUb), a, \_$	$\neg(aUb), \_, \_, b$

Il reste donc 8 états cohérents :

(1) $aUb, \bigcirc(aUb), a, b$	(2) $aUb, \bigcirc(aUb), a, \neg b$
(3) $aUb, \bigcirc(aUb), \neg a, b$	(4) $aUb, \neg\bigcirc(aUb), a, b$
(5) $(aUb), \neg\bigcirc(aUb), \neg a, b$	(6) $\neg(aUb), \bigcirc(aUb), \neg a, \neg b$
(7) $\neg(aUb), \neg\bigcirc(aUb), a, \neg b$	(8) $\neg(aUb), \neg\bigcirc(aUb), \neg a, \neg b$

Arcs. D'après la Définition 3.8, il existe un arc entre les localisations l_1 et l_2 dans les cas suivants : $\bigcirc\rho \in l_1 \Leftrightarrow \rho \in l_2$ donc $\neg\bigcirc\rho \in l_1 \Leftrightarrow \neg\rho \in l_2$. Nous en déduisons la liste d'adjacence suivante :

- $1 \rightarrow 1, 2, 3, 4, 5$
- $2 \rightarrow 1, 2, 3, 4, 5$
- $3 \rightarrow 1, 2, 3, 4, 5$
- $4 \rightarrow 6, 7, 8$
- $5 \rightarrow 6, 7, 8$
- $6 \rightarrow 1, 2, 3, 4, 9$
- $7 \rightarrow 6, 7, 8$
- $8 \rightarrow 6, 7, 8$

Localisations initiales. Les localisations initiales sont celles possédant $f \equiv (aUb)$ comme étiquette. Il s'agit donc des localisations $1, 2, 3, 4, 5$.

Localisations finales. L'ensemble d'ensembles de localisations finales est seulement composé d'un ensemble de localisations étant donné qu'il n'y a qu'un seul opérateur *Until*. Cet ensemble est composé des localisations étiquetées soit par $\neg(aUb)$ soit par b. Il s'agit donc des localisations $1, 3, 4, 5, 6, 7, 8$.

3.3.2 Définition et propriétés

Dans cette section, nous rappelons les propriétés importantes associées à l'automate Event-Clock A_f construit à partir de la formule f.

Théorème 3.1. *[36] L'ensemble des traces temporisées discrètes acceptées par l'automate Event-Clock A_f est exactement l'ensemble des traces temporisées discrètes de la formule f,* i.e. $\tau' \in \mathcal{L}(A_f) \Leftrightarrow (\tau', 0) \not\sim f$.

Nous déduisons du théorème précédent le corollaire suivant :

Corollaire 3.1. $\tau = \{(\varphi_i, t_i)\}_{i\in\mathbb{N}} \in \mathcal{L}(A_f) \Rightarrow \forall i \in \mathbb{N}, (\tau, i) \not\sim f \wedge \neg f$

Preuve. La preuve est faite sur la structure de la formule f.

- Atomes.
  - $f \equiv p$
    Supposons que $(\tau, i) \not\sim p \wedge \neg p$. Par la définition 2.15, $(\tau, i) \not\sim p$, on en déduit que $UpL(p) \subseteq UpL(\varphi_i)$. $(\tau, i) \not\sim \neg p$, par conséquent, $UpL(p) \nsubseteq UpL(\varphi_i)$, où φ_i est la i^{ieme} observation de la trace τ. Contradiction.
  - $f \equiv r \geq r'$
    Supposons que $\tau \not\sim (r \geq r') \wedge \neg(r \geq r')$.
    Tout d'abord, rappelons que seuls les littéraux étiquettent les arcs de A_f et notons que f est ici un littéral. De plus, étant donné que $\tau \in \mathcal{L}(A_f)$, il existe un chemin acceptant (Définition 3.2) tel que :

    $$\gamma = l_0^p \overset{\psi_0}{\rightarrow} l_1 \overset{\psi_1}{\rightarrow} ... l_n \overset{\psi_n^p}{\rightarrow} ..., \text{ où } \forall i, (\tau, i) \not\sim \psi_i \text{ et } UpL(\psi_i) \subseteq UpL(\varphi_i).$$

    D'après la définition 3.8, ψ_0 contient le littéral f. Nous en déduisons que $UpL(f) \subseteq UpL(\psi_0)$. La Définition 3.2 impose que $UpL(\psi_0) \subseteq UpL(\varphi_0)$. Par conséquent, nous concluons que $UpL(f) \subseteq UpL(\varphi_0)$. On en déduit que si $M \models \tau$ alors $M \models_0^{h_\tau} \varphi_0$ et $M \models_0^{h_\tau} r \geq r'$. On conclut alors que $\nexists M | M \models \tau$ et $M \models_i^{h_\tau} \neg(r \geq r')$, ainsi τ ne peut satisfaire $\neg(r \geq r')$. Contradiction.

- Supposons que la propriété soit vraie pour les formules f', f_1, f_2

- Cas général
  - $f \equiv f_1 \wedge f_2$
    Si $(\tau, i) \not\sim (f_1 \wedge f_2) \wedge \neg(f_1 \wedge f_2)$, alors $(\tau, i) \not\sim (f_1 \wedge f_2 \wedge \neg f_1) \vee (f_1 \wedge f_2 \wedge \neg f_2)$ ce qui n'est pas satisfiable par hypothèse d'induction.
  - $f \equiv \bigcirc f'$
    Supposons que $(\tau, i) \not\sim \bigcirc f' \wedge \neg\bigcirc f'$. Ceci peut s'écrire $(\tau, i) \not\sim \bigcirc(f' \wedge \neg f')$, *i.e.* $(\tau, i+1) \not\sim f \wedge \neg f$. Contradiction.
    (similaire pour $f \equiv \ominus f'$)
  - $f \equiv f_1 U f_2$
    Supposons que $(\tau, i) \not\sim f_1 U f_2 \wedge \neg(f_1 U f_2)$. Par la définition 2.15, $(\tau, i) \not\sim f_1 U f_2$ ssi $\exists j \geq i \mid (\tau, j) \not\sim f_2$ et $\forall k \in [i, j[, (\tau, k) \not\sim f_1$, ce qui est en contradiction avec $\tau \not\sim \neg(f_1 U f_2)$. En effet, si $(\tau, i) \not\sim f_1 U f_2$ l'exécution lit un chemin de la forme $f_1, f_1, ..., f_1, f_2$ qui est en contradiction avec la définition de $\neg(f_1 U f_2)$.
    (similaire pour $f \equiv f_1 S f_2$)

- $f \equiv \rhd_{\sim n} f$
  Supposons que $(\tau, i) \not\sim \rhd_{\sim n} f \wedge \neg(\rhd_{\sim n} f)$. Par la définition 2.15, $(\tau, i) \not\sim \rhd_{\sim n} f$ ssi $\exists j > i \mid (\tau, j) \not\sim f$ et $\forall k \in]i, j[, (\tau, k) \not\sim \neg f$ et $t_j - t_i \sim n$, ce qui est en contradiction avec $(\tau, i) \not\sim \neg(\rhd_{\sim n} f)$ (explication similaire à celle présentée ci-dessus).
  (similaire pour $f \equiv \lhd_{\sim n} f'$)

□

3.3.3 Illustration biologique

Notre modèle THPN simule la compétition existant entre deux activités enzymatiques opposées : l'enzyme D2 est un activateur des hormones thyroïdiennes, elle synthétise la forme active de l'hormone (T3) à partir de sa forme inactive (T4), à l'inverse, l'enzyme D3 inhibe les deux types d'hormones (T3 et T4). Nous nous demandons si dans tous les cas de résolution de conflits, l'enzyme inhibitrice prédomine sur l'enzyme activatrice. Cette propriété biologique est traduite en formule CTEL f :

$$f \equiv \Box \diamond (m(T3) = 0)$$

où $\diamond$ signifie "à un moment donné dans l'avenir" et $\Box$ signifie "toujours dans l'avenir". Les deux opérateurs $\Box$ et $\diamond$ correspondent aux abréviations suivantes : soit ϕ une formule CTEL, $\diamond\phi \equiv \top\ U\ \phi$ et $\Box\phi \equiv \neg \diamond \neg\phi$.

La formule f signifie que la concentration de T3 atteindra toujours à un moment donné la concentration 0. Ceci traduit la prédominance de D3 sur D2. La Figure 3.5 montre l'automate Event-Clock de la négation de f : $\neg f \equiv \neg\Box \diamond (m(T3) = 0)$.

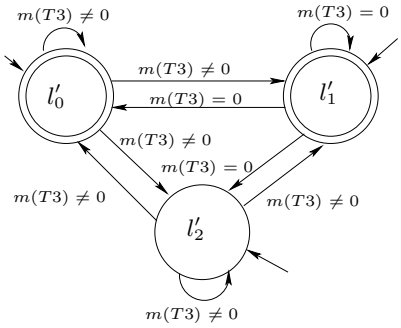

FIG. 3.5 – Automate Event-Clock de la négation de $f \equiv \Box \diamond (m(T3) = 0)$.

3.4 Un produit avec transformations des étiquettes

Usuellement, l'extraction de traces communes à deux automates A_1 et A_2 s'obtient en faisant l'intersection des langages $\mathcal{L}(A_1) \cap \mathcal{L}(A_2)$. Lorsque la classe des automates étudiés possède la propriété de clôture par intersection, cela signifie que l'intersection des langages de deux automates $\mathcal{L}(A_1) \cap \mathcal{L}(A_2)$ correspond au langage résultant du produit des deux automates étudiés $\mathcal{L}(A_1 \times A_2) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$. Alur *et al.* [37] ont montré la

propriété de fermeture par intersection de la classe des automates Event-Clock étiquetés par un ensemble de propositions. Dans notre, cas, l'ensemble des atomes a été étendu avec la notion d'atomes instantanés (Définition 2.3 page 44). Nous devons donc définir un produit spécifique qui intègre des notions quantitatives. Nous définissons le produit avec Transformation de Labels, noté produit-LT. Nous montrons alors que le langage de l'automate produit $A_p = A_1 \times A_2$ correspond bien à l'intersection des langages des deux automates initiaux A_1 et A_2 (Théorème 3.2 et 3.3).

Le produit-LT est appliqué aux automates A_M et $A_{\neg f}$ associés respectivement au modèle THPN et à la négation de la propriété que l'on cherche à tester. La dernière étape consiste alors à vérifier si le langage temporisé de l'automate produit $A_M \times A_{\neg f}$ est vide ou non. Dans le cas d'automates Event-Clock étiquetés par un ensemble de propositions [36, 37], la décision du vide se fait par construction de l'automate de région [38]. L'idée est de construire une machine d'états finis qui accepte le langage atemporisé (*Untimed* en anglais) suivant :

$$Untimed(A_M \times A_{\neg f}) = \{\varphi_0\varphi_1...\varphi_n...|\tau = (\varphi_0, t_0)(\varphi_1, t_1)...(\varphi_n, t_n)... \in \mathcal{L}(A_M \times A_{\neg f})\}$$

3.4.1 Produit-LT

Le produit-LT consiste à faire la conjonction des observations présentes sur les arcs. Ainsi, s'il existe un arc dans l'automate Event-Clock A_M tel que $\Big(l_1, m(T4) = 0,\ l_2\Big)$ et qu'il existe un arc dans l'automate Event-Clock $A_{\neg f}$ tel que $\Big(l'_1, (m(T4) \geq 4) \wedge (x_{NoEvent} = 3), l'_2\Big)$ alors l'arc $\Big((l_1, l'_1), (m(TH4) = 0) \wedge (m(TH4) \geq 4) \wedge (x_{NoEvent} = 3),\ (l_2, l'_2)\Big)$ est un arc de l'automate produit $A_M \times A_{\neg f}$. Formellement, le produit-LT se définit comme suit :

Définition 3.9 (Produit-LT). *Soient* $A_1 = (L^1, L^1_0, \Sigma^1, \mathcal{C}^1, E^1, \mathcal{F}^1)$ *et* $A_2 = (L^2, L^2_0, \Sigma^2, \mathcal{C}^2, E^2, \mathcal{F}^2)$ *deux automates Event-Clock. Le produit-LT de* A_1 *et* A_2 *est l'automate Event-Clock* $A_p = (L^p, L^p_0, \Sigma^p, \mathcal{C}^p, E^p, \mathcal{F}^p)$ *défini comme suit :*

- $L^p = L^1 \times L^2$
- $L^p_0 = L^1_0 \times L^2_0$
- $\Sigma^p = \Sigma^1 \cup \Sigma^2$.
- $\mathcal{C}^p = \mathcal{C}^1 \cup \mathcal{C}^2$
- $\mathcal{F}^p = \{(F^1_i \times L^2)|F^1_i \in \mathcal{F}^1\} \cup \{(L^1 \times F^2_j)|F^2_j \in \mathcal{F}^2\}$
- $E^p = \{((l_{i_1}, l_{j_1}), \psi_1 \wedge \psi_2, (l_{i_2}, l_{j_2}))$ *tel que* $(l_{i_1}, \psi_1, l_{i_2}) \in E^1$ *et* $(l_{j_1}, \psi_2, l_{j_2}) \in E^2\}$.

Il est important de vérifier que toutes les traces de l'automate produit $A_1 \times A_2$ sont initialement reconnues par les automates A_1 et A_2, il s'agit de la preuve de la correction de notre produit (Théorème 3.2). De même, il est important de prouver que toute trace commune aux deux automates A_1 et A_2 est reconnue par l'automate produit $A_1 \times A_2$, il s'agit de la complétude (Théorème 3.3).

Théorème 3.2 (Correction du produit-LT). *Soit* A_p *l'automate Event-Clock résultant du produit-LT des automates Event-Clock* A_1 *et* A_2. *Soit* τ *une trace temporisée discrète, si* $\tau \in \mathcal{L}(A_p)$ *alors* $\tau \in \mathcal{L}(A_1)$ *et* $\tau \in \mathcal{L}(A_2)$.

Preuve. Soit $\tau = \{(\varphi_i, t_i)\}_{i\in\mathbb{N}}$ une trace temporisée telle que $\tau \in \mathcal{L}(A_p)$. Il existe donc nécessairement un chemin acceptant sur A_p, notée γ_p tel que

$$\gamma_p = l_0^p \overset{\psi_0^p}{\rightarrow} l_1^p \overset{\psi_1^p}{\rightarrow} ...l_n^p \overset{\psi_n^p}{\rightarrow} ..., \text{ où } \forall i, (\tau, i) \leftsquigarrow \psi_i^p \text{ et } UpL(\psi_i^p) \subseteq UpL(\varphi_i).$$

Par définition du produit-LT, chaque localisation $l_i^p \in L^p$ est un couple (l_i, l'_i) où $l_i \in L^1$ et $l'_i \in L^2$. l_0^p est aussi un couple (l_0, l'_0) où $l_0 \in L_0^1$ et $l'_0 \in L_0^2$. γ_p peut donc s'écrire :

$$\gamma_p = (l_0, l'_0) \overset{\psi_0^p}{\rightarrow} (l_1, l'_1) \overset{\psi_1^p}{\rightarrow} ...(l_n, l'_n) \overset{\psi_n^p}{\rightarrow} ..., \text{ où } \forall i, (\tau, i) \leftsquigarrow \psi_i^p \text{ et } UpL(\psi_i^p) \subseteq UpL(\varphi_i)$$

Chaque ψ_i^p est la conjonction de deux observations ψ_i et ψ'_i où $(l_i, \psi_i, l_{i+1}) \in E^1$ et $(l'_i, \psi'_i, l'_{i+1}) \in E^2$. Il est alors possible de déduire de γ_p les deux chemins acceptants γ_1 et γ_2 des automates A_1 et A_2.

$$\gamma_1 = l_0 \overset{\psi_0}{\rightarrow} l_1 \overset{\psi_1}{\rightarrow} ...l_n \overset{\psi_n}{\rightarrow} ... \quad \text{et} \quad \gamma_2 = l'_0 \overset{\psi'_0}{\rightarrow} l'_1 \overset{\psi'_1}{\rightarrow} ...l'_n \overset{\psi'_n}{\rightarrow} ...$$

Montrons que γ_1 est un chemin acceptant de la trace temporisée τ :

- $l_i \in L^1$ pour tout i et $l_0 \in L_0^1$
- chaque $F_i \in \mathcal{F}^p$ est soit un couple (F, l) où $F \in \mathcal{F}^1$ et $l \in L^2$ soit un couple (l, F) où $l \in L^1$ et $F \in \mathcal{F}^2$. Etant donné que γ_p respecte la condition généralisée de Büchi pour l'automate A_p, γ_p passe infiniment souvent par $F_i^p = (F_i^1 \times L^2)$ de $\mathcal{F}^p$ pour chaque $F_i^1 \in \mathcal{F}^1$. On en déduit que pour tout $F_i^1 \in \mathcal{F}^1$, il existe une infinité de positions j telles que $l_j \in F_i^1$.
- $(l_i, \psi_i, l_{i+1}) \in E^1$. Etant donné que $(\tau, i) \leftsquigarrow \psi_i \wedge \psi'_i$, nous avons $(\tau, i) \leftsquigarrow \psi_i$. De plus, étant donné que $UpL(\psi_i^p) \subseteq UpL(\varphi_i)$ et $\psi_i^p \equiv \psi_i \wedge \psi'_i$, nous déduisons que $UpL(\psi_i) \subseteq UpL(\psi_i^p) \subseteq UpL(\varphi_i)$.

γ_1 est donc un chemin acceptant de la trace temporisée τ. Une démonstration similaire est effectuée pour le chemin γ_2. □

Théorème 3.3 (Complétude du produit-LT)**.** *Soit A_p l'automate Event-Clock résultant du produit-LT des automates Event-Clock A_1 et A_2. Soit τ une trace temporisée discrète, si $\tau \in \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$ alors $\tau \in \mathcal{L}(A_p)$.*

Preuve. Si $\tau \in \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$ alors il existe deux chemins acceptants γ_1 et γ_2 sur les automates A_1 et A_2 tels que :

$$\gamma_1 = l_0 \overset{\psi_0}{\rightarrow} l_1 \overset{\psi_1}{\rightarrow} ...l_n \overset{\psi_n}{\rightarrow} ... \quad \text{où } \forall i, (\tau, i) \leftsquigarrow \psi_i \text{ et } UpL(\psi_i) \subseteq UpL(\varphi_i) \tag{3.1}$$

$$\gamma_2 = l'_0 \overset{\psi'_0}{\rightarrow} l'_1 \overset{\psi'_1}{\rightarrow} ...l'_n \overset{\psi'_n}{\rightarrow} ... \quad \text{où } \forall i, (\tau, i) \leftsquigarrow \psi'_i \text{ et } UpL(\psi'_i) \subseteq UpL(\varphi_i) \tag{3.2}$$

Soit γ_p le chemin résultant de la composition des deux chemins γ_1 et γ_2 :

$$\gamma_p = (l_0, l'_0) \overset{\psi_0 \wedge \psi'_0}{\rightarrow} (l_1, l'_1) \overset{\psi_1 \wedge \psi'_1}{\rightarrow} ... \ (l_n, l'_n) \overset{\psi_n \wedge \psi'_n}{\rightarrow} ....$$

Montrons que γ_p est un chemin de l'automate A_p :

- $(l_i, l'_i) \in L^1 \times L^2$ pour tout i et $(l_0, l'_0) \in L_0^1 \times L_0^2$,

- Par définition du produit-LT, pour chaque $F_i^1 \in \mathcal{F}^1$ (resp. pour chaque $F_i^2 \in \mathcal{F}^2$) il existe une infinité de positions j (resp. k) telles que $l_j \in F_i^1$ (resp. $l_k \in F_i^2$). Ceci est requis étant donné que γ_1 et γ_2 sont des chemins acceptants des automates A_1 et A_2.
- Enfin, les arcs $((l_i, l'_i), \psi_i \wedge \psi'_i, (l_{i+1}, l'_{i+1}))$ sont tels que $(l_i, \psi_i, l_{i+1}) \in E^1$ et $(l'_i, \psi'_i, l'_{i+1}) \in E^2$.

Par définition du produit-LT, γ_p est alors un chemin de l'automate A_p.

La dernière étape consiste à montrer que γ_p est un chemin acceptant :

- $(l_i, l'_i) \in L^p$ et $(l_0, l'_0) \in L_0^p$,
- Soit $F^p \in \mathcal{F}^p$ de la forme $(F_i^1 \times L^2)$. Montrons que γ_p passe infiniment souvent par les localisations de $F_i^1 \times L^2$. γ_1 passe infiniment souvent par F_i^1 pour tout $F_i^1 \in \mathcal{F}^1$, ainsi γ_p passe infiniment souvent par l'ensemble $(F_i^1 \times L^2)$ pour tout $F_i^1 \in \mathcal{F}^1$. De façon similaire, nous montrons que γ_p passe infiniment souvent par l'ensemble $(L^1 \times F_i^2)$ pour tout $F_i^2 \in \mathcal{F}^2$ de la forme $(L^1 \times F_i^2)$.
- $((l_i, l'_i), \psi_i \wedge \psi'_i, (l_{i+1}, l'_{i+1})) \in E^p$. Par hypothèse, $(\tau, i) \leftsquigarrow \psi_i$ et $(\tau, i) \leftsquigarrow \psi'_i$. Par conséquent, nous avons $(\tau, i) \leftsquigarrow \psi_i \wedge \psi'_i$ avec $UpL(\psi_i \wedge \psi'_i) \subseteq UpL(\varphi_i)$ (voir équations 3.1 et 3.2 ci-dessus).

γ_p est donc un chemin acceptant. □

3.4.2 Illustration biologique

Dans notre procédure de model-checking, le produit-LT est exécuté sur l'automate Event-Clock du modèle THPN A_M (Figure 3.6) et l'automate Event-Clock de la négation de la propriété $A_{\neg f}$, avec $\neg f \equiv \neg \Box \diamond (m(T3) = 0)$ (Figure 3.7).

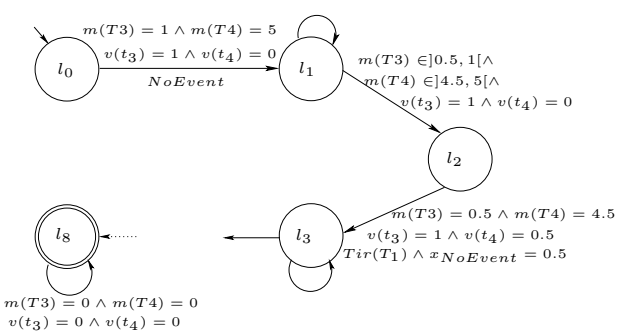

FIG. 3.6 – Automate Event-Clock associé au modèle THPN A_M.

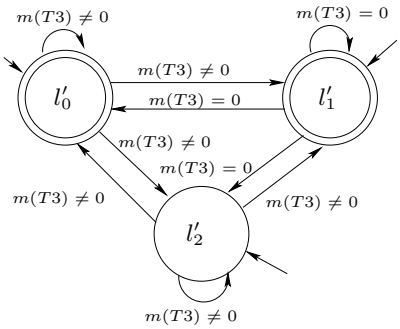

FIG. 3.7 – Automate Event-Clock de la négation de $f \equiv \Box \diamond (m(T3) = 0)$, $A_{\neg f}$.

Lorsque le produit-LT est appliqué sur ces deux automates Event-Clock, nous obtenons l'automate Event-Clock produit $A_p = A_M \times A_{\neg f}$, dont une portion est représentée sur la Figure 3.8.

Détail du produit-LT :

L'automate Event-Clock produit A_p possède 3 localisations initiales : (l_0, l'_0), (l_0, l'_1) et (l_0, l'_2). La Figure 3.8 détaille l'automate produit à partir de la localisation initiale (l_0, l'_0).

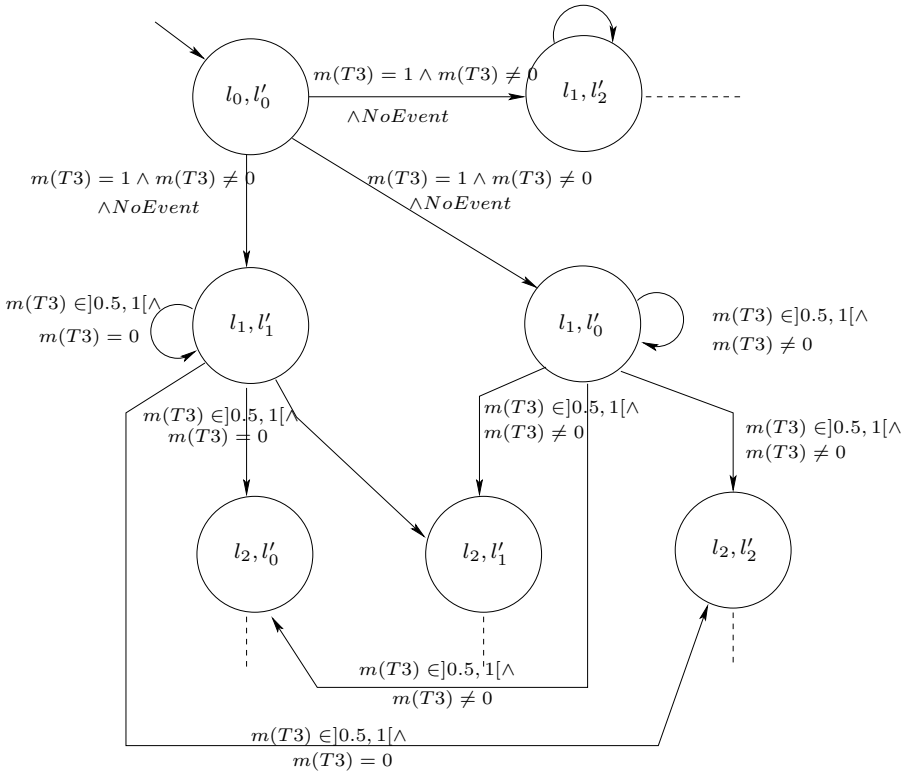

FIG. 3.8 – Portion de l'automate Event-Clock produit $A_p = A_M \times A_{\neg f}$.

De cette localisation trois localisations sont atteignables (l_1, l'_0), (l_1, l'_1) et (l_1, l'_2). Un arc de l'automate produit allant de la localisation (l_i, l'_j) à la localisation (l_{i+1}, l'_{j+1}) est étiqueté par la conjonction des observations ψ_i et ψ_j où ψ_i est l'observation étiquetant l'arc allant de la localisation l_i à la localisation l_{i+1} dans l'automate A_M et où ψ_j est l'observation étiquetant l'arc allant de la localisation l'_j à la localisation l'_{j+1} dans l'automate $A_{\neg f}$. Ainsi, l'arc allant de (l_0, l'_0) à (l_1, l'_0) est étiqueté par l'observation $m(T3) = 1 \wedge m(T4) = 5 \wedge v(t_3) = 1 \wedge v(t_4) = 0 \wedge NoEvent \wedge m(T3) \neq 0$. La Figure 3.8 ne présente qu'une portion des observations étiquetant les arcs. On procède de même pour le reste de l'automate.

3.4.3 Détermination du vide

L'automate produit $A_p = A_M \times A_{\neg f}$ étant à présent construit, il nous reste à décider si le langage temporisé de cet automate est vide ou pas. La détermination du vide est fondée sur la construction d'un automate de Büchi, appelé *l'automate de région*, qui accepte le langage atemporisé de l'automate Event-Clock A_p. Raskin et Schobbens présentent dans [36] la procédure de détermination du vide pour les automates Event-Clock propositionnels. Etant donné que la notion d'automate de région n'est que très peu intuitive, nous allons dans un premier temps rappeler sa définition dans un cadre propositionnel que nous illustrerons sur un exemple assez simple. Nous présenterons alors les modifications que nous avons apportées dans notre procédure de décision pour pouvoir traiter la détermination du vide dans notre cas (les atomes ne sont pas que propositionnels).

3.4.3.1 Cas des automates Event-Clock propositionnels

Dans cette section, nous présentons la construction de l'automate de région associé à l'automate Event-Clock A. On rappelle que l'objectif est de construire un automate qui accepte le langage atemporisé (*Untimed* en anglais) de l'automate Event-Clock, c'est-à-dire :

$$Untimed(A) = \{\varphi_0\varphi_1...\varphi_n...|\tau = (\varphi_0, t_0)(\varphi_1, t_1)...(\varphi_n, t_n)... \in \mathcal{L}(A)\}$$

Définition 3.10 (Etat étendu)**.** *Un* état étendu *d'un automate Event-Clock $A = (L, L_0, \Sigma, \mathcal{C}, E, \mathcal{F})$ est une paire (l, η) où $l \in L$ est une localisation et $\eta : \mathcal{C} \to \mathbb{R}^+ \cup \{\perp\}$ est une valuation d'horloge qui associe une valeur dans $\mathbb{R}^+ \cup \{\perp\}$ à chaque horloge $z \in \mathcal{C}$ de l'automate A.*

La définition suivante formalise l'effet du temps qui passe sur les valuations d'horloge.

Définition 3.11. *La valuation d'horloge η' obtenue à partir de la valuation d'horloge η en laissant passer un temps t, notée $\eta + t$ est définie comme suit :*

- *Pour toute horloge de prédiction y : $(\eta+t)(y) = \eta(y)-t$ si $\eta(y)-t \geq 0$; sinon $\eta+t$ n'est pas définie.*
- *Pour toute horloge historique x : $(\eta + t)(y) = \eta(y) + t$.*

Etant donné que le temps est modélisé par des réels positifs ou nuls, le nombre d'états étendus ne peut pas être énuméré. Néanmoins, pour évaluer les contraintes temps réel étiquetant les arcs de l'automate Event-Clock, seules la partie entière et la partie fractionnaire de l'horloge sont requises. Ainsi, pour savoir quelle horloge changera en premier sa valeur, nous n'avons besoin que d'un ordre sur les parties fractionnaires des valeurs des horloges. De ces remarques, nous rappelons la définition d'équivalence entre valuations. Cette relation d'équivalence partitionne les valuations en nombre fini de classes d'équivalence, appelées *régions*. Deux états dans la même région ont un comportement similaire.

Définition 3.12 (Région)**.** *Deux valuations d'horloge η_1, η_2 sont dans la même région, noté $\eta_1 \approx \eta_2$, pour un automate Event-Clock $A = (L, L_0, \Sigma, \mathcal{C}, E, \mathcal{F})$ si et seulement si les conditions suivantes sont respectées :*

- *η_1 et η_2 sont en accord sur les horloges de valeurs non définies $\perp$. Les autres horloges sont appelées actives.*

- *η_1 et η_2 sont en accord sur la partie entière de toutes les horloges actives valant au plus c, où c est la plus grande constante apparaissant dans les contraintes temps-réel qui étiquettent les arcs de A (en cas de rationnels, toutes les constantes sont multipliées par leur plus petit multiple commun) :*
  - *Pour tout horloge active $z \in \mathcal{C}$ avec la valuation η_1, si $\eta_1(z) \leq c$ ou $\eta_2(z) \leq c$ alors $\lfloor \eta_1(z) \rfloor = \lfloor \eta_2(z) \rfloor$.*

- *η_1 et η_2 sont en accord sur l'ordonnancement de la partie fractionnaire de toutes les horloges actives valant au plus c :*
  - *Pour une horloge de prédiction y, la partie fractionnaire de $\eta_1(y)$, notée $\langle \eta_1(y) \rangle$, vaut $\lceil \eta_1(y) \rceil - \eta_1(y)$,*

- *Pour une horloge historique x, la partie fractionnaire de $\eta_1(x)$, notée $\langle \eta_1(x) \rangle$ vaut $\eta_1(x) - \lfloor \eta_1(x) \rfloor$,*
- *Pour toute horloge active z_1, z_2 avec $\eta(z_1) \leq c$ et $\eta(z_2) \leq c$:*
  - *$\langle \eta_1(z_1) \rangle = 0$ si et seulement si $\langle \eta_2(z_1) \rangle = 0$*
  - *$\langle \eta_1(z_1) \rangle \leq \langle \eta_1(z_2) \rangle$ si et seulement si $\langle \eta_2(z_1) \rangle \leq \langle \eta_2(z_2) \rangle$*

Une *région d'horloge* β est une classe d'équivalence de $\approx$. Deux états étendus (l_1, η_1), (l_2, η_2) sont région-équivalents si $l_1 = l_2$ et $\eta_1 \approx \eta_2$.

Il ne nous reste alors qu'à définir quand est-ce qu'une région d'horloge β_2 est un successeur temporel d'une autre région β_1.

Définition 3.13. *Une région d'horloge β_2 est un successeur temporel d'une région d'horloge β_1 si et seulement si $\forall \eta_1 \in \beta_1, \exists t \in \mathbb{R}^+$ tel que $\eta_1 + t \in \beta_2$.*

L'ensemble de ces définitions peut facilement se comprendre à l'aide d'un exemple. Prenons l'automate Event-Clock de la Figure 3.9. Celui-ci ne contient que deux horloges, toutes les deux historiques x_a et x_b. La plus grande constante c apparaissant dans les contraintes d'horloge est 2. On en déduit l'ensemble de régions présenté sur la Figure 3.10.

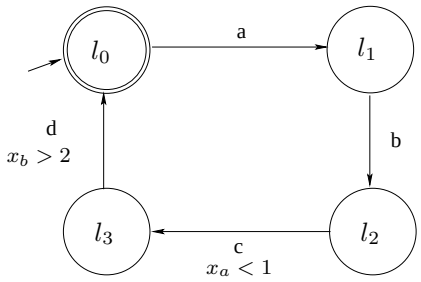

FIG. 3.9 – Exemple d'un automate Event-Clock.

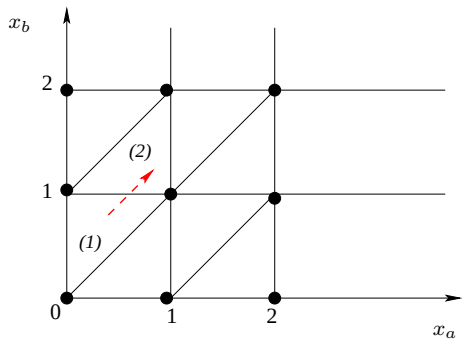

FIG. 3.10 – Ensemble de régions associées aux horloges historiques x_a et x_b.

Chaque trait, chaque point et chaque surface forment une région de la Figure 3.10. Voici quelques exemples de régions :

- **Région ponctuelle :** $(x_a = 0, x_b = 1, \langle x_a \rangle = \langle x_b \rangle)$

- **Région linéaire :** $(0 < x_a < 1, x_b = 1, \langle x_a \rangle > \langle x_b \rangle)$
- **Région surfacique :** $(0 < x_a < 1, 0 < x_b < 1, \langle x_a \rangle < \langle x_b \rangle)$, numérotée (1) sur la Figure 3.10.

Les régions succédant à une autre région s'obtiennent en suivant la direction donnée par la flèche en pointillés sur la Figure 3.10. Ainsi, si nous partons de la région $(0 < x_a < 1, 0 < x_b < 1, \langle x_a \rangle < \langle x_b \rangle)$, notée (1), nous obtenons la succession suivante :

$(0 < x_a < 1, 0 < x_b < 1, \langle x_a \rangle < \langle x_b \rangle) \rightarrow (0 < x_a < 1, x_b = 1, \langle x_a \rangle > \langle x_b \rangle) \rightarrow$
$(0 < x_a < 1, 1 < x_b < 2, \langle x_a \rangle > \langle x_b \rangle) \rightarrow (x_a = 1, 1 < x_b < 2, \langle x_a \rangle > \langle x_b \rangle) \rightarrow ...$

La définition de l'automate de région $R(A)$ de l'automate Event-Clock A peut à présent être donnée.

Définition 3.14 (Automate de région)**.** *L'automate de région de l'automate Event-Clock* $A = (L, L_0, \Sigma, \mathcal{C}, E, \mathcal{F})$ *est l'automate de Büchi* $R(A) = (L^r, L_0^r, \Sigma^r, E^r, \mathcal{F}^r)$ *où :*

- L^r *est l'ensemble de régions* (l, β) *avec* $l \in L$*, et* β *est une classe d'équivalence des valuations d'horloges,*

- L_0^r *est le sous-ensemble de localisations* (l, β) *où* $l \in L_0$ *et pour toute horloge historique* x*,* $\beta(x) = \perp$

- $\Sigma^r = Obs(\Sigma) \cup \{\epsilon\}$

- *Les éléments de* E^r *sont les triplets* $((l_1, \beta_1), s, (l_2, \beta_2))$ *tels que :*

  - *si* $s \in Obs(\Sigma)$*, il y a un arc* (l_1, s, l_2) *dans l'automate* A *et une région d'horloges* β_3 *telle que :*

    - $\beta_1 = \beta_3[y_\alpha := 0 | \alpha \in s]$ *(*β_1 *est en accord avec* β_3 *sur toutes les horloges excepté les horloges de prédiction associées aux atomes apparaissant dans* s *; ces horloges ont la valeurs* 0 *dans* β_1*),*
    - $\beta_2 = \beta_3[x_\alpha := 0 | \alpha \in s]$ *(*β_2 *est en accord avec* β_3 *sur toutes les horloges excepté les horloges historiques associées aux atomes apparaissant dans* s *; ces horloges ont la valeurs* 0 *dans* β_2*),*
    - *Lorsque l'arc est franchi, la valeur des horloges est cohérente avec les contraintes temps-réel de* s*.*

  - *si* $s = \epsilon$ *alors la région* β_2 *est un successeur temporel de la région* β_1 *et* $l_1 = l_2$*.*

- $\mathcal{F}^r = \{F'_1, ..., F'_n\} \cup \{F_{x_\alpha} | \alpha \in \Sigma^r\} \cup \{F_{x_\alpha} | \alpha \in \Sigma^r\}$

  - *pour tout* i*,* $F'_i = \{(l, \beta) | l \in F_i$ *et* β *est une classe d'équivalence de valuations d'horloges*$\}$
  - $F_{x_\alpha} = \{(l, \beta) | \eta(x_\alpha) = 0 \ \vee \ \eta(x_\alpha) > c \ \vee \ \eta(x_\alpha) = \perp, \forall \eta \in \beta\}$
  - $F_{y_\alpha} = \{(l, \beta) | \eta(y_\alpha) = 0 \ \vee \ \eta(y_\alpha) = \perp, \forall \eta \in \beta\}$

Les deux derniers points assurent que les horloges soient remises à zéro, indéfinies ou plus grandes que leur valeur maximale, infiniment souvent. Ceci est imposé par la progression temporelle.

L'automate de région associé à l'automate Event-Clock de la Figure 3.9 est représenté sur la Figure 3.11.

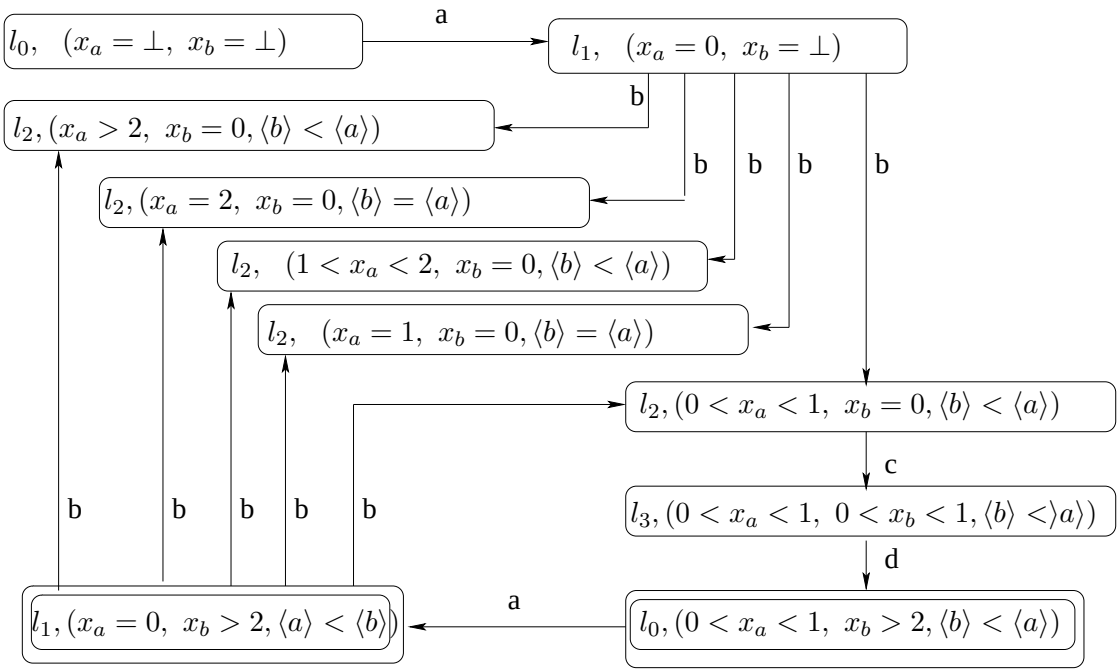

FIG. 3.11 – Automate de région.

Explications :

L'automate Event-Clock ne possède qu'une unique localisation initiale l_0. De plus, initialement, les deux horloges historiques sont indéfinies $x_a = x_b = \perp$. Lors du franchissement de l'arc allant de l_0 à l_1, l'atome a est lu, l'horloge historique x_a devient donc active alors que x_b reste inactive. Nous sommes donc dans l'état $(l_1, x_a = 0, x_b = \perp)$ de l'automate de région.

L'arc allant de l_1 à l_2 dans l'automate Event-Clock reconnaît l'atome b, l'horloge x_b devient donc active et vaut 0. Etant donné qu'aucune contrainte temporelle n'étiquette l'arc allant de l_1 à l_2, toute région successeur de $(x_a = 0, x_b = \perp)$ donne un état dans l'automate de région. On en déduit les 5 états de l'automate de région associées à l_2.

Pour passer de la localisation l_2 à l_3, une contrainte d'horloge impose que $x_a < 1$. Parmi les 5 états précédents de l'automate de région, seul l'état $(l_2, (0 < x_a < 1, x_b = 0, b < a))$ possède un successeur permettant de satisfaire la contrainte temporelle $x_a < 1$, il s'agit de l'état $(l_3, (0 < x_a < 1, 0 < x_b < 1, b < a))$, etc.

L'automate de région $R(A)$ reconnaît un langage non temporisé. Alur *et al.* [37] ont montré que le langage reconnu par cet automate est exactement le langage atemporisé de l'automate Event-Clock A.

Théorème 3.4. *[37] Soit A un automate Event-Clock et $R(A)$ l'automate de région associé, le langage de $R(A)$ est Untimed($\mathcal{L}(A)$).*

On en déduit le corollaire suivant :

Corollaire 3.2. *[36] Le langage temporisé de A est vide si et seulement si le langage de $R(A)$ est vide.*

3.4.3.2 Notre procédure de décision

Dans notre procédure de model-checking, nous avons étendu la notion d'atomes propositionnels à celle d'atomes instantanés (Définition 2.3 page 44). Les atomes instantanés introduisent de la sémantique dans l'étiquetage des arcs. Par conséquent, nous ne pouvons appliquer la procédure de construction de l'automate de région directement sur l'automate résultant du produit-LT des automates A_M et $A_{\neg f}$. Nous rajoutons donc une étape préalable permettant de se rapporter à un cas propositionnel comme décrit ci-dessus.

Vers un étiquetage propositionnel : Modification des étiquettes. Dans un produit usuel d'automates, l'arc $((l_1, l'_1), a, (l_2, l'_2))$ de l'automate produit $A_p = A_1 \times A_2$ ne peut provenir que des arcs (l_1, a, l_2) et (l'_1, a, l'_2) des deux automates initiaux A_1 et A_2. Le produit usuel d'automates nécessite l'égalité des étiquettes sur les arcs.

Dans notre cas, le produit-LT (Définition 3.9 page 69) autorise la construction de beaucoup plus d'arcs. Les théorèmes 3.2 et 3.3 montrent que ces arcs surnuméraires ne peuvent pas engendrer des chemins acceptants autres que ceux provenant de l'intersection des langages des deux automates initiaux A_1 et A_2. Prenons l'exemple des arcs (l_1, a, l_2) et $(l'_1, \neg a, l'_2)$ provenant respectivement des automates A_1 et A_2. D'après la définition du produit-LT, l'arc $((l_1, l'_1), a \wedge \neg a, (l_2, l'_2))$ est un arc de l'automate produit, mais aucun chemin acceptant ne peut passer par cet arc. Nous ajoutons donc deux étapes préliminaires permettant de retirer tous ces arcs surnuméraires incohérents.

Rappelons que chaque observation bien formée ψ_i étiquetant les arcs de l'automate peut s'écrire sous forme de trois sous-observations : $\psi_i \equiv \psi_{i_1} \wedge \psi_{i_2} \wedge \psi_{i_3}$ (voir Notation 2 page 47). De plus, par la Définition 2.15, $(\tau, i) \not\sim \psi_i$ si et seulement si $(\tau, i) \not\sim \psi_{i_1}$ et $(\tau, i) \not\sim \psi_{i_2}$ et $(\tau, i) \not\sim \psi_{i_3}$.

- Comme décrit précédemment, la première étape consiste à chercher une incohérence dans les sous-observations ψ_{i1}. La sous-observation ψ_{i_1} ne considère que les propositions. Il est donc facile de voir si ψ_{i_1} est satisfiable ou pas. *Une sous-observation ψ_{i1} est alors incohérente si et seulement si $p \wedge \neg p$ est une sous-formule de ψ_{i1}, avec $p \in Pr$ une proposition [68].*
  On insiste sur le fait que si ψ_{i_1} est insatisfiable alors aucune trace discrète τ ne pourra satisfaire ψ_{i_1} à la position i.

- Les sous-observations ψ_{i2} concernent les atomes instantanés de la forme $v \geq v'$ où $v, v' \in V \amalg \mathbb{R}$. Etant donné que notre produit-LT fait la conjonction des atomes,

plusieurs informations portant sur la même variable v peuvent être réparties dans plusieurs atomes de l'observation étiquetant l'arc. Par exemple, $v \geq 2 \wedge v \geq 4$. De plus, la conjonction de ces atomes instantanés peut engendrer des incohérences sémantiques, qu'il serait bien de supprimer. Par exemple, si nous reprenons l'automate produit dont une portion est présentée sur la Figure 3.8, l'arc bouclant sur la localisation (l_1, l'_1) est étiqueté par l'observation $m(T3) = 0 \wedge m(T3) \in]0.5; 1[$. Il est évident qu'il n'existe pas d'évaluation possible de la variable $m(T3)$ permettant de satisfaire simultanément les deux atomes. Nous ajoutons donc une étape permettant de vérifier la cohérence de la sous-observation ψ_{i2} et permettant de *modifier les étiquettes* pour qu'au plus une information soit associée à chaque variable.
La sous-observation ψ_{i2} peut être perçue comme un système très simple de contraintes. Chaque atome instantané $v \geq v'$ correspond à une contrainte, qui peut être représentée sous forme de polyèdres [69]. La conjonction de deux atomes instantanés $v \geq v' \wedge v \geq v''$, se traduit par l'intersection des deux polyèdres correspondant respectivement à la contrainte $v \geq v'$ et à la contrainte $v \geq v''$. A chaque arc de l'automate produit est alors associé un système de contraintes $\{c_1, ..., c_n\}$ où les c_i sont des contraintes de la forme $v \geq v'$ avec $v, v' \in V \amalg \mathbb{R}$.
Wilde [69] montre que le polyèdre associé à un système de contraintes $\{c_1, ..., c_n\}$ est vide si et seulement si la conjonction des contraintes c_i est insatisfiable. Ainsi, dans le cas où le polyèdre associé à un arc de l'automate produit est vide, nous pouvons conclure que la sous-observation ψ_{i2} est insatisfiable, c'est-à-dire qu'il n'existe pas de trace temporisée τ qui satisfait ψ_{i_2} à une certaine position i.

- Soit (l_i, ψ_i, l_{i+1}) un arc tel que ψ_i a été montré insatisfiable à cause de ψ_{i_1} ou de ψ_{i_2}. Il n'existe donc pas de trace temporisée τ telle que $(\tau, i) \not\sim \psi_i$. Par la suite, nous considérons l'automate A'_p dans lequel les arcs insatisfiables ont été supprimés et dans lequel les étiquettes ont été modifiés. La Figure 3.12 présente la portion de l'automate produit de la Figure 3.8 modifié.

Les modifications apportées à l'automate produit A'_p modifié permet de ne plus considérer la sémantique des étiquettes. On peut à présent se rapporter à un cas propositionnel.

Traitement du cas propositionnel La sous-observation ψ_{i_3} est composée des contraintes temporelles qui utilisent les horloges historiques et celles de prédiction x_α et y_α, avec α un atome instantané. A cause de la contrainte imposée sur les atomes composés d'horloges (*i.e.* comparaison des horloges avec des rationnels), la procédure classique d'automate de région peut à présent être utilisée. Ainsi, d'après ce qui a été vu précédemment, nous pouvons conclure que le langage temporisé de l'automate produit modifié A'_p est vide si et seulement si le langage de l'automate de région $R(A'_p)$ est vide.

3.4.3.3 Illustration biologique

Nous avons implémenté un logiciel permettant d'automatiser la construction de l'automate de région. Cet outil sera détaillé dans le Chapitre 4. L'automate de région associé à l'automate produit A'_p de la Figure 3.12 étant trop important, celui-ci ne sera pas détaillé ici. Sa construction suit le même procédé que celui utilisé pour l'automate de région de

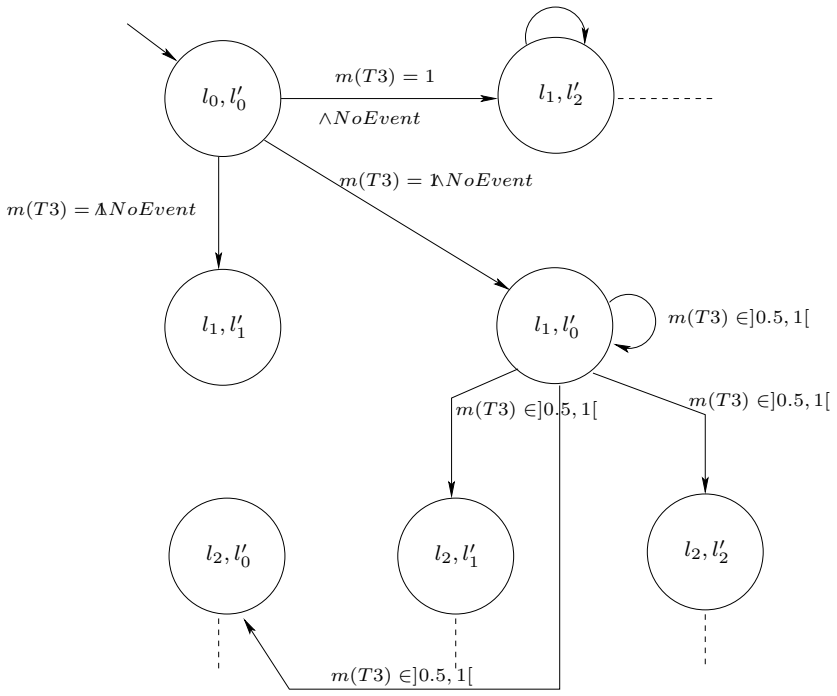

FIG. 3.12 – Portion de l'automate produit modifié.

la Figure 3.11. Dans le cas de notre illustration biologique, nous arrivons à la conclusion que le langage de l'automate produit A'_p est vide.

Pour conclure que notre modèle THPN satisfait la propriété f étudiée, il nous faut donner les démonstrations de correction et complétude de notre procédure. La partie suivante traite de ces deux questions.

3.5 Correction et complétude de la procédure

3.5.1 Théorèmes finaux

Lemme 3.6. *Soient M le modèle THPN en temps continu, f une formule CTEL, τ une trace temporisée discrète et A_M l'automate Event-Clock associé au modèle,*

$$M \models f \Rightarrow \forall \tau \in \mathcal{L}(A_M), \tau \backsim f$$

Preuve. Ce lemme découle directement du corollaire 2.1 (page 49) et du lemme 3.5 (page 64). □

Lemme 3.7 (Correction de la procédure)**.** *Soient M le modèle THPN en temps continu, f une formule CTEL, $A_{\neg f}$ l'automate Event-Clock associé à la négation de f et A_M l'automate Event-Clock associé au modèle,*

$$\mathcal{L}(A_{\neg f} \times A_M) = \emptyset \Rightarrow M \models f$$

Preuve. Supposons que $\mathcal{L}(A_{\neg f} \times A_M) = \emptyset$ et que $M \models \neg f$.
Du lemme 3.6, il découle $\forall \tau \in \mathcal{L}(A_M),\ \tau \not\sim \neg f$.
D'après le théorème 3.1, on a $\tau \not\sim \neg f \Rightarrow \tau \in \mathcal{L}(A_{\neg f})$. Ainsi, $\forall \tau \in \mathcal{L}(A_M),\ \tau \in \mathcal{L}(A_{\neg f})$. Etant donné que le produit-LT est complet, nous déduisons que $\forall \tau \in \mathcal{L}(A_M),\ \tau \in \mathcal{L}(A_{\neg f} \times A_M)$. Etant données les restrictions (résolution de conflits et réseau de Petri borné) qui ont été imposées sur les propriétés des THPN (page 27), nous déduisons que $\mathcal{L}(A_M) \neq \emptyset$. Par conséquent, $\mathcal{L}(A_{\neg f} \times A_M) \neq \emptyset$. Contradiction. □

Lemme 3.8 (Complétude de la procédure)**.** *Soient M le modèle THPN en temps continu, f une formule CTEL, $A_{\neg f}$ l'automate Event-Clock associé à la négation de f et A_M l'automate Event-Clock associé au modèle,*

$$M \models f \Rightarrow \mathcal{L}(A_{\neg f} \times A_M) = \emptyset$$

Preuve. Supposons que $M \models f$ et que $\mathcal{L}(A_{\neg f} \times A_M) \neq \emptyset$.
$\mathcal{L}(A_{\neg f} \times A_M) \neq \emptyset$ donc , $\exists \tau \in \mathcal{L}(A_{\neg f} \times A_M)$. D'après le théorème 3.2 (page 69), il découle $\tau \in \mathcal{L}(A_M)$ *et* $\tau \in \mathcal{L}(A_{\neg f})$.
$M \models f$ donc, d'après le lemme 3.6, on a $\forall \tau \in \mathcal{L}(A_M), \tau \not\sim f$ et d'après le théorème 3.1 (page 67), il découle $\forall \tau \in \mathcal{L}(A_{\neg f}), \tau \not\sim \neg f$.
Toutes les traces temporisées $\tau \in \mathcal{L}(A_{\neg f} \times A_M)$ doivent alors satisfaire f et $\neg f$, *i.e.* : $\forall \tau \in \mathcal{L}(A_{\neg f} \times A_M),\ \tau \not\sim f \wedge \neg f$. Il a été prouvé (corollaire 3.1, page 67) qu'aucune trace $\tau \in \mathcal{L}(A_{\neg f})$ ne pouvait être telle que $\tau \not\sim f \wedge \neg f$. Contradiction. □

Théorème 3.5. *Soit M le modèle THPN en temps continu, f une formule CTEL, $A_{\neg f}$ l'automate Event-Clock associé à la négation de f et A_M l'automate Event-Clock associé au modèle, $M \models \tau \Leftrightarrow \mathcal{L}(A_{\neg f} \times A_M) = \emptyset$*

Preuve. Ce théorème découle directement des deux précédents lemmes. □

3.5.2 Illustration biologique

Etant donné que le langage de l'automate produit A'_p est vide, nous en déduisons que notre modèle THPN satisfait la propriété $f \equiv \Box \diamond (m(T3) = 0)$, ce qui traduit la prédominance de la déiodinase de type 3, inhibitrice des hormones thyroïdiennes par rapport à la déiodinase de type 2, activatrice des hormones thyroïdiennes. L'exploitation biologique de ce résultat sera détaillé dans le chapitre biologique (Chapitre 5).

3.6 Résumé

La Figure 3.13 présente un schéma synthétisant les notations utilisées dans notre procédure ainsi que les liens existant entre elles. La question initiale : «Notre modèle THPN satisfait-il la propriété étudiée ?» est résumée par la double flèche rouge entre les THPN et l'ensemble de formules bien formées For. Pour répondre à cette question, il faut passer la frontière des modèles pour passer à celle de la vérification (F).

Le THPN, soumis à une stratégie de résolution de conflits, produit des exécutions en temps continu (1) qui peuvent être converties en automate Event-Clock A_M (2). Les

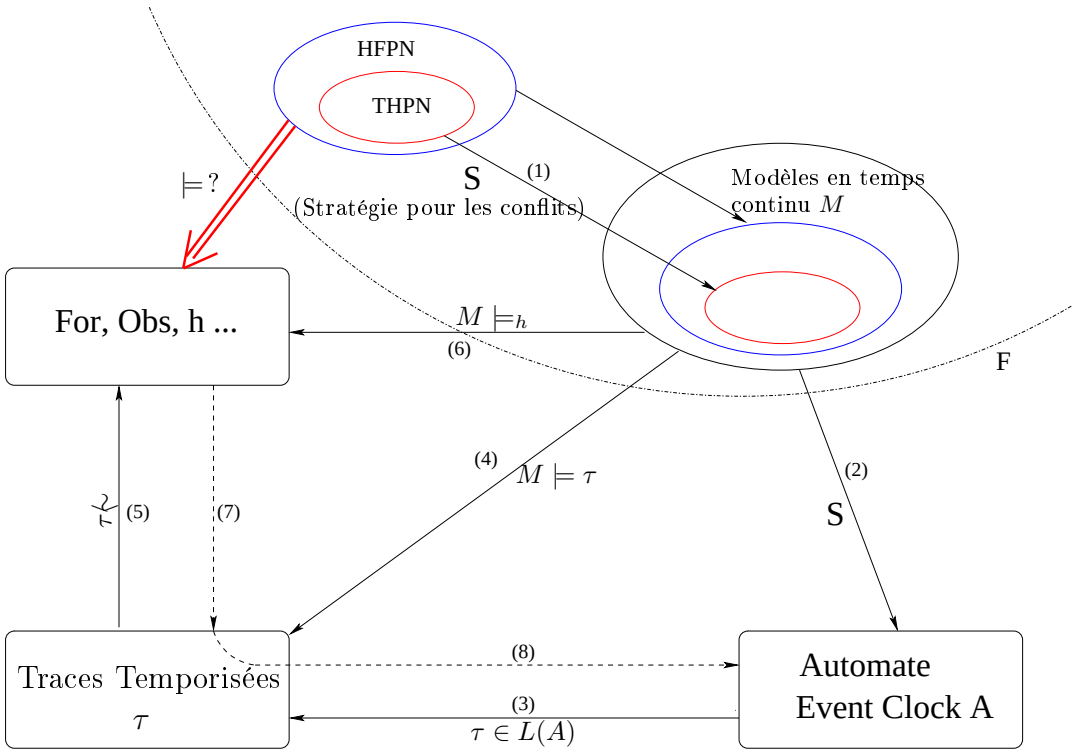

FIG. 3.13 – Résumé de la procédure globale de model-checking pour les THPN.

traces temporisées reconnues par l'automate A_M (3) sont satisfaites par le modèle en temps continu M (4). En effet, on a montré que $M \models \tau$ si τ appartient à $L(A)$, A étant l'automate associé à M. On remarque que (4) = (2)+(3). Nous définissons alors les règles de satisfaction entre une trace temporisée et une formule (5) et entre un modèle en temps continu et une formule (6) étant donnée une séquence temporelle h.

La propriété étudiée f est une formule bien formée de For qui représente un ensemble de traces temporisées. Celle-ci peut être convertie en automate Event-Clock (7) et (8). Le problème de satisfaction initial est donc réduit à l'étude d'automates Event-Clock. Notre procédure permet de conclure que le THPN satisfait la propriété f (double flèche rouge) si et seulement si pour toute trace temporisée de l'automate associé au modèle M (lui-même associé au THPN), on a $M \models \tau$ et $\tau \backsim f$.

Chapitre 4

La Plate-forme logicielle : BioPetri

Notre procédure de model-checking a été implémentée dans le *logiciel BioPetri*, un model-checker pour THPN et formules CTEL qui utilise la conversion en automates Event-Clock. Le but est de décider si un modèle THPN satisfait ou non une propriété f exprimée en CTEL.

Dans un premier temps, nous décrivons la démarche adoptée dans le *logiciel BioPetri*, puis nous détaillons les fichiers et fonctions majeures. Enfin, nous explicitons le mode d'utilisation du logiciel à l'aide de l'exemple biologique traité au cours des précédents chapitres.

4.1 Description du logiciel BioPetri

Pour une stratégie de résolution de conflit donnée, le *logiciel BioPetri* construit un graphe d'évolution à partir du modèle THPN en suivant l'algorithme présenté dans [34] et rappelé dans le Chapitre 1 section 1.4.2.

Du graphe d'évolution, le *logiciel BioPetri* construit un automate Event-Clock A_M suivant les étapes de l'algorithme détaillées dans le Chapitre 3 section 3.2.2.

La procédure pour construire l'automate Event-Clock $A_{\neg f}$ détaillée par Raskin et Schobbens [36] est également implémentée.

Le *logiciel BioPetri* construit l'automate produit-LT des deux automates A_M et $A_{\neg f}$ et nous donne toutes les localisations finales pouvant être atteintes. Si aucune localisation finale n'est atteinte, nous pouvons conclure que le modèle THPN satisfait la propriété étudiée f. Dans le cas contraire, celui-ci ne satisfait pas la propriété.

4.2 Implémentation du logiciel BioPetri

Le code du *logiciel BioPetri* peut être trouvé sur le site web :
http ://stroncale.googlepages.com/home. Deux parties constituent *BioPetri*. La première concerne la construction de graphes d'évolution à partir d'un modèle THPN et la deuxième concerne le model-checker pour automates Event-Clock et logique CTEL. Chaque partie va être décrite séparément.

4.2.1 Construction du graphe d'évolution

Le code de la première partie est regroupé dans un répertoire intitulé *EG_ CONSTRUCTION*. Celui-ci comporte les deux algorithmes permettant la construction du graphe d'évolution à partir d'un THPN (fichier $EG\_from\_THPN.c$), c'est-à-dire :

1. L'algorithme calculant les vitesses instantanées des transitions d'un réseau de Petri continu *(fonction algo54)*,
2. L'algorithme utilisant le précédent pour construire le graphe d'évolution *(fonction algo61)*,

Ces deux algorithmes ont été présentés dans le Chapitre 1 section 1.4.2. Le fichier $EG\_from\_THPN.c$ comprend environ 4500 lignes de code. Le graphe d'évolution est obtenu grâce à l'exécution de la fonction $algo61$.

La compilation et l'exécution du fichier $EG\_from\_THPN.c$ nécessite, outre un compilateur C, un solveur de problèmes de programmation linéaire. En effet, le calcul des vitesses instantanées effectué par l'algorithme 5.4 passe par la résolution de problèmes simples de programmation linéaire. Nous avons utilisé le solveur GLPK disponible à l'adresse suivante : *http ://www.gnu.org/software/glpk*.

4.2.2 Model-Checker pour automates Event-Clock

La deuxième partie correspond au model-checker pour automates Event-Clock et formules CTEL. Les fichiers associés à cette partie sont regroupés dans un répertoire intitulé *ECA_ MODEL_ CHECKING*. On rappelle que l'étape de détermination du vide de notre procédure de model-checking nécessite de déterminer l'incohérence de systèmes de contraintes en les convertissant en ensembles de polyèdres. Notre model-checker utilise donc la bibliothèque de polyèdres développées en C par Wilde [69] (sous-répertoire intitulé *c*). Notre procédure de model-checking comporte 5 étapes détaillées dans le Chapitre 3. Chaque étape est implémentée dans un fichier du sous-répertoire noté *caml* (Figure 4.1) :

- Les automates temps-réel utilisés sont des automates Event-Clock [37] étendus par la notion d'atomes instantanés. Le type de ces automates a été défini dans le fichier *eca.ml* ($\sim$ 300 lignes),
- Le graphe d'évolution, obtenu dans la première partie, est converti en un automate Event-Clock A_M (fonction *construct_ eca* du fichier *petri_ to_ eca.ml* d'environ 225 lignes),
- Soit f la propriété étudiée. La troisième étape de notre procédure consiste à convertir en automate Event-Clock la négation de la propriété. Soit $A_{\neg f}$, l'automate ainsi obtenu (fonction *eca_ of_ for* du fichier *eventclock.ml*, $\sim$ 370 lignes) .
- L'obtention des traces communes aux deux automates A_M et $A_{\neg f}$ se fait grâce au produit-LT (fonction *product* du fichier *product_ sem.ml*, $\sim$ 320 lignes) ,
- La dernière étape consiste à déterminer si le langage de l'automate produit est vide ou non. Nous recherchons alors si des localisations finales sont atteignables ou non (fonction *reachable* du fichier *region.ml*, comprenant environ 1000 lignes de code).

Certains fichiers, non mentionnés ci-dessus (*poly.ml* et *matrix.ml*), sont utilisés pour la conversion en Caml de la bibliothèque de polyèdres. Le code a été inspiré du code source de TEMPO, un model-checker pour les automates Event-Recording propositionnels [70], uniquement composés d'horloges historiques.

L'exécution de ces différents fichiers nécessitent un compilateur Ocaml, disponible à l'adresse suivante : *http ://caml.inria.fr/download.en.html.*

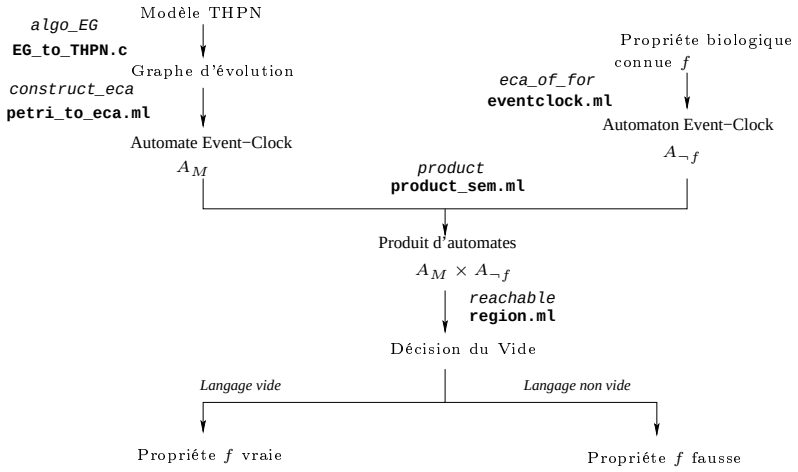

FIG. 4.1 – Résumé des fichiers et fonctions du *Logiciel BioPetri.*

4.3 Développement sur un exemple

Dans cette section, nous détaillons la démarche à suivre pour utiliser le *logiciel BioPetri* sur l'exemple détaillé le long des précédents chapitres.

4.3.1 Construction du graphe d'évolution

- La structure du THPN de la Figure 1.14 est détaillée dans un fichier texte "competition_enzyme.txt" selon le format de la Figure 4.2.

- L'exécutable de *EG_from_THPN.c* (répertoire *EG_CONSTRUCTION*) demande un nom de fichier :
  - `Nom du Fichier`
  - `competition_enzyme.txt`

  Dans le fichier *EG_print* se trouve la version imprimée du graphe d'évolution obtenu. Lors de l'exécution, trois fichiers ont été créés : *nom.lp*, *sortie.txt* et *uj*. Ces fichiers sont utilisés pour la résolution des systèmes de programmation linéaire lors de l'appel du solveur GLPK. Leur contenu ne sera pas détaillé ici.

- Le graphe d'évolution obtenu se définit par :
  - *matmar* : la matrice de marquage pour tous les IB-states,
  - *time_ event* : les temps successifs des transitions du graphe d'évolution récapitulés sous forme de vecteur,
  - *matevt* : la matrice des événements ayant lieu à chaque temps du vecteur *time_ event*,
  - *matvit* : la matrice récapitulant le vecteur de vitesses instantanées de chaque transition du graphe d'évolution et,
  - *nloop* : le numéro de l'IB-state sur lequel boucle le graphe d'évolution. En cas de deadlock, *nloop* correspond au numéro du dernier IB-state.

4.3.2 Model-Checker pour automates Event-Clock

Un fichier *Makefile* a été créé de façon à automatiser l'appel des fonctions décrites ci-dessus les unes à la suite des autres (sous-répertoire *caml* du répertoire *ECA_ MODEL_ CHECKING*). L'exécution du *Makefile* permet l'appel des fonctions suivantes :

- L'appel de la fonction *construt_ eca* du fichier *petri_ eca* se fait comme suit :

$$let\ \ a_M = construt\_eca\ nbTGE\ matmar\ matevt\ matvit\ time\_event \\ nbPd\ nbP\ nbTd\ nbT\ nbloop\ nbIB\ nom$$

Nombre de places 4

Nombres de places discrètes 0

Nombre de transitions 4

Nombre de transitions discrètes 1

Nombre de sous-réseaux-continus 1

Nombre de conflits de type 1, 2, 3 et 4 0 1 2 0

Résolution de conflit de type 1

Résolution de conflit de type 2 3 2 ($T_3 < T_2$)

Résolution de conflit de type 3 1 2 ($T_1 < T_2$)

Résolution de conflit de type 4 3 1 ($T_3 < T_1$)

Matrice Pre

	T_1	T_2	T_3	T_4
$T3$	0.5	1	1	0
$T4$	0	0	1	1
$D2$	0	0	0	1
$D3$	0	0	1	0

Matrice Post

	T_1	T_2	T_3	T_4
$T3$	0.5	1	0	1
$T4$	0	0	0	0
$D2$	0.5	0	0	1
$D3$	0	1	1	0

Marquage initial : $(T3, T4, D2, D3)$

$(1, 5, 0, 0)$

Vitesses maximales : (T_2, T_3, T_4)

$(1, 1, 1)$

Délais : (0.5)

FIG. 4.2 – Format du fichier décrivant la structure du modèle THPN.

Cette fonction renvoie l'automate du modèle THPN a_M de type *Eca*. Elle prend comme arguments, ceux cités précédemment ainsi que $nbTGE$ et $nbIB$, correspondant respectivement au nombre de transitions et d'IB-states du graphe d'évolution, ainsi que $nbPd$, nbP, $nbTd$ et nbT représentant respectivement le nombre de places discrètes, le nombres de places, le nombres de transitions discrètes et le nombre de transitions totales. L'automate a_M est décrit dans le fichier intitulé *formule_ tetard.ml*.

- La négation de la propriété étudiée nf est traduite dans le type *formule* du fichier *Eventclock.ml*. On appelle alors la fonction *eca_ of_ for* :

$$let \;\; a_{nf} = eca\_of\_for \;\; nf \;\; \text{“}nom\_nf\text{”}$$

où *nf* est de type formule et “nom_nf” est le nom de la formule donnée sous forme de chaîne de caractères. La définition du type formule CTEL se comprend facilement d'après le code source. Prenons l'exemple de la formule simplifiée

$$f \equiv \diamond (m(T3) = 0) \equiv \top U (m(T3) = 0)$$

qui s'implémente comme suit :

```
(*Définition de l'atome m(T3)=0*)
let a = {sign = Pos; variable=m(T3); comp=Eq; valeur=0}
let nf = Eventclock.U(Eventclock.Atome True, Eventclock.Atome a)
```

- La fonction *product* a_M a_{nf} du fichier *product_ sem.ml* renvoie l'automate EventClock de type *Eca* résultant du produit des deux automates a_M et a_{nf}.

- La dernière étape consiste à tester l'atteignabilité des états finaux de l'automate obtenu ci-dessus. On appelle la fonction *reachable tab eca_ p* du fichier *reachable.ml* où *tab* correspond à un tableau récapitulant les horloges présentes dans l'automate produit *eca_ p*.

La compilation de toutes ces fonctions est incluse dans le fichier *Makefile*. Néanmoins, de façon à contrôler l'exécution du fichier *formule_ tetard.ml*, celle-ci n'a pas été intégrée dans le *Makefile*. On procède donc comme suit :

- `make`
- `(*Exécution du fichier formule_tetard.ml*)`
- `./formule_tetard.ml`

A l'écran s'affichent alors l'automate produit, ainsi que la conclusion quant au vide du langage de cet automate produit.

Le temps d'exécution de la procédure complète est très variable selon que le langage de l'automate produit est vide ou non. Ainsi, dans notre exemple, le langage final est vide, le logiciel *BioPetri* ne met alors que quelques secondes pour terminer son exécution. Par contre, dans le cas où le langage n'est pas vide, la détermination de tous les états finaux

atteignables peut prendre quelques heures.

L'automatisation de notre procédure de model-checking nous a permis de nous intéresser à des systèmes biologiques beaucoup plus importants, c'est ce que nous détaillons dans le chapitre suivant.

Chapitre 5

Application à la métamorphose amphibienne

5.1 Contexte biologique

5.1.1 Le Xénope : modèle étudié

Le genre Xenopus est composé de 18 espèces naturellement disséminées à travers l'Afrique et l'Amérique Latine [71]. Parmi les différentes espèces que compte ce genre, *Xenopus laevis* est la plus largement employée par les biologistes. Importée en Europe dans les années 1930, elle fut tout d'abord utilisée dans le cadre d'études d'endocrinologie. Par la suite, *Xenopus laevis* a supplanté les autres modèles amphibiens couramment utilisés en Europe. En effet, *Xenopus laevis* est d'une relative facilité d'élevage : l'induction des pontes est possible jusqu'à 5 fois par an. Ces pontes, très abondantes, sont facilement fécondables *in vitro*. On peut ainsi obtenir un grand nombre d'embryons dont le développement est synchronisé. L'embryon devenu têtard met environ deux mois pour se métamorphoser en un juvénile qui deviendra sexuellement mature au bout de 2 ans.

La structure du génome de *X. laevis* est relativement complexe. D'une taille de 3,2 Gpb, ce génome possède 36 chromosomes et est allotétraploïde, c'est-à-dire qu'au lieu d'avoir des paires de chromosomes, l'animal a des groupes de 4 chromosomes. *X. laevis* est un modèle unique pour l'étude de la métamorphose. Toutefois, la complexité de son génome en fait un modèle qui s'intègre difficilement dans le cadre d'études fonctionnelles en génétique et en génomique. C'est dans l'objectif de mener de telles approches tout en conservant le bénéfice des connaissances obtenues chez *X. laevis* que des scientifiques de plusieurs laboratoires ont reporté leurs efforts, depuis maintenant une dizaine d'années, sur une autre espèce de xénope, phylogénétiquement proche, *Xenopus tropicalis*.

En effet, *Xenopus tropicalis* est diploïde, son génome haploïde mesure 1,7 Gpb répartis sur dix chromosomes. Il est possible d'obtenir des individus de cette espèce sexuellement matures en environ 6 mois, ce qui facilite leur étude.

Xenopus tropicalis est bien plus à même de s'inscrire dans des problématiques de génétique et d'études fonctionnelles à grande échelle grâce à la durée de son cycle de vie et à la structure de son génome. Dans cette optique, le séquençage de son génome a été entrepris par le Joint Genome Institute [72].

De nombreuses études [73, 74] ont comparé les 2 espèces de façon à s'assurer, aussi bien au niveau technique que biologique, de la proximité des deux espèces. Par la suite, nous nous intéressons donc à l'élaboration et à l'étude de modèles associés à *X. tropicalis* bien que l'essentiel des données biologiques provenant de la littérature soit relatif à *X. laevis*.

5.1.2 Métamorphose

La métamorphose des amphibiens est un processus de développement postembryonnaire remarquable. Des modifications morphologiques importantes permettent au têtard de passer d'un milieu de vie aquatique à un milieu terrestre. Ce changement de mode de vie nécessite le développement d'organes tels les membres ou les poumons, qui accompliront des rôles physiologiques adaptés à ce nouveau biotope. Par ailleurs, des organes spécifiques à la vie aquatique, comme les branchies ou la queue, seront éliminés. Ces modifications importantes sont toutes gouvernées par les hormones thyroïdiennes. Ainsi, la métamorphose des amphibiens est un modèle de choix pour comprendre le rôle des hormones thyroïdiennes et leur mode de fonctionnement.

5.1.3 Les hormones thyroïdiennes

5.1.3.1 Régulation

La glande thyroïde est la glande endocrine qui produit les hormones thyroïdiennes, notées HT. On trouve les HT sous deux formes majoritaires qui sont la thyroxine ou tétraiodothyronine (T4) et la triiodothyronine (T3) [54, 55]. La T4 est l'HT circulant majoritairement dans l'organisme, mais la forme la plus active biologiquement est la T3. Dès leur sécrétion dans le flux sanguin, les HT se lient à différentes protéines qui permettent leur transport dans tout l'organisme et régulent leur disponibilité plasmatique et cellulaire.

Comme nous l'avons vu dans les chapitres précédents, les HT peuvent subir, dans le cytoplasme, des modifications modulant leur activité.

- La Déiodinase 3, notée D3, est inactivatrice et son action aboutit à la formation de rT3 à partir de T4 ou de T2 à partir de T3 [61, 60].
- La Déiodinase 2, notée D2, est activatrice et permet la catalyse de la T4 en T3. Elle est aussi capable de transformer le rT3 en T2 [59, 58] (Figure 5.1).

5.1.3.2 Mode d'action des hormones thyroïdiennes

Les hormones thyroïdiennes ne peuvent agir seules. Elles ne sont fonctionnelles que par association avec leur récepteur, noté RHT [57, 75]. Ces récepteurs appartiennent à la superfamille des récepteurs nucléaires qui sont des facteurs de transcription. Les RHT sont constitutivement associés à la chromatine. Il existe deux types de RHT chez *Xenopus laevis* : le TR-α et le TR-β. Leur niveau d'expression varie d'un tissu à l'autre et leur distribution est limitée aux tissus capables de répondre aux HT. Ces récepteurs présentent une forte affinité pour T3 mais lient également T4 dans une moindre mesure.

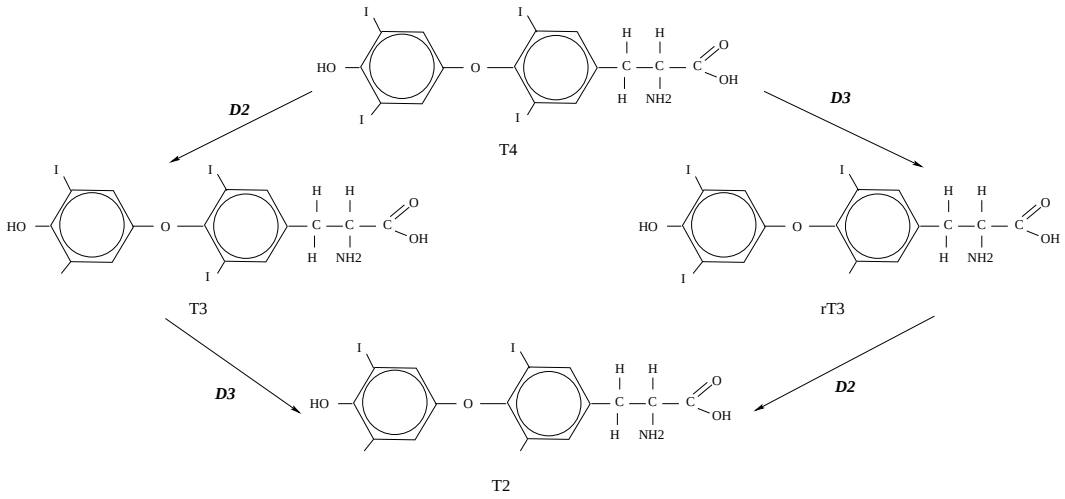

FIG. 5.1 – Les déiodinases, enzymes activatrices et inactivatrices des hormones thyroïdiennes.

5.1.3.3 Fonctions

La compétence du têtard à se métamorphoser dépend de la disponibilité des différents composants intervenant dans la voie de signalisation transcriptionnelle régulée par les HT. C'est la présence concomitante d'hormone active et de ses récepteurs qui permettra le déclenchement et l'accomplissement de la métamorphose. Le développement du xénope est divisé en différents stades de développement numérotés de NF 1 à NF 66 en référence à leurs auteurs P. D. Nieuwkoop et J. Faber [76] (axe des abscisses de la Figure 5.2). La quantification d'HT plasmatiques au cours du développement du xénope montre que T4 et T3 sont respectivement détectables à partir des stades NF 54 et NF 56. Les concentrations plasmatiques de T3 et de T4 augmentent rapidement jusqu'au stade NF 63 pour atteindre un pic qui correspond à l'apogée de la métamorphose. Le pic de T3 apparaît plus tôt et est plus important que celui de T4. Les quantités d'HT diminuent ensuite jusqu'à la fin de la métamorphose (Figure 5.2).

L'expression des différents RHT a également été caractérisée. Il existe une corrélation entre l'expression des RHT et la métamorphose [77], plus précisément celle-ci est liée à TR-β. Or, TR-β est lui-même un gène directement régulé par les HT [78].

Au cours du développement, on observe une expression différente des deux isoformes de RHT [57]. Le transcrit et la protéine de TR-α sont détectables dès les stades les plus précoces de l'embryogenèse [79] correspondant à une contribution d'origine maternelle. L'expression du TR-α zygotique débute ensuite, vers le stade NF 35. Le TR-β est détecté en très faible quantité à partir du stade NF 40, mais sa synthèse ne débute réellement qu'au déclenchement de la métamorphose (stade NF 52), lors de l'apparition des HT plasmatiques (Figure 5.2).

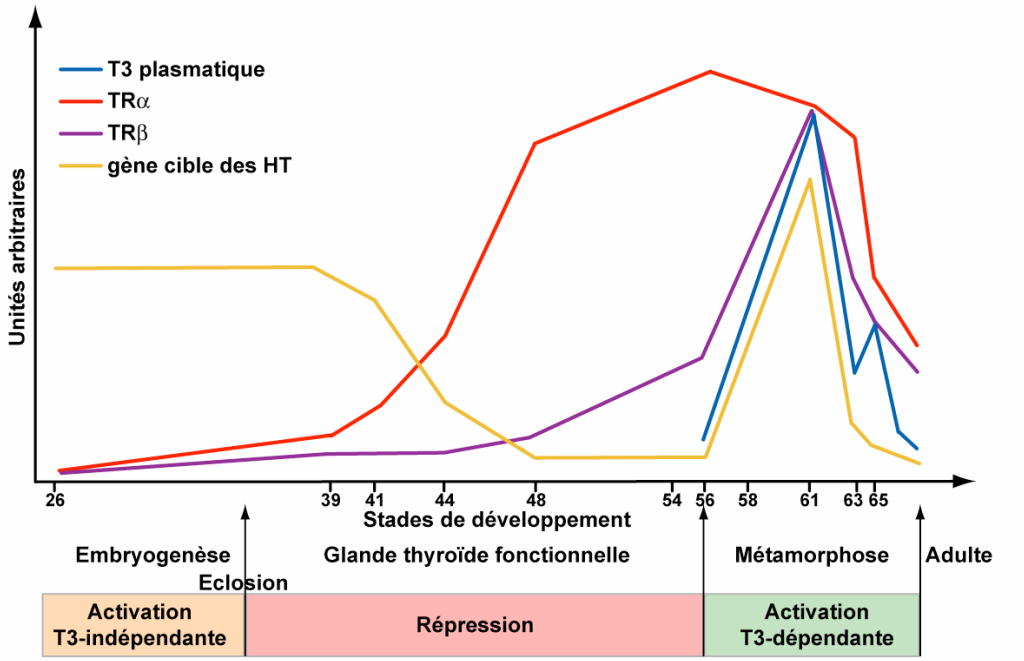

Fig. 5.2 – Expression des acteurs majeurs de la métamorphose au cours du développement.

5.1.3.4 Modifications morphologiques

La métamorphose du xénope correspond à la période délimitée par les stades NF 52 et NF 66. Les différentes étapes de la métamorphose sont identifiables grâce aux modifications morphologiques observées (Figure 5.3). La métamorphose débute après les phases précoces de l'embryogenèse qui vont de la fécondation jusqu'à l'organogenèse (stades NF1 à NF45, environ 4 jours chez le xénope, Figure 5.3-A). La prémétamorphose précède la métamorphose qui peut se décomposer en deux grandes étapes : la prométamorphose et le climax ou apogée de la métamorphose [80]. La prémétamorphose correspond à une phase de croissance (Figure 5.3- B). Elle prend place après l'éclosion, à partir du moment où le têtard commence à se nourrir (stade NF 46). C'est durant cette période que les bourgeons de membres apparaissent et se maintiennent dans un état rudimentaire. Les bourgeons des membres postérieurs sont visibles à l'extrémité de la poche abdominale et les bourgeons antérieurs sont enclos dans les cavités branchiales. La prémétamorphose dure de 3 semaines à un mois suivant les conditions externes (température, alimentation, densité d'individus). La prométamorphose commence au stade NF 54 et correspond au moment où les HT sont détectées dans le plasma (Figure 5.3-C). On remarque, à l'initiation de cette phase, un arrêt de la croissance du têtard. Les modifications morphologiques qui s'opèrent alors correspondent essentiellement à la formation des membres postérieurs et antérieurs qui seront fonctionnels à l'issue de cette étape d'environ une semaine. Le climax métamorphique débute à partir du stade NF 59 (Figure 5.3-D). C'est durant cette troisième phase que les changements morphologiques les plus spectaculaires sont observés, comme la disparition de la queue ou la transformation complète des structures cranio-faciales. L'apogée métamorphique est le moment où la production d'HT est maximale.

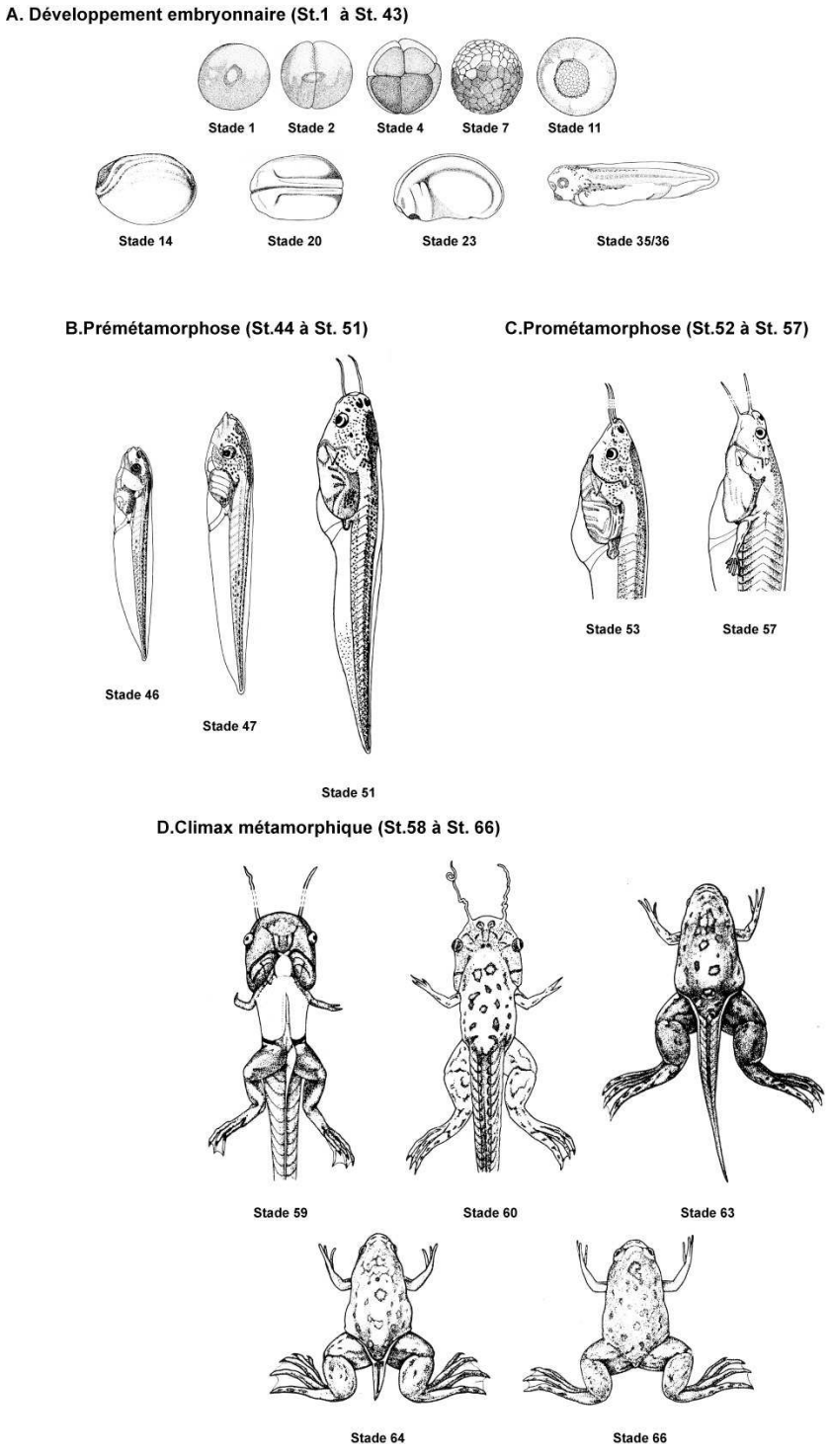

FIG. 5.3 – Les différentes étapes de développement chez le xénope [81].

Au niveau cellulaire, ces effets se matérialisent par des modulations de trois processus biologiques généraux : l'apoptose, la prolifération et la différenciation cellulaire. Ainsi, les tissus spécifiquement larvaires (queue, branchies) disparaissent complètement et les

organes spécifiquement adultes apparaissent (membres, poumons). D'autres, qui existent au sein des deux formes de vie, sont modifiés pour répondre aux nouvelles contraintes environnementales, éthologiques et physiologiques (peau, foie, système nerveux).

5.1.3.5 Programme de la métamorphose

L'étude des gènes impliqués dans la métamorphose est un sujet d'investigation en constante évolution depuis que la capacité des HT à induire la synthèse d'ARN et de protéines a été mise en évidence [82]. Les méthodes de suivi de l'expression des gènes ont permis d'acquérir une vision de plus en plus globale de l'évolution des transcrits régulés par les HT et associés aux modifications morphologiques.

Les gènes dont l'expression est modulée à la métamorphose ont été répartis en deux catégories : les gènes précoces et les gènes tardifs. Cette distinction sépare les transcrits dont la régulation est visible dans les 24 heures suivant un traitement exogène à la T3 de ceux dont la régulation n'est visible qu'après. Une hypothèse décrite dans [83] considère que les gènes précoces sont essentiellement des déiodinases et des facteurs de transcription. Ces facteurs de transcription nouvellement synthétisés, associés aux HT, permettraient l'induction des gènes tardifs. Les produits de ces gènes tardifs engendreraient alors les modifications spécifiques à chaque tissu. Il s'agit par exemple de protéases pour la dégradation de la queue ou de protéines associées au cycle cellulaire pour la prolifération des pattes.

Parmi les nombreux gènes intervenants dans la métamorphose, nous nous sommes concentrés, dans un premier temps, sur le gène codant le récepteur aux HT, et sur les gènes métabolisant les HT, c'est-à-dire les deux déiodinases : D2 et D3. RHT, D2 et D3 sont des gènes précoces induits par les hormones thyroïdiennes. Il existe donc un rétrocontrôle positif sur RHT et sur D2 alors que le rétrocontrôle est négatif sur D3.

5.1.4 Problématique

Les techniques de génomique fonctionnelle permettent l'observation de processus biologiques à une échelle globale du génome difficilement atteignable par l'utilisation d'autres méthodes. Toutefois, ces approches globales présentent le défaut d'aboutir à une production de données tellement gigantesque que leur exploitation exhaustive est rare. Ainsi, les résultats s'accumulent sans que la volonté de les intégrer à une échelle physiologique ou cellulaire ne soit clairement affichée, conférant de manière paradoxale un caractère réductionniste aux études de génomique fonctionnelle.

Afin d'intégrer les résultats obtenus des données expérimentales à une échelle cellulaire, nous avons modélisé les interactions génétiques nécessaires à certaines modifications morphologiques liées à la métamorphose. Notre étude *in silico* se divise en trois parties :

1. La première partie consiste à étudier exclusivement la compétition qui existe entre les deux enzymes D2 et D3. Cette partie correspond à l'exemple biologique détaillé tout au long de ce rapport. Nous récapitulerons donc les principaux résultats et nous discuterons à la fois ces résultats et les raisons qui nous ont poussés à étudier spécifiquement cette compétition.

2. Dans une deuxième partie, nous avons étudié en parallèle deux modèles : le modèle entraînant l'apoptose dans la queue et le modèle déclenchant la prolifération cellulaire dans les pattes arrières. Nous avons alors identifié des propriétés associées à la dynamique temporelle des hormones thyroïdennes dans chacun de ces tissus.
3. Les deux modèles (pattes/queue) se concentrent sur les mécanismes déclenchant soit l'apoptose soit la prolifération. Ainsi, l'apoptose et la prolifération sont modélisés sous forme de "boîtes noires", c'est-à-dire sous la forme d'un ensemble abstrait de gènes. Dans la dernière partie, nous avons étudié plus en détail ces boîtes noires, grâce à une analyse du transcriptome. On aboutit alors à des modèles très simples qui entraînent néanmoins des discussions biologiques riches et complexes.

5.2 Compétition enzymatique

Nicolas Pollet du Laboratoire de Développement et Evolution avec qui nous collaborons s'intéresse essentiellement aux mécanismes transcriptionnels responsables de la résorption de la queue. Nous nous sommes donc intéressés, dans un premier temps, à l'élaboration *in silico* d'un réseau de Petri hybride temporisé qui modélise l'induction des gènes entraînant l'apoptose dans la queue. Ce modèle intègre les entités biologiques suivantes : les deux déiodinases D2 et D3, le récepteur aux HT, RHT, l'hormone thyroïdienne T3, et deux ensembles abstraits de gènes GP et GA traduisant respectivement les gènes précoces et les gènes tardifs apoptotiques. La recherche des paramètres du modèle (vitesses de réaction, concentrations,...) se fait par une étude bibliographique et par discussion avec les biologistes. Dès les premières étapes de construction de notre modèle, notre attention s'est tournée vers les déiodinases. En effet, de prime abord, il semblait clair que D2 permettait l'obtention d'une très forte concentration intra-cellulaire de T3 indispensable au déclenchement de l'apoptose. Or, les données cinétiques fournies par Germain *et al.* [84] indiquent que l'enzyme D3 est plus présente que D2 et que son activité enzymatique est 10 fois plus forte que celle de D2. Ainsi, avant de poursuivre la construction du modèle complet dans les cellules de la queue, nous nous sommes concentrés sur le modèle mettant en compétition les deux enzymes D2 et D3.

5.2.1 Rappels

Le THPN de la Figure 5.4 modélise la compétition existant entre les deux enzymes D2 et D3. Les quatre entités D2, D3, T3 et T4 sont modélisées par des places continues (modélisation d'une concentration réelle). Les transitions t_1 et t_2 modélisent l'induction des enzymes par l'HT T3. On rappelle que la transition t_1 est discrète car l'induction de D2 par T3 se fait de façon indirecte, on observe donc un délai entre l'induction par T3 et la synthèse de D2. Enfin, les transitions t_4 et t_3 modélisent respectivement les réactions enzymatiques de D2 et de D3 : $D2 + T4 \rightarrow D2 + T3$, $D3 + T3 \rightarrow D3 + T2$ et $D3 + T4 \rightarrow D3 + rT3$ où $T2$ et $rT3$ sont inactifs, donc pas représentés dans le modèle.

La question posée à ce modèle peut se formuler comme suit : "L'action de l'enzyme D3 prédomine-t-elle toujours celle de D2 ? ". La formulation logique de cette question s'écrit : $f \equiv \Box \diamond (m(T3) = 0)$, où $\Box$ signifie toujours et $\diamond$ signifie à un moment donné. Par

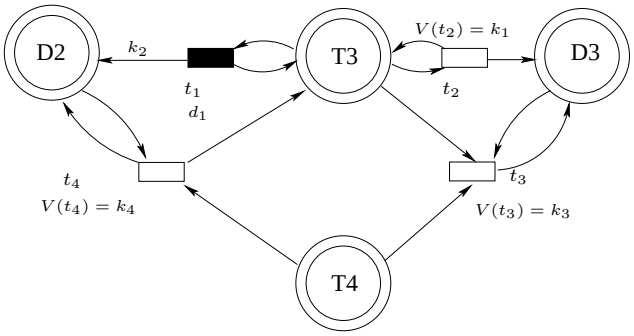

FIG. 5.4 – Modèle THPN modélisant la compétition enzymatique D2/D3.

conséquent, f peut se formuler de la façon suivante : la concentration de T3 finit toujours par atteindre 0, c'est-à-dire l'action de D3 surpasse toujours celle de D2.

Le logiciel *BioPetri* permet de vérifier pour chaque résolution de conflit si le modèle THPN de la Figure 5.4 satisfait ou non la propriété décrite ci-dessus. Dans tous les cas de résolution de conflit, on conclut que le modèle satisfait la propriété, ce qui traduit la continuelle prédominance de D3 sur D2. On insiste sur le fait que ce résultat dépend des valeurs des vitesses mais que celles-ci sont connues dans la littérature [84].

5.2.2 Discussion

De ce modèle très simple, ont découlé de nombreuses questions. Nous avions supposé, dans un premier temps, que toutes les cellules d'un tissu, ici la queue, exprimaient les déiodinases. Les résultats *in silico* nous ont poussés à remettre cette hypothèse en question. De même, étant donnée la faible action de D2, nous nous sommes demandés si certaines cellules exprimaient D2 et d'autres D3. L'action de D2 ne serait donc plus inhibée par celle de D3. Dans le cas où D2 et D3 sont exprimées dans les mêmes cellules, quel est exactement le rôle de D2 ?....

Une étude bibliographique nous a permis de répondre à certaines de ces questions. Cai et Brown [58] ainsi que Berry *et al.* [85] ont analysé le profil d'expression de D2 et de D3 dans la queue, par hybridation *in situ*. Leurs résultats montrent que D2 et D3 sont exprimées très localement, dans les mêmes cellules de la queue. En effet, seules certaines cellules expriment un forte quantité de déiodinases. Elles sont localisées au voisinage des vaisseaux sanguins. Ces cellules recevant un fort apport de HT par le sang pourraient être capables de réguler l'activité thyroïdienne ainsi que la quantité de T3 distribuée aux autres types cellulaires environnants. Deux mécanismes différents pourraient donc coexister au sein d'un même organe, selon la présence ou non de déiodinases dans la cellule. Nakajima *et al.* [86] proposent deux mécanismes de mort dans la queue suivant l'expression ou non de D2 dans les cellules :

- La T3 transformée au sein des cellules exprimant D2 serait transmise aux cellules environnantes, permettant ainsi le déclenchement du programme métamorphique.
- Les cellules environnantes disparaîtraient consécutivement à la mort cellulaire programmée (apoptose) des cellules directement stimulées par T3.

Ces hypothèses accordent à D2 un rôle majeur. Or, notre modèle *in silico* semble contredire cette hypothèse. Afin de répondre à ces nouvelles questions d'une manière expérimentale, un suivi de la quantité de T3 au niveau cellulaire pourrait être réalisé par des expériences d'immuno-marquage des HT corroborées à la détection des déiodinases. Ceci permettrait une visualisation simultanée des HT et des déiodinases dans les différents tissus de la queue. Une approche de transgénèse utilisant le promoteur d'un gène de réponse directe aux HT contrôlant l'expression d'un gène rapporteur pourrait également être envisagée. Cette méthode permettrait de localiser des produits de gènes induits par les HT dans la queue.

En résumé, ce premier modèle *in silico* nous oriente sur le rôle des déiodinases. Ce modèle est, bien entendu, trop restreint pour conclure de façon précise sur l'effet de ces enzymes dans un tissu. Il nous permet juste de mettre un bémol sur l'hypothèse, très répandue dans la littérature, du rôle majeur de D2, lorsque celle-ci est en compétition avec D3. Nous allons à présent, étudier l'effet de chaque enzyme au sein du modèle complet qui déclenche l'apoptose dans les cellules de la queue. Néanmoins, l'étude des déiodinases au sein d'un unique tissu ne permet pas d'être exhaustif vis-à-vis de leur rôle respectif. Ainsi, nous construisons en parallèle le modèle déclenchant la prolifération cellulaire dans les pattes arrières. Comme nous le verrons dans la section suivante, ces deux modèles sont très proches, ce qui nous permet une étude comparative.

5.3 Modèles apoptose *versus* prolifération

La croissance des pattes arrières est l'une des premières modifications morphologiques induites par les hormones thyroïdiennes. Chez *Xenopus tropicalis*, les bourgeons des pattes arrières se forment mais ne peuvent se développer sans HT à partir du stade NF 53. La croissance des pattes induite par les hormones thyroïdiennes a lieu lorsque la concentration en HT est faible dans le plasma (Figure 5.2). Le développement des pattes est terminé avant le climax de la métamorphose.

A l'opposé, la queue est le dernier organe a être modifié. La régulation HT-dépendante à l'origine de la résorption de la queue nécessite une très forte concentration intra-cellulaire d'hormones thyroïdiennes [83]. Cette régulation est déclenchée au climax de la métamorphose quand la concentration en HT est maximale (Figure 5.2).

Bien que ces deux processus biologiques soient totalement opposés, le réseau de régulation responsable de l'activation de la prolifération cellulaire dans les pattes et celui responsable de l'activation de l'apoptose dans la queue sont relativement similaires. Ils impliquent les mêmes enzymes et cofacteurs.

Ainsi, nous allons tout d'abord présenter les modèles THPN associés aux deux tissus. Puis, nous testerons sur ces modèles trois propriétés biologiques connues. Enfin, nous effectuerons une séries d'expériences et de mutations *in silico*.

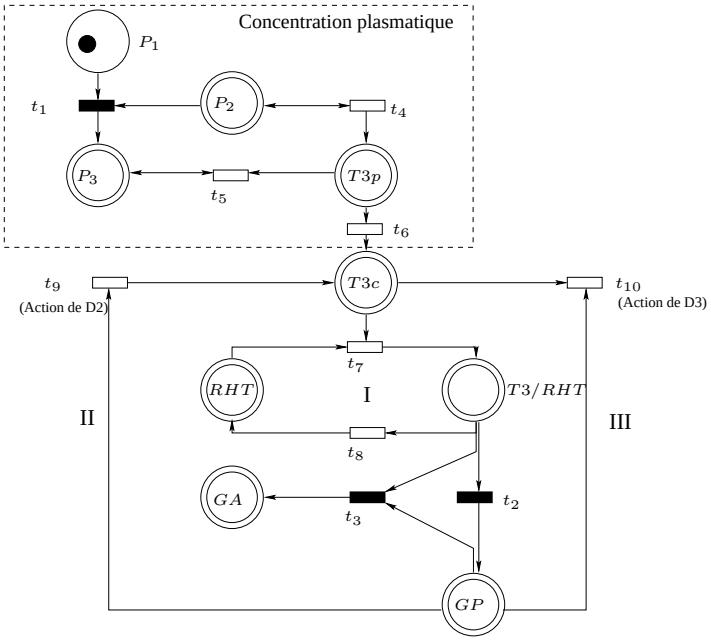

FIG. 5.5 – Modèle THPN du déclenchement de l'apoptose dans la queue du xénope. I : boucle de rétroaction positive sur RHT, II : boucle de rétroaction positive sur T3 et III : boucle de rétroaction négative sur T3.

5.3.1 Construction des modèles

5.3.1.1 Modèle dans les cellules de queue

Etant donné que notre but est d'étudier les causalités menant à la perte de la queue à partir de la concentration plasmatique en HT, ces concentrations plasmatiques doivent être considérées comme résultat de régulation en amont du processus étudié.

Le modèle présenté en Figure 5.5 impose des cinétiques de la concentration de T3 plasmatique (T3p) en accord avec celle donnée sur la Figure 5.2. Ces cinétiques sont considérées comme données d'entrée de notre modèle. Le modèle intègre aussi les régulations associées à la T3 intra-cellulaire, à D2, D3, RHT, aux gènes précoces et aux gènes tardifs apoptotiques en général. La construction de notre modèle s'est effectué par différents sous-réseaux. Nous commençons par le T3 plasmatique.

- **Modélisation de la concentration en T3 plasmatique.** La première étape dans la construction de notre modèle consiste à modéliser une concentration de T3 plasmatique (T3p) au cours du temps similaire à celle donnée dans la Figure 5.2. Cette courbe est utilisée comme base de notre modèle. Ce sous-réseau est modélisé par trois places continues (P_2, P_3 et $T3p$), par une place discrète (P_1), par deux transitions continues (t_4 et t_5) et par une transition discrète (t_1). Initialement, la transition continue t_4 est tirée ce qui remplit la place $T3p$. Une fois le délai de la transition discrète t_1 écoulé, les jetons de P_1 et de P_2 sont consommés. La transition t_4 ne peut alors plus tirer et la place $T3p$ n'est plus remplie. Le climax de la métamorphose est alors atteint, la concentration en $T3p$ décroît de manière continue à cause du tir de

la transition continue t_5.

- **Modélisation du rôle de RHT.** Une part des T3 plasmatiques (place $T3p$) entre dans la cellule (transition t_6) pour former les T3 intra-cellulaires (place $T3c$). Initialement, les cellules de la queue expriment basalement du RHT, il s'agit de la forme α du récepteur. Les T3 intra-cellulaires se lient à leur récepteur RHT pour former le complexe T3/RHT (place $T3/RHT$ et transition continue t_7). RHT est un gène de réponse directe aux T3. Par conséquent, le complexe T3/RHT induit l'expression de RHT (transition continue t_8).

- **Modélisation des gènes précoces.** Lorsque la concentration du complexe T3/RHT est suffisante, les gènes précoces, modélisés par la place continue GP, sont transcrits. La notion de seuil est modélisé par la transition discrète t_2. En effet, on rappelle que les transitions continues n'admettent pas de seuil. Dans notre modèle, nous ne faisons pas apparaître les déiodinases sous forme d'entités biologiques, par contre, nous modélisons leur processus biologiques grâce aux transitions continues t_9 et t_{10}. La production de T3 intra-cellulaire par l'action de D2 est modélisée par la transition t_9 et celle de D3 consommant T3 est modélisée par la transition t_{10}.

- **Modélisation des gènes tardifs apoptotiques.** Les gènes apoptotiques, modélisés par la place continue GA ne peuvent être induits qu'après un délai durant lequel la concentration en gènes précoces et en complexes T3/RHT restent suffisantes. Ce délai et ces seuils sont modélisés par la transition discrète t_3. Notre but n'est pas ici d'étudier l'évolution de la concentration des gènes apoptotiques. Cette place est utilisée comme un interrupteur. En effet, on désire savoir s'il y a apoptose ou non. L'ensemble des gènes entraînant l'apoptose sont abstraits dans la place continue GA. La dernière partie de ce chapitre s'intéresse plus particulièrement à cet ensemble de gènes.

Paramètres

Les vitesses de tir associées aux activités enzymatiques (transitions t_9 et t_{10}) sont obtenues des données cinétiques (Km et Vmax) de Germain *et al.* dans [84] (Tableau 5.1). La concentration initiale de RHT est donnée par Wong et Shi dans [87]. Initialement, nous considérons qu'il n'y a pas de T3 intra-cellulaire. Ranjan *et al.* [57] donnent les paramètres cinétiques associés à la vitesse de synthèse de RHT. Enfin, la constante de liaison maximale (MBC) de T3 et de RHT (paramètre de la transition t_7) est donnée par Pellizas *et al.* dans [75].

Remarque 5.1. *Trois boucles de rétroaction apparaissent dans le modèle.*

- *Tout d'abord, RHT a un rétrocontrôle positif sur lui-même (I sur la Figure 5.5), c'est-à-dire qu'il induit sa propre synthèse.*
- *On retrouve les boucles de rétroaction opposée sur T3 : l'action de D2 induit un rétrocontrôle positif (II) alors que l'action de D3 induit un rétrocontrôle négatif (III).*

Paramètres biologiques	Valeurs	Références
Activité de D2	66.3 $pmol/min.mg$	[84]
Activité de D3	727.0 $pmol/min.mg$	[84]
Concentration initiale de T3c	0 nM	Choix de modélisation
Concentration initiale de RHT dans la queue	1 $U.A$	[87]
Concentration initiale de RHT dans les pattes	4 $U.A$	[83]
Constante d'association T3/RHT	$\approx 2.50\ M^{-1}$	[75]
Vitesse de synthèse de RHT	0.75 $U.A$ par Stade de développement	[57]

TAB. 5.1 – Tableau récapitulatif des paramètres biologiques et de leur valeur.

5.3.1.2 Modèle dans les cellules de pattes

Deux différences majeures existent entre le réseau de régulation dans les cellules de queue et celui des cellules de pattes. Tout d'abord, la déiodinase activatrice D2 n'est pas un gène de réponse aux T3. D2 est ubiquitaire dans les cellules de pattes [58, 59, 61]. De plus, la concentration initiale du récepteur de T3, RHT, est beaucoup plus importante que celle donnée initialement dans la queue [83].

Le modèle THPN de la régulation HT-dépendante dans les pattes est donc le même que celui de la queue (Figure 5.5) excepté que l'arc allant de la place des gènes précoces GP à la transition t_9 traduisant l'action de D2, est retiré. Ceci modélise la nature ubiquitaire de D2. On rappelle que D2 n'est pas explicitement représenté dans la Figure 5.5, seule son action est modélisée par la transition t_9. Les autres modifications se font au niveau des paramètres. La concentration initiale de la place RHT est modifiée, celle-ci a été déterminée d'après une étude comparative de la concentration en RHT dans différents organes du têtard [83].

5.3.2 Propriétés

Un modèle n'est utile que s'il est préalablement capable de reproduire les comportements connus. Si le modèle satisfait bien ces comportements (décrits sous forme de propriétés), celui-ci continuera à être exploité en effectuant d'autres expériences *in silico*. Dans le cas contraire, le modèle est retravaillé en collaboration avec les biologistes, dans le but de faire coïncider les comportements *in silico* et *in vitro*.

L'identification de propriétés biologiques est une tâche ardue. Tout d'abord, il faut sélectionner un (ou un ensemble de) comportement(s) pertinent(s), parmi ceux répertoriés dans la littérature. Puis, il s'agit de "traduire" correctement ce ou ces comportements en propriétés logiques.

Nous avons recherché des propriétés biologiques vraies dans les deux tissus (pattes

et queue). Nous avons isolé trois comportements différents du système correspondant à trois phases dépendantes du temps et de la concentration en HT. Etant donné que seule la concentration plasmatique en HT peut être expérimentalement mesurée, seule cette concentration est utilisée dans les propriétés biologiques.

- **Première phase :** La concentration plasmatique en hormones thyroïdiennes n'est pas suffisante pour déclencher les processus HT-dépendants (apoptose dans la queue et prolifération cellulaire dans les pattes).
- **Deuxième phase :** Les gènes précoces sont transcrits dès que la concentration plasmatique atteint un seuil s_1.
- **Troisème phase :** Dès que le seuil s_1 a été atteint, il faut attendre un délai d_2 avant d'observer l'induction des gènes tardifs (gènes apoptotiques ou gènes du cycle cellulaire).

Après avoir identifié les propriétés comportementales de notre réseau de régulation, nous devons les traduire en logique CTEL.

5.3.2.1 Expression formelle des phases 1 et 2

Les phases 1 et 2 sont fortement corrélées. En effet, la première phase peut être exprimée comme suit : "tant que la concentration en T3 plasmatique est sous le seuil s_1, les gènes précoces ne sont pas transcrits". La seconde phase correspond, d'une certaine façon, à l'opposé. Elle peut s'exprimer comme suit : "les gènes précoces sont transcrits (noté $m(GP) > 0$) depuis que la concentration en T3 plasmatique a atteint le seuil s_1 (noté $(m(T3p) \geq s_1)$)". Par conséquent, l'expression de la phase 2 inclut la première phase. La phase 2 écrite en CTEL donne :

$$(\Box m(GP) > 0) S (m(T3p) \geq s_1)$$

On rappelle que $\Box$ signifie toujours et que $\Box\phi$ est une abréviation de $\neg(\top U \neg\phi)$. Enfin, S est l'opérateur *Since*, il s'agit de l'opérateur miroir du *Until*.

5.3.2.2 Expression formelle de la phase 3

La phase 3 inclut des notions de temps-réel : à la première occurrence de l'événement $(m(T3p) \geq s_1)$, un délai minimum d_2 doit être attendu avant l'induction des gènes tardifs, notée $(m(GT) > 0)$. La première occurrence de $(m(T3p) \geq s_1)$ peut être formellement exprimée comme suit :

$$(m(T3p) \geq s_1) \wedge \boxminus (m(T3p) < s_1)$$

où $\boxminus$ est l'opérateur miroir de $\Box$. Cette expression signifie que la concentration en T3 plasmatique $(m(T3p))$ atteint le seuil s_1 et que dans le passé, $m(T3p)$ était inférieur à ce seuil.

Si la formule précédente est vraie, alors les gènes tardifs seront transcrits dans un délai minimum d_2, ce qui s'écrit : $\rhd_{\geq d_2}(m(GT) > 0)$.

La propriété biologique complète associée à la phase 3 s'écrit donc :

$$((m(T3p) \geq s_1) \wedge \boxminus(m(T3p) < s_1)) \Rightarrow \rhd_{\geq d_2}(m(GT) > 0)$$

5.3.2.3 Estimation du seuil s_1 et du délai d_2 dans la queue et dans les pattes

Les paramètres s_1 et d_2 sont extraits d'études cinétiques dans la queue [83] et dans les pattes [60]. Nous obtenons un délai d_2 équivalent dans la queue et dans les pattes : $d_2(queue) = d_2(pattes) = 48\ heures$, un seuil $s_1 = 75\ UA$ (UA :Unités Arbitraires) dans la queue et un seuil $s_1 < 10\ UA$ dans les pattes.

5.3.2.4 Résultats

Aussi bien dans les pattes que dans la queue, les deux modèles satisfont les deux propriétés biologiques décrites précédemment. Ainsi, nos modèles sont en accord avec le découpage en trois phases de régulation. Deux réseaux de régulations quasiment similaires sont donc responsables de deux modifications morphologiques opposées (apoptose et prolifération cellulaire). Ces deux processus sont déclenchés à des concentrations très différentes de T3 plasmatique (concentration correspondant au stade NF 56 pour les pattes et concentration correspondant au stade NF 61 pour la queue, Figure 5.2).

Cette étude de propriétés biologiques connues ne nous a pas permis de réfuter nos modèles. Ceux-ci peuvent, à présent, être utilisés pour effectuer diverses expériences *in silico*. Nous avons effectué les deux types d'étude suivants, présentés respectivement dans les sections 5.3.3 et 5.3.4 :

- L'étude du rôle des déiodinases est poursuivie dans chaque tissu. Nous effectuons, pour cette étude, des mutations *in silico* engendrant l'inhibition ou la sur-expression de chacune des deux enzymes D2 et D3.

- La nature précise des hormones thyroïdiennes reste incertaine. En effet, les HT sont très souvent associées à des morphogènes mais aucune expérience n'a permis de valider ou réfuter cette hypothèse. Or, d'autres substances, comme l'acide rétinoïque [88], a longtemps été désignée comme morphogène de façon erronée. Les hypothèses biologiques concernant la nature des HT sont détaillées ci-dessous.

5.3.3 Etude *in silico* du rôle des déiodinases

Dans cette section, nous étudions le rôle *in silico* des deux déiodinases D2 et D3 dans les deux tissus : pattes et queue. Deux types d'expériences peuvent être effectués : l'inactivation de l'enzyme et la sur-expression ubiquitaire de l'enzyme.

5.3.3.1 Effet de mutations de D2

La première partie de ce chapitre suggérait un rôle mineur de D2 par rapport à celui de D3. Suite à ces résultats, on se demande quelle serait l'évolution des deux réseaux sans l'activité de D2. L'inhibition de l'enzyme D2 est modélisée en retirant la transition

continue t_9 du modèle THPN de la Figure 5.5.

Dans les pattes, l'effet de D2 est indégnable. L'inactivation de D2 dans ce type cellulaire empêche les phases 2 et 3 d'avoir lieu. L'enzyme D2 semble donc compenser le faible apport plasmatique en hormones thyroïdiennes. L'action de l'enzyme permet l'obtention d'une concentration intra-cellulaire de T3 suffisamment importante pour déclencher le processus T3-dépendant de prolifération cellulaire.

A l'inverse, l'inactivation de l'enzyme D2 dans les cellules de la queue n'engendre pas de modifications *in silico* dans le déroulement de la métamorphose. En effet, malgré l'absence de D2, les trois phases détaillées ci-dessus ont encore lieu. Ainsi, malgré l'absence d'activateur des hormones thyroïdiennes, les gènes précoces et les gènes tardifs sont transcrits. *In silico*, l'apoptose a donc lieu avec ou sans D2.

5.3.3.2 Effet de mutations de D3

L'action de l'enzyme D3 peut être inhibée dans les deux modèles en retirant la transition continue t_{10}.

Etant donné le rôle majeur de D2 dans les cellules de pattes, l'inhibition de D3 n'a pas de conséquence sur la prolifération cellulaire dans ce tissu. A l'inverse, l'inhibition *in silico* de cette enzyme dans les cellules de queue a pour conséquence l'induction des gènes tardifs apoptotiques avant le délai d_2 de la troisième phase. Ainsi, sans l'enzyme D3, la queue du têtard disparaît précocement.

On peut se demander si une sur-expression de D3 dans la queue aurait des conséquences inverses ? La sur-expression de D3 est modélisée en augmentant la vitesse d'action de D3 dans la transition continue t_{10}. De plus, sa synthèse est rendue ubiquitaire en retirant l'arc sortant de la place des gènes précoces GP et entrant dans cette transition. Dans ces conditions *in silico*, les phases 2 et 3 ne sont jamais atteintes, ce qui signifie que l'apoptose n'a jamais lieu dans les cellules de queue. Huang *et al.* [60] ont exécuté cette expérience *in vivo*. De telles expériences sont très difficiles à mettre en œuvre expérimentalement. Ainsi, la grande majorité des animaux sur-exprimant D3 n'ont pas survécu. Seul un animal a atteint l'âge adulte et celui-ci a gardé sa queue de têtard tout au long de son développement. Nos résultats *in silico* sont donc en accord avec ceux obtenus *in vivo*.

5.3.3.3 Discussion

Les déiodinases semblent jouer un rôle opposé dans les deux tissus. En effet, le rôle de D2 est primordial pour l'obtention des pattes arrières. A l'inverse la queue disparaît *in silico* avec ou sans l'activateur de HT. Ainsi, dans la queue l'apport plasmatique en HT semble suffisant pour déclencher *in silico* la métamorphose. La pousse des pattes arrière ayant lieu bien avant le climax de la métamorphose, l'apport en HT ne permet pas le déclenchement de la prolifération cellulaire.

De même, le rôle de D3 est différent selon le tissu étudié. Les mutations de D3 n'ont pas d'effet dans les pattes arrières. A l'inverse, l'inhibition de D3 engendre une perte précoce de la queue et la sur-expression de D3 empêche l'apoptose.

L'apport plasmatique en T3 est très important dans la queue. Ainsi, le rôle de D2 devient minoritaire. Par contre, le rôle de D3 semble primordial, l'enzyme permet de retarder la disparition de la queue. Ceci permettrait peut-être de s'assurer de la pousse complète des pattes arrières avant la disparition de la queue.

Parmi les expériences proposées *in silico*, seule la sur-expression de D3 a été réalisée *in vivo*. Ce sont des expériences longues et difficiles à mettre en œuvre (un seul animal survivant). De façon à étudier plus facilement ces nouveaux résultats et nouvelles hypothèses, Nicolas Pollet du laboratoire de Développement et Evolution a mis en place un modèle de culture cellulaire *in vitro*. Ce modèle correspond à un intermédiaire entre le modèle *in silico* et le modèle complet *in vivo*. Une expérience, qui sera réalisée dans l'avenir, consiste à obtenir une culture de cellules de queue dépourvues de l'enzyme D2 et d'étudier leur évolution à différents temps de la métamorphose. Le modèle *in vitro* est un modèle indispensable pour pouvoir exploiter les résultats de nos modèles *in silico*.

5.3.4 Etude *in silico* de la nature des hormones thyroïdiennes

5.3.4.1 Hypothèses biologiques

Nous nous posons la question de la nature des hormones thyroïdiennes. Dans la littérature, les HT sont usuellement comparées à des morphogènes. Un morphogène est une molécule sécrétée qui a la propriété d'induire des types cellulaires différents à différentes concentrations. Ainsi, à une concentration spécifique d'HT correspondrait un processus biologique spécifique. Dans notre cas d'étude, cette hypothèse reviendrait à penser que c'est la faible concentration en HT qui permet la croissance des pattes arrières et que c'est la forte concentration en HT qui engendre la disparition de la queue.

Nous opposons à cette hypothèse celle de l'interrupteur moléculaire. Sous cette hypothèse, le processus biologique (prolifération ou apoptose) ne serait pas une conséquence directe de la concentration en HT. Dans chaque organe, il suffirait que la concentration en HT atteigne un seuil pour que le processus biologique soit déclenché. La notion d'interrupteur moléculaire est fortement corrélée à celle de switch épigénétique. René Thomas [53] définit l'épigénèse comme une modification transitoire de l'environnement responsable d'une multi-stationnarité. Dans notre cas, une épigénèse sur T3 pourrait exister dans la queue. Celle-ci mènerait aux deux états stables suivants :

- Si la concentration intra-cellulaire de T3 est faible, elle reste faible,
- Si la concentration intra-cellulaire de T3 est suffisante pour s'induire elle-même, alors elle reste élevée, ce qui permet d'induire les différents processus biologiques spécifiques de chaque tissu.

5.3.4.2 Tests *in silico*

La deuxième hypothèse peut être testée *in silico* grâce à un pulse de T3. Un pulse de T3 consiste à ajouter une quantité saturante de T3 pendant un délai variant d'une à plusieurs heures, puis à retirer le T3 ajouté (étape de lavage du pulse).

Pour cette expérience *in silico*, on ne considère plus l'apport plasmatique. On se place dans un contexte de culture cellulaire de queue. Si T3 est au centre d'un switch épigénétique, l'ajout d'une concentration saturante de T3 est suffisante pour amorcer le processus biologique (apoptose), même après le lavage.

Différents temps de pulse court ont été testés (de 1 heure à 5 heures). Dans tous les cas, les phases 2 et 3 ne sont jamais observées *in silico*.

Lorsque des périodes beaucoup plus longues sont testées (plusieurs jours), on n'observe pas de différences avec la métamorphose "naturelle".

5.3.4.3 Tests *in vivo*

Suite à ces résultats, le Laboratoire de Développement et Evolution a commencé à mettre en place des méthodes de traitements transitoires aux HT. A l'heure actuelle, une seule expérience a été effectuée. Des têtards sont traités durant 24 heures par 10nM de T3, puis laissés 24 heures sans traitement. Les transcriptomes induits par ce traitement sont ensuite analysés sur puces à ADN. Il a été observé que les têtards ayant subi un tel traitement présentent, au bout de 48 heures, les mêmes caractéristiques morphologiques que les têtards de stades de développement équivalents traités avec la même dose d'hormone pendant 48 heures d'affiler.

Les résultats *in silico* et *in vivo* concordent pour des temps très longs de pulses. Les pulses de durée plus courte sont actuellement en cours. On peut néanmoins commencer à discuter des résultats d'après les expériences proposées par Wang et Brown dans [83]. Ceux-ci concluent qu'entre 24 et 48 heures après l'ajout de HT, le programme génétique responsable de le perte de la queue est activé. Avant ce délai, un traitement continu au HT est nécessaire pour induire la résorption de la queue.

Ces résultats, en accord avec notre modèle, vont dans le sens de l'hypothèse de morphogènes devant être continuellement présents pour agir.

5.3.5 Vers un Xénope partiel *in silico*

L'obtention d'un modèle *in silico* permet d'effectuer très facilement de nombreuses expériences et tests. Ces expériences *in silico* ont abouti à de nouvelles questions ou hypothèses biologiques qui doivent être testées expérimentalement pour pouvoir être validées ou réfutées. Le travail de modélisation présenté dans cette section, est le résultat d'une collaboration très proche avec le Laboratoire de Développement et Evolution, et en particulier Nicolas Pollet et Raphaël Thuret. En effet, ceux-ci ont mis en place des cultures cellulaires de façon à faciliter les échanges entre les modèles *in silico* et *in vitro*.

Dans ce travail de modélisation, nous nous sommes concentrés sur 2 organes : la queue qui est le centre d'étude du Laboratoire de biologie et les pattes dont le processus biologique est opposé à celui ayant lieu dans la queue. L'objectif associé à la modélisation de ces 2 modifications morphologiques est de comprendre comment une même hormone

(T3) peut contrôler des changements si différents. Ce travail permet de regrouper au sein d'un même modèle des données provenant très souvent d'expériences indépendantes.

Une idée consisterait donc à poursuivre cette méthodologie dans le but d'ajouter d'autres organes tels que les pattes avant, le cerveau, la peau ou encore l'intestin dont le développement lors de la métamorphose correspond à un troisième processus biologique appelé la différenciation. Un tel travail permettrait d'obtenir un génome de xénope partiel en activité *in silico*, permettant l'étude d'une métamorphose complète. Son étude permettrait donc de mieux comprendre les enchaînements ayant lieu lors de la métamorphose.

Le modèle présenté dans la Figure 5.5 comporte deux boîtes noires : la boîte noire associée aux gènes précoces et celle associée aux gènes tardifs. L'objectif de notre modèle étant de vérifier si l'induction des gènes tardifs est déclenchée ou non par les hormones thyroïdiennes, le détail de ces boîtes noires ne s'est pas avéré utile. Or, l'étude détaillée de l'ensemble des gènes tardifs entraînant l'apoptose peut être, bien sûr, très informative sur le programme génétique associé à l'apoptose et/ou à la prolifération. Dans la dernière section de ce chapitre, nous nous sommes donc concentrés sur leur étude. Etant donné que le Laboratoire de Développement et Evolution travaille sur l'étude transcriptionnelle à grande échelle chez le têtard *Xenopus tropicalis*, nous débutons ce travail par l'étude des résultats de leurs puces à ADN.

5.4 Confrontation de modèles à des données transcriptionnelles

5.4.1 Contexte biologique

Raphaël Thuret a étudié durant sa thèse [81] la cinétique de 2902 transcrits intervenant dans différents processus biologiques lors de la métamorphose du xénope. Cette analyse de la cinétique d'expression de gènes a été réalisée par des expériences de puces à ADN sur trois tissus en cours de métamorphose : la queue, le système nerveux central et le foie. Pour chaque organe, six stades de développement ont été comparés et les profils d'expression de 802 gènes différentiellement exprimés ont été sélectionnés.

Parmi ces 802 gènes différentiellement exprimés, 140 sont sélectionnés dans le foie, 513 dans le système nerveux central et 393 dans la queue. Dans ce travail, nous nous intéressons exclusivement aux 393 gènes de la queue. Les résultats de ces puces à ADN n'ont pas été étudiés tels quels. Ceux-ci ont préalablement été structurés par apprentissage statistique. Ce travail a été fait par Yang Zhou [89]. Yang a identifié, par clustering spectral, 5 clusters de gènes dans la queue. Ceux-ci sont récapitulés dans la Figure 5.6. On observe 142 gènes non régulés (cluster 1), 112 gènes surrégulés de manière transitoires (cluster 2), 11 gènes fortement sous-régulés (cluster 3), 10 gènes fortement surrégulés (cluster 4) et 118 gènes sous-régulés de manière transitoire (cluster 5).

Les clusters 3 et 4 et les clusters 2 et 5 ont des profils d'expression moyens relativement symétriques. L'explication biologique pour les groupes de gènes ayant un profil

d'expression symétrique est qu'ils sont régulés simultanément : l'augmentation du niveau d'expression d'un groupe de gènes s'accompagne d'une diminution du niveau d'expression d'un autre groupe de gènes ou inversement. Etant donné le devenir de la queue durant la métamorphose, on extrapole des observations précédentes que les gènes régulés positivement sont sûrement des gènes pro-apoptotiques alors que les gènes régulés négativement sont sûrement, en grande majorité, des gènes anti-apoptotiques.

Notre objectif est de trouver le réseau de régulation induit par les hormones thyroïdiennes permettant une telle balance entre les gènes régulés positivement et négativement. Les réseaux de régulation possibles sont très simples et sont au nombre de 6 (nous verrons plus loin comment les dénombrer). L'objectif est alors d'identifier des couples pertinents de gènes régulés positivement et négativement. Ces couples seront à la base de nouvelles expériences dont le but sera d'identifier parmi les 6 réseaux, le plus probable. L'identification de tels couples est permis par une étude bibliographique approfondie, suivie de longues discussions avec les biologistes.

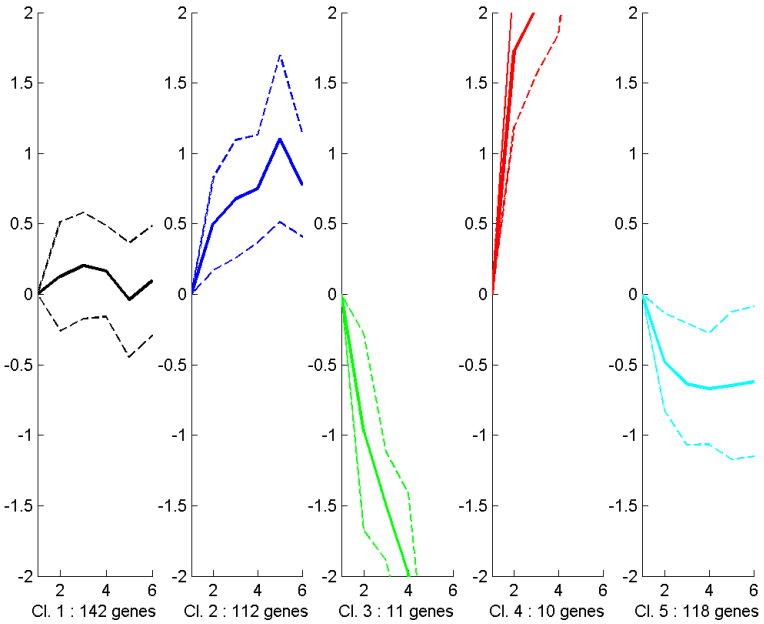

FIG. 5.6 – Visualisation du profil moyen pour chaque cluster de gènes de la queue.

5.4.2 Construction des modèles

5.4.2.1 Inhibition et activation

Dans les modèles précédents, l'inhibition de T3 par D3 était d'ordre enzymatique et non directement génétique. Ainsi, une inhibition correspond, dans ce cas, à une consommation de T3 quand l'inhibiteur est présent. En l'absence d'inhibiteur, la concentration en T3 n'est pas modifiée.

De même, l'activation enzymatique par D2 se traduit dans le modèle, par une augmentation de la concentration en T3 quand D2 est présent. En l'absence de l'activateur la concentration en T3 n'est pas modifiée.

Dans le cadre d'activation ou d'inhibition génétique, nous adoptons un raisonnement différent [90] :

- Activation : Lorsque A active B ($A \xrightarrow{+} B$), si A est au dessus du seuil d'activation, alors la concentration de B augmente. Par contre, si la concentration de A est sous le seuil, l'absence d'activation provoque la diminution de la concentration de B.

- Inhibition : Lorsque A inhibe B ($A \xrightarrow{-} B$), si A est au dessus du seuil d'inhibition, la concentration de B diminue. Par contre, si la concentration de A est insuffisante pour que l'inhibition ait lieu, la concentration de B augmente.

Le formalisme des réseaux de Petri hybrides temporisés (THPN) ne permet pas de modéliser ces définitions directement à l'aide d'une seule transition. Comme nous allons le voir ci-dessous, chaque action (inhibition ou activation) est modélisée par deux transitions.

- Activation : L'activation ($A \xrightarrow{+} B$) se définit donc comme suit (Figure 5.7) :

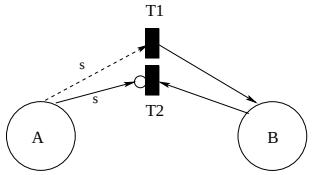

FIG. 5.7 – Modélisation d'une activation génique.

Lorsque A est au dessous du seuil s, seule la transition T_2 est activée, B est alors consommé. Lorsque A est au dessus du seuil s, seule le transition T_1 est active, son tir entraîne l'augmentation de B.

- Inhibition : L'inhibition ($A \xrightarrow{-} B$) se définit donc comme suit (Figure 5.8) :

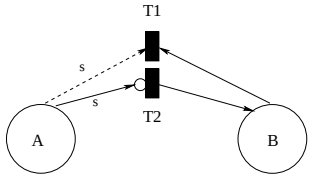

FIG. 5.8 – Modélisation d'une inhibition génique.

Lorsque A est au dessus du seuil s, seule la transition T_1 est active, elle entraîne la consommation de B, ce qui traduit l'inhibition de B par A. Lorsque A est au dessous du seuil, l'inhibition n'est pas active, seule la transition T_2 est tirée, elle engendre l'augmentation de B.

5.4.2.2 Obtention de 6 modèles

On considère trois entités dans nos modèles :

1. Les hormones thyroïdiennes (HT),
2. Les gènes positivement régulés, noté *Ad* pour gènes adultes. En effet, ces gènes sont, *a priori*, des gènes pro-apoptotiques permettant au têtard de devenir adulte,
3. Inversement, les gènes négativement régulés sont notés *Te* pour têtard. Ce sont *a priori*, des gènes anti-apoptotiques dont le rôle va être de protéger la queue de sa disparition.

A partir de ces trois entités, nous obtenons 6 modèles différents. Les réseaux de régulation sont présentés sur la Figure 5.9 et les modèles THPN associés sont représentés sur la Figure 5.10. La deuxième Figure est donnée à titre d'illustration. Par la suite, nous nous référencerons essentiellement à la première figure beaucoup plus lisible.

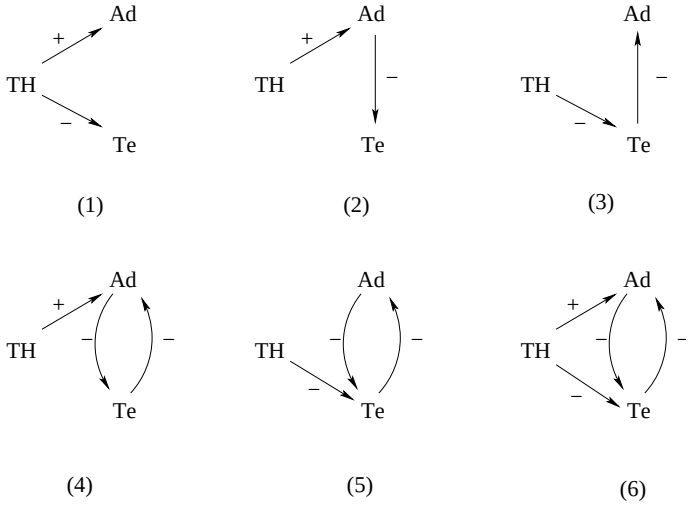

FIG. 5.9 – Les six modèles possibles engendrant une balance entre gènes positivement et négativement régulés.

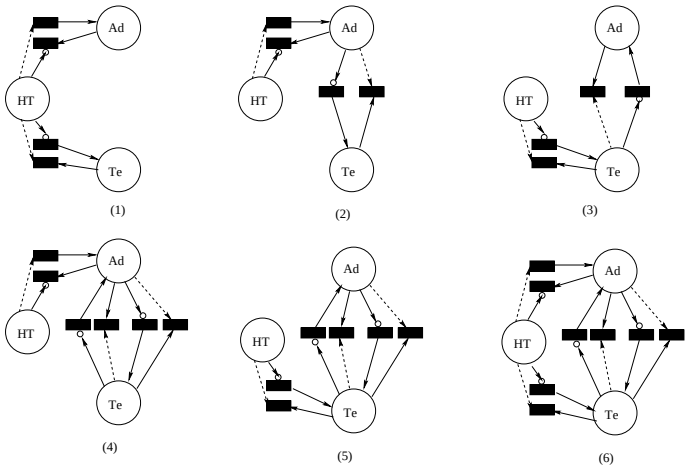

FIG. 5.10 – Les 6 modèles THPN associés aux 6 réseaux de régulation possibles.

5.4.3 Analyse biologique des modèles

Dans cette section, nous recherchons et discutons plusieurs couples (Ad/Te) pertinents sur lesquels il serait possible d'effectuer des expériences nous permettant de choisir parmi les 6 modèles, ou au moins d'en réfuter quelques uns. Une recherche bibliographique nous a permis de mettre en évidence l'existence de certaines flèches et donc de commencer à s'intéresser spécifiquement à un nombre réduit de modèles. Elle a également permis d'identifier deux programmes distincts de mort cellulaire dans la queue du têtard : un premier programme concerne les muscles de la queue et le deuxième programme concerne les fibroblastes.

5.4.3.1 Muscles de la queue

La mort des muscles de la queue est obtenue par apoptose. Il s'agit, par conséquent, d'une mort cellulaire-autonome. Les membres de la famille Bcl2 sont très souvent associés à ce processus de dégradation [91]. Les membres de cette famille activent la voie mitochondriale de l'apoptosome. Ceux-ci libèrent le cytochrome c dans la mitochondrie, ce qui déclenche une cascade de caspases entraînant la mort cellulaire.

Parmi les nombreux membres de la famille Bcl2, nous nous sommes particulièrement intéressés au couple Bax et XR11 [91]. Alors que Bax est un gène pro-apoptotique, XR11 est un anti-apoptotique. Ce couple, très souvent cité dans la littérature, semble être à la base d'une balance entre "vie et mort", balance régulée par les HT. En effet, Bax est un gène de réponse aux hormones thyroïdiennes [92]. A l'heure actuelle, rien ne permet de dire qu'il en est de même pour XR11. Au niveau protéique et non transcriptionnel, il est reconnu que XR11 inhibe Bax [93, 94]. Cory *et al.* [91] mettent également en avant l'inhibition inverse. De ces données, on en déduit un réseau de régulation potentiel (Figure 5.11) : les HT activent la synthèse de la protéine pro-apoptotique Bax (régulation transcriptionnelle), et les gènes Bax/XR11 s'inhibent respectivement (régulation protéique).

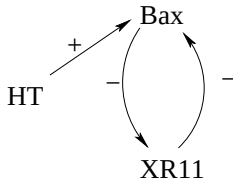

Fig. 5.11 – Réseau de régulation potentiel associant le couple Bax/XR11.

En 2006, Das *et al.* [95] ont étudié l'expression de nombreux gènes impliqués dans la résorption de la queue lors de la métamorphose naturelle et lors d'une métamorphose induite par l'ajout d'hormones thyroïdiennes. Ceux-ci ont identifié une majorité de gènes sous-régulés dans les muscles de la queue contre une minorité de gènes sur-régulés. Ils ont identifié les gènes sous-régulés comme des gènes impliqués dans les voies métaboliques, telles que le métabolisme énergétique..., les gènes sur-régulés sont essentiellement associés à des protéases, comme la glycine déhydrogénase ou la cytosolic dipeptidase. Dans cet article, seules des régulations de type transcriptionnel sont mises en évidence. On en déduit

le réseau de régulation de la Figure 5.12. Ce réseau laisse supposer que les hormones thyroïdiennes "éteignent" les grandes voies métaboliques de la cellule (respiration, énergie,...) et activent en parallèle des protéases responsables de la dégradation de l'environnement cellulaire.

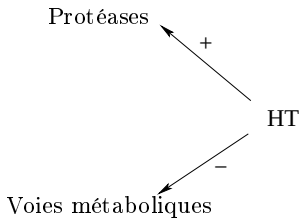

FIG. 5.12 – Réseau de régulation potentiel découlant des résultats de Das *et al.* dans les muscles.

Des deux points précédents, nous pouvons émettre l'hypothèse qu'il n'existe pas un réseau unique responsable de la perte des muscles de la queue. Une hypothèse serait de combiner les deux réseaux de régulation présentés sur les Figures 5.11 et 5.12. Le réseau de régulation ainsi formé aurait à son centre les HT, d'où sortiraient trois régulations différentes : une positive vers Bax, une positive vers les protéases et une négative vers les voies métaboliques.

5.4.3.2 Fibroblastes de la queue

Le deuxième type cellulaire étudié dans la littérature concerne les fibroblastes de la queue. Ces cellules ont été identifiées comme la source de nombreuses enzymes protéolytiques jouant un rôle dans la dissolution de la queue. Nous ne sommes donc pas ici dans un cas d'apoptose cellulaire-autonome, comme présenté dans les muscles. L'étude transcriptionnelle des gènes impliqués dans la résorption de la queue met en avant une majorité de gènes sur-régulés contre une minorité de gènes sous-régulés [95]. Nous sommes donc dans le cas opposé à celui des muscles. Parmi les gènes sur-régulées, on compte une grande majorité d'enzymes hydrolytiques comme l'intégrine ou des enzymes lysosomales. Les quelques gènes sous-régulés correspondent à des protéines de la matrice extra-cellulaire ou à différents types de collagènes. Ces régulations (activation ou inhibition) étant accentuées par l'ajout d'hormones thyroïdiennes, on en déduit que ces processus sont activés parallèlement par les HT, il s'agit ici de régulations transcriptionnelles (Figure 5.13). Ainsi, la dissolution de la queue peut s'expliquer par les deux aspects suivants : la structure et le maintien des cellules ne sont plus assurés à cause de l'inhibition de protéines de la matrice extra-cellulaire, et en parallèle, des enzymes hydrolytiques assurent la dissolution des fibroblastes.

5.4.3.3 Discussion

Deux réseaux de régulation se distinguent parmi les six proposés dans la Figure 5.9 :

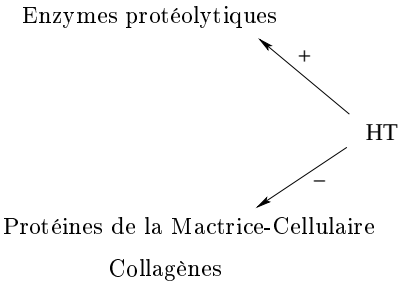

FIG. 5.13 – Réseau de régulation potentiel découlant des résultats de Das *et al.* dans les fibroblastes.

- Le premier (1) qui permet d'inhiber transcriptionnellement des processus biologiques indispensables à la survie cellulaire (respiration,...) et d'activer en parallèle des processus biologiques destructeurs (protéases,...).
- Le quatrième (4) correspond au modèle en relation avec l'hypothèse d'une balance entre pro et anti-apoptotiques.

Ce travail a permis d'identifier certains couples pertinents sur lesquels des expériences sont à prévoir, comme Bax et XR11. Dans l'avenir, nous chercherons donc à agrandir et à exploiter ces deux réseaux avec de nouvelles données, comme celles obtenues par une étude des interactions spécifiques à Bax et XR11.

Chapitre 6

Extension paramétrique

6.1 Motivation

Dans le Chapitre 3, nous avons présenté une procédure de vérification de propriétés pour les modèles THPN dans lesquels chaque paramètre est numérique. Notre procédure permet donc de répondre à la question suivante : "Pour un jeu de paramètres donné, notre modèle THPN satisfait-il la propriété étudiée ?". Il serait bien sûr intéressant d'étendre notre procédure à un ensemble de combinaisons de paramètres. Etant donné que les paramètres d'un THPN sont soit des entiers (marquage discret), soit des rationnels (vitesses instantanées, délai,...), soit des réels (marquage continu), il existe donc une infinité de jeux de paramètres possibles. Il n'est donc pas envisageable de mettre en place un approche exhaustive qui a été envisagé dans le cas des modèles uniquement discrets [4] et qui consiste à énumérer chacun des paramétrages possibles et de lancer pour chacun d'eux une étape de model-checking.

Les paramètres d'un système biologique peuvent se classer en deux catégories : certains paramètres sont connus (données cinétiques, concentrations,...) et leurs valeurs sont disponibles dans la littérature, d'autres sont incertains voire inconnus et sont estimés de façon plus ou moins arbitraires. Etant donné qu'il nous est impossible de tester l'ensemble des valeurs possibles des paramètres inconnus, nous nous sommes orientés vers une extension paramétrique du model-checking qui autorise une définition symbolique de certains paramètres. Cette extension paramétrique permet de répondre à la question suivante : "Soient α, β et γ les paramètres inconnus de notre modèle THPN. Pour quelles valeurs de paramètres notre modèle THPN satisfait-il la propriété biologique et pour quelles valeurs de paramètres ne la satisfait-il pas ?" Cette procédure que nous appellerons par la suite, *model-checking paramétrique*, nous permet, pour un système biologique étudié, d'estimer la ou les valeurs possibles de paramètres inconnus, en confrontant notre *modèle paramétrique* à une ou plusieurs propriétés biologiques connues.

Ce chapitre se divise en quatre sections. La première section définit le cadre paramétrique en présentant le formalisme des THPN paramétriques. L'exécution d'un THPN paramétrique dépend de la valeur des paramètres symboliques, il n'est donc pas possible de construire directement le graphe d'évolution. Nous définissons donc dans la deuxième section, un algorithme permettant de construire l'ensemble des graphes d'évolution pos-

sibles d'un THPN paramétrique que nous regroupons sous la forme d'un pseudo-arbre appelé par la suite le *graphe d'évolution symbolique*. La troisième section présente la procédure complète de *model-checking paramétrique*. Enfin, la dernière section se focalise sur les propriétés associées à notre extension paramétrique.

6.2 Cadre paramétrique

Dans cette section, nous autorisons la définition symbolique des paramètres d'un THPN. Un THPN composé de paramètres symboliques est appelé par la suite THPN paramétrique (Définition 6.2). La Définition 6.1 distingue quatre types de symboles paramétriques.

Définition 6.1 (Symboles paramétriques)**.** *On appelle un ensemble de symboles, un quadruplet* $\mathbb{P} = (\alpha, \beta, \gamma, \delta)$ *où* α, β, γ *et* δ *sont des ensembles de symboles tels que :*

- *Les symboles* $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ *représentent le marquage des places continues définis dans* $\mathbb{R}^+$,
- *Les symboles* $\beta = \{\beta_1, \beta_2, ..., \beta_n\}$ *représentent le marquage des places discrètes définis dans* $\mathbb{N}$,
- *Les symboles* $\gamma = \{\gamma_1, \gamma_2, ..., \gamma_i\}$ *représentent les vitesses maximales des transitions continues définies dans* $\mathbb{Q}^+$ *et*
- *Les symboles* $\delta = \{\delta_1, \delta_2, ..., \delta_j\}$ *représentent les délais des transitions discrètes définis dans* $\mathbb{Q}^+$.

Définition 6.2 (THPN paramétrique)**.** *Un Réseau de Petri Hybride Temporisé paramétrique est un 9-uplet* $(\mathcal{P}, \mathcal{T}, \zeta, Pre, Post, \mathbb{P}, m_0, delai, V)$ *où :*

- $\mathcal{P}$ *et* $\mathcal{T}$ *sont des ensembles disjoints, respectivement de places et transitions,*
- $\zeta : \mathcal{P} \cup \mathcal{T} \rightarrow \{D, C\}$ *appelée fonction hybride indique pour chaque nœud s'il est discret ou continu,*
- $Pre : \mathcal{P} \times \mathcal{T} \rightarrow \mathbb{Q}^+ \amalg \mathbb{N}$ *est la matrice d'incidence d'entrée. Si* $T \in T^D$ *alors* $Pre(P, T) \in \mathbb{N}$ *sinon* $Pre(P, T) \in \mathbb{Q}^+$,
- $Post : \mathcal{T} \times \mathcal{P} \rightarrow \mathbb{Q}^+ \amalg \mathbb{N}$ *est la matrice d'incidence de sortie. Si* $T \in T^D$ *alors* $Post(T, P) \in \mathbb{N}$ *sinon* $Post(T, P) \in \mathbb{Q}^+$,
- $\mathbb{P} = (\alpha, \beta, \gamma, \delta)$ *est un ensemble de symboles paramétriques,*
- $m_0 : \mathcal{P} \rightarrow \mathbb{R}^+ \amalg \mathbb{N} \amalg \alpha \amalg \beta$ *est le marquage initial vérifiant la propriété : si* $P \in P^D$ *alors* $m_0(P) \in \mathbb{N} \amalg \beta$ *sinon* $m_0(P) \in \mathbb{R}^+ \amalg \alpha$,
- $delai : T^D \rightarrow \mathbb{Q}^+ \amalg \delta$ *est le temps associé à* T,
- $V : T^C \rightarrow \mathbb{Q}^+ \amalg \gamma$ *représente la vitesse de tir maximal.*

Remarque 6.1. *Nous avons fait le choix de garder les matrices Pre et Post non paramétriques. Ce choix est justifié par le type de questions posées lors de l'étude d'un système biologique : elles concernent essentiellement les concentrations initiales (marquage paramétrique) et les vitesses des réactions biochimiques étudiées (vitesses maximales paramétriques).*

Remarque 6.2. *La définition d'un THPN paramétrique inclut celle d'un THPN non paramétrique dans lequel l'ensemble des paramètres $\mathbb{P}$ est vide.*

La Figure 6.1 présente un exemple de THPN paramétrique, dans lequel tous les paramètres sont symboliques.

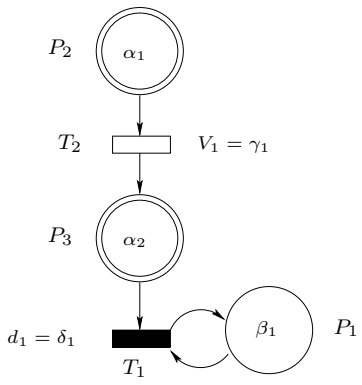

FIG. 6.1 – Exemple de THPN paramétrique

La Définition 6.2 est une définition syntaxique des THPN paramétriques. La sémantique des THPN paramétriques dépend des paramètres. Par exemple, pour décider si une transition discrète T_i est permise, on regarde si sa place prédécesseur P_j satisfait $m(P_j) \geq Pre(P_j, T_i)$. Si le marquage de la place P_j est symbolisé par le paramètre β_j, il faut envisager les deux cas, permise et non permise, en associant à chacun des deux cas la contrainte adaptée. Ainsi, la transition T_i est permise sous la contrainte $\beta_j \geq Pre(P_j, T_i)$ et la transition T_i n'est pas permise sous la contrainte $\beta_j < Pre(P_j, T_i)$.

L'étude dynamique d'un THPN paramétrique nécessite donc le traitement de tous les cas possibles en fonction des valeurs de paramètres. Chaque cas est alors caractérisé par un ensemble de contraintes. Les définitions présentées ci-dessous ont donc pour but de définir formellement les expressions paramétriques et les contraintes que nous allons utiliser pour construire le graphe d'évolution symbolique.

Définition 6.3 (Expression paramétrique)**.** *Etant donné un ensemble de paramètres $\mathbb{P}$, une expression paramétrique* ep *est une expression de la forme* $\sum_{k=1}^{n}(\prod_{j=1}^{n_k} t_j^k.p_j^k) + t_0$ *avec* $t_j^k \in (\mathbb{N}, \mathbb{Q}^+, \mathbb{R}^+)$ *selon que les* $p_j^k \in (\alpha, \beta, \gamma, \delta)$*. L'ensemble des expressions paramétriques est noté* $Exp(\mathbb{P})$*.*

La définition précédente autorise des relations non linéaires entre les paramètres. Ce cas général est à considérer étant donné que la sémantique des THPN (construction du graphe d'évolution) nécessite de calculer des produits entre vitesses instantanées et marquages et de faire des opérations non linéaires pour calculer les événements futurs qui sont susceptibles de se produire. Néanmoins, les expressions linéaires étant beaucoup plus

manipulables, nous définissons ce sous-ensemble d'expressions paramétriques. Nous discuterons par la suite, des contraintes à imposer sur les paramètres permettant de n'obtenir que des expressions linéaires.

Définition 6.4 (Expression paramétrique linéaire)**.** *Une expression paramétrique linéaire* elp *est une expression de la forme* $t_1p_1 + ... + t_np_n + t_0$*, avec* $t_j \in (\mathbb{N}, \mathbb{Q}^+, \mathbb{R}^+)$ *selon que les* $p_j \in (\alpha, \beta, \gamma, \delta)$*. L'ensemble des expressions paramétriques linéaires est noté* $ExpL(\mathbb{P})$*.*

Les expressions paramétriques peuvent former des contraintes grâce aux comparateurs $\{<, >, \geq, \leq\}$. Nous distinguons alors les contraintes non linéaires des contraintes linéaires (Définition 6.5).

Définition 6.5 (Contrainte)**.** *Une contrainte est une inégalité de la forme* $e \sim e'$ *où* e, e' *sont des expressions paramétriques et* $\sim \in \{<, >, \geq, \leq\}$*.*

Si de plus, e *et* e' *sont des expressions paramétriques linéaires, la contrainte est dite linéaire.*

La négation d'une contrainte (linéaire ou non) c, notée $\neg c$ est obtenue en remplaçant les signes $<, \leq, >, \geq$ respectivement par $\geq, >, \leq, <$.

Une contrainte $c = e \sim e'$, avec $\sim \in \{<, >, \geq, \leq\}$ est satisfiable s'il existe une valuation de paramètres pour laquelle l'inégalité $e \sim e'$ est vérifiée. La notion de valuation de paramètres est donnée ci-dessous.

Définition 6.6 (Valuation de paramètres)**.** *Une valuation de paramètres est une fonction* $\nu : \mathbb{P} \to \mathbb{N} \amalg \mathbb{Q}^+ \amalg \mathbb{R}^+$*, telle que :*

- *Si* $p \in \alpha$ *alors* $\nu(p) \in \mathbb{R}^+$*, si* $p \in \gamma \cup \delta$ *alors* $\nu(p) \in \mathbb{Q}^+$ *sinon* $\nu(p) \in \mathbb{N}$*.*
- *Si* ep *est une expression paramétrique et* ν *une valuation, alors* $ep[\nu]$ *dénote la valeur obtenue en remplaçant chaque paramètre* p *de* ep *par* $\nu(p)$*.*
- *De même, nous définissons* $c[\nu]$ *pour une contrainte* c*. La valuation* ν *satisfait la contrainte* c*, noté* $\nu \models c$*, si* $c[\nu]$ *est évalué à vrai.*
- *Une valuation* ν *satisfait un ensemble de contraintes* $\mathbb{C}$*, noté* $\nu \models \mathbb{C}$ *si et seulement si pour toute contrainte* c *de* $\mathbb{C}$*,* $\nu \models c$*.*

Définition 6.7 (Sémantique d'une contrainte)**.** *La sémantique d'une contrainte* c*, notée* $\langle c \rangle$*, est l'ensemble des valuations qui satisfont* c*. La sémantique d'un ensemble de contraintes* $\mathbb{C}$ *est donnée par* $\langle \mathbb{C} \rangle = \bigcap_{c \in \mathbb{C}} \langle c \rangle$*.*

6.3 Graphe d'évolution symbolique

Dans cette section, nous détaillons l'algorithme de construction du graphe d'évolution symbolique (noté GES) associé à un modèle paramétrique THPN. Cet algorithme se divise en deux étapes :

- La première étape consiste à construire l'ensemble des IB-states successeurs d'une transition donnée (Section 6.3.1),
- La deuxième construit, à partir d'un IB-state, l'ensemble des transitions qui lui succèdent (Section 6.3.2).

Comme nous le verrons plus tard, à chaque nœud (transition ou IB-state) sera associé un ensemble de contraintes. Nous discuterons également le mode de traitement de ces contraintes en distinguant le cas des contraintes linéaires et non linéaires. Enfin, l'algorithme complet sera détaillé avant d'être illustré sur des exemples. Nous discuterons alors les propriétés d'un tel graphe d'évolution symbolique.

6.3.1 Obtention des IB-states

Comme dans le cas de graphes d'évolution numériques, le graphe d'évolution symbolique possède une unique transition initiale, notée T_0^{GES}. Celle-ci est annotée du marquage initial de chaque place $P_i \in \mathcal{P}$ (éventuellement paramétrique) et d'un ensemble d'événements venant d'avoir lieu et se résumant à $(Evt(T_0^{GES}) = \{NoEvt\})$ pour la première transition. On cherche l'ensemble des IB-states succédant à la transition T_0^{GES} et plus généralement à une transition T_k^{GES} du graphe d'évolution symbolique. Etant donné que chaque IB-state est caractérisé par le marquage des places discrètes et par les vitesses instantanées des transitions continues, nous allons traiter ci-dessous chacun de ces deux points séparément.

6.3.1.1 Calcul du marquage discret

Soit $m^D = (\beta_1, \beta_2, ..., \beta_n)$ le marquage de chaque place discrète. Ce marquage est modifié dans le nouvel IB-state uniquement si un événement de type D1 (tir d'une transition discrète) appartient à l'ensemble des événements $Evt(T_k^{GES})$. Soit $D1(T_i) \in Evt(T_k^{GES})$ un événement de type D1 impliquant la transition discrète T_i du THPN-paramétrique, le marquage de chaque place prédécesseur $P_j \in {}^\circ T_i$ est diminuée de $Pre(P_j, T_i)$ jetons et chaque place successeur $P_j \in T_i^\circ$ est incrémentée de $Post(T_i, P_j)$ jetons. Si dans l'IB-state courant $m(P_j) = \beta_j$ et que $Pos(T_i, P_j) = 1$ alors le tirage de T_i mène à un nouvel IB-state dans lequel $m(P_j) = \beta_j + 1$. Si dans l'IB-state courant $m(P_j) = \beta_j$ et que $Pre(P_j, T_i) = 1$ alors le tirage de T_i mène à un IB-state dans lequel $m(P_j) = \beta_j - 1$.

6.3.1.2 Calcul des vitesses instantanées

Le calcul des vitesses instantanées dépend du marquage continu. Ce marquage permet de déterminer la permission de chaque transition continue. Nous avons vu dans le Chapitre 1 que :

- lorsqu'une transition continue est *fortement permise*, elle tire à sa vitesse maximale,
- lorsqu'elle est *faiblement permise*, elle tire à une vitesse comprise entre 0 et sa vitesse maximale,
- enfin si la *transition n'est pas permise*, elle ne peut tirer et sa vitesse vaut 0.

L'algorithme de calcul des vitesses instantanées présenté dans un cadre purement numérique dans le Chapitre 1 peut être étendu au cas paramétrique. Nous procédons alors en deux temps : lors d'un *pré-traitement*, nous déterminons l'ensemble des vitesses instantanées possibles. Le *post-traitement* permet ensuite retirer toutes les combinaisons incohérentes.

Pré-traitement : La permission d'une transition continue dépend du marquage continu qui peut être paramétrique. Pour chaque place paramétrique continue P_j prédécesseur de la transition continue T_i, nous devons distinguer le cas où $\alpha_j > 0$ et où $\alpha_j = 0$. On doit alors considérer tous les cas possibles. Par exemple, si on a deux places continues paramétriques P_1 et P_2 prédécesseurs de la transition continue T, on obtient $\{\alpha_1 > 0, \alpha_1 = 0\}$ et $\{\alpha_2 > 0, \alpha_2 = 0\}$. On a alors 4 combinaisons possibles : $\{\alpha_1 = 0, \alpha_2 = 0\}$, $\{\alpha_1 > 0, \alpha_2 = 0\}$, $\{\alpha_1 = 0, \alpha_2 > 0\}$ et $\{\alpha_1 > 0, \alpha_2 > 0\}$.

1. Pour toute transition continue T_i fortement permise, $v_i = V_i = \gamma_i$. Ce point correspond au premier cas de l'algorihtme pour un THPN non paramétrique.

2. Soit T_j une transition continue faiblement permise à cause d'un ensemble de places $Q_j(t)$ de marquage nul. Si T_j n'est pas impliquée dans un conflit ou que T_j a le premier niveau de priorité, alors $v_j(t)$ se calcule de la façon suivante :

$$v_j(t) = min\left(\min_{P_i \in Q_j(t)}\left(\frac{I_i}{Pre(P_i, T_j)}\right) \ , \ V_j\right)$$

   Ce point correspond au deuxième cas de l'algorithme pour un THPN non paramétrique. Cependant, comme le calcul précédent est paramétrique, on doit considérer trois cas selon l'ordre entre $\min_{P_i \in Q_j(t)} \frac{I_i}{Pre(P_i, T_j)}$ et V_j. Les prémisses de l'implication ($\Rightarrow$) forment les contraintes. Si celles-ci sont vérifiées alors on en déduit la valeur de la vitesse instantanée indiquée en conclusion. Prenons l'exemple d'un IB-state dans lequel $v_j = \min_{P_i \in Q_j(t)} \frac{I_j}{Pre(P_i, T_j)}$, celui-ci sera alors annoté de la contrainte $\min_{P_i \in Q_j(t)} \frac{I_i}{Pre(P_i, T_j)} < V_j$.

$$(A)\begin{pmatrix} \min\limits_{P_i \in Q_j(t)} \frac{I_i}{Pre(P_i, T_j)} < V_j & \Rightarrow & v_j = \min\limits_{P_i \in Q_j(t)} \frac{I_j}{Pre(P_i, T_j)} \\ \min\limits_{P_i \in Q_j(t)} \frac{I_i}{Pre(P_i, T_j)} = V_j & \Rightarrow & v_j = \min\limits_{P_i \in Q_j(t)} \frac{I_i}{Pre(P_i, T_j)} = V_j \\ \min\limits_{P_i \in Q_j(t)} \frac{I_i}{Pre(P_i, T_j)} > V_j & \Rightarrow & v_j = V_j \end{pmatrix}$$

3. Soit le conflit $K_m =< P_m, \{T_1, T_2, ..., T_n\} >$ tel que $T_1 < T_2 < ... < T_n$, c'est-à-dire T_1 est prioritaire sur T_2, elle-même prioritaire sur T_3, etc. La transition T_1 a le premier niveau de priorité, v_1 est donc calculée comme précédemment. La transition T_2 a le second niveau de priorité. Soit $Q_k(t)$, un autre ensemble de places de marquage nul entrant dans T_2, $v_2(t)$ est calculée comme suit :

$$(B) \quad v_2(t) = min\begin{pmatrix} (1) & I_m - Pre(P_m, T_1).v_1(t), \\ (2) & V_2, \\ (3) & \min\limits_{P_i \in Q_k(t)} \frac{I_i}{Pre(P_i, T_2)} \end{pmatrix}$$

   De cette expression, on déduit 7 cas possibles :

$$(C)\left(\begin{array}{lcl}
(1) < (2)\ et\ (3) & \Rightarrow & v_2(t) = I_m - Pre(P_m, T_1).v_1(t) \\
(2) < (1)\ et\ (3) & \Rightarrow & v_2(t) = V_2 \\
(3) < (1)\ et\ (2) & \Rightarrow & v_2(t) = \min\limits_{P_i \in Q_k(t)} \dfrac{I_i}{Pre(P_i, T_2)} \\
(1) = (2) < (3) & \Rightarrow & v_2(t) = I_m - Pre(P_m, T_1).v_1(t) = V_2 \\
(1) = (3) < (2) & \Rightarrow & v_2(t) = I_m - Pre(P_m, T_1).v_1(t) = \min\limits_{P_i \in Q_k(t)} \dfrac{I_i}{Pre(P_i, T_2)} \\
(2) = (3) < (1) & \Rightarrow & v_2(t) = V_2 = \min\limits_{P_i \in Q_k(t)} \dfrac{I_i}{Pre(P_i, T_2)} \\
(1) = (2) = (3) & \Rightarrow & v_2(t) = V_2 = \min\limits_{P_i \in Q_k(t)} \dfrac{I_i}{Pre(P_i, T_2)}) = I_m - Pre(P_m, T_1).v_1(t)
\end{array}\right)$$

On procède de même pour les autres transitions $T_3, ..., T_n$.

Post-traitement : On combine alors les ensembles de contraintes obtenus à chaque étape. Chaque ensemble combiné nous donne un IB-state. Une combinaison peut être incohérente, comme par exemple $\{\gamma_1 = \gamma_2,\ \gamma_1 > \gamma_2\}$. Dans le cas général, déterminer l'incohérence d'un ensemble de contraintes est indécidable. Néanmoins, comme nous le détaillerons dans la section 6.3.4, nous avons mis en évidence une condition suffisante permettant de ne se rapporter qu'à des contraintes linéaires. L'incohérence d'un ensemble de contraintes linéaires est alors décidable. Il nous sera ainsi possible de retirer les combinaisons de contraintes incohérentes.

6.3.1.3 Illustration

Illustrons cet algorithme sur le THPN paramétrique de la Figure 6.2.

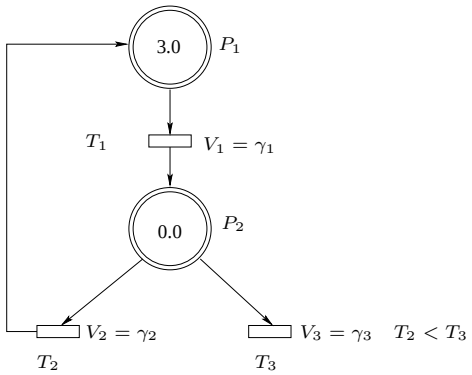

FIG. 6.2 – Exemple de THPN paramétrique.

Pré-traitement : Le marquage des places n'est pas paramétrique.

1. Seule la transition T_1 est fortement permise, on en déduit $v_1 = V_1 = \gamma_1$.

2. La transition T_2 a le premier niveau de priorité. Etant donnée que la seule source de la place P_2 est la transition T_1, on a $\frac{I_2}{Pre(P_2,T_2)} = \gamma_1$. Par conséquent, $v_2(t) = min(\gamma_1, \gamma_2)$. On obtient donc les trois cas suivants :

$$\left(\begin{array}{lcl} \gamma_1 < \gamma_2 & \Rightarrow & v_2 = \gamma_1 \\ \gamma_1 = \gamma_2 & \Rightarrow & v_2 = \gamma_1 = \gamma_2 \\ \gamma_1 > \gamma_2 & \Rightarrow & v_2 = \gamma_2 \end{array} \right)$$

3. Il ne reste que la transition T_3 à traiter. Celle-ci a le deuxième et dernier niveau de priorité. Etant donné qu'il n'y pas d'autres ensembles de places entrant dans T_3, on ne considère pas le cas (3) de l'équation (B). La comparaison des cas (1) et (2) de l'équation (B) amène donc à traiter uniquement les trois cas suivants correspondant à (1) < (2), (1) = (2) et (1) > (2) où (1) correspond à $\gamma_1 - v_2$ et (2) correspond à γ_3 :

$$\left(\begin{array}{lcl} \gamma_3 < \gamma_1 - v_2 & \Rightarrow & v_3 = \gamma_3 \\ \gamma_3 = \gamma_1 - v_2 & \Rightarrow & v_3 = \gamma_3 = \gamma_1 - v_2 \\ \gamma_3 > \gamma_1 - v_2 & \Rightarrow & v_3 = \gamma_1 - v_2 \end{array} \right)$$

Post-traitement : On fait toutes les combinaisons possibles de chaque ensemble de cas, en prenant soin de retirer toutes les combinaisons incohérentes :

$$\left(\begin{array}{lcl} \gamma_1 < \gamma_2 \; et \; \gamma_3 < \gamma_1 - v_2 & \Rightarrow & v_1 = v_2 = \gamma_1, \; v_3 = \gamma_3 \\ \gamma_1 < \gamma_2 \; et \; \gamma_3 = \gamma_1 - v_2 & \Rightarrow & v_1 = v_2 = \gamma_1, \; v_3 = \gamma_3 = \gamma_1 - v_2 \\ \gamma_1 < \gamma_2 \; et \; \gamma_3 > \gamma_1 - v_2 & \Rightarrow & v_1 = v_2 = \gamma_1, \; v_3 = \gamma_1 - v_2 \\ & ... & \end{array} \right)$$

6.3.2 Obtention des transitions

La deuxième étape de l'algorithme de construction du graphe d'évolution symbolique consiste à partir d'un IB-state, à déduire les différentes transitions qui peuvent lui succéder. Une transition T_i^{GES} du graphe d'évolution est annotée d'un temps $t_{T_i^{GES}}$, d'un ensemble d'événements et du marquage des places continues au temps $t_{T_i^{GES}}$. Trouver les transitions succédant à un IB-state consiste donc à trouver les ensembles d'événements futurs possibles. Bien entendu, les futurs événements dépendent des paramètres et des relations entre eux.

Comme dans le cadre numérique, les futurs événements sont listés dans une séquence ordonnée par le temps, notée TOS. La méthode consiste à calculer le temps des futurs événements C1, D2 et D1, de les rentrer dans TOS et de les comparer. Dans un cadre numérique, le plus petit temps de TOS nous donne l'événement le plus proche, c'est lui qui formera la future transition du graphe d'évolution. Dans un cadre paramétrique, les temps de TOS peuvent être paramétriques, on obtient alors une transition pour chaque ordonnancement *pertinent* des temps de TOS. La notion d'*ordonnancements pertinents* est détaillée ci-après.

La méthode générale, étant à présent décrite, revenons en détail sur le calcul des futurs événements C1, D2 et D1.

Futurs événements C1 : Pour chaque place continue $P_i \in P^C$ de balance négative $B_i < 0$, P_i se vide (événement C1) au temps t tel que : $0 = m(P_i) + B_i.t$ soit $t = -\frac{m(P_i)}{B_i}$.
$m(P_i)$ et B_i peuvent être paramétriques, ce qui donne un temps t paramétrique.

Futurs événements D2 : On rappelle qu'un événement D2 est associé à une place continue dont le changement de marquage modifie le degré de permission d'une transition discrète. Deux cas se distinguent. Le marquage de la place continue P_i est au dessus du seuil s permettant à la transition discrète T_j de tirer, sa balance étant négative $B_i < 0$, le marquage de P_i redescendra sous le seuil s au temps t tel que : $s = m(P_i) + B_i.t$, soit $t = \frac{s-m(P_i)}{B_i}$.
Dans le deuxième cas, le marquage de la place P_i est au dessous du seuil s et sa balance est positive $B_i > 0$. Le marquage de la place P_i atteindra donc le seuil s au temps $t = \frac{s-m(P_i)}{B_i}$.

Futurs événements D1 : Si une place continue vient d'atteindre le seuil s permettant à une transition discrète T_j de tirer, alors on peut ajouter dans TOS le futur tir de la transition T_j (événement D1) au temps $t + delai(T_j)$. Sinon, le marquage des places discrètes, fourni dans l'IB-state, nous indique quelles transitions discrètes sont permises ou pas, on en déduit facilement les futurs événements D1.

Supposons que TOS ne contienne que 3 éléments $(D1(t_{D1}), D2(t_{D2}), C1(t_{C1}))$, dans lesquels les temps t_{D1}, t_{D2} et t_{C1} sont paramétriques. Comme dans le cas numérique, le but est ici de déterminer quel(s) événement(s) va (vont) avoir lieu en premier. Les temps étant paramétriques, nous déterminons les premiers événements en ordonnançant les temps t_{D1}, t_{D2} et t_{C1}. On en déduit que seuls les ordonnancements donnant des ensembles différents d'événements ayant lieu en premier sont pertinents. On obtient alors les 7 *ordonnancements pertinents* suivants. Chacun de ces ordonnancements donne une nouvelle transition dans le graphe d'évolution symbolique.

(1) $t_{D1} < t_{D2}$ et $t_{D1} < t_{C1}$ (5) $t_{D2} = t_{D1}$ et $t_{D1} < t_{C1}$
(2) $t_{D2} < t_{D1}$ et $t_{D2} < t_{C1}$ (6) $t_{D2} = t_{C1}$ et $t_{C1} < t_{D1}$
(3) $t_{C1} < t_{D1}$ et $t_{C1} < t_{D2}$ (7) $t_{D1} = t_{C1}$ et $t_{C1} < t_{D2}$
(4) $t_{D2} = t_{D1} = t_{C1}$

Chaque transition du graphe d'évolution succédant au même IB-state est, à présent, annotée du temps t des prochains événements ainsi que de l'ensemble des événements qui vont avoir lieu. La dernière étape consiste à obtenir le marquage de places continues à ce nouveau temps. Le marquage de chaque place continue P_i décroît linéairement à une vitesse correspondant à la balance de la place (B_i). Par conséquent, le nouveau marquage de chaque place P_i est obtenu comme ceci : $m(P_i) = m(P_i)_0 + B_i.t$ où $m(P_i)_0$ correspond au dernier marquage de la place P_i (obtenu sur la dernière transition du GES), B_i est la balance de la place obtenue grâce aux vitesses instantanées de l'IB-state et t est le temps de TOS qui vient d'être calculé. Là encore, le marquage peut être paramétrique.

La construction du graphe d'évolution symbolique correspond donc à un appel alterné

de construction d'IB-states (Section 6.3.1) et de construction de transitions (Section 6.3.2). La section suivante présente l'algorithme de construction du graphe d'évolution dans sa globalité.

6.3.3 Algorithme

De façon à assurer la terminaison de l'algorithme, nous imposons la restriction suivante :

Restriction 1. *Chaque paramètre $p_i \in \mathbb{P}$ du THPN paramétrique doit être borné $p_i \in (min_{p_i}, max_{p_i})$ (p_i appartient à un intervalle pouvant être ouvert ou fermé à droite et/ou à gauche). Sans perte de généralité, nous considérons par la suite que la borne minimale est 0. L'association de deux bornes à chaque paramètre, forme un ensemble de contraintes que nous appellerons $\mathbb{C}_0$, vrai à la racine, c'est-à-dire à la première transition du graphe d'évolution symbolique.*

Algorithme :

1. Initialisation
   (a) Entrer les données relatives au THPN, marquage initial, résolution locale des conflits et ordonnancement des sous-réseaux par ordre de priorité,
   (b) Initialisation du marquage continu. Initialisation de TOS par les D1-event possibles **(dépend des paramètres de type α et β)** : Soit T_i une transition discrète, et $P_j \in {}^\circ T_i$, si le marquage de P_j est paramétrique α_j (ou β_j), on considère deux cas $\alpha_j \geq Pre(P_j, T_i)$ et $\alpha_j < Pre(P_j, T_i)$ où $Pre(P_j, T_i)$ est le seuil de permission de la transition T_i.
   Soit $t_s = t_0 = 0$.

2. Si l'ensemble des événements D2 au temps t_s n'est pas vide, c'est-à-dire $D2(t_s) \neq \emptyset$ alors ajouter le D1-event correspondant dans TOS.

3. Si $D1(t_s) = \emptyset$ alors **ALLER A L'ETAPE 4** sinon tirer les transitions correspondantes (ou un sous-ensemble s'il existe des conflits de type 1), mettre à jour TOS et **ALLER A L'ETAPE 2**.

4. Construire les sous-réseaux de Petri continus sous-jacents.

5. Si tous les niveaux de priorité des sous-réseaux ont été traités alors **ALLER A L'ETAPE 6** sinon calculer les vitesses instantanées de chaque transition continue (Section 6.3.1). On subdivise le GES au niveau des IB-states. La contrainte associée à chaque IB-state correspond à la conjonction de la contrainte du nœud père avec la contrainte spécifique de l'IB-state. Tout IB-state incohérent par son ensemble de contraintes est retiré. **ALLER A L'ETAPE 5**.

6. Mise à jour de TOS en déterminant les temps des prochains événements C1, D2 et D1 (Section 6.3.2). On subdivise le GES au niveau des transitions. La contrainte associée à chaque transition correspond à la conjonction de la contrainte du nœud père avec la contrainte spécifique de la transition. Toute transition incohérente par son ensemble de contraintes est retirée.

7. Calcul du marquage au nouveau temps dans chaque branche.

8. Si les paramètres de l'IB-state sont déjà présents dans un précédent IB-state de la branche alors FIN sinon **ALLER A L'ETAPE 2** pour cette branche.

Remarque 6.3. *L'algorithme de construction d'un graphe d'évolution est un cas particulier de l'algorithme de construction du GES dans lequel l'ensemble des paramètres* $\mathbb{P}$ *est vide. En effet, l'application de cet algorithme sur un modèle THPN non paramétrique donne le même graphe d'évolution que celui obtenu avec l'algorithme numérique (Chapitre 1 Section 1.4.2.3).*

6.3.4 Traitement des contraintes

Deux étapes de l'algorithme permettent de simplifier le graphe d'évolution symbolique en retirant les nœuds (transitions ou IB-states) incohérents par leur ensemble de contraintes. Dans cette section, nous discutons donc des méthodes nous permettant de gérer ces ensembles de contraintes. Nous allons différencier le cas où nous n'avons que des contraintes linéaires du cas général.

6.3.4.1 Traitement des contraintes linéaires

Nous avons retenu deux méthodes pour vérifier si un ensemble de contraintes linéaires est cohérent.

Programmation linéaire. L'ensemble des contraintes d'un nœud du graphe d'évolution symbolique peut former les contraintes d'un système de programmation linéaire. La fonction objective importe alors peu dans ce cas présent. Etant donnée la complétude des algorithmes de programmation linéaire comme le simplex [96], si l'algorithme ne trouve pas de solution c'est qu'il n'existe pas de valuations permettant de satisfaire toutes les contraintes. On peut alors conclure que la contrainte n'est pas satisfiable et donc que le nœud n'est pas cohérent.

Opération sur des polyèdres. Chaque contrainte d'un nœud du graphe d'évolution symbolique peut être converti en polyèdres [69]. Selon que l'intersection de tous les polyèdres est vide ou non (algorithme dans [69]), on conclut à l'incohérence ou à la cohérence de l'ensemble de contraintes. S'il est vide, c'est qu'il n'existe pas de valuation permettant de satisfaire l'ensemble des contraintes de ce nœud, il est donc incohérent.

6.3.4.2 Traitement des contraintes non linéaires

Lorsque les contraintes ne sont pas linéaires, leur traitement devient plus difficile. On peut les traiter par système de programmation non-linéaire. Néanmoins, ces problèmes sont non décidables, il existe néanmoins des algorithmes de décision définis au cas par cas [97]. Un exemple est la programmation par contrainte [98], elle renvoie un intervalle sur lequel il est possible mais pas certain qu'il existe une solution. Ce type de contraintes étant beaucoup plus délicat à traiter, nous allons essentiellement nous rapporter à des cas linéaires. Le paragraphe suivant précise les restrictions à apporter pour n'étudier que des contraintes linéaires.

6.3.4.3 Cas d'obtention de contraintes linéaires et non linéaires

La présence d'un seul type de paramètres dans le THPN paramétrique (soit α, soit β, soit γ soit δ) est une condition suffisante pour l'obtention de contraintes linéaires. Les vitesses des transitions continues sont soit des constantes, soit des expressions paramétriques. Dans les deux cas, leur valuation reste constante entre deux événements. Le marquage évolue ainsi linéairement et le coefficient directeur est proportionnel aux vitesses. Si le marquage est paramétrique, les vitesses restent des constantes et inversement. Les contraintes restent dans ce cas linéaires.

Dans un des exemples traités ci-dessous, les deux types de paramètres α (marquage continu) et δ (délai) coexistent au sein d'un même THPN paramétrique. Les contraintes obtenues restent néanmoins linéaires. Ceci est cohérent étant donné que les délais paramétriques n'interviennent ni dans le calcul des vitesses instantanées ni dans le calcul du marquage (voir l'exemple détaillé de la Figure 6.4 combinant ces deux types de paramètres). Néanmoins, nous n'avons pas généralisé ce résultat, nous garderons ainsi comme condition suffisante la présence d'un seul type de paramètre. Cette condition est applicable en biologie, celle-ci permet de traiter séparément les questions concernant les concentrations (marquage), celles traitant des vitesses de réaction (vitesses instantanées) et celles traitant des délais.

La combinaison des paramètres de type α et de type γ (vitesses maximales) fait très facilement perdre la linéarité des contraintes. Par exemple, soient P_1 et P_2 deux places dont le marquage vaut respectivement α_1 et α_2 et P_2 se vide à une vitesse paramétrique γ. La place P_2 se vide au temps $t = \frac{\alpha_2}{\gamma}$ où γ est la vitesse de consommation de ressources des places P_1 et P_2. Au temps $t = \frac{\alpha_2}{\gamma}$, le marquage de la place P_1 vaut donc $\alpha_1 - \frac{\alpha_2}{\gamma}$, on a perdu la linéarité.

6.3.5 Exemples

Dans cette section, nous illustrons notre algorithme de construction du graphe d'évolution symbolique sur deux exemples : le premier exemple est un modèle paramétrique de THPN possédant plusieurs paramètres, et le deuxième traite spécifiquement la présence de conflits dans le modèle.

6.3.5.1 Exemples avec plusieurs paramètres

Considérons le THPN paramétrique de la Figure 6.3, celui-ci comporte deux paramètres : le marquage de la place continue P_2 vaut initialement α_1 et le délai de la transition discrète T_1 vaut δ_1. Etant donné que chaque paramètre doit être initialement borné (Restriction 1), nous imposons que $\alpha_1 \in [0, 5]$ et $\delta_1 \in]0, 5]$.

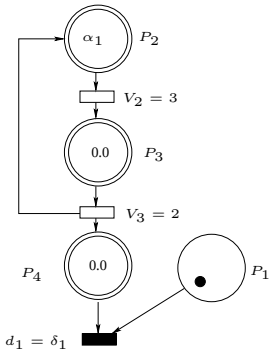

FIG. 6.3 – THPN paramétrique numéro 1.

Le graphe d'évolution associé au THPN de la Figure 6.3 est présenté sur la Figure 6.4. La branche annotée CN correspond à un Cas Numérique, elle ne comporte donc aucune difficulté, c'est pourquoi celle-ci n'est pas détaillée. Le numéro au début de chaque item correspond à l'étape de l'algorithme à considérer.

1. Marquage initial $m_0 = (1, \alpha_1, 0, 0)$. Il n'y a pas d'événement D1 au temps $t_0 = 0$, $D1(t_0) = \emptyset$.

5. Les vitesses instantanées dépendent du marquage. On distingue 2 cas : le cas où $\alpha_1 = 0$ et le cas où $\alpha_1 \in]0; 5]$. Dans le cas où $\alpha_1 \in]0; 5]$ la transition T_2 est fortement permise et tire donc à sa vitesse maximale, sinon la transition n'est pas permise, les transitions continues ne peuvent tirer ($v_2 = v_3 = 0$) et l'algorithme est terminé.

6. Deux types d'événements peuvent avoir lieu dans le cas où $\alpha_1 \in]0; 5]$.
   - Le marquage de la place P_4 atteint le seuil 1 permettant à T_1 de tirer. Cet événement a lieu au temps 0.5.
   - Le marquage de la place P_2 se vide. La place P_2 se vide au temps $t = \alpha_1$.

   L'ordonnancement des événements dépend donc de la valeur du paramètre α_1, on distingue donc trois cas : $\alpha_1 < 0.5$, $\alpha_1 = 0.5$ et $\alpha_1 > 0.5$.

Détaillons le cas où $\alpha_1 > 0.5$. Les autres branches sont explorées de façon similaire. On trouvera l'arbre complet à la Figure 6.4.

7. Le prochain événement est donc $m(P_4) = 1$ au temps $t = 0.5$ et le marquage au temps $t = 0.5$ vaut $(1, \alpha_1 - 0.5, 0.5, 1)$ respectivement pour les places P_1, P_2, P_3 et P_4.

5. La place P_2 n'étant pas vide, la transition T_2 tire toujours à sa vitesse maximale : $v_2 = 3$, $v_3 = 2$.

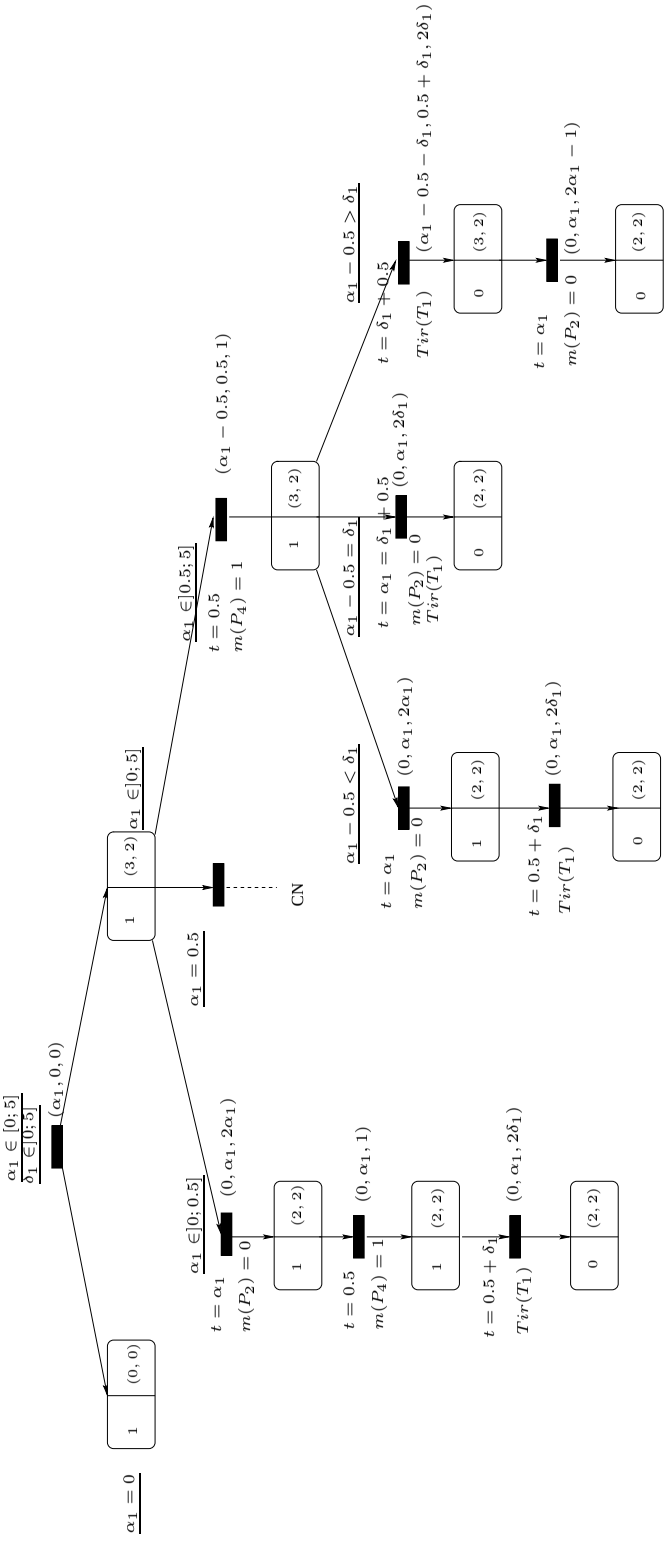

FIG. 6.4 – Graphe d'évolution symbolique du THPN paramétrique numéro 1.

6. Deux types d'événements peuvent avoir lieu : le tir de T_1 au temps $t = 0.5 + \delta_1$ ou la place P_2 se vide à $t = \alpha_1$. On distingue donc à nouveau 3 cas : $\alpha_1 < 0.5 + \delta_1$, $\alpha_1 =$

$0.5 + \delta_1$ et $\alpha_1 > 0.5 + \delta_1$.

On se focalise à présent sur la sous-branche $\alpha_1 - 0.5 < \delta_1$.

7. Le prochain événement a donc lieu au temps $t = \alpha_1$, il s'agit de la place P_2 qui se vide. Le marquage au temps $t = \alpha_1$ vaut $(1, 0, \alpha_1, 2\alpha_1)$.
5. La place P_3 n'étant pas vide, la transition T_3 tire à sa vitesse maximale $v_3 = 2$. La place P_2 est à présent vide, la transition T_2 ne peut tirer à sa vitesse maximale sans engendrer un marquage de P_2 négatif, on a donc $v_2 = v_3 = 2$.
6. Etant données les balances nulles des places P_2 et P_3 aucun événement C1 n'est à prévoir. Le seul événement à venir est le tir de la transition discrète T_1 qui aura lieu au temps $t = 0.5 + \delta_1$.
7. La transition T_1 est tirée au temps $t = 0.5 + \delta_1$, consommant l'unique jeton de la place discrète P_1. Le marquage vaut alors $(0, 0, \alpha_1, 2\delta_1)$.
5. Le calcul des vitesses donne comme précédemment $v_2 = v_3 = 2$.
6. Il n'y a pas de futurs événements possibles. Nous sommes arrivés à la fin de l'algorithme.

6.3.5.2 Exemple contenant des conflits

Considérons à présent le THPN paramétrique de la Figure 6.5 dans lequel les transitions continues T_2 et T_3 sont en conflit. On choisit de résoudre ce conflit en donnant la priorité à la transition T_2 dont la vitesse maximale est paramétrique $V_2 = \gamma_1 \in]0, 5]$. On obtient alors le graphe d'évolution symbolique de la Figure 6.6.

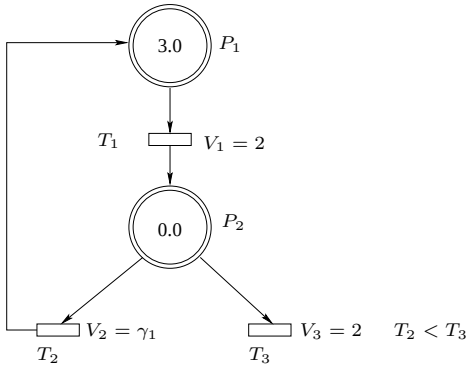

FIG. 6.5 – THPN paramétrique numéro 2.

La place P_2 étant vide, les transitions T_2 et T_3 sont faiblement permises. Les ressources de P_2 sont $V_1 = 2$. D'après l'algorithme de calcul de vitesses instantanées, $v_2 = min(2, \gamma_1)$, et $v_3 = min(2 - v_2, 2)$. On obtient 3 cas pour v_2 et 3 cas pour v_3 :

$$\left(\begin{array}{lcl} \gamma_1 < 2 & \Rightarrow & v_2 = \gamma_1 \\ \gamma_1 = 2 & \Rightarrow & v_2 = \gamma_1 = 2 \\ \gamma_1 > 2 & \Rightarrow & v_2 = 2 \end{array} \right)$$

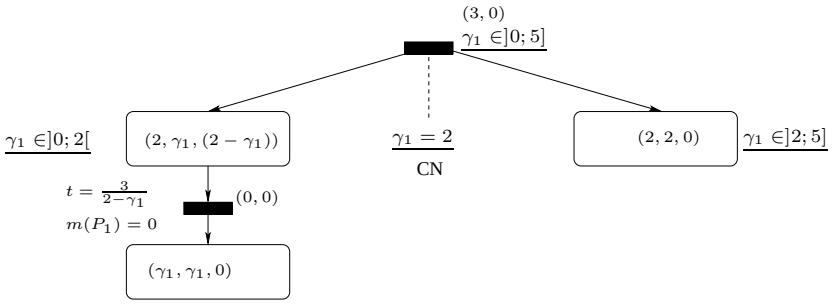

FIG. 6.6 – Graphe d'évolution symbolique du THPN paramétrique numéro 2.

$$\begin{pmatrix} 2 - v_2 < 2 & \Rightarrow & v_3 = 2 - v_2 \\ 2 - v_2 = 2 & \Rightarrow & v_3 = 2 - v_2 = 2 \\ 2 - v_2 > 2 & \Rightarrow & v_3 = 2 \end{pmatrix}$$

Parmi les 3 cas de v_3, seul le premier est cohérent. En effet, v_2 vaut soit γ_1 soit 2, v_2 est donc strictement positive, seule la contrainte $2 - v_2 < 2$ est cohérente. La combinaison des deux ensembles nous donne donc les trois cas de la Figure 6.6.

6.3.6 Propriétés du graphe d'évolution symbolique

Chaque branche du graphe d'évolution symbolique correspond à un graphe d'évolution, noté GE_i. Chaque graphe d'évolution GE_i est défini par un ensemble de contraintes $\mathbb{C}_i$ où $\mathbb{C}_i$ est la contrainte du dernier IB-state (deadlock) ou de la dernière transition (bouclage) du graphe d'évolution GE_i.

Lemme 6.1. *Soit N_i un nœud du graphe d'évolution symbolique donnant les nœuds fils $N_{i1}, N_{i2}, ..., N_{in}$ suivant les contraintes $\mathbb{C}_{i1}, \mathbb{C}_{i2}, ..., \mathbb{C}_{in}$ alors pour tout j, $l \in [1, n], l \neq j$, $\langle \mathbb{C}_{ij} \rangle \cap \langle \mathbb{C}_{il} \rangle = \emptyset$*

Preuve : Considérons tout d'abord le cas où le nœud N_i est une transition du GES qui engendre les IB-states $N_{i1}, N_{i2}, ..., N_{in}$. La subdivision en IB-states dépend du calcul des vitesses instantanées. L'algorithme présenté dans la Section 6.3.1 montre que 3 cas sont à considérer :

1. chaque place continue paramétrique P_i fournit 2 contraintes $\alpha_i = 0$ et $\alpha_i > 0$,
2. les transitions faiblement permises de premier niveau de priorité ou non impliquées dans un conflit fournissent 3 contraintes détaillées dans la matrice (A) (Section 6.3.1.2 page 121),
3. les transitions continues de degré supérieur à 1 fournissent 7 contraintes suivant le schéma détaillé dans la matrice (C) (Section 6.3.1.2 page 121).

Chaque point (1, 2 ou 3) est formé de contraintes deux à deux incompatibles. Chaque IB-state successeur correspond à une combinaison des trois ensembles de contraintes décrits dans les items ci-dessus. Par conséquent, deux IB-states IB_{ij} et IB_{il} diffèrent au moins au niveau d'une contrainte d'un des trois items. Ces deux contraintes étant incompatibles, on en déduit que $\langle \mathbb{C}_{ij} \rangle \cap \langle \mathbb{C}_{il} \rangle = \emptyset$.

Dans le deuxième cas, le nœud N_i est un IB-state qui engendre les transitions $N_{i1}, N_{i2}, ..., N_{in}$ dans le GES. La subdivision en transition se fait selon l'ordre des prochains événements (D1, D2 ou C1) à venir. Le nombre d'événements futurs possibles est fini et tous les ordonnancements pertinents sont énumérables (Section 6.3.2). Etant donné que ces ordonnancements sont deux à deux incompatibles, on en déduit la propriété voulue.
□

Lemme 6.2. *Soit N_i un nœud dans le graphe d'évolution symbolique donnant les nœuds $N_{i1}, N_{i2}, ..., N_{in}$ suivant les contraintes $\mathbb{C}_{i1}, \mathbb{C}_{i2}, ..., \mathbb{C}_{in}$ alors $\langle\mathbb{C}_{i1}\rangle \cup \langle\mathbb{C}_{i2}\rangle \cup ... \cup \langle\mathbb{C}_{in}\rangle = \langle\mathbb{C}_i\rangle$*

Preuve :

Considérons tout d'abord le cas où le nœud N_i est une transition du GES qui engendre les IB-states $N_{i1}, N_{i2}, ..., N_{in}$. La subdivision en IB-states dépend du calcul des vitesses instantanées. L'algorithme présenté dans la Section 6.3.1 montre que 3 cas sont à considérer :

1. chaque place continue paramétrique P_i fournit 2 contraintes $\alpha_i = 0$ et $\alpha_i > 0$,
2. les transitions faiblement permises de premier niveau de priorité ou non impliquées dans un conflit fournissent 3 contraintes détaillées dans la matrice (A) (Section 6.3.1.2 page 121),
3. les transitions continues de degré supérieur à 1 fournissent 7 contraintes suivant le schéma détaillé dans la matrice (C) (Section 6.3.1.2 page 121).

Chaque point (1, 2 ou 3) est formé de contraintes deux à deux incompatibles dont la disjonction est toujours vraie (évaluation à $\top$), quelle que soit la valuation des paramètres. La contrainte associée à un IB-state IB_{ij} correspond à la conjonction de la contrainte $\mathbb{C}_j$ spécifique de l'IB-state IB_{ij} et de la contrainte $\mathbb{C}_i$ du nœud père. Ainsi, on obtient pour les n IB-states :

$$(\mathbb{C}_i \wedge \mathbb{C}_1) \vee ... \vee (\mathbb{C}_i \wedge \mathbb{C}_n)$$

ce qui permet d'écrire :

$$\mathbb{C}_i \wedge (\mathbb{C}_1 \vee ... \vee \mathbb{C}_n)$$

ce qui équivaut à :

$$\mathbb{C}_i \wedge \top \equiv \mathbb{C}_i$$

Dans le deuxième cas, le nœud N_i est un IB-state qui engendre les transitions $N_{i1}, N_{i2}, ..., N_{in}$ dans le GES. La subdivision en transition se fait selon l'ordre des prochains événements (D1, D2 ou C1) à venir. Le nombre d'événements futurs possibles est fini et tous les ordonnancements sont énumérables (Section 6.3.2), leur disjonction vaut également $\top$.
□

Lemme 6.3. *Soit $\mathbb{C}_0$, la contrainte de la première transition du graphe d'évolution symbolique. Pour toute valuation ν telle que $\nu \models \mathbb{C}_0$, il existe un unique graphe d'évolution GE_i tel que $\nu \models \mathbb{C}_i$.*

Preuve : Les lemmes 6.1 et 6.2 montrent respectivement que chaque branche est unique et que toutes les branches existent.
□

Théorème 6.1. *Soit $\mathbb{C}_0$, la contrainte de la première transition du graphe d'évolution symbolique. Soit ν une valuation, telle que $\nu \models \mathbb{C}_0$, le graphe d'évolution GE du THPN dans lequel chaque paramètre $p_i \in \mathbb{P}$ a été initialement remplacé par $\nu(p_i)$, est exactement le graphe d'évolution GE_i^ν obtenu à partir du graphe d'évolution $GE_i \in GES$ tel que $\nu \models \mathbb{C}_i$, dans lequel chaque expression paramétrique e est remplacée par sa valeur $e[\nu]$.*

Preuve : Le Lemme 6.3 montre que pour une valuation $\nu \models \mathbb{C}_0$, il existe un unique graphe d'évolution GE_i dans le GES tel que $\nu \models \mathbb{C}_i$. Lorsque les expressions paramétriques e de GE_i sont remplacées par leur valeur $e[\nu]$, on obtient le graphe d'évolution non paramétrique GE_i^ν. Etant donné que l'algorithme de construction d'un graphe d'évolution est un cas particulier de l'algorithme de construction du GES lorsqu'il n'y a pas de paramètres (même calcul de vitesses, même événements,...). On en déduit donc que $GE_i^\nu = GE$.
□

6.4 Model-checking paramétrique

Le graphe d'évolution symbolique correspond à l'exécution symbolique d'un THPN paramétrique. Notre objectif est, à présent, de mettre en place une méthode de model-checking paramétrique permettant de répondre à la question suivante : "Soit ϕ une propriété biologique donnée. Pour quelles valeurs de paramètres le modèle satisfait-il la propriété ? ". Le but est d'obtenir une contrainte $\mathbb{C}$ telle que si une valuation de paramètres ν satisfait $\mathbb{C}$ alors le THPN paramétrique satisfait, pour cette valuation, la propriété étudiée et inversement. Etant donné notre objectif, nous n'intégrons les paramètres symboliques que dans le modèle THPN et non dans la propriété étudiée. Néanmoins, d'autres procédures comme [99], intègrent des paramètres au niveau de la propriété et non du modèle. Une question biologique pourrait alors être de la forme suivante : "Quelle est la concentration de la molécule x quand notre système biologique atteint son état stationnaire ?".

Alur *et al.* ont développé un raisonnement paramétrique pour les automates temporisés dans [100]. Ils ont montré que la décision du vide d'un automate temporisé paramétrique n'est pas décidable. Suite à ces résultats, de nombreuses méthodes de model-checking paramétriques décidables ont été développées sur des sous-classes d'automates temporisés [99, 101, 102]. Les sous-classes ainsi définies sont très spécifiques et ne peuvent être facilement utilisées. Nakata *et al.* [102] développèrent les automates à intervalles de temps dans lesquels le temps de chaque arc doit être déterminé. Hune *et al.* [99] utilisèrent les automates L/U (pour Lower and Upper Bound). Ces automates obligent chaque horloge à posséder une borne minimale et une borne maximale bien distinctes. A l'heure actuelle,

aucun raisonnement paramétrique n'a été fait sur les automates Event-Clock.

Dans ce chapitre, nous décrivons notre procédure de model-checking paramétrique pour les THPN paramétriques. Les étapes de la procédure paramétrique sont similaires à celles développées dans le cadre numérique :

- Le graphe d'évolution symbolique est converti en automate. La présence de paramètres dans le GES nous oblige à définir les automates Event-Clock paramétriques,
- La propriété étudiée est convertie en automate Event-Clock. Etant donné qu'il n'y a pas de paramètres dans la propriété étudiée, cette étape reste la même.

De façon à ne pas déterminer le vide d'un automate Event-Clock paramétrique, nous ajoutons une étape intermédiaire qui convertit l'automate Event-Clock paramétrique du THPN paramétrique en automate Event-Clock non paramétrique. On se rapporte alors à un cas numérique. Le produit des deux automates Event-Clock (THPN paramétrique et propriété) est effectué et le vide est déterminé.

La première section présente le formalisme des automates Event-Clock paramétriques. La procédure complète de model-checking paramétrique est alors présentée dans la deuxième section. Enfin, les propriétés de notre procédure sont discutées dans la dernière section.

6.4.1 Automates Event-Clock paramétriques

Notre procédure de model-checking paramétrique convertit, comme dans le cas numérique, le THPN paramétrique ainsi que la négation de la propriété en deux automates. La présence de paramètres dans le THPN nous oblige à étendre les définitions d'atomes et d'automates Event-Clock aux notions d'atomes paramétriques, notés p-atomes (Définition 6.8) et aux notions d'automates Event-Clock paramétriques, notés p-ECA (Définition 6.9).

Définition 6.8 (p-Atome)**.** *Etant donnés $\mathbb{P}$, un ensemble de paramètre et $\Sigma = (V, Pr)$ une signature, un p-atome est une expression de la forme $r \geq r'$, p ou leur négation, où r, r' sont soit des termes soit des expressions paramétriques et $p \in Pr$, tel que si r (resp. r') est de la forme x_α ou y_α, l'autre terme r' (resp. r) est nécessairement un rationnel ou une expression paramétrique.*

Notons que les atomes peuvent être vus comme des p-atomes.

Définition 6.9 (Automate Event-Clock paramétrique)**.** *Un automate Event-Clock paramétrique, noté p-ECA, sur la signature $\Sigma = (V, Pr)$ est un 8-tuple $A = (L, L_0, \mathbb{P}, \Sigma, \mathcal{C}, E, \mathcal{F}, cnstr)$ où :*

- *L est un ensemble fini de localisations et $L_0 \subseteq L$ est un sous-ensemble de localisations initiales,*
- *$\mathbb{P}$ est un ensemble de paramètres,*
- *$\Sigma = (V, Pr)$ est la signature,*
- *$\mathcal{C}$ est un ensemble d'horloges historiques ou de prédiction,*
- *E est un ensemble fini d'arcs. Un arc est un quadruplet $(l_1, \psi, l_2, \mathbb{C}_{c1})$ où $l_1 \in L$ est la localisation source, $l_2 \in L$ est la localisation cible, $\psi \in Obs(\Sigma)$ décrit la localisation l_1 et $\mathbb{C}_{c1}$ est une contrainte sur le franchissement de l'arc,*

- $\mathcal{F} = \{F_1, ..., F_n\}$ *où* $F_i \subseteq L$, *est un ensemble d'ensembles de localisations acceptantes,*
- $cnstr : L \rightarrow \mathbb{C}$ *est une fonction qui associe à chaque localisation un ensemble de contraintes sur les paramètres.*

Nous distinguons une sous-classe des automates paramétriques (p-ECA), dans laquelle il n'y a que des atomes et non des p-atomes. On appellera cette sous-classe celle des automates Event-Clock sans atomes paramétriques, notés *sp-ECA* (Définition 6.10).

Définition 6.10 (sp-ECA)**.** *Un automate Event-Clock sans atomes paramétriques A, noté sp-ECA, est un automate paramétrique dans lequel les observations étiquetant les arcs ne sont composées que d'atomes sans paramètres.*

Remarque 6.4. *On observe la relation suivante entre les trois classes d'automates Event-Clock : les automates Event-Clock non paramétrique (notés ECA), ceux sans atomes paramétriques (sp-ECA) et ceux paramétrique (p-ECA) :*

$$ECA \subseteq sp\text{-}ECA \subseteq p\text{-}ECA$$

La Figure 6.7 montre un exemple de p-ECA et de sp-ECA. Nous ne considérons, dans cet exemple, aucun lien entre les deux automates. Que ce soit dans le p-ECA ou dans le sp-ECA, les contraintes associées aux localisations sont les suivantes : $cnstr(l_1) = \{\alpha_1 \in [0; 0.5[\}$ et $cnstr(l_2) = \{\alpha_1 \in [0; 0.5[\}$ et celles associées aux arcs sont $\{m = \alpha\}$ pour l'arc allant de l_1 à l_2, et $\{m \geq \alpha\}$ pour l'arc allant de l_2 à l_1. La différence essentielle entre A_1 et A_2 est que les observations étiquetant les arcs de A_1 contiennent des paramètres (le p-atome $m = \alpha$ étiquette l'arc allant de l_1 à l_2) alors que les observations étiquetant les arcs de A_2 ne contiennent pas de paramètres, ce sont des atomes ($m \in]0, 0.5[$ étiquette l'arc allant de l_1 à l_2).

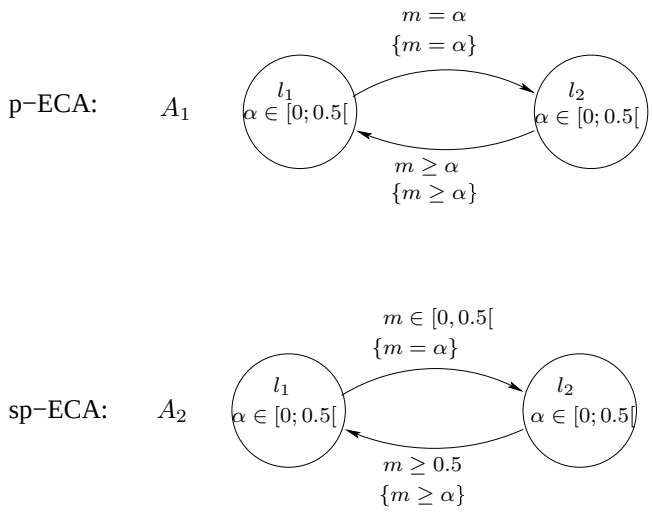

FIG. 6.7 – Exemple d'un p-ECA et d'un sp-ECA. Les contraintes associées aux arcs sont notées entre accolades.

La notion de trace sur un automate sans atomes paramétriques reste la même que dans le cas non paramétrique. Rappelons néanmoins la définition.

Définition 6.11 (Trace). *Une trace $\tau = \{(\varphi_i, t_i)\}_{i\in\mathbb{N}}$ est reconnue par un sp-ECA $A = (L, L_0, \mathbb{P}, \Sigma, \mathcal{C}, E, \mathcal{F}, cnstr)$ s'il existe un chemin acceptant infini $\gamma = l_0 \xrightarrow[\mathbb{C}_{c0}]{\psi_0} l_1 \xrightarrow[\mathbb{C}_{c1}]{\psi_1} ...l_n \xrightarrow[\mathbb{C}_{cn}]{\psi_n} ...$ où :*

- *chaque $l_i \in L$ et $l_0 \in L_0$,*
- *$(l_i, \psi_i, l_{i+1}, \mathbb{C}_{ci}) \in E$ et $(\tau, i) \leftsquigarrow \psi_i$ et $UpL(\psi_i) \subseteq UpL(\varphi_i)$,*
- *pour tout $F_i \in \mathcal{F}$, il existe infiniment beaucoup de positions j telles que $l_j \in F_i$.*

Le langage temporisé de l'automate sp-ECA A, noté $\mathcal{L}(A)$ correspond à l'ensemble des traces reconnues par A.

On remarque que la notion de langage temporisé sur les sp-ECA ne fait pas intervenir les contraintes $\mathbb{C}_{ci}$ des arcs. Celles-ci ne seront utilisés qu'au niveau du produit des deux automates.

6.4.2 Conversion du graphe d'évolution symbolique en automate

Le graphe d'évolution symbolique (GES) est converti en automate Event-Clock paramétrique, noté A_M. L'algorithme de conversion est similaire à celui utilisé dans le cadre non-paramétrique :

- Toute transition (T_i^{GES}) du GES donne une localisation l_i dans l'automate A_M. La localisation l_i est étiquetée par la contrainte $\mathbb{C}_i$ de la transition T_i^{GES} du GES. On a ainsi $cnstr(l_i) = \mathbb{C}_i$.
  Les arcs sortant de ces localisations sont étiquetés par les observations $\phi_2(T_i^{GES})$ (voir la définition de ϕ_2 page 59). On rappelle que $\phi_2(T_i^{GES})$ est une observation bien formée associant une valeur à chaque variable au moment où l'exécution passe par la transition T_i^{GES} du GES (vitesses, marquages,...).

- Tout couple (IB-state/transition) donne une localisation l_j dans l'automate A_M. La localisation l_j est étiquetée par la contrainte $\mathbb{C}_j$ de l'IB-state : $cnstr(l_j) = \mathbb{C}_j$.
  Les arcs sortant de ces localisations sont étiquetés par les observations ϕ_1 des IB-states (voir la définition de ϕ_1 page 59). On rappelle que ϕ_1 est une observation bien formée associant une valeur à chaque variable (vitesses, marquages,...) au moment où l'exécution passe par l'IB-state.

- Il existe un arc entre deux localisations dans l'automate A_M s'il existe un arc entre les nœuds associés dans le GES. Les localisations provenant des couples (IB-states/transitions) ont également un arc bouclant sur ces localisations (modélisation de la durée de l'IB-state).

- Les observations ϕ_1 et ϕ_2 étiquetant les arcs sont des conjonctions d'atomes. Ces observations peuvent donc également être considérées comme une conjonction de contraintes où chaque contrainte correspond à un atome. Ainsi, les contraintes associées à chaque arc correspondent exactement aux observations ϕ_1 et ϕ_2. L'information contenue dans ces observations apparaît donc deux fois au niveau des arcs : sous

forme d'observations et sous forme de contrainte. Cette duplication sera utile lors de la conversion du p-ECA en sp-ECA, opération qui nous permet de nous rapporter à un cas numérique (Section 6.4.3).

Si nous appliquons cet algorithme au graphe d'évolution symbolique de la Figure 6.4, nous obtenons l'automate Event-Clock paramétrique A_M dont une portion est présentée dans la Figure 6.8.

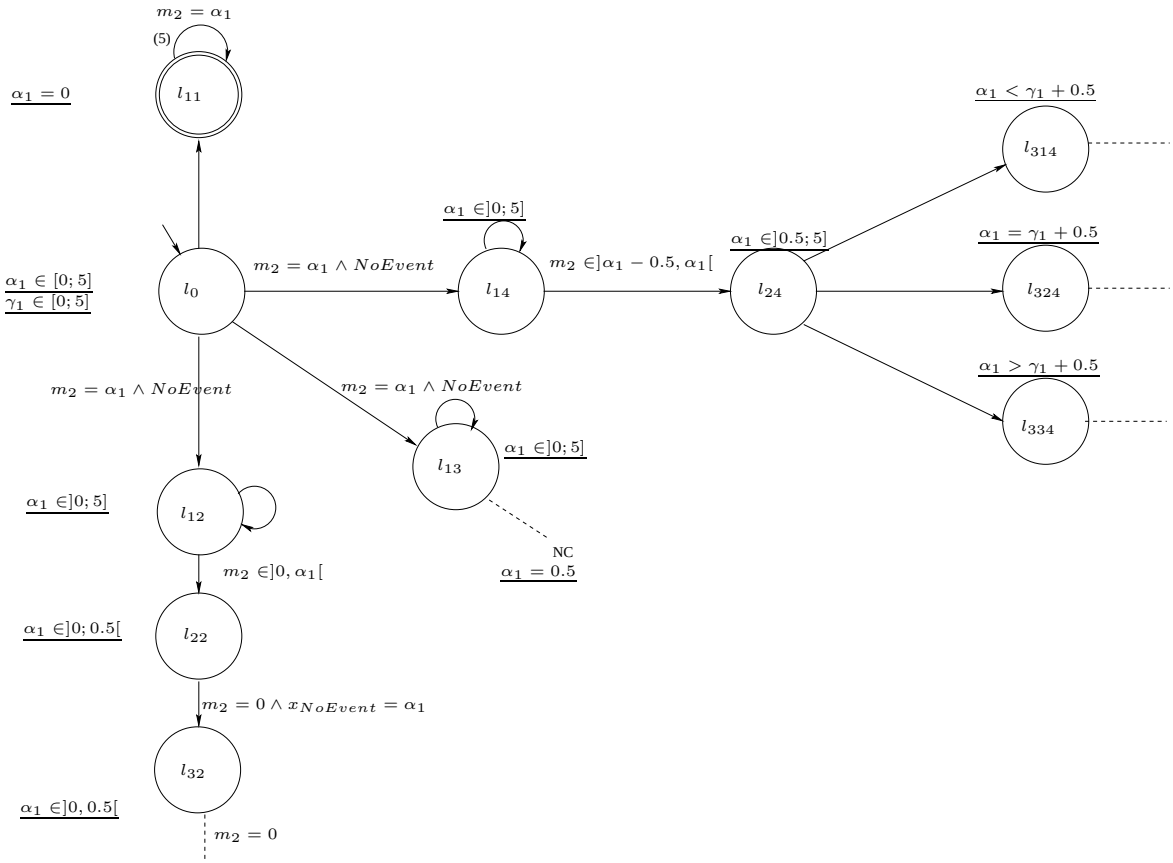

FIG. 6.8 – Automate Event-Clock paramétrique associé au graphe d'évolution symbolique du THPN numéro 1.

Déroulement de l'algorithme :

- La localisation l_0 représente la première transition du GES (les contraintes associées aux localisations sont celles qui sont soulignées). Les arcs sortant de cette localisation sont étiquetés par l'observation ϕ_2 composée de la conjonction de tous les marquages, de tous les événements et de toutes les vitesses de la première transition :

$$\phi_2 \equiv (m_1 = 1) \wedge (m_2 = \alpha_1) \wedge (m(P_3) = m(P_4) = 0) \wedge ...$$

De façon à faciliter la lisibilité de la Figure 6.8, seule une portion des observations

étiquette les arcs.

- La première subdivision du graphe d'évolution donne 4 couples (IB-state/transition), on obtient donc les localisations $l_{11} - l_{14}$ dans l'automate Event-Clock paramétrique A_M. La localisation l_{11} correspond à l'IB-state dans lequel le paramètre α_1 vaut 0. Etant donné que cet IB-state correspond à un deadlock, la localisation l_{11} est une localisation finale (double cercle). Les trois autres localisations ($l_{12} - l_{14}$) correspondent à la partition suivante : $\alpha_1 \in]0, 0.5[$, $\alpha_1 = 0.5$ et $\alpha_1 \in]0.5, 5]$. Les arcs sortant de ces localisations sont étiquetés par des observations décrivant les IB-states qui leur sont associés (marquages, vitesses,...).

- La Définition 6.9 des p-ECA associe à chaque arc un ensemble de contraintes. L'algorithme précise que chaque observation étiquetant les arcs est dédoublée sous forme de contrainte. Ainsi, l'arc allant de la localisation l_0 à la localisation l_{14} est étiqueté de l'observation $m_2 \in [0, 5] \wedge NoEvent$ et de la contrainte $\mathbb{C}_{c0} = \{m_2 \in [0, 5] \wedge NoEvent\}$.

6.4.3 Transformation en un sp-ECA

La présence de paramètres sur les arcs de l'automate A_M nous contraindrait, dans la suite de la procédure, à déterminer si le langage paramétrique d'un automate est vide ou non. Dans notre procédure de model-checking paramétrique, cette étape est évitée. Nous nous rapportons à un cas numérique en convertissant tous les p-atomes de l'automate paramétrique en atomes. Ainsi, le p-ECA A_M est modifié en sp-ECA.

Cette étape de conversion consiste à remplacer chaque paramètre $p \in \mathbb{P}$ par un encadrement numérique. Montrons que dans chaque localisation de l'automate Event-Clock paramétrique, il y a un encadrement numérique des paramètres. Chaque localisation l_i est annotée par la contrainte $cnstr(l_i)$. On note $\mathbb{C}_{NB_i}$ le sous-ensemble de contraintes bornant numériquement chaque paramètre. $\mathbb{C}_{NB_i}$ existe nécessairement étant donné que chaque paramètre est borné initialement par la contrainte $\mathbb{C}_0 = cnstr(l_0)$ qui annote la première transition du graphe d'évolution symbolique (Restriction 1 page 126). De plus, soit un nœud N_i de ce graphe d'évolution annoté par la contrainte $\mathbb{C}_i$, donnant les nœuds fils $N_{i1}, N_{i2}, ..., N_{in}$. D'après l'algorithme de construction du GES, chaque nœud fils N_{ij} est annoté de la contrainte $\mathbb{C}_{ij} = \mathbb{C}_i \cap \mathbb{C}_d$, où $\mathbb{C}_d$ est la contrainte spécifique du nœud N_{ij}. La contrainte $\mathbb{C}_0$ est donc transmise à chaque noeud fils N_{ij}. Les bornes numériques du noeud N_{ij} correspondent donc soit $\mathbb{C}_0$ soit à une contrainte restreinte par rapport à $\mathbb{C}_0$, on a donc : $\langle \mathbb{C}_0 \rangle \supseteq \langle \mathbb{C}_{NB_i} \rangle$. De plus, d'autres contraintes non numériques ont pu être ajoutées au noeud N_{ij} restreignant ainsi la contrainte sur les bornes numériques, on a donc : $\langle \mathbb{C}_{NB_i} \rangle \supseteq \langle cnstr(l_i) \rangle$ où l_i est la localisation associée au noeud N_{ij}.

Remarque 6.5 (Condition d'application)**.** *L'étape consistant à encadrer numériquement chaque paramètre n'est possible que si les expressions paramétriques forment des fonctions monotones. Etant donné que ces expressions paramétriques peuvent être non linéaires, elles peuvent engendrer des comportements non monotones. Nous nous restreignons donc*

à l'étude des THPN paramétriques qui ne possèdent qu'un seul type de paramètres symboliques (α, β, γ ou δ). En effet, il a été vu précédemment que ces THPN paramétriques engendraient des expressions paramétriques linéaires grâce auxquelles chaque paramètre peut être encadré numériquement.

Soit A_M^E le sp-ECA obtenu du p-ECA A_M en encadrant numériquement tous les paramètres étiquetant les arcs. L'automate A_M^E provenant de l'encadrement numérique de l'automate A_M de la Figure 6.8 est présenté sur la Figure 6.9. Prenons l'exemple de l'arc allant de la localisation l_0 à la localisation l_{14}. L'observation associée à cet arc ne comporte qu'un paramètre dans le p-atome $m = \alpha_1$. Etant donnée la borne numérique $\alpha_1 \in [0, 5]$ de la localisation l_0, on en déduit l'encadrement numérique $m \in [0, 5]$ qui étiquette ce même arc dans l'automate sans atome paramétrique.

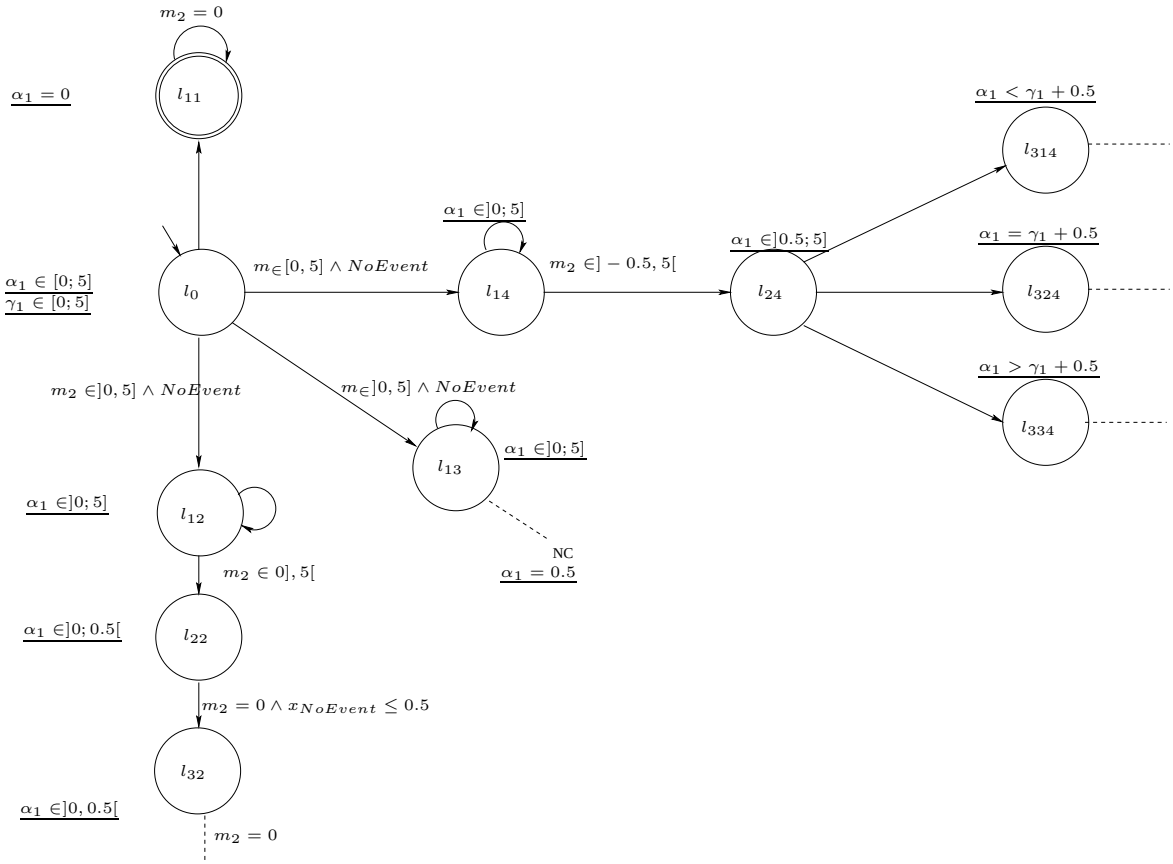

FIG. 6.9 – Automate Event-Clock sans atomes paramétriques associé au graphe d'évolution symbolique du THPN numéro 1.

Remarque 6.6. *On remarque que l'encadrement numérique des paramètres a pour conséquence l'élargissement du langage accepté par l'automate A_M^E. En effet, l'observation paramétrique $m_2 = \alpha_1$ où $\alpha_1 \in [0, 5]$ n'est pas équivalente à l'observation $m_2 \in [0, 5]$. Les contraintes associées aux arcs ont pour but de se souvenir de l'observation initiale. Ainsi, la contrainte associée à l'observation $m_2 = \alpha_1$ est justement $\mathbb{C} = \{m_2 = \alpha_1\}$. Cette*

contrainte n'est pas modifiée lors de l'encadrement numérique. L'arc est alors annoté de l'observation $m_2 \in [0,5]$ avec la contrainte $\{m_2 = \alpha_1\}$. Ainsi, même si la transformation en sp-ECA induit l'élargissement du langage, les contraintes imposées sur les arcs vont nous permettre de nous ramener à un langage restreint, composé de traces respectant ces contraintes.

6.4.4 Construction de l'automate associé à la propriété

Notre procédure de model-checking a pour but de répondre à la question suivante : "Pour quelles valeurs de paramètres le modèle THPN satisfait-il la propriété étudiée ϕ ?". Ainsi, les paramètres symboliques ne sont présents que dans le modèle THPN et non dans la propriété étudiée.

La construction de l'automate Event-Clock associé à la négation de la propriété ϕ reste donc similaire à celle présentée dans le cas numérique (Section 3.3 page 64). Néanmoins, étant donné que l'automate du modèle est un sp-ECA, l'automate Event-Clock $A_{\neg\phi}$ est transformé en sp-ECA. Cette transformation ne consiste qu'à rajouter des contraintes au niveau des localisations et des arcs :

- **Contraintes au niveau des localisations :** Pour chaque localisation l_i de l'automate $A_{\neg\phi} = (L, L_0, \mathbb{P}, \Sigma, \mathcal{C}, E, \mathcal{F}, cnstr)$, on associe la contrainte $\top$, c'est-à-dire $cnstr(l_i) = \top,\ \forall l_i \in L$.

- **Contraintes au niveau des arcs :** Les arcs de l'automate $A_{\neg\phi}$ sont étiquetés par des observations qui sont des conjonctions d'atomes. Ces observations sont "recopiées" sous forme de contraintes. Ainsi, pour tout arc $(l_i, \psi_i, l_j, \mathbb{C}_i)$, on a la contrainte $\mathbb{C}_i \equiv \psi_i$.

Prenons l'exemple d'une propriété très simple $\neg\phi \equiv \diamond(m_2 = 1)$. On obtient le sp-ECA de la Figure 6.10.

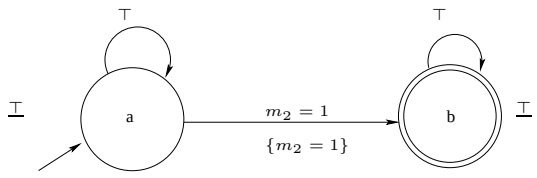

FIG. 6.10 – Automate Event-Clock sans atomes paramétriques de $\neg\phi \equiv \diamond(m_2 = 1)$ (Version simplifiée par rapport à l'algorithme de Schobbens [36]).

6.4.5 Produit des deux automates

Nous avons, à présent, deux automates Event-Clock sans atomes paramétriques : celui du modèle A_M^E et celui de la négation de la propriété $A_{\neg\phi}$. L'obtention des traces communes à ces deux automates est possible grâce au produit de ces deux automates. Etant donné

que le produit est effectué sur des sp-ECA et non plus sur des ECA, nous devons étendre la définition du produit-LT.

Définition 6.12 (Produit-LT étendu)**.** *Soient* $A_1 = (L^1, L_0^1, \mathbb{P}, \Sigma^1, \mathcal{C}^1, E^1, \mathcal{F}^1, cnstr_1)$ *et* $A_2 = (L^2, L_0^2, \mathbb{P}, \Sigma^2, \mathcal{C}^2, E^2, \mathcal{F}^2, cnstr_2)$ *deux sp-ECA. Le produit-LT étendu de* A_1 *et* A_2 *est le sp-ECA* $A_p = (L^p, L_0^p, \mathbb{P}, \Sigma^p, \mathcal{C}^p, E^p, \mathcal{F}^p, cnstr_p)$ *défini comme suit :*

- $L^p = L^1 \times L^2$, *et* $\forall (l_{i1}, l_{i2}) \in L^p,\ cnstr_p(l_{i1}, l_{i2}) = cnstr_1(l_{i1}) \wedge cnstr_2(l_{i2})$
- $L_0^p = L_0^1 \times L_0^2$,
- $\Sigma^p = \Sigma^1 \cup \Sigma^2$,
- $\mathcal{C}^p = \mathcal{C}^1 \cup \mathcal{C}^2$,
- $\mathcal{F}^p = \{(F_i^1 \times L^2) | F_i^1 \in \mathcal{F}^1\} \cup \{(L^1 \times F_j^2) | F_j^2 \in \mathcal{F}^2\}$,
- $E^p = \{((l_{i_1}, l_{j_1}), \psi_1 \wedge \psi_2, (l_{i_2}, l_{j_2}), \mathbb{C}_{c1} \wedge \mathbb{C}_{c2})$ *tel que* $(l_{i_1}, \psi_1, l_{i_2}, \mathbb{C}_{c1}) \in E^1$ *et* $(l_{j_1}, \psi_2, l_{j_2}, \mathbb{C}_{c2}) \in E^2\}$

Remarque 6.7. *La seule différence existant entre la Définition 3.9 page 69 du produit-LT avec celle du produit-LT étendu réside dans la conjonction des contraintes au niveau des arcs et des localisations. Or, nous avons montré que la définition du langage pour les sp-ECA (Définition 6.11) reste la même que dans le cas non paramétrique. Les propriétés associées au produit-LT s'étendent donc aussi au produit-LT étendu.*

Lorsque le produit-LT étendu est appliqué aux automates des Figures 6.9 et 6.10, nous obtenons l'automate produit A_p^E dont une portion est présentée sur la Figure 6.11 (seule la branche où $\alpha_1 \in]0, 0.5[$ est développée).

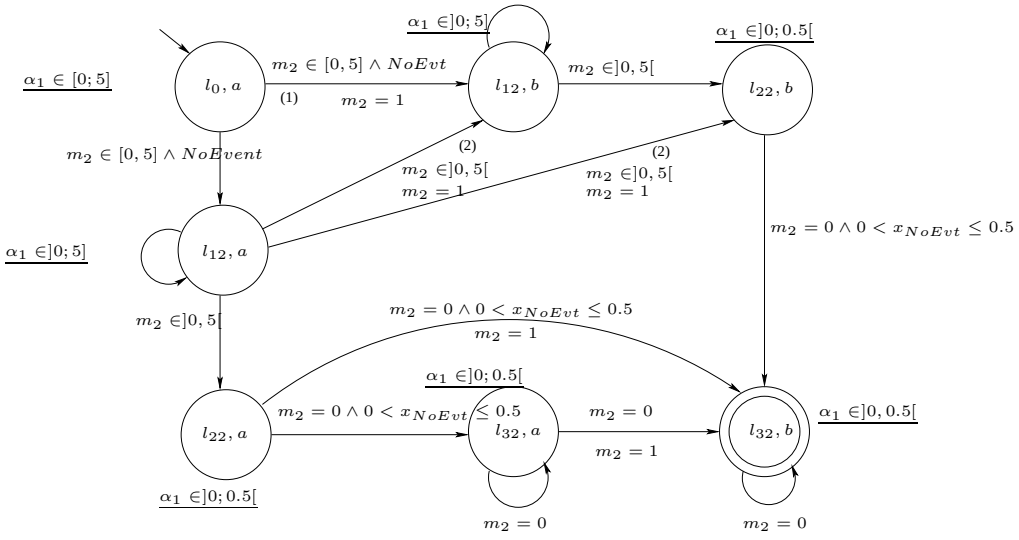

FIG. 6.11 – Portion de l'automate Event-Clock produit sans atomes paramétriques A_p^E.

6.4.6 Détermination du vide

Notre objectif est d'obtenir une contrainte $\mathbb{C}$ telle que pour toute valuation ν satisfaisant $\mathbb{C}$, le modèle THPN satisfait la propriété et pour toute valuation ν ne satisfaisant pas la contrainte $\mathbb{C}$, le modèle THPN ne satisfait pas la propriété.

Dans un cadre numérique, l'existence d'*une* trace dans l'automate produit nous permet de conclure à la non satisfaction de la propriété. Cette trace correspond à un contre exemple indiquant qu'il existe au moins un chemin commun entre le modèle biologique et la négation de la propriété étudiée.

Dans le cas paramétrique, il nous faut *toutes* les traces de l'automate produit, c'est-à-dire qu'il faut déterminer tous les chemins acceptants. Chaque chemin acceptant nous fournit une contrainte $\mathbb{C}_i$ telle que si une valuation ν satisfait $\mathbb{C}_i$, alors il existe un chemin commun à notre modèle et à la négation de la propriété. La propriété n'est donc pas satisfaite pour cette valuation. En possédant *toutes* les traces de l'automate produit, si une valuation ν satisfait l'union des contraintes $\mathbb{C}_i$, alors pour cette valuation, le langage de l'automate produit n'est pas vide et donc la propriété ϕ n'est pas satisfaite. Par contre, si la valuation ν ne satisfait aucune de ces contraintes $\mathbb{C}_i$ alors le langage de l'automate produit est vide pour cette valuation, la propriété ϕ est donc satisfaite. La procédure de décision du vide est donc modifiée par rapport au cas numérique. Détaillons les différentes étapes :

1. **Retour à un cas propositionnel :** Tout comme dans le cas numérique (Section 3.4.3.2 page 77), nous nous rapportons à un cas propositionnel en traitant les incohérences sémantiques pouvant exister au niveau des arcs. On rappelle que les arcs sont étiquetés par des observations $\psi_i \equiv \psi_{i1} \wedge \psi_{i2} \wedge \psi_{i3}$. La sous-observation ψ_{i1} est une conjonction de propositions $p \in Pr$. ψ_{i1} est donc incohérent s'il contient la sous-formule $p \wedge \neg p$. La sous-observation ψ_{i2} correspond à la conjonction des atomes instantanés de la forme $v \geq v'$ où $v, v' \in V \amalg \mathbb{R}$. ψ_{i2} est donc incohérent si le système d'équations linéaires associé à ψ_{i2} n'admet pas de solution. On se situe ici au niveau de l'évaluation des variables $v \in V$, on rappelle qu'il n'y a pas de paramètres dans les observations des sp-ECA.

   Dans l'automate produit A_p^E de la Figure 6.11, seuls les arcs allant de la localisation (l_{22}, a) à la localisation (l_{32}, b) et de la localisation (l_{32}, a) à la localisation (l_{32}, b) sont incohérents. En effet, il n'existe pas de valuations possibles de m_2 permettant de satisfaire $m_2 = 0$ et $m_2 = 1$. Ces arcs sont donc retirés de l'automate.

   Ces modifications nous permettent (comme dans le cas numérique) de nous rapporter à un automate propositionnel, sur lequel il est possible de construire l'automate de région.

2. **Construction de l'automate de région :** On obtient $R(A_p^E)$, l'automate de région associé à l'automate A_p^E. La construction de cet automate ne diffère pas de celle présentée dans le Chapitre 3 page 72. Les contraintes associées à la localisation l_i de l'automate A_p^E sont recopiées sur les états (l_i, β) de l'automate de région $R(A_p^E)$. De même, les contraintes associées à l'arc allant de la localisation l_i à la localisation l_j dans l'automate A_p^E sont recopiées sur les arcs allant d'un état (l_i, β_i) à un état (l_j, β_j) dans l'automate de région.

3. **Dépliage en arbre :** De façon à obtenir toutes les traces de l'automate de région, celui-ci est déplié sous-forme d'arbre. Chaque branche de l'arbre ainsi obtenue correspond à un chemin sur l'automate produit, caractérisé par une contrainte $\mathbb{C}$.

Détaillons la procédure de dépliage :

Procédure de dépliage en arbre :

- L'automate de région $R(A_p^E)$ est un automate de Büchi généralisé, c'est-à-dire qu'il possède un ensemble d'ensembles d'états acceptants. La première étape consiste à dégénéraliser l'automate, de façon à ce qu'il ne possède plus qu'un ensemble d'états acceptants. Cette procédure est usuelle et est rappelée dans [103].
- Partir de l'état initial s_0, s_0 est la racine de notre arbre.
- Pour chaque état successeur du nœud s_i, on crée une branche.
- Si s_i boucle sur lui-même, on garde la boucle.
- On arrête le dépliage dans deux cas :
  - On est arrivé à un état de deadlock,
  - On atteint un état s_j qui a déjà été visité dans cette même branche (coupure de la branche).
- Chaque branche est étiquetée par deux informations :
  - On garde l'information de la cause de l'arrêt du dépliage : deadlock ou coupure.
  - En cas de coupure, on garde l'information de la présence d'un état final dans la branche entre les deux états répétés responsables de la coupure. Par exemple, si on a $s_1, s_2, ..., s_1$, on coupe car on retrouve s_1, si s_2 est un état final, alors on annote la branche. On sait que la branche bouclera infiniment souvent sur l'état final s_2.

L'automate et le nombre d'arcs étant finis, on en déduit que le dépliage est nécessairement fini. En effet, si n est le nombre d'arcs de l'automate à déplier chaque branche a au plus une longueur de n, on arrive alors nécessairement soit à un état déjà visité soit à un état de deadlock.

Etant donné la taille de l'automate de région associé à l'automate produit A_p^E de notre exemple biologique, nous appliquons notre algorithme sur le sp-ECA et l'automate de région de la Figure 6.12. Une fois le dépliage effectué, on obtient l'arbre de la Figure 6.13.
L'arbre possède trois branches. La branche B_1 est arrêtée par coupure, mais celle-ci ne possède pas d'état final, contrairement à la branche B_2 qui a atteint l'état final $(l_3, x_a > 1, x_d = 0)$. Enfin, la branche B_3 est arrêtée par deadlock. Intuitivement, il est facile de voir sur l'exemple de la Figure 6.13 que seule la branche B_2 est un chemin acceptant de l'automate de région. Seule la branche B_2 peut engendrer des traces qui bouclent infiniment souvent sur un état final. On retient néanmoins que cette branche peut également engendrer des traces infinies bouclant un nombre fini de fois sur un état final. Ceci importe peu étant donné que nous recherchons toutes branches susceptibles de donner des traces acceptantes et que dans notre exemple seule la branche B_2 satisfait ce critère.

4. **Obtention des contraintes de chaque branche :** Nous définissons pour chaque branche trois ensembles de contraintes.
   - Chaque état (l_i, β_i) de l'automate de région est annoté de la contrainte $cnstr(l_i)$

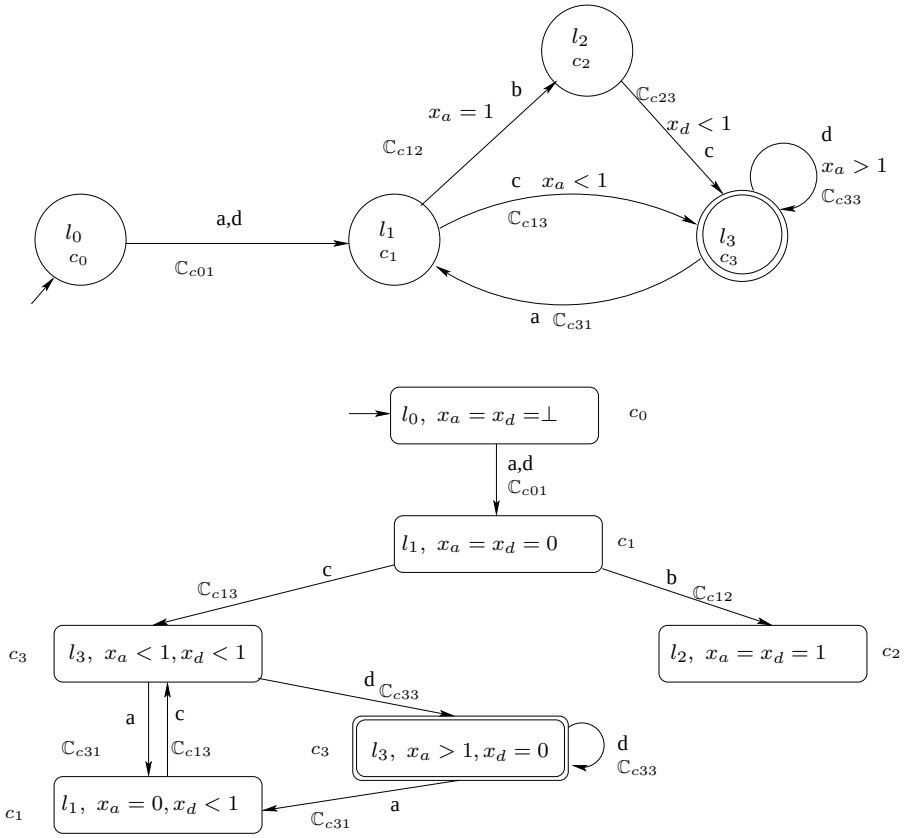

FIG. 6.12 – Exemple d'un sp-ECA et de son automate de région.

(Figure 6.12). Une branche B_j de l'arbre peut donc être annotée par la contrainte résultant de la conjonction des contraintes sur les états. Par exemple, la branche B_2 de la Figure 6.13 est annotée de la contrainte $c_0 \wedge c_1 \wedge c_3$. Formellement, cette contrainte sur les localisations, notée $\mathbb{C}_L(B_j)$ est définie comme suit :

Définition 6.13. *Pour chaque branche B_i de l'arbre résultant du dépliage d'un automate de région $R(A)$, on note $\mathbb{C}_L(B_i)$ la contrainte résultant de la conjonction des contraintes sur les localisations de la branche. $\langle\mathbb{C}_L(B_i)\rangle$ correspond à l'ensemble de valuations ν des paramètres permettant de satisfaire la contrainte $\mathbb{C}_L(B_i)$.*

- Chaque état de l'automate de région fournit un ensemble de contraintes sur les horloges du modèle. Par exemple, dans la Figure 6.13, l'état $(l_3, x_a < 1, x_d < 1)$ conduit à la contrainte $c_r \equiv (x_a < 1) \wedge (x_d < 1)$. Bien que ces contraintes soient placées au niveau des états de l'automate de région, elles concernent en réalité les arcs, car ce sont eux qui temporisent l'automate. Ainsi, la contrainte $c_r \equiv (x_a < 1) \wedge (x_d < 1)$ de l'état $(l_3, x_a < 1, x_d < 1)$ concerne l'arc entrant dans cet état.

- Les arcs de l'automate de région sont également annotés d'une autre contrainte.

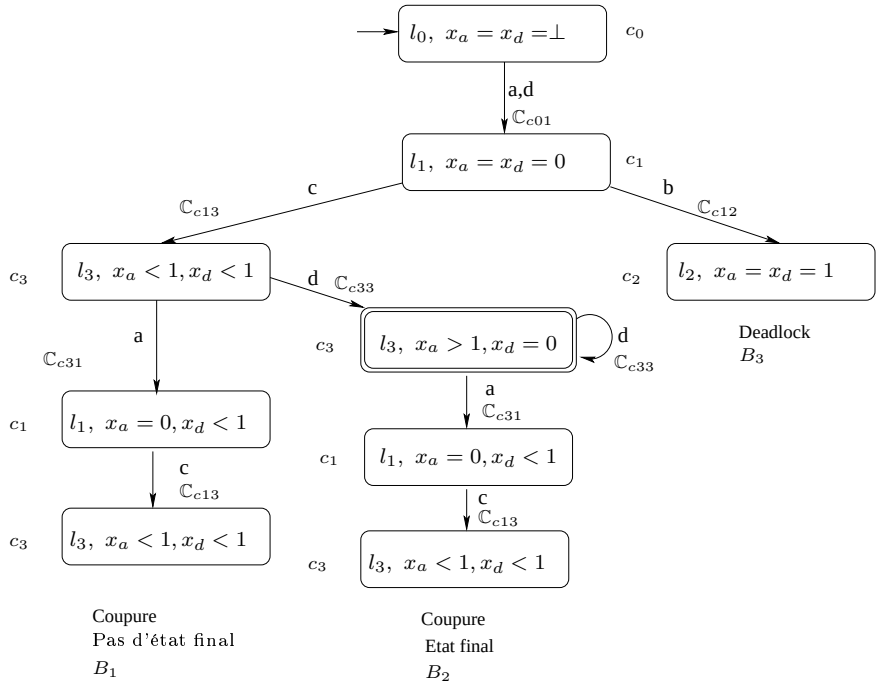

FIG. 6.13 – Dépliage de l'automate de région en arbre.

En effet, tout arc de l'arbre allant de l'état (l_i, β_i) à l'état (l_{i+1}, β_{i+1}) est annoté de la contrainte $\mathbb{C}_{ci}$ qui étiquette l'arc allant de la localisation l_i à la localisation l_{i+1} dans l'automate produit A_p^E. Ainsi, l'arc entrant dans l'état $(l_3, x_a < 1, x_d < 1)$ de la Figure 6.13 est étiqueté par deux types de contraintes :
∘ la contrainte associée à la région $c_r \equiv (x_a < 1) \wedge (x_d < 1)$,
∘ et la contrainte associée à l'arc $\mathbb{C}_{c13}$.
Nous considérons alors que cet arc e_j entrant dans l'état $(l_3, x_a < 1, x_d < 1)$ est étiqueté par une contrainte c_{aj} correspondant à la conjonction des deux contraintes citées ci-dessus. On a alors $c_{aj} \equiv ((x_a < 1) \wedge (x_d < 1)) \wedge \mathbb{C}_{c13}$. La définition suivante présente dans un cadre générale les contraintes c_{aj} des arcs de l'arbre.

Définition 6.14. *Chaque arc e_j de l'arbre résultant du dépliage d'un automate de région $R(A)$ est annoté d'une contrainte c_{aj} résultant de la conjonction des contraintes associées aux régions c_{rj} et celles associées à l'arc de l'automate $\mathbb{C}_{c_j}$. Ainsi, $\langle c_{aj} \rangle$ correspond à l'ensemble de valuations ν de paramètres permettant de satisfaire la contrainte c_{aj} de l'arc e_j.*

On peut généraliser la définition précédente à tous les arcs d'une branche de l'arbre :

Définition 6.15. *Soit B_i, une branche de l'arbre résultant du dépliage d'un automate de région $R(A)$. Soit $\langle \mathbb{C}_a(B_i) \rangle$ l'ensemble des valuations ν de paramètres permettant de satisfaire l'ensemble des contraintes c_{aj} des arcs e_j de la branche B_i. Ceci revient à écrire : $\langle \mathbb{C}_a(B_i) \rangle = \bigcap \langle c_{aj} \rangle$, pour tous les arcs e_j de la branche*

B_i.

Il est à présent possible de généraliser à toute la branche :

Définition 6.16. *Chaque branche B_i de l'arbre résultant du dépliage d'un automate de région $R(A)$ se définit par un ensemble de valuations de paramètres $\langle \mathbb{C}(B_i) \rangle$ correspondant à l'intersection des valuations possibles sur les arcs $\langle \mathbb{C}_a(B_i) \rangle$ avec les valuations possibles des états $\langle \mathbb{C}_L(B_i) \rangle$. On a alors $\langle \mathbb{C}(B_i) \rangle = \langle \mathbb{C}_a(B_i) \rangle \cap \langle \mathbb{C}_L(B_i) \rangle$.*

Nous allons montrer que chaque branche B_i de l'arbre possédant au moins un état final et ayant été arrêtée par coupure est un préfixe d'un chemin acceptant. Chacune de ces branches B_i fournit un ensemble de valuations $\langle \mathbb{C}(B_i) \rangle$ et nous montrerons que pour toute valuation ν de $\langle \mathbb{C}(B_i) \rangle$, le modèle THPN ne satisfait pas la propriété étudiée étant donné que pour cette valuation, il existe un chemin acceptant dans l'automate de région. Ces lemmes et théorèmes seront énoncés formellement dans la section 6.5. Dans un premier temps, nous allons illustrer ces idées sur un exemple.

Reprenons la portion de l'automate produit A_p^E présentée dans la Figure 6.14, dans laquelle nous avons retiré les arcs incohérents formés de l'observation $m_2 = 0 \wedge m_2 = 1$. Concentrons nous, dans un premier temps, sur les contraintes associées aux arcs. L'arc allant de la localisation (l_0, a) à la localisation (l_{12}, b) possède la contrainte $\mathbb{C}_{c0} \equiv \{m_2 = \alpha_1 \wedge NoEvent \wedge m_2 = 1\}$. Il est facile de voir que l'ensemble de valuations de paramètres permettant de satisfaire $\mathbb{C}_{c0}$ est restreint à $\alpha_1 = 1$, noté (1) sur la Figure 6.14. La contrainte associée à l'arc allant de la localisation (l_{12}, b) à la localisation (l_{22}, b) s'écrit $m_2 \in]0, \alpha_1[$. Ainsi, quelle que soit la valuation de α_1, la contrainte $m_2 \in]0, \alpha_1[$ reste cohérente.

Nous procédons de même pour tous les arcs de l'automate de la Figure 6.14 et nous obtenons les deux ensembles de valuations restreints annotés (1) et (2) sur la Figure 6.14 :

- (1) : $\alpha_1 = 1$
- (2) : $\alpha_1 > 1$

L'étape de construction de l'automate de région n'est pas présentée ici à cause de la taille de ce dernier. Néanmoins, étant donné que la propriété étudiée $\neg\phi \equiv \diamond(m_2 = 1)$ n'utilise pas d'opérateurs temps réel, la seule contrainte d'horloge apparaissant dans l'automate ($x_{NoEvent} \leq 0.5$) ne peut pas entraîner d'incohérence temporelle. En effet, une incohérence temporelle ne peut avoir lieu que si plusieurs contraintes temporelles coexistent mais ne possèdent pas solutions communes. Prenons le cas de deux arcs successifs étiquetés respectivement par $y_a > 1 \wedge y_b < 1$ et par $a \wedge b$. Il est impossible que le délai à attendre avant le prochain a soit supérieur à 1 et que celui à attendre avant le prochain b soit inférieur à 1. Dans notre cas, la présence de l'unique contrainte temporelle ($x_{NoEvent} \leq 0.5$) ne fait qu'imposer un délai inférieur à 0.5 depuis l'observation du dernier $NoEvent$, sans entraîner d'incohérences temporelles.

Nous pouvons donc déplier l'automate produit A_p^E et non son automate de région. Nous obtenons l'arbre de la Figure 6.15.

Dans un cadre non paramétrique, le langage de A_p^E n'est pas vide. Le dépliage en arbre nous indique qu'il y a trois chemins acceptantes notés γ_1, γ_2 et γ_3 sur la Figure 6.15.

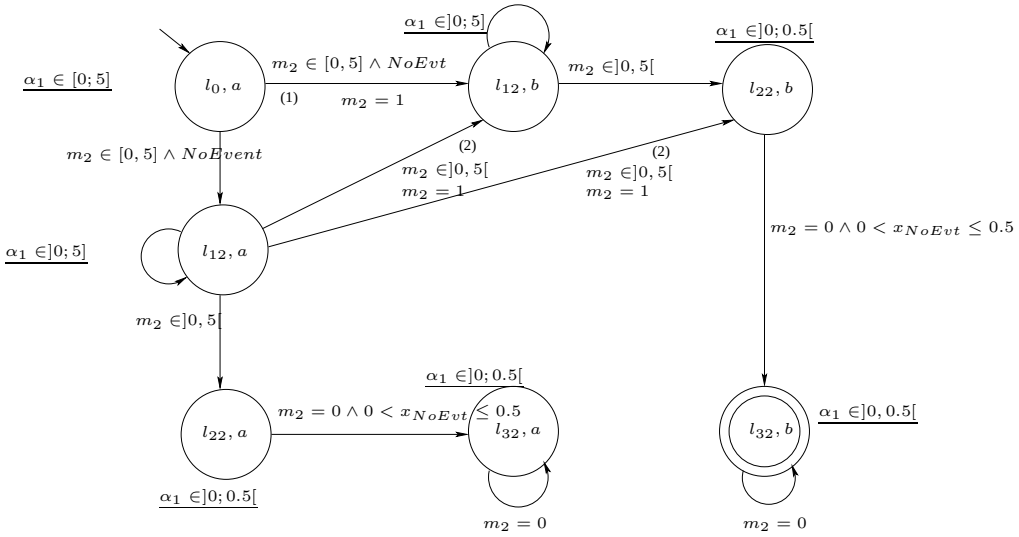

FIG. 6.14 – Portion du sp-ECA produit A_p^E.

Etudions les contraintes associées à chacune de ces branches. Pour les trois branches, la contrainte associée aux localisations vaut $\alpha_1 \in]0, 0.5[$, c'est-à-dire $\langle \mathbb{C}_L(B_1) \rangle = \langle \mathbb{C}_L(B_2) \rangle = \langle \mathbb{C}_L(B_3) \rangle = \{\alpha_1 \in]0, 0.5[\}$.

Les valuations de paramètres possibles pour les arcs des branches B_1 et B_2 sont $\langle \mathbb{C}_a(B_1) \rangle = \langle \mathbb{C}_a(B_2) \rangle = \{\alpha_1 > 1\}$ et celles de la branches B_3 sont $\langle \mathbb{C}_a(B_3) \rangle = \{\alpha_1 = 1\}$.

On en déduit que $\langle C(B_i) \rangle = \langle C_L(B_i) \rangle \cap \langle C_a(B_i) \rangle = \emptyset$ pour les 3 branches B_1, B_2 et B_3. Ainsi, pour toute valuation ν satisfaisant les contraintes sur les localisations $\alpha_1 \in]0, 0.5[$, le langage de l'automate produit est vide, ce qui nous permet de conclure que le modèle THPN satisfait la propriété pour ces valuations ν.

La section suivante présente formellement toutes les propriétés dont nous venons de donner l'idée.

6.5 Propriétés

Dans la section ci-dessus, il a été précisé que chaque branche B_i de l'arbre résultant du dépliage de l'automate de région $R(A_p^E)$ est annotée de deux informations :

- L'information de la cause de l'arrêt du dépliage : deadlock ou coupure,
- En cas d'arrêt par coupure, l'information de la présence ou non d'un état final entre les deux états de coupure.

Nous allons montrer qu'une branche arrêtée par coupure et possédant un état final correspond à un chemin acceptant sur l'automate d'origine A_p^E (Lemme 6.4). De plus, nous montrerons que tout chemin acceptant de l'automate passe nécessairement par une de ces branches (Lemme 6.5).

Définition 6.17 (Branche cons)**.** *Une branche B_i est annotée de l'étiquette* cons *(pour consistent en anglais) si elle a été arrêtée par coupure au niveau d'un état s_i et que entre*

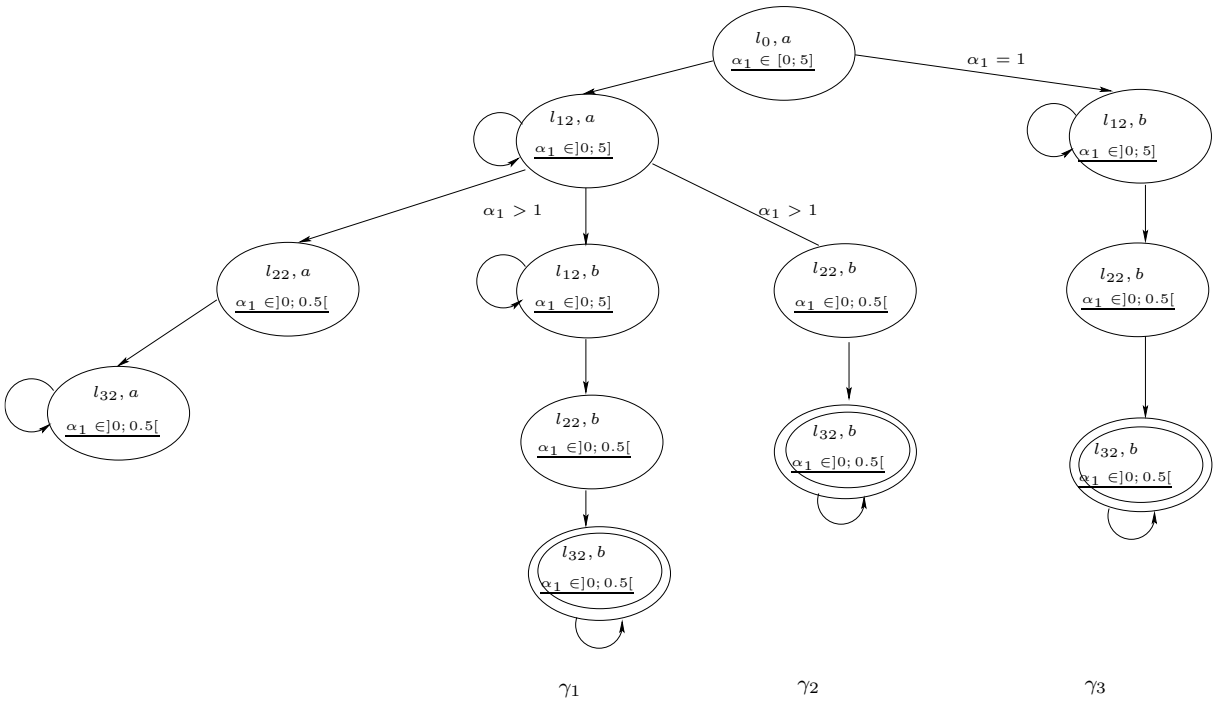

FIG. 6.15 – Dépliage de la portion de l'automate produit A_p^E.

les deux occurrences de l'état s_i, il y a un état final.

Lemme 6.4. *Une branche* cons *de l'arbre résultant du dépliage de l'automate de région $R(A_p^E)$ correspond à un chemin acceptant sur A_p^E.*

Preuve :

La notation *cons* indique entre autre que la coupure de l'arbre a eu lieu lors de la visite d'un état s_i qui avait déjà été visité dans la même branche. L'arbre découlant de l'état s_i revisité est donc déjà déplié en amont, ce qui permet d'assurer que la branche boucle infiniment souvent. De plus, la notation *cons* indique qu'il existe au moins un état final entre les deux états s_i, étant donné que l'on manipule un automate de Büchi, il est possible de boucler infiniment souvent sur un des états finaux même si cette branche peut également donner des traces infinies qui bouclent un nombre fini de fois sur un état final.

Une branche *cons* indique donc l'existence d'au moins un chemin acceptant sur A_p^E.

□

Lemme 6.5. *Tout chemin acceptant de l'automate produit A_p^E passe nécessairement par une branche* cons *de l'arbre résultant du dépliage de l'automate de région $R(A_p^E)$.*

Preuve :

Chaque branche est soit stoppée par deadlock soit arrêtée par bouclage, c'est-à-dire que la poursuite du dépliage engendrerait un sous-arbre qui a déjà été déplié en amont. Considérons deux branches B_1 et B_2^{cons} ayant un nœud commun s_i. Supposons que B_1 boucle infiniment souvent sur s_i. Il existe donc un chemin passant un certain nombre de fois par B_1 puis par B_2^{cons}. Pour que ce chemin soit acceptant, il faut nécessairement

passer par B_2^{cons} qui permet de boucler infiniment souvent pas un état final (condition d'acceptation de Büchi).
□

Chaque branche B_i^{cons} fournit un ensemble de valuations de paramètres $\langle \mathbb{C}(B_i^{cons}) \rangle$. Notre objectif est de montrer que si une valuation ν appartient à $\langle \mathbb{C}(B_i^{cons}) \rangle$, alors le langage de l'automate produit pour cette valuation n'est pas vide (Théorème 6.2). A l'inverse, si une valuation ν n'appartient à aucun ensemble $\langle \mathbb{C}(B_i^{cons}) \rangle$, alors le langage de l'automate produit pour cette valuation est vide (Théorème 6.3).

Par la suite, on notera GE_ν le graphe d'évolution du graphe d'évolution symbolique tel que la valuation ν satisfait les contraintes associées à ce graphe d'évolution $\mathbb{C}_{GE_\nu}$. On rappelle que le Théorème 6.1 montre qu'il existe un unique graphe d'évolution GE_ν tel que ν satisfait les contraintes de GE_ν, $\mathbb{C}_{GE_\nu}$. L'ensemble des traces du graphe d'évolution GE_ν, noté TGE_ν est converti en un sp-ECA A_M^ν.

Les lemmes 6.6 à 6.8 montrent que les propriétés démontrées dans un cadre numérique restent vraies dans un cadre paramétrique où on ne considère qu'une valuation ν.

Lemme 6.6. *Soit A un sp-ECA et A' l'automate Event-Clock associé à A dans lequel il n'y a ni contraintes sur les arcs ni contraintes sur les localisations. Le langage du sp-ECA A est exactement le langage de l'ECA A' qui lui est associé.*

Preuve : La définition de langage (Définition 6.11) ne fait pas intervenir les contraintes supplémentaires qui apparaissent dans le sp-ECA.
□

Lemme 6.7. *Soit TGE_ν, l'ensemble des traces du graphe d'évolution GE_ν extrait du graphe d'évolution symbolique et A_M^ν, le sp-ECA résultant de la conversion de TGE_ν. On a comme dans le cas numérique égalité des langages, c'est-à-dire $TGE_\nu = \mathcal{L}(A_M^\nu)$.*

Preuve : On a montré que le graphe d'évolution GE_ν du graphe d'évolution symbolique était exactement le graphe d'évolution GE obtenu dans un cadre non paramétrique lorsque tous les paramètres du THPN $p_i \in \mathbb{P}$ ont été remplacés par $\nu(p_i)$ (Théorème 6.1), on a donc $TGE_\nu = TGE$, où TGE correspond à l'ensemble des traces du graphe d'évolution GE. De plus, on a montré que dans un cadre non paramétrique, $TGE = \mathcal{L}(A_M)$. Enfin, le lemme 6.6 montre que $\mathcal{L}(A_M) = \mathcal{L}(A_M^\nu)$.
□

Lemme 6.8. $M_\nu \models \phi \Leftrightarrow \mathcal{L}(A_p^\nu) = \emptyset$

Preuve : Nous avons montré ce théorème dans un cadre non paramétrique. De plus, avec le lemme 6.6, on montre que la propriété reste vérifiée.
□

Théorème 6.2. *Soit B_i^{cons}, une branche* cons *de l'arbre résultant du dépliage de l'automate de région $R(A_p^E)$, caractérisée par l'ensemble de valuations $\langle \mathbb{C}(B_j^{cons}) \rangle = \langle \mathbb{C}_L(B_j^{cons}) \rangle \cap \langle \mathbb{C}_a(B_j^{cons}) \rangle$. Si une valuation ν appartient à $\langle \mathbb{C}(B_j^{cons}) \rangle$ alors $\mathcal{L}(A_p^\nu) \neq \emptyset$.*

Preuve :

D'après le lemme 6.4, B_i^{cons} correspond à un chemin acceptant γ sur A_p^E :

$$\gamma = (l_0, l_0^2) \xrightarrow[\mathbb{C}_{c0}]{\psi_0} (l_1, l_1^2) \xrightarrow[\mathbb{C}_{c1}]{\psi_1} ...(l_n, l_n^2) \xrightarrow[\mathbb{C}_{cn}]{\psi_n} ...$$

On en déduit les deux chemins γ_1 et γ_2 acceptants respectivement sur A_M^E et $A_{\neg\phi}$:

$$\gamma_1 = l_0 \xrightarrow[\mathbb{C}_{c0}^1]{\psi_0^1} l_1 \xrightarrow[\mathbb{C}_{c1}^1]{\psi_1^1} ...l_n \xrightarrow[\mathbb{C}_{cn}^1]{\psi_n^1} ...$$

et

$$\gamma_2 = l_0^2 \xrightarrow[\mathbb{C}_{c0}^2]{\psi_0^2} l_1^2 \xrightarrow[\mathbb{C}_{c1}^2]{\psi_1^2} ...l_n^2 \xrightarrow[\mathbb{C}_{cn}^2]{\psi_n^2} ...$$

Etant donné que la valuation $\nu \in \langle \mathbb{C}_L(B_i^{cons}) \rangle$, on en déduit qu'il existe un chemin sur A_M^ν, noté γ_1' (Théorème 6.1) :

$$\gamma_1' = l_0 \xrightarrow[\mathbb{C}_{c0}^1]{\psi_0'^1} l_1 \xrightarrow[\mathbb{C}_{c1}^1]{\psi_1'^1} ...l_n \xrightarrow[\mathbb{C}_{cn}^1]{\psi_n'^1} ...$$

Soit γ' le chemin résultant du produit de γ_1' et de γ_2.

$$\gamma' = (l_0, l_0^2) \xrightarrow[\mathbb{C}_{c0}]{\psi_0'^1 \wedge \psi_0^2} (l_1, l_1^2) \xrightarrow[\mathbb{C}_{c1}]{\psi_1'^1 \wedge \psi_1^2} ...(l_n, l_n^2) \xrightarrow[\mathbb{C}_{cn}]{\psi_n} ...$$

Montrons que les $\psi_i' \equiv \psi_i'^1 \wedge \psi_i^2$ sont cohérents quel que soit i, c'est-à-dire qu'il existe une valuation des variables $v \in V$ permettant de satisfaire les observations étiquetant les arcs de l'automate A_M^ν (On rappellle que la notion de cohérence mentionnée se rapporte à celle mentionnée dans le Chapitre 3 page 77). D'après la notation 2, l'observation ψ_i' s'écrit sous la forme de trois sous-observations $\psi_i' \equiv \psi_{i1}' \wedge \psi_{i2}' \wedge \psi_{i3}'$, où $\psi_{i1}' \wedge \psi_{i2}'$ sont des conjonctions d'atomes instantanés et ψ_{i3}' correspondent aux contraintes d'horloge.

$\nu \in \langle \mathbb{C}_a(B_j^{cons}) \rangle$ donc pour tout arc e_i de la branche B_j^{cons}, $\nu \in \langle \mathbb{C}_{ci} \rangle$ et $\nu \in \langle c_{ri} \rangle$. On rappelle que les $\mathbb{C}_{ci}$ correspondent aux contraintes étiquetant les arcs de l'automate et que les c_{ri} correspondent aux contraintes associées aux régions de l'automate de région (cf. page 146).

- On sait que ν satisfait les contraintes associées à chaque arc e_i : $\nu \in \langle \mathbb{C}_{ci} \rangle$. Par conséquent, la contrainte $\mathbb{C}_{Ci}$ évaluée par ν, notée $\mathbb{C}_{ci}[\nu]$, est une contrainte satisfiable. Etant donné que les contraintes associées aux arcs $\mathbb{C}_{ci}$ correspondent exactement aux observations $\psi_{i1}' \wedge \psi_{i2}'$ des arcs (d'après l'algorithme de construction de l'automate p-ECA). On en déduit que la conjonction d'atomes instantanés $\psi_{i1}' \wedge \psi_{i2}'$ sont satisfiables quel que soit i.

- De même, ν satisfait les contraintes des régions de l'automate de région : $\nu \in c_{ri}$, ainsi $c_{ri}[\nu]$ est satisfiable. Etant donné que ν appartient aussi à $\mathbb{C}_{ci}$, on en déduit que les contraintes d'horloges de $\mathbb{C}_{ci}$ sont également satisfiables. Etant donné que les observations sur les arcs correspondent exactement aux contraintes étiquetant ces arcs, on en déduit que les contraintes d'horloge ψ_{i3}' sont satisfiables quel que soit i.

On en déduit que γ' est un chemin acceptant sur A_p^ν, c'est-à-dire $\mathcal{L}(A_p^\nu) \neq \emptyset$.
□

On déduit du lemme 6.8 et du théorème 6.2 le corollaire suivant :

Corollaire 6.1. *Soit ν une valuation et ϕ la propriété étudiée, $\nu \in \bigcup\langle\mathbb{C}(B_j^{cons})\rangle \Rightarrow M_\nu \not\models \phi$*

Théorème 6.3. *Soit ν une valuation, pour toute branche B_j^{cons} de l'arbre résultant du dépliage de l'automate de région $R(A_p^E)$ étiquetée par l'ensemble de valuations $\langle\mathbb{C}(B_j^{cons})\rangle = \langle\mathbb{C}_L(B_j^{cons})\rangle \cap \langle\mathbb{C}_a(B_j^{cons})\rangle$:*

$$\nu \in \langle\mathbb{C}_L(B_j^{cons})\rangle \text{ et } \nu \notin \bigcup\langle\mathbb{C}(B_j^{cons})\rangle \Rightarrow M_\nu \models \phi$$

Preuve :

$\nu \in \langle C_L(B_i^{cons})\rangle$, donc on s'intéresse à l'automate A_M^ν.

Tout chemin acceptant tiré des branches B_i^{cons} est de la forme :

$$\gamma = (l_0, l_0^2) \xrightarrow[\mathbb{C}_{c0}]{\psi_0} (l_1, l_1^2) \xrightarrow[\mathbb{C}_{c1}]{\psi_1} ...(l_n, l_n^2) \xrightarrow[\mathbb{C}_{cn}]{\psi_n} ...$$

$\nu \in \langle\mathbb{C}_L(B_i^{cons})\rangle$ et $\nu \notin \bigcup\langle\mathbb{C}(B_i^{cons})\rangle$, on en déduit que $\nu \notin \langle\mathbb{C}_a(B_i^{cons})\rangle$. On rappelle que $\mathbb{C}_a(B_i^{cons})$ correspond à la contrainte associée aux arcs de la branche B_i^{cons} (Définition 6.15).

Pour cette branche B_i^{cons}, il existe au moins un arc e_i tel que soit $\nu \notin \langle\mathbb{C}_{ci}\rangle$ soit $\nu \notin \langle c_{ri}\rangle$.

Si $\nu \notin \langle\mathbb{C}_{ci}\rangle$, la contrainte $\mathbb{C}_{ci}$ évaluée par ν, $\mathbb{C}_{ci}[\nu]$, n'est pas satisfaite. Etant donné que par construction de l'automate, la contrainte $\mathbb{C}_{ci}$ correspond exactement aux observations $\psi_{i1} \wedge \psi_{i1}$, on en déduit que la conjonction d'atomes instantanés $\psi_{i1} \wedge \psi_{i2}$ n'est pas cohérent, c'est-à-dire qu'il n'y a pas de valuations possibles des variables $v \in V$ permettant de rendre cohérentes les observations sur les arcs de l'automate A_M^ν.

Si $\nu \notin \langle c_{ri}\rangle$ ou que $\nu \notin \langle\mathbb{C}_{ci}\rangle$, ν ne permet pas aux contraintes d'horloge ψ_{i3} d'être satisfiables.

Par conséquent, aucun chemin de A_p^E ne peut être acceptant dans A_p^ν.

Peut-il exister d'autres chemins acceptants non représentés dans l'arbre ?

Le lemme 6.5 montre que tout chemin acceptant de A_p^E passe nécessairement par une branche B_i^{cons}. Ainsi, considérons 2 branches B_1 et B_2^{cons}. Seule la branche 2 est de type *cons*. Supposons qu'il existe un chemin passant d'abord par B_1 puis par B_2^{cons}, celle-ci est caractérisée par l'ensemble de valuations $\langle\mathbb{C}(B_1)\rangle \cap \langle\mathbb{C}(B_2^{cons})\rangle$. Ce chemin n'est pas explicitement détaillé dans l'arbre. Néanmoins, si $\nu \notin \mathbb{C}(B_2^{cons})$, $\nu \notin \langle\mathbb{C}(B_1)\rangle \cap \langle\mathbb{C}(B_2^{cons})\rangle$, par conséquent, le chemin passant d'abord par B_1 puis par B_2^{cons} ne peut pas donner un chemin acceptant sur A_p^ν.
□

6.6 Bilan

Ce chapitre a présenté une extension paramétrique de notre procédure de model-checking. Cette extension nous permet de définir certains paramètres biologiques inconnus sous forme symbolique. Grâce à la connaissance de propriétés comportementales, on peut à présent déterminer *in silico* la ou les valeurs des paramètres inconnus permettant au modèle biologique de satisfaire ces propriétés. Ces nouvelles données *in silico* pourront alors être validées ou précisées expérimentalement.

Conclusion

Dans le cadre de la modélisation formelle des réseaux de régulation biologiques, deux grandes tendances coexistent : celle qui privilégie le pouvoir d'expression du formalisme en vue de simulation mais ne permet pas d'effectuer des preuves formelles, et celle qui permet l'application de preuves mais après de fortes abstractions dans les modèles.

Ce manuscrit présente un compromis entre pouvoir d'expression et capacité de preuves formelles. Nous nous sommes concentrés sur les THPN : une sous-classe de HFPN, à l'heure actuelle l'un des formalismes les plus puissants pour la biologie des systèmes parmi ceux fournissant des simulations automatisées. L'originalité du travail détaillé dans ce manuscrit est liée à l'association de deux méthodes venant de deux communautés différentes de l'informatique. Nous avons su associer la méthodologie de construction d'un graphe d'évolution venant de la communauté des réseaux de Petri [34] aux méthodes de manipulation d'automates temps-réel venant de la communauté des systèmes formels temps-réels [37, 36]. Cette combinaison nous a alors permis d'étendre les deux théories existantes, soit en ajoutant une notion quantitative par des atomes spécifiques (atomes instantanés) et un produit spécifique (produit-LT), soit en autorisant des définitions symboliques de certains paramètres aussi bien dans le graphe d'évolution que dans les automates temps-réel utilisés. Autoriser la preuve formelle sur un formalisme aussi expressif que les THPN a donc nécessité la définition précise et détaillée d'un cadre logique.

Il n'aurait pas été possible d'appliquer cette méthode à des exemples biologiques non triviaux sans une plate-forme logicielle adaptée. Nous avons donc développé le logiciel *BioPetri* dans lequel chaque étape de notre procédure de model-checking pour THPN et formules CTEL a été implémentée.

Un point fort de ce travail correspond à l'extension paramétrique de notre procédure de model-checking. Cette extension permet la manipulation symbolique des paramètres inconnus. Ainsi, étant données des propriétés biologiques supposées vraies, il est à présent possible de déterminer les valeurs des paramètres inconnus permettant au modèle THPN paramétrique de satisfaire ces propriétés. Cette méthode permet ainsi d'acquérir de nouvelles données sur le modèle biologique étudié, ou d'évaluer les valeurs de paramètres qui satisfont une hypothèse biologique donnée.

Enfin, une proche collaboration avec Nicolas Pollet du laboratoire de Développement et Evolution de Paris Sud nous a permis d'étudier finement les mécanismes moléculaires HT-dépendants, responsables des modifications morphologiques, lors de la métamorphose du têtard *Xenopus tropicalis*. Nous nous sommes particulièrement intéressés à deux chan-

gements opposés : la pousse des pattes arrières et la résorption de la queue. De nos modèles *in silico*, ont découlé de nouvelles hypothèses et questions biologiques, concernant essentiellement le rôle des enzymes métabolisant les hormones thyroïdiennes. De façon à valider ou réfuter ces nouvelles hypothèses, le laboratoire de Développement et Evolution a mis en place un modèle de cultures cellulaires *in vitro* qui correspond à un intermédiaire indispensable entre les modèles *in silico* et *in vivo*.

Plusieurs perspectives s'ouvrent à l'issue de ce travail.

- Etude de la métamorphose

  L'élaboration d'un modèle cellulaire *in vitro* va nous permettre à court terme de tester expérimentalement les questions et hypothèses soulevées par le modèle *in silico*. Nous avons ainsi effectué *in silico* des pulses courts qui n'ont pas permis de déclencher l'apoptose dans les cellules de la queue (voir page 106). Il est important de valider ou de réfuter ce résultat expérimentalement. Alors qu'une expérience de pulse peut s'avérer difficilement contrôlable dans un contexte *in vivo* impliquant un animal ou un organe entier, celle-ci se trouve facilitée au sein d'un modèle *in vitro* de cultures cellulaires (possibilité de contrôle de la concentration intra-cellulaire, possibilité d'utiliser des colorants,...). Notre collaboration avec le laboratoire de Développement et Evolution se définit donc par un aller-retour entre les modèles *in silico* et *in vitro*, qui ne peut s'avérer que fructueux dans l'avenir.

- Inférence de THPN

  Dans la dernière partie du chapitre biologique, nous commençons à nous intéresser à la construction de réseaux de régulation d'après des données transcriptionnelles. Il serait intéressant, à long terme, de développer des méthodes d'inférence de réseaux de régulation modélisés par THPN. A l'heure actuelle, il existe différentes méthodes de construction de réseaux dans la littérature qui diffèrent principalement dans la condition dictant quand un gène est relié à un autre [104, 105]. Il serait donc pertinent de coupler ce type de méthode à notre formalisme de modélisation. Les réseaux ainsi inférés donneraient la structure d'un modèle THPN qui serait étudié suivant la méthodologie décrite dans ce manuscrit.

- Aspect paramétrique

  Le dernier chapitre de ce manuscrit présente une extension paramétrique de notre procédure de model-checking. Nous envisageons, dans un avenir proche, d'appliquer cette théorie à des exemples biologiques pertinents. Ainsi, nous pourrions, par exemple, déterminer les valeurs de l'activité de l'enzyme D2 permettant à la boucle positive D2 qui lui est associée d'être active dans la queue. Cette valeur *in silico* serait alors comparée aux valeurs expérimentales de Germain *et al.* dans [84].

- Extension fonctionnelle

Dans ce manuscrit, nous nous sommes orientés vers les THPN, une sous-classe des HFPN dans laquelle l'aspect fonctionnel a été retiré. Nous avons commencé à nous intéresser à la réintroduction de certaines fonctions. Les réseaux de Petri variables, notés VHPN [34] autorisent la définition des vitesses des transitions continues sous forme de fonctions dépendant du marquage continu. Ce formalisme autorise des relations de proportionnalité. Par contre, son étude est beaucoup plus complexe étant donnée l'évolution exponentielle du marquage continu, induite par de telles vitesses. David et Alla ont alors étudié le comportement asymptotique des VHPN, il s'agit des réseaux de Petri hybrides asymptotiques (AHPN). Ces AHPN permettent de se rapporter à des vitesses constantes et donc de construire un graphe d'évolution tout en gardant une relation de proportionnalité au niveau des vitesses. Néanmoins, ce formalisme induit, par son étude asymptotique, des approximations. Celles-ci font perdre la complétude et la correction de notre procédure de model-checking. En effet, certaines propriétés pourraient être satisfaites par le modèle approximé mais pas par le modèle réel et inversement.
Deux choix s'offrent à nous dans l'avenir. Nous pouvons déterminer une catégorie de propriétés qui sont vraies dans le modèle approximé si et seulement si elles le sont dans le modèle réel. Nous pouvons également nous orienter vers l'exploitation des VHPN et non des AHPN, ce qui obligera à modifier le cadre formel de façon à prendre en compte des équations différentielles.

Ce manuscrit présente une méthode pertinente pour la vérification des THPN et prometteuse pour la vérification des HFPN. En collaboration avec les biologistes, elle saura apporter de nouvelles questions et hypothèses, comme elle a déjà su le faire dans l'étude de la métamorphose amphibienne.

Bibliographie

[1] T. Ideker, T. Galitski, and L. Hood. A new approach to decoding life : Systems biology. *Annu. Rev. Genomics Hum. Genet.*, 2 :343–372, 2001.

[2] H. Kitano. Systems biology : a brief overview. *Science*, 295(5560) :1662–4, 2002.

[3] H. Kitano. Computational systems biology. *Nature*, 420(6912) :206–10, 2002.

[4] G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks : Extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3) :339–347, 2004.

[5] R. Thomas and M. Kaufman. Analyse logique des circuits de rétroaction. *Mathématiques pour Modéliser la Dynamique du Vivant*, 2000.

[6] J.J. Tyson, A. Csikasz-Nagy, and B. Novak. The dynamics of cell cycle regulation. *Bioessays*, 24(12) :1095–109, 2002.

[7] W. Sha, J. Moore, K. Chen, A.D. Lassaletta, C.S. Yi, J.J. Tyson, and J.C. Sible. Hysteresis drives cell-cycle transitions in xenopus laevis egg extracts. *Proc. Natl Acad. Sci. U S A.*, 100(3) :975–80, 2003.

[8] K.C. Chen, L. Calzone, A. Csikasz-Nagy, F.R. Cross, B. Novak, and J.J. Tyson. Integrative analysis of cell cycle control in budding yeast. *Mol. Biol. Cell*, 15 :3841–3862, 2004.

[9] S. Schuster, T. Pfeiffer, F. Moldenhauer, I. Koch, and T. Dandekar. Structural analysis of metabolic networks : Elementary flux modes, analogy to Petri nets and application to mycoplasma pneumoniae. In *German Conference on Bioinformatics 2000*, pages 115–120, 2000.

[10] S. Schuster, T. Pfeiffer, F. Moldenhauer, I. Koch, and T. Dandekar. Exploring the pathway structure of metabolism : decomposition into subnetworks and application to mycoplasma pneumoniae. *Bioinformatics*, 18(2) :351–361, 2002.

[11] A. von Kamp and S. Schuster. Metatool 5.0 : fast and flexible elementary modes analysis. *Bioinformatics*, 22 :1930–1931, 2006.

[12] S. Pérès and J.-P. Comet. Contribution of computational tree logic to biological regulatory networks : example from pseudomonas aeruginosa. In *International workshop on Computational Methods in Systems Biology*, volume 2602 of *LNCS*, pages 47–56, February 24-26, 2003.

[13] H. de Jong. Modeling and simulation of genetic regulatory systems : a literature review. *J. Comput. Biol.*, 9(1) :67–103., 2002.

[14] M. Nagasaki, A. Doi, H. Matsuno, and S. Miyano. Genomic object net : a platform for modeling and simulating biopathways. *Applied Bioinformatics*, 2 :181–184, 2003.

[15] A. Doi, M. Nagasakin, H. Matsuno, and S. Miyano. Genomic object net : II. modelling biopathways by hybrid functional Petri net with extension. *Applied Bioinformatics*, 2(3) :185–188, 2003.

[16] H. Matsuno, Y. Tanaka, H. Aoshima, A. Doi, M. Matsui, and S. Miyano. Biopathways representation and simulation on hybrid functional Petri nets. *In Silico Biology*, 3 :389–404, 2003.

[17] A. Doi, S. Fujita, H. Matsuno, M. Nagasaki, and S. Miyano. Constructing biological pathway models with hybrid functional Petri nets. *In Silico Biology*, 4 :271–291, 2004.

[18] W. Reisig. *Petri Nets*. Springer-Verlag, 1985.

[19] S. Troncale, D. Campard, F. Tahi, J. Guespin, and J.-P. Vannier. Modeling and simulation with hybrid functional Petri nets of the role of interleukin-6 in human early haematopoiesis. In *Proceedings of the Pacific Symposium of Biocomputing*, volume 11, pages 427–438, 2006.

[20] S. Troncale, F. Tahi, D. Campard, and JP. Vannier. Modélisation et simulation de la régulation de l'hématopoïèse précoce grâce aux réseaux de Petri hybrides fonctionnels. *Technique des Sciences Informatiques*, 26 :99–122, 2006.

[21] S. Troncale, R. Thuret, A.-C. Fierro, N. Pollet, J.-P. Comet, and G. Bernot. Modelling of TH-dependent regulation of tadpole tail resorption. *Journal of Biological Physics and Chemistry*, 7 :45–50, 2007.

[22] J. Ahmad, O. Roux, G. Bernot, J.-P. Comet, and A. Richard. Analysing formal models of genetic regulatory networks with delays : Applications to lambda phage and t-cell activation systems. *Int. J. Bioinformatics Research and Applications*, 2008 [to appear].

[23] H. de Jong, J. Geiselmann, G. Batt, C. Hernandez, and M. Page. Qualitative simulation of the initiation of sporulation in bacillus subtilis. *Bulletin of Mathematical Biology*, 66(2) :261–300, 2004.

[24] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page, and D. Schneider. Validation of qualitative models of genetic regulatory networks by model checking : Analysis of the nutritional stress response in escherichia coli. *Bioinformatics*, 21(suppl. 1) :i19–i28, 2005.

[25] E. Fanchon, F. Corblin, L. Trilling, B. Hermant, and D. Gulino. Modeling the molecular network controlling adhesion between human endothelial cells : Inference and simulation using constraint logic programming. In *CMSB 2004*, pages 104–118, 2004.

[26] Workshop on Constraint Based Methods for Bioinformatics (WCB–ICLP'05). *Constraint Logic Programming for Modeling a Biological System described by a Logical Network*, 2005.

[27] H. Siebert and A. Bockmayr. Relating attractors and singular steady states in the logical analysis of bioregulatory networks. In *Algebraic Biology 2007*, pages 36–50, 2007.

[28] R. Thomas, A.M. Gathoye, and L. Lambert. A complex control circuit. Regulation of immunity in temperate bacteriophages. *Eur. J. Biochem.*, 71(1) :211–27, 1976.

[29] M. Kaufman, J. Urbain, and R. Thomas. Towards a logical analysis of the immune response. *J. Theor. Biol.*, 114(4) :527–61, 1985.

[30] E.H Snoussi. Qualitative dynamics of a piecewise-linear differential equations : a discrete mapping approach. *Dynamics and stability of Systems*, 4 :189–207, 1989.

[31] E.H. Snoussi and R. Thomas. Logical identification of all steady states : the concept of feedback loop caracteristic states. *Bull. Math. Biol.*, 55(5) :973–991, 1993.

[32] D. Thieffry and R. Thomas. Qualitative analysis of gene networks. In *Proceedings of the Pacific Symposium of Biocomputing*, pages 77–88, 1998.

[33] R. Thomas and M. Kaufman. Multistationarity, the basis of cell differentiation and memory. I. Structural conditions of multistationarity and other nontrivial behavior. *Chaos*, 11 :170–179, 2001.

[34] R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets.* Springer, 2005.

[35] R. David and H. Alla. *Du Grafcet aux Réseaux de Petri.* Hermès Ed., 1992.

[36] J-F. Raskin and P-Y. Schobbens. The logic of event clocks. *Journal of Automata, Languages and Combinatorics*, 4 :247–282, 1999.

[37] R. Alur, L. Fix, and T. Henzinger. Event-clock automata : a determinizable class of timed automata. *Theor. Computer Science*, 211 :253–273, 1999.

[38] R. Alur and D. Dill. A theory of timed automata. *Theor. Computer Science*, pages 183–235, 1994.

[39] C.-A. PEtri. Interpretation of net theory. *Computer Science : Net theory and applications*, pages 1–19, 1979.

[40] J.-M. Proth. Petri nets for modelling and evaluating deterministic and stochastic manufacturing systems. In *International workshop on PEtri Nets and Performance Models*, pages 2–15, 1997.

[41] G. Wheeler. The modelling and analysis of IEEE802.6's configuration. In *Application of PEtri Nets to Communication Networks*, volume 1605, pages 69–92, 1999.

[42] V.-M. Reddy, M.-L. Mavrovouniotis, and M.-N. Liebman. Petri net representations in metabolic pathways. In *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, volume 1, pages 328–336, 1993.

[43] R. Hofstädt. A Petri net application to model metabolic processes. In *Systems Analysis Modelling Simulation*, volume 16, pages 113–122, 1994.

[44] H. Genrich, R. Küffner, and K. Voss. Executable Petri net models for the analysis of metabolic pathways. *STTT*, 3 :394–404, 2001.

[45] K. Voss, M. Heiner, and I. Koch. Steady state analysis of metabolic pathways using Petri nets. *In Silico Biology*, 3 :367–387, 2003.

[46] P.J.E. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. In *Proc. Natl. Acad. Sci. USA*, volume 95, pages 6750–6755, 1998.

[47] R. Srivastava, L. You, J. Summers, and J. Yin. Stochastic versus deterministic modeling of intracellular viral kinetics. *Journal of Theoretical Biology*, 218 :309–321, 2002.

[48] H. Matsuno, A. Doi, M. Nagasaki, and Saturo. Miyano. Hybrid Petri net representation of gene regulatory network. In *Proceedings of the Pacific Symposium of Biocomputing*, pages 341–352, 2000.

[49] H. Alla and R. David. Continuous and hybrid Petri nets. *Journal of Circuits, Systems and Computers*, 8 :159–188, 1998.

[50] P. Mendes. Biochemistry by numbers : simulation of biochemical pathways with gepasi 3. *Trends in Biochemical Sciences*, 22 :361–363, 1997.

[51] S. Troncale, D. Campard, F. Tahi, J. Guespin, and J.-P. Vannier. Modeling and simulation with hybrid functional Petri nets of the role of interleukin-6 in human early haematopoiesis. In *Proceedings of the Pacific Symposium of Biocomputing*, pages 427–438, 2006.

[52] E. Fanchon, F. Corblin, L. Trilling, B. Hermant, and D. Gulino. Modeling the molecular network controlling adhesion between human endothelial cells : inference and simulation using constraint logic programming. In *CMSB*, volume 14, pages 104–118, 2004.

[53] R. Thomas and R. d'Ari. Biological feedback. *CRC Press*, 1990.

[54] C. Rose. Integrating ecology and developmental biology to explain the timing of frog metamorphosis. *TRENDS in Ecolgy and Evolution*, 20 :129–135, 2005.

[55] J.D. Furlow and E.S. Neff. A developmental switch induced by thyroid hormone : Xenopus laevis metamorphosis. *TRENDS in Endocrinology and Metabolism*, 17 :40–47, 2006.

[56] J. Leloup and M. Buscaglia. Triiodothyronine, hormone of amphibian metamorphosis. *C.R. Acad. Sci.*, pages 2261–2263, 1977.

[57] M. Ranjan, J. Wong, and Y.-B. Shi. Transcriptional repression of xenopus TR-β gene is mediated by a thyroid hormone response element located near the start site. *Journal of Biological Chemistry*, 269 :24699–24705, 1994.

[58] L. Cai and D. Brown. Expression of type 2 iodothyronine deiodinase marks the time that a tissue responds to thyroid hormone-induced metamorphosis in Xenopus laevis. *Developmental Biology*, 266 :87–95, 2003.

[59] H. Huang, L. Cai, B. Remo, and D. Brown. Timing of metamorphosis and the onset of the negative feedback loop between the thyroid gland and the pituitary is controlled by type 2 iodothyronine deiodinase in Xenopus laevis. *Proc. Nat. Acad. Sci.*, 98 :7348–7353, 2001.

[60] H. Huang, N. Marsh-Armstrong, and D. Brown. Metamorphosis is inhibited in transgenic Xenopus laevis tadpoles that overexpress type 3 deiodinase. *Developmental Biology*, 96 :962–967, 1999.

[61] K. Becker, K. Stephens, J. Davey, M. Scheider, and V-A. Galton. The type 2 and 3 iodothyronine deiodinases play important roles in coordinating development in rana catesbeiana tadpoles. *Endocrinology*, 138 :2989–2997, 1997.

[62] Y.-B. Shi. *Amphibian Metamorphosis : From Morphology to Molecular Biology*. Wiley, 1999.

[63] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool Kronos. In *Hybrid Systems III : Verification and Control*, volume 1066, pages 208–219. Springer, 1995.

[64] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi. UPPAAL - a tool suite for automatic verification of real-time systems. In *Hybrid Systems III : Verification and Control*, pages 232–243. Springer, 1996.

[65] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 4, pages 133–191. Elsevier Science Publishers, 1990.

[66] T. Henzinger, J-F. Raskin, and P-Y. Schobbens. The regular real-time langages. In *ICALP 97 : Automata, Languages and Programming*, pages 580–591, 1998.

[67] P. Wolper. The tableau method for temporal logic. In *Logique et Analyse*, pages 119–136, 1985.

[68] J. H. Gallier. *Logic for Computer Science*. Harper and Row, 1986.

[69] D. K. Wilde. A library for doing polyhedral operations. Technical Report RR-2157, IRISA, 1993.

[70] M. Sorea. Tempo : A model checker for event-recording automata. Technical report, SRI International, Computer Science Laboratory, 2001.

[71] E. Burki and M. Fischberg. Evolution of globin expression in the genus Xenopus. *Mol. Biol. Evol.*, 2 :270–277, 1985.

[72] P. Richardson and J. Chapman. The xenopus tropicalis genome project. *Current Genomics*, 4 :645–652, 2003.

[73] CH. Thiebaud and M. Fischberg. DNA content in the genus xenopus. *Chromosoma*, 59 :253–257, 1977.

[74] CA. Bisbee, MA. Baker, AC. Wilson, I. Haji-Azimi, and M. Fischberg. Albumin phylogeny for clawed frogs. *Science*, 195 :785–787, 1977.

[75] C.-G. Pellizas, A.-H. Coleoni, M.E. Costamagna, M. Di Fulvio, and A.M. Masini-Repiso. Insulin-like growth facto I reduces thyroid hormone receptors in the rat liver. evidence for a feed-back loop regulating the peripheral thyroid hormone action. *Journal of Endocrinology*, 158 :87–95, 1998.

[76] P.D. Nieuwkoop and J. Faber. *Normal Table of Xenopus laevis*. Garland Publishing Inc, 1956.

[77] Y. Yaoita and DD. Bown. A correlation of thyroid hormone receptor gene expression with amphibian metamorphosis. *Genes Dev.*, 4 :1917–1924, 1990.

[78] A. Kanamori and D.-D. Brown. The regulation of thyroid hormone receptor beta genes by thyroid hormone in Xenopus laevis. *J. Biol. Chem.*, 267 :739–745, 1992.

[79] A. Kawahara, B.-S. Baker, and J.-R. Tata. Developmental and regional expression of thyroid hormone receptor genes during Xenopus metamorphosis. *Development*, 112 :933–943, 1991.

[80] M.-H.-I. Dodd and J.-M. Dodd. The biology of metamorphosis. *Physiology of amphibia*, 1976.

[81] R. Thuret. *Etudes des transcriptomes de Xenopus tropicalis au cours de la métamorphose.* PhD thesis, Laboratoire de Dévelopment et Evolution, Univers. Paris Sud, 2007.

[82] J.-R. Tata, L. Ernster, O. Lindberg, E. Arrhenius, S. Pedersen, and R. Hedman. The action of thyroid hormones at the cell level. *Biochem. J.*, 86 :408–428, 1963.

[83] Z. Wang and D. Brown. Thyroid hormone-induced gene expression program for amphibian tail resorption. *Journal of Biological Chemistry*, 268 :16270–16278, 1993.

[84] D. Germain, R. Schwartzman, W. Croteau, A. Kanamori, Z. Wang, D. Brown, and V Galton. A thyroid hormone-regulated gene in *Xenopus laevis* encodes a type III iodothyronine 5-deiodinase. *Developmental Biology*, 91 :7767–7771, 1994.

[85] D. Berry, R. Schwartzman, and D. Brown. The expression pattern of thyroid hormone response genes in the tadpole tail identifies multiple resorption programs. *Developmental Biology*, 203 :12–23, 1998.

[86] K. Nakajima and Y. Yaoita. Dual mechanisms governing muscle cell death in tadpole tail during amphibian metamorphosis. *Developmental Dynamics*, 227 :246–255, 2003.

[87] J. Wong and Y.-B. Shi. Coordinated regulation of and transcriptional activation by xenopus thyroid hormone and retinoid X receptors. *Journal of Biological Chemistry*, 270 :18479–18483, 1995.

[88] M. Kerszberg. Accurate reading of morphogen concentrations by nuclear receptors : a formal model of complex transduction pathways. *J. Theo. Biol.*, 183 :95–104, 1996.

[89] Y. Zhou. Analyse transcriptionnelle des gênes impliqués dans la métamorphose de Xenopus tropicalis. Master's thesis, IBISC, Univers. Evry, 2007.

[90] C. Chaouiya, E. Remy, and D. Thieffry. Petri net modelling of biological regulatory networks. In *Elsevier Science*, 2005.

[91] S. Cory and J. Adams. The Bcl2 family : regulators of the cellular life-or-death switch. *Nature review*, 2 :647–656, 2002.

[92] I. Rowe, K. LeBlay, D. Du Pasquier, K. Palmier, G. Levi, B. Demeneix, and L. Coen. Apoptosis of tail musclle during amphibian metamorphosis involves a caspase-9-dependent mechanism. *Developmental Dynamics*, 233 :76–87, 2005.

[93] J. Johnston, R. Chan, M. Calderon-Segura, S. McFarlane, and L. Browder. The roles of Bcl-XL in modulating apoptosis during development of Xenopus laevis. *BMC Developmental Biology*, 5 :20–36, 2005.

[94] I. Rowe, L. Coen, K. LeBlay, S. Le Mevel, and B. Demeneix. Autonomous regulaton of muscle fibre fate during metamorphosis in Xenopus tropicalis. *Developmental Dynamics*, 224 :381–390, 2002.

[95] B. Das, L. Cai, M. Carter, YL. Piao, A. Sharov, M. Ko, and D. Brown. Gene expression changes at metamorphosis induced by thyroid hormone in Xenopus laevis tadpoles. *Developmental Biology*, 291 :342–355, 2006.

[96] I. Maros. *Computational Techniques of the Simplex Method.* Kluwer Academic Publishers, 2003.

[97] J. Hollman and L. Langemyr. Algorithms for non-linear constraints. In *WCLP*, pages 113–131, 1991.

[98] V. Saraswat. *Principles and pratice of constraint programming.* MIT Press, 1995.

[99] H. Thomas, Romijn. J., M. Stoelinga, and F.-W. Vaandrager. Linear parametric model checking of timed automata. In *TACAS*, pages 189–203, 2001.

[100] R. Alur, T.-A. Henzinger, and M.-Y. Vardi. Parametric real-time reasoning. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 592–601, 1993.

[101] V. Bruyère, E. Dall'Olio, and J.-F. Raskin. Durations, parametric model-checking in timed automata. In *STACS '03 : Proceedings of the 20th Annual Symposium on Theorical Aspects of Computer Science*, pages 687–698. Springer-Verlag, 2003.

[102] A. Nakata, T. Tanimoto, S. Sasaki, and T. Higashino. A timed failure equivalence preserving abstraction for parametric time-interval automata. *IJFCS : International Journal of Foundations of Computer Science*, 17 :833–850, 2006.

[103] W. Thomas. Complementation of Büchi automata revisited. In Juhani Karhumäki et al., editors, *Jewels are forever – Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 109–122. Springer, 1999.

[104] G.-F. Berriz, O.D. King, B. Bryant, C. Sander, and F.-P. Roth. Characterising gene sets with funcassociate. *Bioinformatics*, 19 :2502–2504, 2003.

[105] A.-J. Butte, P. Tamayo, D. Slonim, T.-R. Golub, and I.-S. Kohane. Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks. *PNAS*, 97 :12182–12186, 2000.