Table des matières

Introduction

La biologie des systèmes propose d'étudier les êtres vivants du point de vue du *système*. Les systèmes biologiques étudiés par cette discipline sont divers, et s'organisent à des échelles variées, qui vont du macroscopique au microscopique : l'étude d'un écosystème entier aussi bien que celle d'un mécanisme moléculaire faisant entrer en jeu un petit nombre de molécules relèvent de cette discipline. La biologie des systèmes vise à comprendre un système biologique en étudiant les propriétés qui *émergent* de l'ensemble des propriétés des entités qui le constituent. Les relations qu'entretiennent les entités d'un système donné sont souvent représentées sous la forme de *réseaux biologiques*. En particulier, les *réseaux moléculaires* représentent les processus moléculaires, plus ou moins abstraits, qui interviennent dans un processus biologique donné. Ils permettent de représenter divers processus biologiques, comme des voies métaboliques, des voies de signalisation ou encore des processus de régulation génétique.

En biologie des systèmes, étudier un processus biologique, c'est donc avant tout étudier le réseau moléculaire sous-jacent à ce processus. Deux types de réseaux moléculaires existent, et diffèrent par la nature des processus moléculaires qu'ils représentent : les réseaux de réactions permettent de représenter des processus moléculaires concrets tels que des réactions chimiques ou des translocations, et les graphes d'influences des processus moléculaires plus abstraits tels que les activités moléculaires. Si les graphes d'influences sont généralement plus simples que les réseaux de réactions (dans le sens où ils comportent moins de relations), la différence entre ces deux types de représentation réside principalement dans le point de vue qu'ils permettent d'adopter : les réseaux de réactions donnent les mécanismes moléculaires sous-jacents à un processus biologique, alors que les graphes d'influences représentent les fonctions moléculaires qui entrent en jeu dans ce processus. Par exemple, dans la voie de signalisation induite par la protéine G, on sait que la fonction de MEK est d'activer ERK. Le mécanisme de cette activation est également connu : MEK active ERK en catalysant sa phosphorylation. Par conséquent, dans un réseau de réactions représentant les processus moléculaires sous-jacents à cette voie, la réaction de phosphorylation d'ERK ainsi que sa catalyse par MEK seront représentées, alors que dans le graphe d'influences correspondant, la fonction de MEK sera représentée simplement par une influence positive de l'activité de MEK sur l'activité d'ERK, sans en détailler le mécanisme. Les points de vue mécaniste d'une part, et fonctionnel d'autre part, ne sont donc pas antagonistes mais complémentaires. Un même processus biologique peut parfois aussi bien être représenté par un réseau de réactions que par un graphe d'influences, et c'est en particulier le cas pour les processus de signalisation.

Deux tâches fondamentales de la biologie des systèmes sont donc la construction des réseaux moléculaires et leur analyse. Avec l'augmentation du nombre de données expérimentales, ces tâches ne peuvent plus être réalisées manuellement. Par conséquent, le développement de méthodes automatiques pour la construction et l'analyse des réseaux est devenu nécessaire. Afin de produire des résultats qui soient compréhensibles et exploitables par les biologistes, ces méthodes doivent refléter aussi fidèlement que possible les raisonnements entrepris par ces derniers lorsqu'ils réalisent l'une ou l'autre tâche. Par conséquent, il est nécessaire qu'elles prennent en compte l'ensemble des concepts couramment utilisés pour décrire les processus biologiques, et que le sens apporté à ces concepts soit proche de celui consi-

déré par les biologistes. C'est dans cette optique que nous avons développé un ensemble de méthodes pour la construction et l'analyse des réseaux de réactions et des graphes d'influences. Ces méthodes sont basées sur la formalisation à la fois des différents concepts biologiques utilisés (données expérimentales, relations contenues dans les réseaux de réactions et les graphes d'influences) - leur donnant ainsi un sens explicite - et des raisonnements entrepris par les biologistes pour la construction et l'analyse des réseaux moléculaires.

Les relations représentées dans les réseaux moléculaires sont principalement obtenues grâce aux expériences menées par les biologistes. Au fur et à mesure de ces expériences, les connaissances des processus moléculaires sous-jacents aux processus biologiques sont complétées, jusqu'à ce qu'elles soient suffisamment nombreuses et exhaustives pour être organisées sous la forme d'un réseau moléculaire. Plusieurs réseaux exhaustifs ont ainsi été reconstruits. Nous pouvons par exemple citer les réseaux induits par le récepteur de l'EGF [Oda+05] et de la FSH [Glo+11], le réseau de régulation du cycle cellulaire par RB/E2F [Cal+08], ou encore le réseau du métabolisme humain [Thi+13]. Avec les expériences haut-débit, le nombre de données expérimentales relatives aux processus cellulaires a explosé et, vu leur nombre, ces dernières ne peuvent plus être raisonnablement interprétées à l'aide de méthodes manuelles. C'est pourquoi diverses méthodes automatiques de construction des réseaux moléculaires à partir de données expérimentales ont vu le jour (pour des surveys, voir [Ban+07 ; MS07]). Ces méthodes sont principalement statistiques et proposent le plus souvent de construire des réseaux de régulation génétique ou des réseaux de signalisation à partir de données temporelles d'expression de gènes ou de quantités de protéines (voir p. ex. [Eis+98 ; Pe'+01] pour les réseaux de régulation génétique, et [Sac+05 ; Hil+12] pour les réseaux de signalisation). Les relations qu'entretiennent les gènes ou les protéines entre eux sont alors obtenues en mesurant la corrélation entre les données temporelles relatives à ces gènes ou protéines. Ces méthodes ne prennent le plus souvent en compte que quelques types de résultats expérimentaux, et ne permettent pas de découvrir les mécanismes moléculaires précis qui régissent les influences entre les gènes et protéines entrant en jeu dans les processus cellulaires. Elles permettent donc de découvrir des relations cachées au plus profond de données expérimentales quantitatives, mais, de par leur nature, ne construisent pas de réseaux exhaustifs et détaillés à l'échelle des mécanismes moléculaires.

D'autres méthodes sont basées sur la formalisation et l'automatisation, à l'aide de formalismes logiques et de moteurs d'inférence, du raisonnement qualitatif mené par les biologistes lorsqu'ils interprétent des résultats expérimentaux provenant d'expériences classiques de la biologie des systèmes (p. ex. [Zup+03 ; Nig+15]). Ces méthodes ont deux avantages, comparées aux méthodes statistiques. D'une part, elles permettent la prise en compte de données expérimentales qualitatives qui sont encore produites massivement par les biologistes en réalisant des expériences classiques de biologie moléculaire (p. ex. des ELISA, qui permettent de doser la présence de protéines à l'aide d'anticorps). D'autre part, elles permettent de lier formellement les connaissances inférées aux résultats expérimentaux dont elles proviennent, et ce de manière explicite. Cependant, comme pour les approches statistiques, les méthodes proposées jusqu'à maintenant ne prennent souvent en compte qu'un nombre limité de types d'expériences, et ne permettent pas d'inférer de connaissances à l'échelle des mécanismes moléculaires. C'est pourquoi nous proposons dans ce manuscrit une méthode qualitative de construction des réseaux moléculaires, à l'échelle des mécanismes réactionnels, basée sur l'interprétation automatique de résultats expérimentaux provenant d'une grande variété de types d'expérience.

Enfin, les graphes d'influences peuvent aussi être construits par transformation des réseaux de réactions. En effet, les graphes d'influences étant plus abstraits que les réseaux de réactions, il est parfois possible d'interpréter un ensemble de réactions sous la forme d'activités moléculaires et d'influences, et de les représenter sous la forme d'un graphe. La principale contribution dans ce domaine est la méthode introduite dans [VCS13] et implémentée dans le logiciel [CKS10]. Cette méthode est basée

sur des templates prédéfinis et modifiables par l'utilisateur, et offre ainsi une grande liberté dans la transformation. Cependant, elle ne permet pas de transformer certains mécanismes fondamentaux de la biologie des systèmes, et notamment des réseaux de signalisation. Par conséquent, nous introduisons une nouvelle méthode de transformation des réseaux de signalisation représentés sous la forme de réseaux de réactions, en graphes d'influences.

Identifier les mécanismes moléculaires plus ou moins précis qui entrent en jeu dans un processus biologique ne permet pas d'en avoir une compréhension totale. En effet, les processus biologiques sont par nature *dynamiques*, c'est-à-dire qu'ils évoluent au cours du temps, sous l'impulsion de l'évolution des entités moléculaires qu'ils font entrer en jeu. Pour comprendre la dynamique de ces processus, il faut donc d'abord analyser la dynamique des réseaux sous-jacents. La dynamique de tels réseaux peut être vue comme l'évolution, au cours du temps, de l'état des entités moléculaires qui constituent ces réseaux. Les réseaux moléculaires étant par nature des représentations statiques de processus moléculaires, le calcul de la dynamique d'un réseau passe généralement par la construction d'un *modèle mathématique*, obtenu en attribuant une dimension dynamique aux relations de ce réseau, à l'aide d'une *sémantique*. Différentes sémantiques et différents formalismes ont été proposés pour modéliser la dynamique des réseaux moléculaires (pour un survey, voir [HS14]). Deux types de sémantiques peuvent être distinguées : les *sémantiques quantitatives*, qui décrivent la dynamique des entités d'un réseau à l'aide de variables continues ; et les *sémantiques qualitatives*, qui décrivent cette même dynamique à l'aide de variables discrètes bornées.

Si les sémantiques quantitatives sont celles qui reproduisent le plus fidèlement l'évolution temporelle des systèmes biologiques, elles font le plus souvent entrer en jeu des paramètres cinétiques qui sont difficiles à obtenir. Au contraire, les sémantiques qualitatives, même si elles décrivent moins précisément la dynamique des réseaux, permettent d'en capturer les propriétés les plus importantes, comme les attracteurs ou des propriétés d'atteignabilité, et ne nécessitent pas de connaître des paramètres cinétiques.

La sémantique qualitative la plus simple pour la construction de modèles dynamiques à partir de réseaux moléculaires est la *sémantique Booléenne*. Elle a été introduite par Stuart Kauffman à la fin des années 1960 [Kau69], pour la modélisation des graphes d'influences. Le formalisme de choix pour la formalisation de la dynamique Booléenne des graphes d'influences est le *réseau Booléen*. Cependant, quelques travaux (comme [Fay+11]) ont proposé des méthodes de calcul de la dynamique Booléenne des graphes d'influences à l'aide de *programmes logiques* du premier ordre. Katsumi Inoue a récemment montré, de manière formelle, que les *points attracteurs* d'un réseau Booléen pouvaient être calculés très simplement à l'aide d'un programme logique normal propositionnel obtenu à partir de ce réseau Booléen [Ino11]. En s'appuyant sur ces résultats, nous proposons une méthode de calcul des points attracteurs et des traces finies de réseaux Booléens paramétrés à l'aide de principes généraux, basée sur des programmes logiques du premier ordre.

Une sémantique Booléenne a également été proposée pour modéliser la dynamique des réseaux de réactions [CFS06]. Cette sémantique est définie pour des réseaux de réactions comportant différents types de processus moléculaires, comme les réactions chimiques, les associations/dissociations ou les translocations, et les catalyses. Si cette sémantique est formellement bien définie et est une abstraction d'autres sémantiques bien connues [FS08], elle souffre de deux limitations majeures. D'une part, elle ne prend pas en compte les inhibitions, qui sont pourtant courantes dans les réseaux de signalisation par exemple. D'autre part, les modèles qu'elle permet de construire sont parfois éloignés de la façon dont nous appréhendons les processus moléculaires, en particulier les changements d'états d'une même entité moléculaire. Pour remédier à ces manques, nous proposons deux nouvelles sémantiques qualitatives pour l'analyse de la dynamique de réseaux de réactions.

Un nombre conséquent de méthodes développées jusqu'à maintenant, visant à construire ou à analyser la dynamique des réseaux moléculaires, reposent sur des formalismes qualitatifs. Ces méthodes, dites qualitatives, ont l'avantage majeur de leur simplicité de mise en œuvre, notamment en raison du fait qu'elles ne nécessitent pas de paramètres ou de mesures numériques (comme les paramètres cinétiques pour les sémantiques quantitatives de la dynamique des réseaux), qui sont parfois difficiles à obtenir. Les formalismes sur lesquels elles reposent (comme celui de la programmation logique, des réseaux Booléens ou encore des réseaux d'automates) ont en outre été largement étudiés à travers différents travaux théoriques, et un nombre grandissant de logiciels permettent leur utilisation. C'est dans ce cadre que nous avons effectué l'ensemble de nos travaux. Ainsi, toutes les méthodes que nous présentons dans ce manuscrit sont des méthodes qualitatives. De plus, elles reposent toutes sur la Systems Biology Notation (SBGN), qui est l'un des standards de la biologie des systèmes que nous introduisons ci-dessous.

Une particularité de la biologie des systèmes est en effet qu'elle s'appuie maintenant sur un nombre toujours plus grand de ces standards. Ils ont été développés à partir du début des années 2000 avec l'accroissement du nombre des données obtenues, de connaissances produites et des méthodes proposées. La Gene Ontology (GO) [Ash+00] et la Systems Biology Ontology (SBO) [JN13] ont été développées afin de définir et d'organiser les termes du domaine. La Systems Biology Graphical Notation (SBGN) [LN+09] et BioPax [Dem+10] standardisent la représentation des réseaux moléculaires. Quant au Systems Biology Markup Language (SBML) [Huc+03] et au Simulation Experiment Description Markup Language (SEDML) [Wal+11b], ils permettent la représentation des modèles mathématiques et de leurs simulations, respectivement. L'ensemble des ces standards facilitent en particulier l'échange et la compréhension des réseaux moléculaires construits par la discipline, ainsi que la réutilisation des modèles mathématiques. Ils sont maintenant pris en compte par un nombre toujours grandissant de logiciels, et deviennent incontournables dans le développement des méthodes de la biologie des systèmes. Parmi ces standards, SBGN nous intéresse tout particulièrement. C'est un ensemble de trois langages graphiques permettant la représentation des réseaux moléculaires. Le langage Process Description (SBGN-PD) [Moo+11] permet la représentation des réseaux de réactions ; le langage Activity Flow (SBGN-AF) [Mi+09] permet la représentation des graphes d'influences ; et le langage Entity Relationship (SBGN-ER) [LN+11] permet de représenter des relations entre des entités moléculaires d'un système biologique, sans aspects temporels. Ce standard, en plus de servir à la représentation des réseaux moléculaires, fournit l'ensemble des concepts qui sont le plus souvent utilisés par les biologistes pour exprimer les connaissances relatives à un système biologique étudié à l'échelle moléculaire. Ainsi, plus grande partie des réseaux contenus dans les articles de biologie ou dans divers entrepôts de réseaux peuvent être exprimés en SBGN. C'est notamment le cas des réseaux de la base KEGG [KG00], qui peuvent être automatiquement transformés en réseaux SBGN grâce à la méthode présentée dans [Cza+13] et implémentée dans SBGN-ED [CKS10].

Nous présentons dans ce manuscrit nos travaux de thèse, qui ont été réalisés au sein de l'équipe de Bioinformatique du Laboratoire de Recherche en Informatique (LRI), sous la direction de Christine Froidevaux. Ces travaux font suite à notre formation initiale en biologie et ont été initiés au cours de différents stages de recherche que nous avons effectués au sein de cette même équipe, et de celle de Katsumi Inoue au National Institute of Informatics (NII) de Tokyo. Les travaux qui sont présentés dans ce manuscrit sont le fruit de collaborations nationales et internationales.

Nous avons développé un ensemble de méthodes qualitatives pour la construction et l'analyse de la dynamique des réseaux de réactions et des graphes d'influences. L'ensemble de notre démarche repose sur l'explicitation, à l'aide de la formalisation, du sens apporté aux différents concepts biologiques considérés (données expérimentales, réseaux de réactions, graphes d'influences) et des liens qui

les unissent. Pour chacune des deux tâches considérées, nous avons veillé à être au plus proche des raisonnements réalisés par les biologistes. Un certain nombre des méthodes que nous proposons sont d'ailleurs une formalisation, en logique du premier ordre, de raisonnements couramment entrepris par ces derniers. L'ensemble de nos méthodes offrent donc différents points de vue sur les réseaux moléculaires qui dépendent à la fois de la nature de ces réseaux, et de la tâche considérée. Avec l'utilisation systématique de la notation SBGN, elles reposent toutes sur des bases conceptuelles communes, et les différents points de vue considérés peuvent être comparés. La figure 6.15 schématise l'ensemble de nos contributions. Leur détail est donné ci-après.

Nous proposons d'abord deux langages de la logique du premier ordre, construits à partir des langages SBGN-PD et SBGN-AF, que nous appelons SBGNLog-PD et SBGNLog-AF, respectivement. Ces deux langages permettent de représenter les réseaux de réactions et les graphes d'influences sous la forme d'ensembles d'atomes instanciés, et de raisonner automatiquement sur ces réseaux. Nous montrons ensuite comment ces langages peuvent être utilisés pour raisonner sur les réseaux, en introduisant une nouvelle méthode de transformation des réseaux de signalisation exprimés en SBGN-PD, en graphes d'influences exprimés en SBGN-AF. Une telle transformation permet d'obtenir une représentation simplifiée des processus moléculaires intervenant dans une voie de signalisation donnée, et offre un nouveau point de vue sur la voie en question.

Nous proposons ensuite une méthode de construction des réseaux de signalisation basée sur l'interprétation automatique de résultats expérimentaux provenant d'une grande variété de types d'expériences. Notre méthode repose sur un ensemble de règles d'interprétation formalisées en logique du premier ordre, qui permettent d'imiter le raisonnement réalisé par les biologistes lorsqu'ils interprètent ce type de résultats expérimentaux. Nos règles permettent d'établir de nouvelles connaissances et d'émettre de nouvelles hypothèses qui portent sur des mécanismes moléculaires précis, tout en prenant en compte un grand nombre de types d'expérience. Nous montrons également comment nos règles peuvent être utilisées pour proposer des plans expérimentaux afin de valider une hypothèse biologique donnée. Ces travaux sont le fruit d'une collaboration avec différents membres de l'équipe BIOS (INRA Centre Val de Loire) : les règles d'interprétation ont été construites en collaboration avec Pauline Gloaguen et Anne Poupon (voir la thèse de Pauline Gloaguen [Glo12] et [AE+12]), et diverses expériences en vue de valider nos hyptohèses ont été réalisées par Nathalie Langonné et Pascale Crépieux, .

Nous montrons ensuite comment la dynamique Booléenne d'un graphe d'influences SBGN-AF peut être calculée à l'aide de deux programmes logiques du premier ordre, qui formalisent des principes généraux décrivant la dynamique locale des activités constituant les graphes d'influences [Rou+14]. Ces travaux étendent ceux de Katsumi Inoue sur la relation formelle entre les réseaux Booléens et les programmes logiques normaux propositionnels. Ils ont été réalisés en collaboration avec Katsumi Inoue (NII) et Yoshitaka Yamamoto (université de Yamanashi).

Enfin, nous proposons deux nouvelles sémantiques qualitatives pour l'analyse de la dynamique des réseaux de réactions SBGN-PD, exprimées à l'aide de réseaux d'automates [Rou+16]. Contrairement aux sémantiques introduites jusqu'alors, ces deux sémantiques prennent en compte la plupart des concepts qui peuvent être représentés en SBGN-PD. La première de ces sémantiques étend la sémantique qualitative de Biocham en prenant notamment en compte les inhibitions. Quant à la deuxième de ces sémantiques, elle offre un nouveau point de vue sur la dynamique des entités moléculaires qui subissent des changements d'états par transformations successives. Elle est basée sur le concept d'histoires, qui permet de modéliser différents états physiques d'une même entité moléculaire à l'aide d'une unique variable. Ce travail a été réalisé en collaboration avec Loïc Paulevé (équipe Bioinformatique du LRI), et Laurence Calzone (Institut Curie).

Le manuscrit est organisé en quatre parties, comme suit.

- La première partie est consacrée à l'état de l'art.

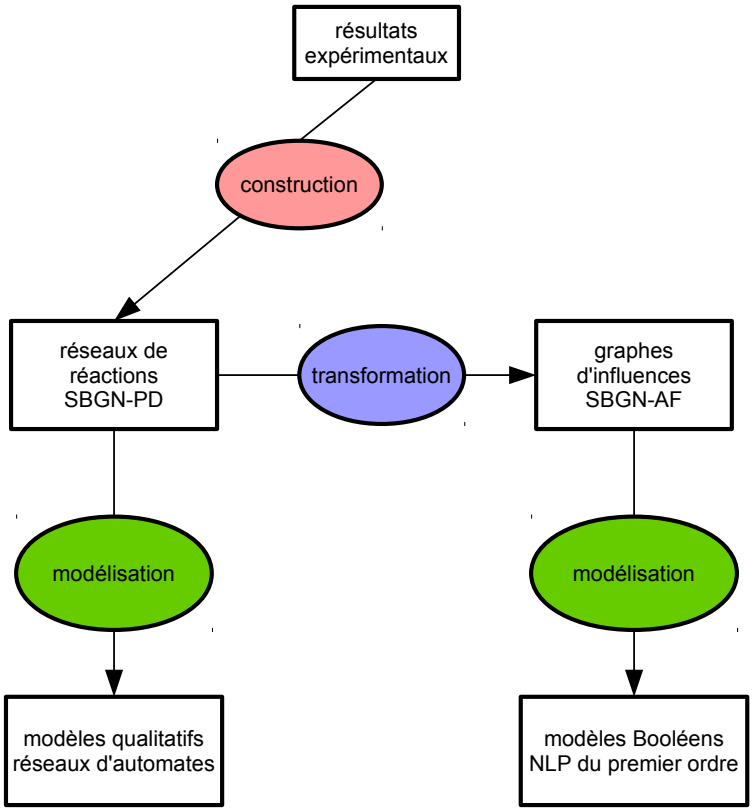


FIGURE 0.1 – **Schéma de l'ensemble de nos travaux.** Les rectangles représentent des objets d'étude de la biologie des systèmes, et les ellipses des processus, qui font chacun l'objet d'une méthode que nous avons développée.

  - Le chapitre 1, est une introduction à la biologie des systèmes et comporte un état de l'art sur la construction des réseaux moléculaires et sur l'analyse de leur dynamique. Nous y présentons différents standards de la biologie des systèmes, ainsi que les principaux entrepôts de réseaux et de modèles mathématiques.
  - Dans le chapitre 2, nous présentons les éléments de programmation logique qui seront utilisés dans ce manuscrit.
- La deuxième partie est dédiée à la représentation et à la construction des réseaux moléculaires.
  - Dans le chapitre 3, nous introduisons deux langages logiques pour la représentation des réseaux moléculaires SBGN qui permettent de raisonner sur ces réseaux. Nous illustrons une telle utilisation de ces langages en proposant une méthode de transformation des réseaux de réactions SBGN-PD représentant des voies de signalisation en graphes d'influences SBGN-AF.
  - Dans le chapitre 4, nous proposons une méthode automatique de construction des réseaux de signalisation qui imite le raisonnement effectué par les biologistes lorsqu'ils interprétent des expériences. Cette méthode permet de construire des réseaux à l'échelle des mécanismes moléculaires, et prend en compte un grand nombre de types d'expériences.
- La troisième partie se focalise sur la construction de modèles mathématiques pour l'étude de la dynamique des graphes d'influences et des réseaux de réactions.
  - Dans le chapitre 5, nous montrons comment les traces synchrones et les points attracteurs de la dynamique Booléenne d'un graphe d'influences SBGN-AF peuvent être calculés à l'aide de programmes logique normaux du premier ordre.
  - Dans le chapitre 6, nous introduisons deux nouvelles sémantiques qualitatives pour l'analyse de la dynamique des réseaux de réactions SBGN-PD. La première de ces sémantiques étend la sémantique Booléenne des réseaux de réactions en prenant notamment en compte les inhibitions. Quant à la deuxième, elle introduit le concept d'histoire, qui offre un nouveau point de vue sur les processus biologiques faisant intervenir des transformations successives d'une même entité moléculaire.
- Enfin, dans la dernière partie, nous donnons une conclusion à nos travaux, et indiquons quelques perspectives générales.

Première partie

État de l'art

Chapitre 1

Biologie des Systèmes

Sommaire

1.1 Introduction

La biologie des systèmes s'est révélée comme discipline majeure à partir du début des années 2000. Elle vise à étudier les êtres vivants du point de vue du *système*. L'étude en biologie des systèmes procède ainsi d'une démarche intégrative, qui s'oppose à une forme de réductionnisme. L'idée centrale de cette démarche est qu'il n'est pas possible d'expliquer le fonctionnement d'un système en expliquant, de façon isolée, le fonctionnement de chacune de ses parties, mais plutôt que le fonctionnement du système *émerge* du fonctionnement des parties.

Si cette idée n'est pas nouvelle et apparaît avec la notion d'organisme à la fin du XVIII^{e} siècle [Jac87], il faut attendre l'avènement de la biologie moléculaire, dans la deuxième moitié du XX^{e} siècle, pour que la biologie des systèmes s'organise en une discipline spécifique. Durant toute la fin de ce siècle, la biologie des systèmes reste cependant relativement marginale, et ce n'est qu'avec la possibilité d'obtenir des jeux de données exhaustifs, notamment en raison des avancées de la biologie moléculaire tant du point de vue de la génomique que de la protéomique, que la biologie des systèmes émerge comme la discipline majeure que l'on connaît aujourd'hui [Kit02].

La biologie des systèmes telle que considérée aujourd'hui a pour principal objet d'étude les systèmes biologiques décrits à une échelle moléculaire. Elle s'efforce de découvrir et de décrire, par l'expérience, les mécanismes et fonctions moléculaires sous-jacents aux systèmes biologiques. Ces connaissances, obtenues par l'expérience, sont organisées sous la forme de *réseaux moléculaires*. Ces réseaux décrivent le plus souvent des sous-systèmes biologiques qui font partie intégrante de systèmes plus vastes, comme les cellules, les tissus ou les organismes. Ils correspondent notamment à des processus biologiques précis comme des voies métaboliques, des voies de signalisation ou encore des sysèmes de régulation génétique. Les réseaux moléculaires décrivant ces processus sont d'ailleurs dénommés suivant la nature des processus qu'ils décrivent, et se focalisent sur différents aspects des processus moléculaires : les réseaux métaboliques décrivent des échanges de matière et d'énergie ; les réseaux de signalisation, la transmission et l'amplification de signaux ; et enfin, les réseaux de régulation génétique, des activations et des inhibitions de gènes.

Afin de comprendre un système biologique, il faut donc d'abord en découvrir les composants, et les relations que ceux-ci entretiennent, c'est-à-dire construire les réseaux moléculaires sous-jacents à ces systèmes. Mais il faut aussi découvrir les propriétés qui émergent de l'interaction de ces composants, et les expliquer par les propriétés des composants eux-mêmes. Si les propriétés émergentes des systèmes ne peuvent être établies que par l'expérience, l'explication de ces propriétés par l'interaction des composants est elle établie à l'aide de leur *modélisation*, le plus souvent mathématique.

Dans le reste de ce chapitre, nous donnons d'abord les deux types de réseaux moléculaires que nous avons considérés dans nos travaux. Puis nous revenons sur les principales approches qui ont été proposées pour la construction des réseaux moléculaires. Ensuite, nous présentons différentes sémantiques communément utilisées pour modéliser la dynamique des réseaux moléculaires. Enfin, nous décrivons différentes initiatives entreprises par la communauté de la biologie des systèmes pour standardiser les représentations et les concepts du domaine, ainsi que pour centraliser les réseaux moléculaires et les modèles disponibles. Nous décrivons notamment en détail la Systems Biology Graphical Notation, qui est au cœur de nos différents travaux.

1.2 Réseaux moléculaires

En biologie des systèmes, on s'intéresse principalement à construire et modéliser des *réseaux moléculaires*. Un réseau moléculaire est une description d'un processus biologique, comme une voie de signalisation ou une voie métabolique, en termes de processus moléculaires plus ou moins abstraits.

Les réseaux moléculaires sont le plus souvent (mais pas seulement) représentés sous la forme de graphes. Au sens large, un réseau moléculaire se compose d'un ensemble d'entités moléculaires ou d'activités émanant d'entités moléculaires qui forment les noeuds du réseau, et d'un ensemble de relations entre ces entités ou ces activités, qui forment les arcs du réseau. Dans ce manuscrit, nous nous focaliserons sur deux types de réseaux : les *réseaux de réactions* et les *graphes d'influences*. Ces derniers sont parfois également appelés *graphes d'interactions*, qui ne doivent pas être confondus avec les réseaux d'interactions protéine-protéine, qui ont une autre signification.

Les réseaux de réactions représentent un ensemble de processus moléculaires concrets, comme des réactions chimiques ou des translocations. Les noeuds de ces réseaux représentent des entités moléculaires ou des pools d'entités, et les arcs les processus moléculaires et leurs modulations. Quant aux graphes d'influences, ils représentent un ensemble d'influences moléculaires. Les noeuds représentent des entités moléculaires ou des activités opérées par des entités moléculaires, et les arcs des influences qui ont lieu entre ces entités ou activités.

Si les réseaux de réactions semblent être des versions détaillées des graphes d'influences, la différence entre ces deux types de réseaux est avant tout conceptuelle : ils ne visent pas à représenter les mêmes concepts, et participent à des points de vue distincts et complémentaires. Les réseaux de réactions, composés de processus physico-chimiques concrets, ont pour but de représenter des *mécanismes* précis, alors que les graphes d'influences font abstraction des mécanismes pour en représenter les résultats : les activités, qui sont l'actualisation de *fonctions moléculaires*. Traditionnellement, les réseaux de réactions permettent le mieux de décrire les voies métaboliques, alors que les graphes d'influences sont plus propices à la description des voies de signalisation et systèmes de régulation génétique. Cependant, lorsque les mécanismes moléculaires d'une voie de signalisation sont connus, il est tout à fait possible de représenter cette voie par un réseau de réactions. Ce sera d'ailleurs le cas pour plusieurs des réseaux de signalisation étudiés dans ce manuscrit. Par contre, les réseaux métaboliques sont, selon nous, difficilement représentables sous la forme de graphes d'influences. En effet, il nous semble difficile de conceptualiser quelle pourrait être l'activité d'un métabolite, étant donné que les métabolites n'ont pas à proprement parler de fonction moléculaire. Certes, il serait possible d'associer une fonction biologique à un métabolite. Par exemple, dans la respiration aérobie, le dioxygène joue le rôle de comburant ; mais nous ne pouvons pas dire que sa fonction moléculaire soit d'oxyder l'ubiquinol. Le dioxygène aurait donc une activité à un niveau systémique, celui de la chaîne respiratoire, sans avoir d'activité au niveau d'un processus moléculaire précis.

Il est parfois possible d'interpréter *a priori* un réseau de réactions en terme d'activités et d'influences, et par conséquent, de transformer un réseau de réactions en un graphe d'influences. Nous montrerons une telle méthode de transformation dans le chapitre 3.

1.3 Construction des réseaux moléculaires

Nous avons défini un réseau moléculaire comme un ensemble de processus moléculaires sous-jacents à un système biologique. La construction d'un réseau moléculaire consiste à découvrir, à partir de résultats expérimentaux, les processus moléculaires sous-jacents à un processus biologique donné, et à les rassembler en un réseau qui peut ensuite éventuellement être représenté de manière graphique.

1.3.1 Approche manuelle

Les biologistes réalisent des expériences qu'ils interprétent ensuite sous la forme de processus moléculaires. Souvent, une même étude montre un petit ensemble des processus moléculaires sous-jacents à un processus biologique. Ces processus moléculaires sont parfois regroupés en un réseau et représentés sous une forme graphique, en tant que conclusion de l'étude. L'ensemble des processus moléculaires

sous-jacents à un réseau sont alors établis par une multitude d'études, et la construction d'un réseau nécessite de regrouper ces processus moléculaires en un même objet (le réseau).

Ce regroupement peut être réalisé manuellement. Il convient alors d'extraire de la littérature l'ensemble des processus moléculaires sous-jacents au processus biologique d'intérêt qui ont été établis par l'expérience. Cette méthode a été employée pour construire un nombre grandissant de cartes représentant divers réseaux moléculaires, qui se veulent plus ou moins exhaustifs.

La plupart de ces réseaux peuvent être retrouvés dans des entrepôts (comme KEGG [KG00]), sur lesquels nous revenons à la section 1.7. Parmi les réseaux reconstruits à partir d'un grand nombre de résultats expérimentaux, et dont le processus de reconstruction a donné lieu à une publication, nous pouvons citer les réseaux suivants : le réseau induit par le récepteur de l'EGF [Oda+05] ; le réseau induit par le récepteur FSH [Glo+11] ; le réseau de la régulation du cycle cellulaire par RB/E2F [Cal+08] ; et enfin, le réseau du métabolisme humain [Thi+13]. Si les premiers réseaux cités ne comportent que quelques centaines de molécules, ce dernier réseau comporte pas moins de 2600 métabolites et 7400 réactions.

1.3.2 Approches automatiques

Diveses méthodes automatiques ont été proposées pour la construction des réseaux moléculaires. La plupart d'entre elles proposent de reconstruire des graphes d'influences. Nous pouvons distinguer parmi ces différentes méthodes celles qui ne reposent pas directement sur l'analyse de résultats expérimentaux, et celles qui ont comme coeur cette analyse.

1.3.2.1 Méthodes ne reposant pas sur l'analyse de données expérimentales

Les méthodes ne reposant pas sur l'analyse de résultats expérimentaux construisent le plus souvent des graphes d'influences à partir de la littérature ou de réseaux d'interaction protéine-protéine. Nous nous focalisons ici sur la construction des réseaux de régulation génétique ou des réseaux de signalisation.

Approche par fouille de la littérature. Certaines méthodes proposent de fouiller la littérature afin d'extraire des relations causales entre protéines ou gènes. Par exemple, dans [CS04], les auteurs fouillent les résumés des articles PubMed avec des techniques de traitement du langage naturel afin de trouver des relations de stimulation et d'inhibition entre des gènes, protéines et phénotypes faisant partie d'un ensemble prédéfini par l'utilisateur. Ils construisent ensuite un graphe d'influences à partir des relations extraites.

La même technique est employée par les auteurs de [Yur+06], qui proposent de construire des réseaux de signalisation en révisant des relations obtenues par fouille de la litérature. Ils utilisent la technologie MedScan [NED03] pour la fouille, qui est un système d'analyse du langage naturel de résumés d'articles. Ils éliminent ensuite un certain nombre des relations obtenues qui sont considérées comme erronées. Finalement, les relations obtenues constituent un réseau de signalisation qu'ils représentent de manière graphique.

Approche par analyse des réseau d'interaction protéine-protéine. La plupart des autres méthodes que nous avons recensées reposent sur l'analyse de la structure des graphes d'interactions protéine-protéine. Ces méthodes permettent de découvrir des voies de signalisation dans ces graphes en analysant diverses propriétés, telles que l'ensemble des chemins ou le degré des noeuds [Prz04 ; Sco+06].

1.3.2.2 Méthodes reposant sur l'analyse de données expérimentales

Approches statistiques. Les méthodes reposant sur l'analyse de données expériementales sont principalement statistiques (voir [Ban+07] et [MS07] pour des reviews de telles méthodes). Parmi ces méthodes, nous présentons deux approches : l'approche par clustering et l'approche Bayésienne. Comme ces méthodes ont d'abord été proposées pour construire des réseaux de régulation génétique à partir de données de microarray, nous expliquons les différentes méthodes en considérant des mesures d'expression de gènes.

L'approche par clustering repose sur l'idée que des gènes qui présentent les mêmes profils d'expression sont probablement co-régulés. Il s'agit alors de regrouper des profils d'expression similaires en mesurant la corrélation entre les profils deux à deux. Le résultat de ces tests de corrélation est un graphe non orienté, dit *réseau de co-expression*. Chacun des gènes est associé à un noeud de ce réseau, et deux gènes ayant des profils d'expression similaires sont reliés par un arc dans le réseau. Cette méthode ne permet toutefois pas de découvrir les mécanismes sous-jacents aux relations de co-expression inférées, et plus particulièrement de dériver des relations causales : il est par exemple possible de découvir que deux gènes A et B sont co-exprimés, mais pas que A stimule B, ou inversement. Par conséquent, cette méthode permet de découvrir à travers les données, des relations de co-expression, sans en expliquer les mécanismes biologiques.

Alors que l'approche par clustering propose de mesurer la similarité des profils d'expression de paires de gènes, les méthodes Bayésiennes reposent sur cette même similarité mais en prenant en compte les profils de tous les sous-ensembles des gènes n'appartenant pas à la paire.

Un *réseau Bayésien* est un graphe acyclique orienté, dont les noeuds sont des variables aléatoires (ici associées aux gènes), et les arcs représentent des relations de dépendance conditionnelle entre ces variables : étant donnée une variable A d'un réseau Bayésien, A est conditionnellement indépendante des autres variables qui ne sont pas des descendantes de A dans le réseau, sachant les parentes de A dans le réseau. La distribution de probabilité jointe des variables du réseau peut alors être calculée à l'aide des seules probabilités des variables conditionnées à leurs parentes.

La construction d'un modèle Bayésien à partir d'un ensemble de profils d'expression de gènes consiste à calculer le réseau Bayésien qui explique le mieux les données. L'ensemble des réseaux Bayésiens candidats sont générés et classés par une fonction de score, qui mesure la probabilité du réseau candidat étant donné les données. Le réseau avec le meilleur score est alors celui qui explique le mieux les données. Comme les arcs de ce réseau sont orientés, ils modélisent des relations causales entre ces gènes, et le réseau Bayésien peut être confondu avec un graphe d'influences. Ainsi, contrairement à l'approche par clustering, l'approche Bayésienne permet le plus souvent de découvrir les mécanismes sous-jacents aux co-expressions de gènes observées.

Si les méthodes statistiques ont été d'abord proposées pour construire des réseaux de régulation génétique (p. ex. [Eis+98],[Pe'+01]), ces méthodes ont également été utilisées pour construire des réseaux de signalisation à partir de données de phosphoprotéomique (p. ex. [Sac+05],[Hil+12]).

Approches déductives. D'autres méthodes adoptent un point de vue radicalement différent, et construisent des réseaux moléculaires en automatisant l'interprétation de résultats expérimentaux.

Dans [Zup+03], les auteurs proposent d'interpréter des résultats expérimentaux obtenus par des expériences réalisées avec des mutants. Ils utilisent pour ce faire un ensemble de règles expertes qui permettent des déduire des relations de modulation entre gènes à partir de ces résultats. Les réseaux qu'ils infèrent sont donc des réseaux de régulation génétique.

Finalement, dans [Nig+15], les auteurs proposent de construire des modèles exécutables (qui modélisent des réseaux de réactions) en interprétant automatiquement, à l'aide de règles ASP, des résultats expérimentaux classiques de la biologie.

1.4 Des réseaux statiques aux modèles dynamiques

Les réseaux moléculaires sont des représentations statiques des processus moléculaires plus ou moins concrets qui entrent en jeu dans des processus biologiques divers, comme le métabolisme ou la signalisation. Ils informent simplement, pour un processus biologique donné, de quels processus moléculaires peuvent avoir lieu, des relations de modulation qu'ils entretiennent, et parfois des conditions dans lesquelles ils ont lieu. Par conséquent, les représentations usuelles des réseaux moléculaires n'apportent aucune information *a priori* sur la dynamique des processus moléculaires : dans un réseau, rien n'est dit sur l'ordre dans lequel ces processus s'enchaînent, et encore moins sur l'évolution au cours du temps des quantités ou activités des molécules qu'ils font entrer en jeu. Or un des buts de la biologie des systèmes est de comprendre la dynamique des processus biologiques, et donc, en particulier, des réseaux moléculaires sous-jacents, qui émergerait de la dynamique des processus moléculaires qu'ils font entrer en jeu. Par conséquent, afin de connaître la dynamique d'un réseau, il est d'abord nécessaire d'interpéter en terme de dynamique les processus qu'il contient.

C'est en partie le rôle du modélisateur, qui vise le plus souvent, en biologie des systèmes, à construire des modèles mathématiques ou informatiques qui décrivent la dynamique de chaque processus moléculaire d'un réseau par l'intermédiaire d'une sémantique. Plusieurs sémantiques ont été proposées. Par exemple, la sémantique des équations différentielles ordinaires (ODE) interprète chaque molécule ou activité d'un réseau par une quantité (comme une concentration), et l'évolution au cours du temps de ces quantités est régie par des ODE. Une fois qu'un modèle dynamique d'un réseau moléculaire a été construit à l'aide d'une sémantique, il peut être simulé, calculé ou vérifié, et le résultat pourra être interprété comme une information sur la dynamique du réseau, et donc du processus qu'il décrit.

Nous pouvons distinguer deux types de sémantiques, qui permettent de construire deux types de modèles dynamiques : les sémantiques quantitatives, qui décrivent des variations de quantités de molécules ou de mesures d'activités au cours du temps à l'aide d'un ensemble de valeurs non borné, et les sémantiques qualitatives, qui décrivent ces mêmes propriétés mais de manière qualitative, par un ensemble de valeurs fini.

1.4.1 Sémantiques quantitatives

Les sémantiques quantitatives cherchent à décrire l'évolution quantitative des molécules ou activités au cours du temps. Nous présentons deux de ces sémantiques quantitatives : celle des ODE et celle des populations.

Sémantique des équations différentielles ordinaires (ODE). La sémantique quantitative la plus courante pour modéliser de tels réseaux est la sémantique des ODE. Cette sémantique, qui est déterministe, propose d'interpréter les variations des quantités des espèces chimiques d'un système de réactions par un ensemble d'ODEs. La variation de la quantité d'une espèce S du système est alors égale à la somme des vitesses des réactions produisant S soustraite à la somme des vitesses des réactions consommant S, le tout pondéré par les constantes stoechiométriques adéquates. La vitesse d'un réaction étant modélisée, de manière générale,par une fonction sur les quantités de ses réactifs, la variation de la quantité d'une espèce est modélisée par une équation différentielle.

Les équations différentielles modélisant les variations de quantités des espèces de la réaction réversible $A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} C + D$, en considérant la loi d'action de masse, sont les suivantes :

$$\frac{d[A]_t}{dt} = \frac{d[B]_t}{dt} = k_2.[C]_t.[D]_t - k_1.[A]_t.[B]_t$$
$$\frac{d[C]_t}{dt} = \frac{d[D]_t}{dt} = k_1.[A]_t.[B]_t - k_2.[C]_t.[D]_t$$

Cet ensemble d'équations différentielles forme un *système d'équations différentielles* dont les constantes k_1 et k_2, appelées constantes cinétiques de réactions, sont des paramètres. Certaines lois, comme la loi de Michaelis-Menten ou la loi de Hill, permettent de prendre en compte des cinétiques particulières impliquant des enzymes ou des phénomènes allostériques, et donnent lieu à des systèmes d'équations plus complexes.

Les systèmes n'étant pas toujours résolubles analytiquement, ils sont souvent *simulés* : étant donné un état initial et un pas de temps, les quantités des différentes espèces chimiques du système sont calculées pour chaque pas de temps à l'aide des équations.

Les équations différentielles ont été utilisées pour modéliser une variété de réseaux moléculaires. Par exemple, dans [Sch+02a], les auteurs modélisent la voie MAPK induite par le récepteur EGF à l'aide d'un système de 94 équations pour 95 paramètres ; dans [Cha+02], les auteurs modélisent le métabolisme central du carbone ; finalement, les auteurs de [Li+08] modélisent la division cellulaire de *Caulobacter crescentus* à l'aide de 16 équations différentielles et 44 paramètres, tirés de la littérature et normalisés.

Sémantique stochastique des populations. Si les équations différentielles modélisent fidèlement des réactions chimiques pour des systèmes biologiques comportant un grand nombre de molécules d'une même espèce, ce n'est pas forcément le cas pour des systèmes comportant peu de molécules.

La sémantique stochastique permet de contourner cet écueil. Elle prend en considération la *stochasticité* des réactions chimiques, en modélisant les réactions entre de simples molécules plutôt qu'entre des ensembles de molécules. Dans cette sémantique, chaque réaction chimique a une certaine probabilité de se produire dans une temps imparti, qui dépend du nombre de molécules de chaque espèce et d'un paramètre lié à cette réaction.

Gillespie a proposé un algorithme exact de simulation d'un ensemble de réactions chimiques interprété par une sémantique stochastique, appelé SSA (pour Stochastic Simulation Algorithm) [Gil77]. Cet algorithme permet de simuler, au cours du temps, l'évolution des quantités des espèces chimiques d'un système, en prenant en compte la stochasticité des réactions impliquées.

Les deux sémantiques que nous avons présentées reposent sur l'utilisation de paramètres cinétiques, qui peuvent être difficiles à obtenir. Ces paramètres sont soit mesurés expérimaentalement, soit estimés par des expériences *in silico*. Par exemple, dans [Hei+12], les auteurs modélisent les voies G et β-arrestine induites par le récepteur de l'angiotensine à l'aide d'un systèmes d'équations différentielles, et estiment les paramètres cinétiques liés à ces équations en ajustant les évolutions temporelles obtenues à celles observées expérimentalement dans différentes conditions.

Analyse de flux. L'analyse de flux permet de calculer le flux de métabolites à travers un réseau métabolique, à l'état stationnaire. Les modèles d'analyse de flux ne permettent donc pas, contrairement aux modèles présentés précédemment, de calculer l'ensemble des comportements dynamiques des réseaux. Cependant, ce type d'analyse permet de découvrir les voies métaboliques d'un réseau métabolique fonctionnant à l'état stationnaire. Elles permettent donc une meilleure compréhension des

réseaux métaboliques, et ont des applications en ingénierie métabolique, notamment pour augmenter artificiellement la production d'un certain métabolite d'intérêt dans un organisme donné.

L'analyse de flux se sub-divise en un certain nombre de techniques différentes. Nous présentons ici deux de ces techniques : celle dite des *modes élémentaires de flux* (EFM) et l'*analyse de balance des flux* (FBA).

Un flux élémentaire d'un réseau de réactions est un ensemble minimal de réactions de ce réseau qui peuvent fonctionner ensemble à l'état stationnaire, et dont les réaction irréversibles fonctionnent dans le bon sens [SDF99]. L'analyse des flux permet par exemple de découvrir des voies métaboliques parallèles produisant *in fine* le même métabolite d'intérêt, et de déterminer laquelle de ces voies produit le plus de ce métabolite. Il est également possible de prédire l'effet du Knock-Down d'une certaine enzyme sur une voie métabolique en analysant les flux de cette voie dans laquelle la réaction catalysée par cette enzyme ne peut pas se produire.

Quant à l'analyse de balance des flux, c'est une technique permettant de calculer le rendement des flux [OTP10]. En particulier, cette technique permet de calculer l'ensemble des flux d'un réseau métabolique maximisant la production d'un certain métabolite ou un certain phénotype (comme la croissance).

Contrairement aux modèles construits avec la sémantique des ODE et la sémantique stochastique, l'analyse des flux ne nécessite pas de paramètres cinétiques. Ces analyses peuvent donc être utilisées telles quelles à partir d'un réseau métabolique.

1.4.2 Sémantiques qualitatives

Alors que les sémantiques quantitatives associent aux molécules ou activités moléculaires d'un réseau des quantités, les sémantiques qualitatives décrivent ces mêmes entités à l'aide de valeurs discrètes (comme le couple présent/absent). Ces sémantiques sont plus abstraites que les sémantiques quantitatives, et permettent de décrire la dynamique des systèmes avec moins de détails. Elles permettent cependant de capturer des propriétés importantes de la dynamique, comme des propriétés d'atteignabilité ou les états stables. De plus, ces sémantiques ne reposent pas sur des paramètres numériques comme des paramètres cinétiques, et peuvent donc être utilisées plus directement que les sémantiques quantitatives. Nous présentons trois sémantiques qualitatives : la sémantique Booléenne pour les graphes d'influences, la sémantique Booléenne de Biocham pour les réseaux de réactions, et la sémantique des réseaux de Thomas.

Sémantiques Booléennes. Des sémantiques Booléennes ont été proposées aussi bien pour modéliser les graphes d'influences que les réseaux de réactions. C'est Kauffman qui, en 1969, a proposé le premier d'interpréter des réseaux de régulation génétique par une sémantique Booléenne, formalisée à l'aide de réseaux Booléens [Kau69]. Cette sémantique a dès lors été appliquée à la modélisation des réseaux de régulation génétique (p. ex. [AO03] et [GG10]) et aux réseaux de signalisation (p. ex. [Zha+08],[SR+09] et [Flo+15]).

Dans la sémantique Booléenne appliquée aux graphes d'influences, les molécules ou activités peuvent être dans deux états : présent et absent. Une molécule ou activité peut alors passer d'un état absent à un état présent, et vice-versa, en fonction des états des molécules ou activités qui la modulent. Nous présenterons plus en détail cette sémantique dans le chapitre 5.

Une sémantique Booléenne pour les réseaux de réactions a également été proposée par les auteurs du logiciel Biocham [CFS06]. Dans la sémantique Booléenne de Biocham, chaque molécule peut également être soit absente soit présente. Chaque réaction chimique de la forme $A + B \to C + D$ est interprétée par quatre transitions, qui sont représentées sous la forme de formules Booléennes (ici, le symbole $\wedge$ représente l'opérateur logique de la conjonction) :

- $A \wedge B \rightarrow A \wedge B \wedge C \wedge D$
- $A \wedge B \rightarrow \neg A \wedge B \wedge C \wedge D$
- $A \wedge B \rightarrow A \wedge \neg B \wedge C \wedge D$
- $A \wedge B \rightarrow \neg A \wedge \neg B \wedge C \wedge D$

Les variables A et B dans le membre gauche expriment le fait que tous les réactifs de la réaction doivent être présents pour que la réaction ait lieu ; les variables C et D dans le membre droit expriment le fait que tous les produits de la réaction deviennent présents quand la réaction a lieu ; enfin, la combinaison des variables A et B, ou de leur négation, dans le membre droit exprime le fait que les réactifs peuvent être complètement ou partiellement consommés par la réaction.

Cette sémantique prend en compte les stimulations, et en particulier les catalyses, en ajoutant la variable Booléenne associée au stimulateur (ou catalyseur) dans les membres gauche et droit de chaque transition. Ainsi, la présence du stimulateur est nécessaire pour que la réaction se passe, au même titre que les réactifs, mais celui-ci n'est pas consommé pour autant.

Sémantique des réseaux de Tomas. Les *réseaux de Thomas* sont une extension des réseaux Booléens de Thomas [Tho73] apportée par Snoussi [Sno89].

Dans la sémantique Booléenne, les molécules ou activités peuvent prendre uniquement deux valeurs, présent ou absent. Or la modélisation par deux valeurs peut être insuffisante dans le cas où des molécules ont différentes activités suivant leur quantité dans le système. Il est alors plus réaliste de modéliser ces molécules ou activités par un ensemble fini de valeurs discrètes. Les molécules ou activités du graphe d'influence sont alors chacune modélisées par un ensemble fini de valeurs et, comme dans la sémantique Booléenne, une molécule ou activité passe d'un état à l'autre en fonction des états des molécules ou activités qui la modulent. À la différence de la sémantique Booléenne, la sémantique des réseaux de Thomas repose sur des paramètres numériques associés aux modulations, appelés *seuils*. Un seuil, associé à une modulation, indique les états dans lesquels doit être la molécule ou l'activité à la source de la modulation pour que cette modulation soit opérée.

Les réseaux de Thomas ont principalement été utilisés pour modéliser des réseaux de régulation génétique (p. ex. [TT95]).

1.4.3 Différents formalismes mathématiques

Les sémantiques que nous avons introduites donnent un sens dynamique aux réseaux moléculaires. Les modèles construits avec ces sémantiques peuvent être formalisés à l'aide d'un certain nombre de formalismes mathématiques bien connus :

- Les modèles construits avec la sémantique stochastique des populations peuvent être formalisés à l'aide de chaînes de Markov à temps continus ou de réseaux de Petri stochastiques [MRH12].
- Les modèles d'analyse des flux sont le plus souvent formalisés à l'aide de matrices représentant le réseau de réactions. Ils peuvent également être formalisés sous la forme de réseaux de Petri, dont les T-invariants minimaux sont par exemple les flux élémentaires du réseau de réactions [Sch+02b].
- Les modèles construits avec la sémantique Booléenne des graphes d'influences peuvent être formalisés de manière équivalente par des réseaux Booléens, des réseaux de Petri saufs (*safe*) [Cha+11], ou des réseaux d'automates [PAK13] ; et ceux construits avec la sémantique Booléenne de BIOCHAM par des règles logiques, des réseaux de Petri saufs, ou des réseaux d'automates (voir le chapitre 6).
- Les modèles construits avec la sémantique des réseaux de Thomas peuvent être formalisés de manière équivalente par des réseaux de Petri saufs [Cha+11] ou des réseaux d'automates [PMR11].

1.4.4 Analyse dynamique des modèles qualitatifs

Les modèles qualitatifs sont caractérisés par un ensemble fini d'*états globaux* et de *transitions* entre ces états. De manière générale, une suite d'états globaux de tels modèles atteints par des transitions successives est appelé une *trace*. La dynamique d'un modèle est alors l'ensemble des traces du modèle. Ces traces peuvent être représentées par un graphe de transitions, dont les noeuds correspondent aux états et les arcs aux transitions possibles.

À partir de certains états, plus aucun état ne peut être atteint. Ces états sont appelés *points-attracteurs*, et correspondent aux états stables du système biologique modélisé. De même, à partir de certains ensembles d'états, plus aucun état en dehors de l'ensemble ne peut être atteint. Ces ensembles sont appelés des *attracteurs cycliques*, et correspondent à des oscillations du système modélisé.

L'analyse dynamique des modèles qualitatifs comprend plusieurs problèmes : la *simulation* du modèle permet d'obtenir une trace de sa dynamique à partir de conditions initiales ; le *calcul des attracteurs* permet d'obtenir les états stables et les oscillations du système modélisé ; le *calcul du graphe de transitions* du modèle (*modèle checking*) permet d'obtenir la dynamique exhaustive du système ; et enfin, la *vérification* du modèle permet de vérifier si la dynamique du système modélisé contient un ensemble de comportements d'intérêt. Nous développons dans la suite la notion de vérification des modèles, que nous utiliserons dans le chapitre 6.

La vérification de modèles réfère à un ensemble de techniques informatiques visant à à vérifier la présence ou l'absence de comportements dans des modèles dynamiques. Les propriétés dynamiques à vérifier sont le plus souvent formalisées en logique temporelle [CE81], formalisme qui permet de décrire une trace ou un arbre d'exécution (i.e. un ensemble de traces). Des algorithmes génériques ont été produits pour vérifier l'adéquation d'un modèle dynamique à une propriété dynamique formalisée en logique temporelle [BKL08]. La vérification de modèles a été extensivement utilisée pour analyser les modèles dynamiques construits à partir de divers types de réseaux moléculaires (voir par exemple [RCB06] pour les réseaux de régulation génétique, et [Kwi+06] pour les réseaux de signalisation). Des exemple de propriétés dynamiques pertinentes pour les modèles dynamiques en biologie des systèmes comprennent l'atteignabilité d'un état comportant une certaine molécule ou activité, l'existence d'oscillations, ou encore leur période.

La complexité intrinsèque de la vérification de modèles limite, en théorie, son application à de grands réseaux. En effet, la vérification de propriétés dynamiques, mêmes des plus simples comme l'atteignabilité d'un état, est P-SPACE complète : en pratique, la mémoire nécessaire pour vérifier de telles propriétés est donc exponentielle par rapport au nombre de molécules des réseaux modélisés. Certains des algorithmes de vérification des modèles qualitatifs reposent sur les graphes de transitions engendrés par ces modèles. Pour des modèles d'une certaine taille, ce graphe est trop grand pour être contenu en mémoire. Par conséquent, des techniques de réduction de ces graphes, reposant notamment sur des représentations symboliques, ont été développées (voir p. ex. [CTM05]). D'autre part, nombre de récents travaux proposent des techniques améliorant l'applicabilité de la vérification en exploitant la concurrence des transitions [Cha+14], en utilisant des interprétations abstraites [PMR12], ou encore en réduisant préalablement les modèles tout en conservant leurs propriétés dynamiques [Nal+11].

1.5 Standards de la biologie des systèmes

Avec l'apparition de la biologie des systèmes comme une discipline à part entière au début des années 2000, de nombreux standards visant à en normaliser les concepts et représentations ont été développés. Ces standards facilitent la compréhension et l'échange des réseaux moléculaires et des mo-

dèles mathématiques produits par la discipline. Nous présentons dans la suite, sans être exhaustifs, les standards qui nous semblent les plus importants. Ces standards concernent la définition et l'organisation des termes du domaine sous forme d'ontologies, la représentation des réseaux moléculaires, la représentation des modèles mathématiques, et la représentation des simulations de modèles.

Ontologies. Les deux ontologies les plus importantes sont la Gene Ontology (GO) [Ash+00] et la Systems Biology Ontology (SBO) [JN13]. La GO définit un vocabulaire structuré et contrôlé qui organise les concepts utilisés pour décrire les gènes et leurs produits. Elle fournit trois ontologies : une pour les processus biologiques, une pour les composants cellulaires et une pour les fonctions moléculaires. De plus, la GO comporte une base de données d'annotations (GOA) de produits de gènes avec les termes de ses ontologies. Cette base comporte plus d'un million d'annotations pour environ 120000 produits de gènes. Quant à la SBO, c'est une ontologie regroupant les concepts utilisés en biologie des systèmes. Contrairement à la GO, les concepts définis et organisés dans cette ontologie ne sont pas restreints à ceux formalisant les gènes et produits de gènes, et intègrent des termes propres à la modélisation mathématique des systèmes. Cette ontologie a en effet été développée afin de pouvoir annoter les modèles mathématiques décrits en SBML (langage standard que nous introduisons plus loin). Elle est organisée en sept vocabulaires différents, relatifs aux concepts principaux suivants : expression mathématique, représentation des méta-données, framework de modélisation, représentation des entités qui se produisent, rôle des participants, représentation des entités physiques et description des paramètres des systèmes. Chaque terme de ces différents vocabulaires est associé à une courte définition.

Représentation des réseaux. La Systems Biology Graphical Notation (SBGN) [LN+09] est un langage standard visant à normaliser la représentation graphique des réseaux moléculaires. Elle comporte trois langages, chacun utilisé pour représenter un type différent de réseaux. Comme ce standard est à la base de l'ensemble des travaux présentés dans ce manuscrit, nous le décrivons en détail dans la prochaine section.

Le langage Biological Pathway Exchange (BioPAX) [Dem+10] vise à standardiser la représentation textuelle des réseaux moléculaires. Ce langage est implémenté sous la forme d'une ontologie, dont l'ensemble des termes permettent de représenter les entités (au sens large) et les processus moléculaires concrets rencontrés dans les différents types de réseaux (p.ex. une sous-voie, un pool d'entité physiques, une complexation, une catalyse). Chaque élément représenté peut être associé à divers attributs : par exemple, une réaction biochimique pourra être associée à une expérience ou une publication qui l'a mise en évidence, ou à une constante d'équilibre. Notons que ce standard ne permet pas de représenter les graphes d'influences : l'unité de base est le processus moléculaire en tant que transformation d'entités biologiques en d'autres entités biologiques, et les pools d'entités physiques ne peuvent influencer que des processus moléculaires.

Représentation des modèles mathématiques. Le Systems Biology Markup Language (SBML) [Huc+03] est un format standard qui permet d'écrire et échanger les modèles mathématiques de la biologie des systèmes. C'est un format XML qui a d'abord été développé pour représenter des modèles dynamiques quantitatifs de réseaux de réactions. Un modèle écrit au format SBML se compose d'un ensemble d'espèces chimiques et de réactions qui décrivent le réseau modélisé, et d'un ensemble de règles et de contraintes, associées à des paramètres, des fonctions mathématiques et des valeurs initiales, qui décrivent la dynamique des réactions. SBML offre également un ensemble de *packages* qui permettent le support de contenus additionnels en étendant le langage de base. Nous citerons en particulier SBML-qual [Cha+13] qui permet de prendre en compte les modèles qualitatifs. Avec ce package, les réactions ne représentent plus des réactions chimiques mais des transitions, et les valeurs initiales appartiennent

à un domaine discret et fini. Cette extension permet entre autres d'écrire des modèles Booléens, des réseaux de Thomas et des réseaux de Petri.

Représentation des simulations de modèles. Finalement, dans le cas particulier de la modélisation, nous pouvons également citer le Simulation Experiment Description Markup Language (SED-ML) [Wal+11b], qui fournit un format standard pour échanger et conserver des simulations de modèles dynamiques. Ce format est basé sur les recommandations données par le Minimum Information About a Simulation Experimentde (MIASE) [Wal+11a], qui décrit l'ensemble minimal d'informations qui doit être fourni pour qu'une simulation de modèle puisse être reproductible, et donc échangée.

1.6 Le standard SBGN

La Systems Biology Graphical Notation (SBGN) est un ensemble de trois langages graphiques permettant de représenter les réseaux moléculaires [LN+09].

Ces trois langages se nomment respectivement Process Description (SBGN-PD) [Moo+11], Activity Flow (SBGN-AF) [Mi+09] et Entity Relationship (SBGN-ER) [LN+11]. Chacun de ces trois langages permet de représenter un type de réseau particulier. Une représentation d'un réseau dans l'un ou l'autre de ces langages est appelée une *carte*.

Le langage SBGN-PD, dont les bases ont été données par la notation de Kitano [Kit03], permet de représenter des réseaux de réactions, ou plus généralement des réseaux comportant des processus moléculaires. Les différents types de processus pouvant être représentés en SBGN-PD sont les suivants : réactions chimiques, complexations, dissociations et translocations. Ces processus transforment des pools d'entités, qui représentent le plus souvent des pools de molécules. Enfin, d'éventuelles modulations de processus par les pools d'entités peuvent également être représentées. SBGN-PD est particulièrement approprié pour la représentation de changements d'états de pools d'entités, comme les phosphorylations ou les activations, qui peuvent être rencontrées dans les réseaux de signalisation, mais également pour la représentation de voies métaboliques, qui transforment, successivement, des métabolites en d'autres métabolites.

Le langage SBGN-AF, quant à lui, permet de représenter les graphes d'influences. Les objets biologiques représentés par ce langage sont des activités de molécules. Les activités d'une molécule sont à distinguer, de manière générale, de la molécule elle-même. Par exemple, une même molécule peut avoir à la fois une activité de complexation et une activité kinase. Ces deux activités pourront être représentées différemment en SBGN-AF. Cependant, il est également possible de représenter toutes les activités distinctes d'une même molécule par la même activité SBGN-AF. Dans ce cas-là, les activités de la molécule et la molécule elle-même peuvent être confondues. Les activités représentées en SBGN-AF peuvent s'influencer : telle activité aura une influence positive sur telle autre, et telle autre une influence négative sur une troisième. Ces influences sont représentées en SBGN-AF par des arcs d'influence. SBGN-AF est quant à lui approprié pour la représentation des réseaux de signalisation et des réseaux de gènes.

Finalement, le langage SBGN-ER, construit à partir du langage MIM (Molecular Interaction Map) [Koh99] de Kohn, permet de représenter des relations entre des entités biologiques. Ce langage est à part des deux autres, étant donné que chaque élément du langage est associé à une sémantique formelle (logique). Ainsi, ce langage n'a pas pour but de représenter des connaissances biologiques à proprement parler, mais propose déjà une interprétation logique de ces connaissances. L'interprétation d'une carte SBGN-ER n'est donc pas libre, contrairement aux cartes représentées avec les deux autres langages.

Notons que les réseaux de signalisation peuvent être représentés à la fois en SBGN-PD et en SBGN-AF. Nous étudierons la relation entre ces deux représentations, pour les réseaux de signalisation dans le chapitre 3.

L'unité syntaxique de base de chacun des trois langages est le *glyphe*. Certains glyphes, comme les opérateurs logiques ou certains arcs de modulations, sont communs aux trois langages. Nous pouvons distinguer trois types de glyphes : les noeuds, qui représentent les objets biologiques d'intérêt ; les décorateurs, qui représentent des attributs de ces objets ; et enfin les arcs, qui représentent les relations partagées par les objets biologiques représentés par les noeuds. Chaque langage est défini par un ensemble de glyphes généraux, qui représente l'alphabet du langage, et par une grammaire qui régit l'utilisation des glyphes du langage. Chaque glyphe de chaque langage est associé à un terme de SBO. Ainsi, les concepts généraux représentés par les glyphes (p.ex. une macromolécule) ont une définition biologique précise.

Nous nous concentrons, dans cette section et dans le reste du manuscrit, sur les langages SBGN-AF et SBGN-PD. Dans la suite, nous présentons en détail ces deux langages. Nous présentons le langage SBGN-AF Level 1 Version 1.0 et le langage SBGN-PD Level 1 Version 1.3.

1.6.1 SBGN Activity Flow

L'ensemble des glyphes du langage SBGN-AF est montré dans la carte de référence du langage donnée dans la figure 1.1.

Noeuds d'activité. Un noeud d'activité représente une activité opérée par une entité ou un pool d'entités. Les activités sont de trois types : *activité biologique*, *phénotype* et *perturbation*. Une activité peut être nommée par une étiquette contenue dans le glyphe représentant l'activité.

Noeud conteneur. Le seul noeud conteneur est le *compartiment*. Un compartiment peut être nommé par une étiquette.

Unité auxiliaire. La seule unité auxiliaire est l'*unité d'information*.

Lorsqu'une unité d'information décore un noeud d'activité, elle représente le type du pool d'entités qui opère cette activité. Le nom de l'entité ou du pool d'entités qui opère l'activité peut être indiqué par une étiquette contenue dans le glyphe représentant l'unité d'information. Les unités d'informations décorant les activités sont de cinq types : *macromolécule*, *espèce chimique simple*, *acide nucléique*, *non-spécifié* et *complexe*.

Lorsqu'une unité d'information décore un compartiment, elle représente une information abstraite relative à ce compartiment.

Arcs de modulation. Un arc de modulation représente l'influence d'une activité ou d'un ensemble d'activités sur une autre activité. Les arcs de modulation sont de quatre types : *influence positive*, *influence négative*, *influence inconnue*, et *stimulation nécessaire*.

Opérateurs logiques. Un opérateur logique permet de donner une condition nécessaire ou suffisante pour qu'une influence ait lieu. Les opérateurs logiques sont de quatre types : *AND*, *OR*, *NOT* et *delay*.

Arc logique. Un *arc logique* lie une activité ou un opérateur logique à un autre opérateur logique.

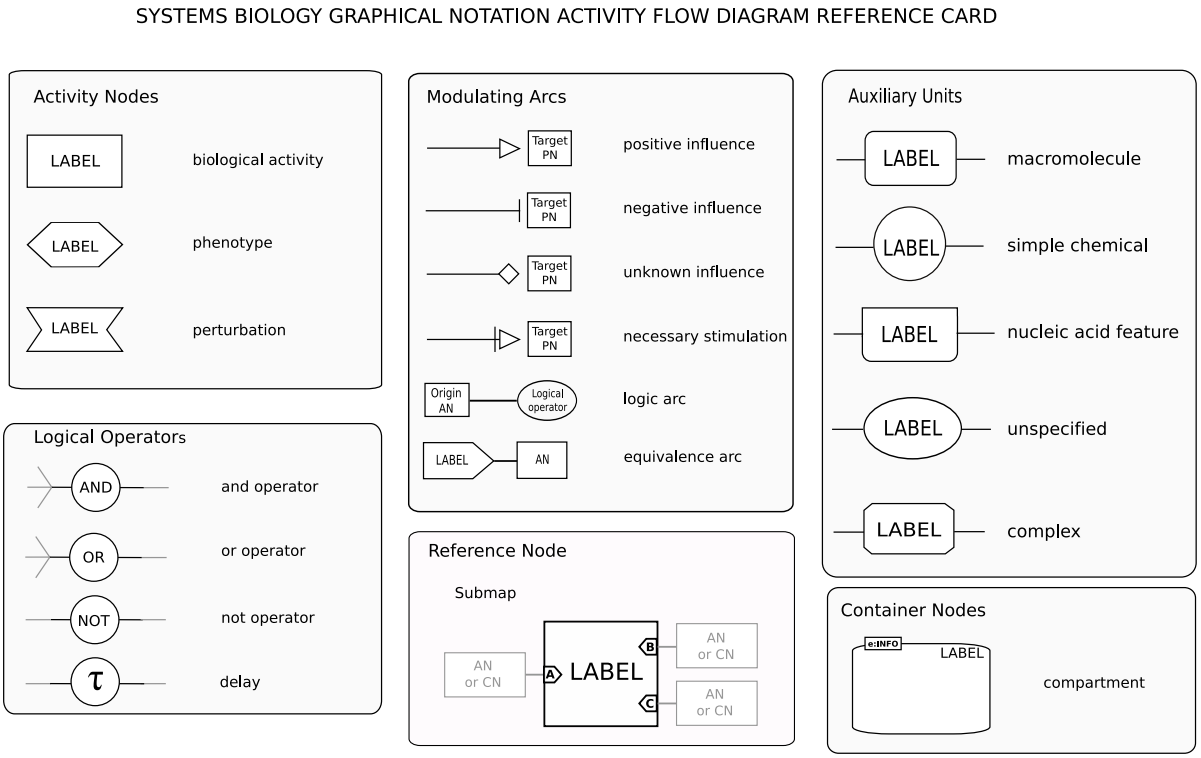


FIGURE 1.1 – **Carte de référence du langage SBGN-AF.**

Noeud de référence. Le seul noeud de référence est la *sous-carte*. Une sous-carte permet d'encapsuler un ensemble de noeuds d'activtés, d'arcs de modulations et d'opérateurs logiques dans un seul glyphe. Le contenu d'une sous-carte est ainsi caché par le glyphe qui la représente. Les noeuds d'activités qui sont des entrées de la sous-carte sont montrés sous la forme de glyphes décorant la sous-carte (par exemple, les glyphes A, B et C dans l'item *Reference Node* de la figure 1.1.

Arc d'équivalence. Un *arc d'équivalence* lie une activité d'une carte à une entrée d'une sous-carte, représentant l'équivalence entre l'activité de la carte et l'entrée de la sous-carte.

1.6.2 SBGN Process Description

L'ensemble des glyphes du langage SBGN-PD est montré dans la carte de référence du langage donnée dans la figure 1.2. Ce langage diffère de SBGN-AF par les concepts qu'il permet de représenter, qui sont des pools d'entités et des processus moléculaires précis.

Noeuds de pool d'entités Un *noeud de pool d'entités* (EPN) représente un pool d'entités toutes identiques. Les noeuds de pool d'entités sont de dix types : *entité non-spécifiée*, *espèce chimique simple*, *macromolécule*, *acide nucléique*, *agent perturbateur*, *source ou puits*, *complexe*, *multimère de macromolécules*, *multimère de complexes*, *multimère d'espèces chimiques simples* et *multimère d'acides nucléiques*.

Unités auxiliaires. Les unités auxiliaires sont des trois types suivants : *unité d'information*, *variable d'état* et *marqueur de clone*.

Une unité d'information décorant un glyphe représente une information abstraite relative au pool d'entités représenté. Cette unité contient une chaîne de caractères qui est le plus souvent de la forme "pre :label" où "pre" est un préfixe et "label" une étiquette, et qui représente par exemple le type matériel ou conceptuel du pool d'entités.

Lorsqu'une unité d'information décore un compartiment, elle représente une information abstraite relative à ce compartiment.

Une variable d'état représente un état physique ou biologique d'un pool d'entités. Elle contient une chaîne de caractères de la forme "Val@Var" où "Val" est la valeur affectée à la variable "Var". La valeur "Val" ou la variable "Var" peuvent être omises, et si c'est le cas, le symbole "@" est également omis.

Enfin, un glyphe représentant un pool d'entités peut être décoré par un marqueur de clone, qui indique que ce glyphe comporte un glyphe équivalent ailleurs sur la carte.

Noeud conteneur. Le seul noeud conteneur est le *compartiment*. Un compartiment peut être nommé par une étiquette.

Opérateurs logiques. Les opérateurs logiques sont de quatre types : *opérateur AND*, *opérateur OR*, *opérateur NOT* et *opérateur de délai*.

Arc logique. Un *arc logique* lie une activité ou un opérateur logique à un autre opérateur logique.

Noeuds de processus. Un *noeud de processus* représente un processus moléculaire. Les noeuds de processus sont de six types : *processus*, *processus omis*, *processus incertain*, *association*, *dissociation* et *phénotype*.

Arcs de flux. Un arc de processus lie un EPN à un noeud de processus. Les arcs de flux sont de deux types : *arc de consommation* et arc de *production*. Un arc de flux peut être décoré d'une unité d'information qui représente la stœchiométrie du processus relative à l'EPN lié au processus par l'arc.

Arcs de modulation. Un arc de modulation représente une modulation exercée par un EPN ou un ensemble d'EPNs sur un processus. Les arcs de modulation sont de quatre types : *modulation*, *stimulation*, *catalyse*, *inhibition* et *stimulation nécessaire*.

Noeuds de référence. Le seul noeud de référence est la *sous-carte*. Une sous-carte permet d'encapsuler un ensemble d'EPNs, de processus, d'arcs logiques et d'arcs de modulations. Le contenu d'une sous-carte est ainsi caché par le glyphe qui la représente. Les EPNs qui sont des entrées de la sous-carte sont montrés sous la forme de glyphes décorant la sous-carte.

Arc d'équivalence. Un *arc d'équivalence* lie un EPN d'une carte à une entrée d'une sous-carte, représentant l'équivalence entre l'EPN de la carte et l'entrée de la sous-carte.

1.7 Entrepôts de réseaux et de modèles dynamiques

Avec l'augmentation de la production de données cellulaires et moléculaires, le nombre de réseaux moléculaires construit à partir de ces données a explosé. Afin de rendre disponibles ces réseaux, ainsi que les modèles mathématiques construits à partir de ces réseaux, un certain nombre d'entrepôts de réseaux et de modèles ont été construits.

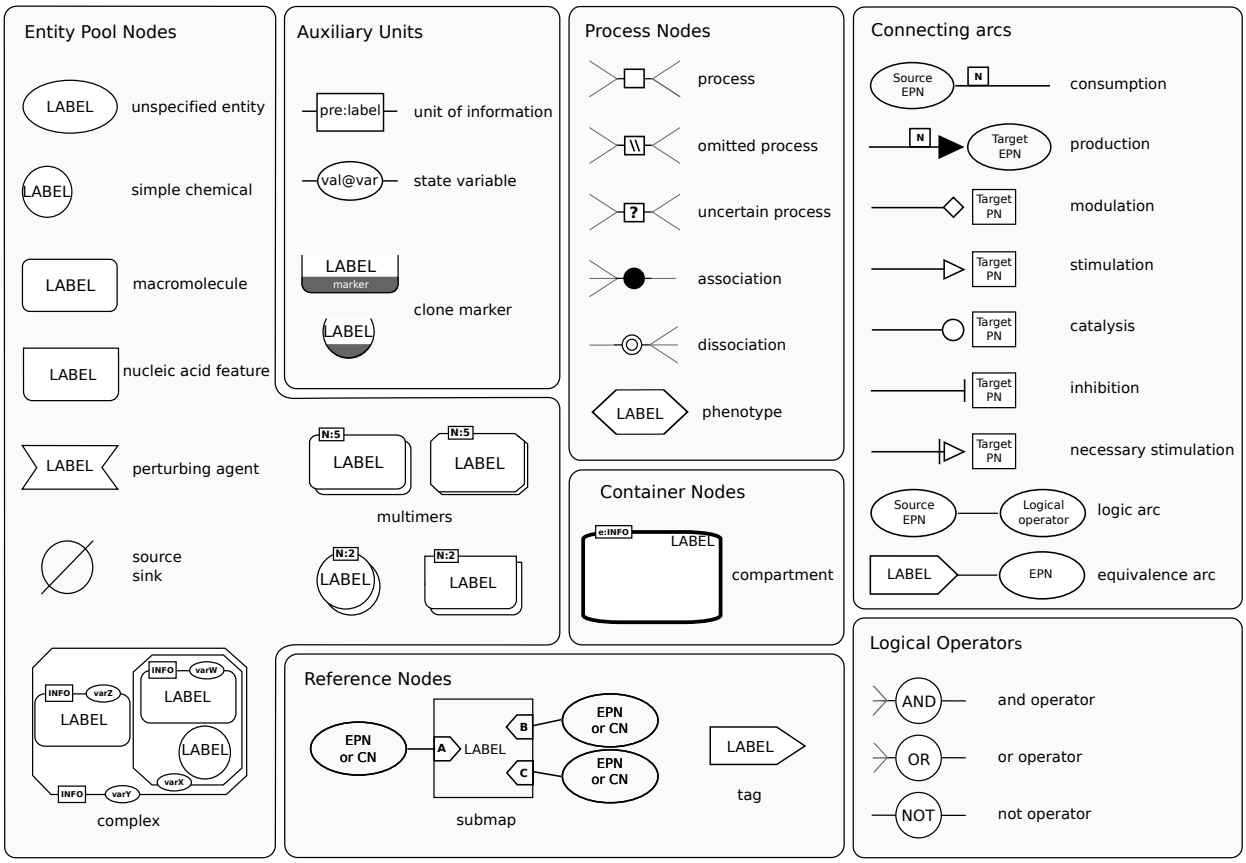


FIGURE 1.2 – **Carte de référence du langage SBGN-PD.**

Entrepôt de réseaux. Le plus important des entrepôts de réseaux moléculaires est sans doute la Kyoto Encyclopedia of Genes and Genomes (KEGG) [KG00]. Cet entrepôt regroupe un nombre considérable de réseaux métaboliques, ainsi qu'un nombre croissant de réseaux de signalisation. Les réseaux métaboliques y sont représentés sous la forme de réseaux de réactions, alors que les réseaux de signalisation sont représentés sous la forme de graphes d'influences. Les réseaux de KEGG sont interactifs : chaque élément de chaque réseau est cliquable et renvoie vers les autres bases de données de KEGG. Il est ainsi par exemple possible de connaître les orthologues d'un gène codant pour une enzyme, d'avoir des précisions sur une réaction chimique ou encore la séquence d'une protéine donnée d'un réseau. La navigation entre les réseaux est organisée sous forme de catégories désignant des grandes classes de processus biologiques, comme le métabolisme des acides aminés ou encore la biosynthèse des métabolites secondaires. Chaque réseau est également décliné sous la forme d'un réseau de référence et de réseaux spécifiques à des espèces données.

L'autre principal entrepôt de réseaux métaboliques est MetaCyc [Cas+08]. Cet entrepôt se focalise sur des voies métaboliques simples, là où KEGG représente des réseaux pouvant faire interagir plusieurs voies métaboliques distinctes (pour une comparaison détaillée des deux entrepôts, voir [Alt+13]). Nous pouvons aussi citer BRENDA [SCS02], qui se focalise également sur les réseaux métaboliques.

D'autres entrepôts que KEGG contiennent des réseaux de signalisation. C'est par exemple le cas de WikiPathways [Pic+08], qui est une plateforme collaborative de mise en ligne de réseaux moléculaires vérifiés manuellement, ou encore de Reactome, dont les réseaux sont représentés sous forme de cartes SBGN, et rendus disponibles en BioPAX. Les réseaux de ce dernier entrepôt ont également été vérifiés

à la main. Ils peuvent être explorés à différentes échelles. Dans la vue d'ensemble, les différents réseaux sont représentés par des noeuds et liés entre eux lorsqu'ils participent aux mêmes processus biologiques. Ces réseaux peuvent ensuite être cliqués pour être montrés dans leurs détails.

Finalement, de nombreux réseaux de signalisation sont diponibles dans BioCarta [Nis01]. Les réseaux sont tantôt représentés sous la forme de réseaux de réactions, tantôt sous la forme de graphes d'influences, et sont parfois un mélange des deux. Chaque voie de signalisation disponible est représentée sous la forme d'un réseau pour l'homme et d'un réseau pour la souris.

Entrepôts de modèles mathématiques. À ce jour, le principal entrepôt de modèles dynamiques (mathématiques) est Biomodels [LN+06]. Cet entrepôt utilise le format SBML pour conserver pas moins de 144546 modèles (au 29/04/2016), provenant de différentes sources. Parmi tous ces modèles, 1476 proviennent de la littérature. 610 d'entre eux ont été révisés manuellement, les autres étant disponibles tels que construits par leurs auteurs. Les quelques 143070 autres modèles de l'entrepôt ont été générés automatiquement à partir de réseaux biologiques par l'initiative Path2Models [Büc+13]. Les réseaux biologiques utilisés pour la génération de ces modèles sont de différentes natures, et proviennent de différents entrepôts de réseaux : de KEGG pour les réseaux métaboliques, de KEGG et MetaCyc pour les réseaux complets d'organismes, et de KEGG *non-metabolic* et BioCarta pour les réseaux de signalisation. Les modèles dynamiques de réseaux métaboliques ont été construits en interprétant chaque processus moléculaire par une ODE, et en important les paramètres cinétiques des processus à partir de la base de données de paramètres cinétiques SABIO-RK [Wit+12].

Chapitre 2

Programmation logique

Sommaire

2.1 Introduction

L'ensemble de nos travaux sont basés sur des formalismes qualitatifs, et en particulier sur des formalismes logiques. Nous introduisons ci-après la programmation logique, en définissant notamment la sémantique des modèles supportés et des modèles stables pour les programmes logiques, sémantiques que nous utiliserons par la suite. Enfin, nous introduisons l'Answer Set Programming, qui est une forme particulière de la programmation logique, et donnons quelques exemples de programmes. Nous renvoyons le lecteur à [CL14] pour les notions fondamentales du calcul propositionnel et de la logique du premier ordre.

2.2 Programmation logique

La programmation logique s'intéresse à exprimer un problème sous la forme de règles logiques et de contraintes explicites formant un *programme logique*. De tels programmes peuvent ensuite être exploités par des moteurs d'inférence ou des *solveurs*.

Alors que la logique du premier ordre utilise la négation dite *classique*, la programmation logique utilise une négation différente, dite *négation par l'échec* ou *négation par défaut*. Intuitivement, étant donnée une variable propositionnelle p, $\neg p \Rightarrow q$ ne signifie plus que si p est faux, alors on peut déduire q, mais que si on n'arrive pas à déduire p, alors on peut déduire q. Cette négation apporte à la logique du premier ordre la possibilité d'une inférence non monotone, et a été implémentée dans divers systèmes d'inférence logique. Un des premiers systèmes à avoir été développé est le système Prolog [CM03], au début des années 1970.

À la fin des années 1980, de nombreux travaux visant à la convergence des logiques dites non-monotones, et en particulier des logiques des défauts et auto-épistémique, et réalisés par Bidoit et Froidevaux d'une part, et Gelfond et Lifschitz d'autre part, ont mené à une caractérisation de la sémantique des programmes logiques utilisant la négation par défaut. Cette sémantique, appelée *sémantique des modèles stables*, a donné lieu à une nouvelle forme de programmation logique, dite *Answer Set Programming*.

Dans la suite, nous définissons d'abord la notion de programme logique, puis nous introduisons les sémantiques des modèles supportés et des modèle stables pour ces programmes (voir [Bid91] pour une revue). Enfin, nous introduisons l'Answer Set Programming.

2.2.1 Programmes logiques

Définition 2.1 (Programme logique)**.** Un *programme logique* est un ensemble de règles logiques de la forme :

$$H \leftarrow B_1 \wedge \cdots \wedge B_k \wedge \neg B_{k+1} \wedge \cdots \wedge \neg B_n$$

où $0 \leq k \leq n$, les B_i sont des atomes et H est un atome ou la constante $\bot$.

Ici, le symbole de négation $\neg$ réfère à la négation par défaut. Par tradition, l'implication de la logique classique est remplacée par le symbole $\leftarrow$, cependant, les programmes logiques sont d'un point de vue syntaxique des théories de la logique propositionnelle ou du premier ordre, et tous les concepts concernant la syntaxe de ces logiques (variables propositionnelles, atomes, littéraux, termes ...) restent valables pour les programmes logiques.

Si, pour chaque règle d'un programme, les B_i sont des variables propositionnelles et H est une variable propositionnelle ou la constante $\bot$, alors ce programme est un *programme logique propositionnel*. Si, au contraire, pour chacune des règles, les B_i sont des atomes formés à l'aide d'un symbole de prédicat et H est également un atome formé à partir d'un symbole de prédicat ou la constante $\bot$, alors ce programme est un *programme logique du premier ordre*.

Soit R une règle logique. Nous dénotons la tête de R par

$$head(R) = H$$

et son corps par

$$body(R) = \{B_1, \ldots, B_k, \neg B_{k+1}, \ldots, \neg B_n\}.$$

Nous dénotons par $body^+(R) = \{B_1, \ldots, B_k\}$ l'ensemble des atomes apparaissant positivement dans le corps de R, et par $body^-(R) = \{B_{k+1}, \ldots, B_n\}$ l'ensemble des atomes apparaissant négativement dans le corps de R. De façon analogue aux notations pour le corps d'une règle, pour une conjonction de littéraux $C = A_1 \wedge \ldots A_k \wedge \neg A_{k+1} \wedge \cdots \wedge \neg A_n$, nous dénotons par $C^+ = \{A_1, \ldots, A_k\}$ l'ensemble des atomes apparaissant positivement dans C, et par $C^- = \{A_{k+1}, \ldots, A_n\}$ l'ensemble des atomes apparaissant négativement dans C.

Nous avons les définitions suivantes, caractérisant R :

- si $body(R) = \emptyset$, alors R est un *fait* ;
- si $head(R) = \bot$, alors R est une *contrainte d'intégrité* ;
- si $head(R)$ est un atome, alors R est une *règle normale* ;

Un programme logique Π qui ne contient que des règles normales est un *programme logique normal*.

Exemple 2.1 (Programme logique normal)**.** L'ensemble des règles suivantes est un programme logique normal du premier ordre :

$$\begin{aligned} P(a) &\leftarrow \neg Q(a) \\ Q(X) &\leftarrow Q(X) \wedge R(X) \\ P(b) &\leftarrow \\ R(a) &\leftarrow \end{aligned}$$

Les règles $P(b) \leftarrow$ et $R(a) \leftarrow$ sont des faits. Pour des règles définissant des faits, le symbole $\leftarrow$ peut être omis.

Pour un programme logique propositionnel P donné, nous dénotons par $var(P)$ l'ensemble des variables propositionnelles de P. Pour un programme logique du premier ordre Π donné, nous dénotons par $const(\Pi)$ l'ensemble des symboles de constante apparaissant dans Π, par $func(\Pi)$ l'ensemble des symboles de fonction apparaissant dans Π, par $pred(\Pi)$ l'ensemble des symboles de prédicat apparaissant dans Π, et par $atom(\Pi)$ l'ensemble des atomes apparaissant dans Π. Pour un symbole de fonction ou de prédicat s de Π, nous dénotons par $ar(s)$ l'arité de s.

Les définitions suivantes sont relatives aux programmes logiques normaux (NLP) du premier ordre.

Définition 2.2 (Univers de Herbrand)**.** L'univers de Herbrand d'un NLP Π, noté $H_U(\Pi)$ est l'ensemble des termes instanciés construits avec les symboles de constante et symboles de fonctions de Π. Il est construit récursivement de la façon suivante :

- si $a \in const(\Pi)$, alors $a \in H_U(\Pi)$;
- si $f \in func(\Pi)$, $ar(f) = n$ et $t_1, \ldots, t_n$ sont des termes de $H_U(\Pi)$, alors $f(t_1, \ldots, t_n) \in H_U(\Pi)$.

Exemple 2.2 (Univers de Herbrand)**.** L'univers de Herbrand du NLP de l'exemple 2.1 est l'ensemble de termes $\{a, b\}$.

Définition 2.3 (Base de Herbrand)**.** La *base de Herbrand* d'un NLP Π, notée $H_B(\Pi)$, est l'ensemble des atomes instanciés construits à partir des symboles de prédicat de Π et des termes de l'univers de Herbrand de Π : $H_B(\Pi) = \{p(t_1, \ldots, t_n) \mid p \in pred(\Pi), ar(p) = n, t_1 \in H_U(\Pi), \ldots, t_n \in H_U(\Pi)\}$.

Exemple 2.3 (Base de Herbrand)**.** La base de Herbrand du NLP de l'exemple 2.1 est l'ensemble d'atomes suivant :

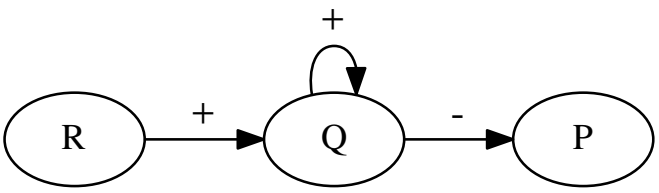
$$\{P(a), P(b), Q(a), Q(b), R(a), R(b)\}$$

Définition 2.4 (Interprétation de Herbrand)**.** Une *interprétation de Herbrand* d'un NLP Π est un sous-ensemble de sa base de Herbrand.

Définition 2.5 (Graphe de dépendance des prédicats)**.** Le graphe de *dépendance des prédicats* d'un NLP Π est un graphe orienté $G_{pred}(\Pi) = (V, A)$ construit de la manière suivante :

- à chaque symbole de prédicat $p \in pred(\Pi)$ est associé un noeud dans V ;
- à chaque couple de noeuds $(v_1, v_2) \in V^2$ est associé un arc positif (resp. négatif) de v_1 vers v_2 dans A *ssi* il existe une règle $R \in \Pi$ telle que le prédicat associé à v_1 apparaît dans un littéral positif (resp. négatif) du corps de R et le prédicat associé à v_2 apparaît dans la tête de R.

Exemple 2.4 (Graphe de dépendance des prédicats)**.** Le graphe de dépendance des prédicats du NLP de l'exemple 2.1 est le graphe suivant :



Le graphe de dépendance des prédicats d'un NLP permet de définir la notion de stratification d'un programme :

Définition 2.6 (Programme stratifié)**.** Un NLP Π est *stratifié ssi* son graphe de dépendance des prédicats $G_{pred}(\Pi)$ n'a pas de circuit comportant un arc étiqueté négativement.

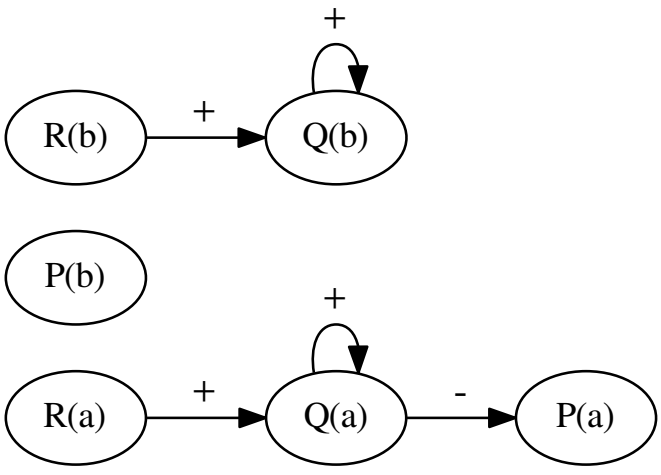
À chaque NLP du premier ordre Π correspond un programme instancié, obtenu en instanciant chaque règle de Π, c'est-à-dire en substituant, pour chaque règle, les variables de cette règle par des termes de l'univers de Herbrand de Π. Si Π est *finiement instanciable* (*finitely ground*), alors sa version instanciée est un ensemble fini de règles instanciées.

Dans la suite, nous considérons un NLP du premier ordre Π finiement instanciable, et assimilons ce programme à sa version instanciée.

Définition 2.7 (Graphe de dépendance des atomes)**.** Le *graphe de dépendance* des atomes d'un NLP Π est un graphe orienté $G_{atom}(\Pi) = (W, E)$ construit de la manière suivante :

- à chaque atome $a \in H_B(\Pi)$ est associé un noeud dans W ;
- à chaque couple de noeuds $(v_1, v_2) \in W^2$ est associé un arc positif (resp. négatif) de v_1 vers v_2 dans E *ssi* il existe une règle $R \in \Pi$ telle que l'atome associé à v_1 apparaît dans un littéral positif (resp. négatif) du corps de R et l'atome associé à v_2 apparaît dans la tête de R.

Exemple 2.5 (Graphe de dépendance des atomes)**.** Le graphe de dépendance des atomes du NLP de l'exemple 2.1 est le graphe suivant :



Le graphe de dépendance des atomes d'un NLP permet de définir une seconde notion de stratification :

Définition 2.8 (Programme fortement stratifié)**.** Le NLP Π est *fortement stratifié ssi* son graphe de dépendance des atomes $G_{atom}(\Pi)$ n'a pas de circuit.

2.2.1.1 Orbites, Modèles supportés, modèles stables

Nous avons donné la définition des programmes logiques et en particulier des programmes logiques normaux, et étudié quelques propriétés de leur structure. Nous étudions maintenant leur sémantique. Cette sémantique est valable pour les NLP propositionnels comme pour les NLP du premier ordre. Pour un NLP du premier ordre, nous considérons toujours une interprétation de Herbrand.

Définition 2.9 (Opérateur de conséquence immédiate)**.** Soit Π un NLP, et I une interprétation de Π. L'opérateur de conséquence immédiate de Π, noté T_Π, est défini de la façon suivante :

$$T_\Pi(I) \triangleq \{head(R) \mid R \in \Pi, body^+(R) \subseteq I, body^-(R) \cap I = \emptyset\}$$

Cet opérateur peut être appliqué récursivement. Nous définissons l'opérateur de conséquence récursif, noté T_Π^k, de la manière suivante :

- $T_\Pi^0(I) = I$;
- $T_\Pi^{k+1}(I) = T_\Pi(T_\Pi^k(I))$, pour $k \geq 0$.

Exemple 2.6 (Opérateur de conséquence immédiate)**.** Soit Π le NLP de l'exemple 2.1, et $I = \{Q(b), R(b)\}$ une interprétation de Π. L'interprétation de Π obtenue en appliquant l'opérateur de conséquence immédiate T_Π à I est l'ensemble d'atomes suivant :

$$T_\Pi(I) = \{P(b), R(a), P(a), Q(b)\}$$

Définition 2.10 (Orbite)**.** Soit Π un NLP, et I une interprétation de Π. L'orbite de I par rapport à Π est la séquence suivante :

$$\langle I, T_\Pi(I), T_\Pi^2(I), \ldots \rangle$$

Exemple 2.7 (Orbite)**.** Soit Π le NLP de l'exemple 2.1, et $I = \{Q(b), R(b)\}$ une interprétation de Π. L'orbite de I par rapport à Π est la séquence d'interprétations suivante :

$$\langle \{Q(b), R(b)\}, \{P(b), R(a), P(a), Q(b)\}, \{P(b), R(a), P(a)\}, \{P(b), R(a), P(a)\}, \ldots \rangle$$

avec $T_\Pi^k(I) = T_\Pi^2(I)$ pour $k \geq 2$.

Nous définissons trois sémantiques différentes pour les NLP : celle des modèles, celle des modèles supportés, et celle des modèles stables.

Définition 2.11 (Modèle)**.** Soit Π un NLP, et I une interprétation de Π. I est un *modèle* de Π *ssi* :

$$\forall R \in \Pi, body^+(R) \subseteq I \wedge body^-(R) \cap I = \emptyset \Rightarrow head(R) \in I$$

est vérifiée.

Exemple 2.8 (Modèle minimal)**.** Les modèles minimaux (au sens de l'inclusion) du NLP de l'exemple 2.1 sont les ensembles d'atomes suivants :

$$\{P(b), R(a), P(a)\} \text{ et } \{P(b), R(a), Q(a)\}$$

Définition 2.12. Soit Π un NLP, et I une interprétation de Π. I est un *modèle supporté* de Π *ssi* :

$$T_\Pi(I) = I$$

Exemple 2.9 (Modèle supporté)**.** Les modèles supportés du NLP de l'exemple 2.1 sont les ensembles d'atomes suivants :

$$\{P(b), R(a), P(a)\} \text{ et } \{P(b), R(a), Q(a)\}$$

Définition 2.13 (Programme logique normal réduit par rapport à une interprétation)**.** Soit Π un NLP, et I une interpétation de Π. La réduction de Π par rapport à I, notée Π^I, est le NLP construit à partir de Π en supprimant :

- chaque règle de Π contenant un littéral $\neg A$, tel que $A \in I$;
- les littéraux négatifs du corps des règles restantes.

Exemple 2.10 (Programme logique normal réduit par rapport à une interprétation)**.** Nous considérons le NLP de l'exemple 2.1, et l'interprétation $I = \{P(b), R(a), P(a)\}$ de ce NLP. Son programme réduit par rapport à I est l'ensemble de règles suivant :

$$\begin{aligned} &P(a)\\ &Q(a) \leftarrow Q(a) \wedge R(a)\\ &Q(b) \leftarrow Q(b) \wedge R(b)\\ &P(b)\\ &R(a) \end{aligned}$$

Définition 2.14 (Modèle stable)**.** Soit Π un NLP, et I une interprétation de Π. I est un *modèle stable* de Π *ssi* c'est un modèle minimal (au sens de l'inclusion) du programme Π^I.

Exemple 2.11 (Modèle stable)**.** L'unique modèle stable du NLP de l'exemple 2.1 est l'ensemble d'atomes $\{P(b), R(a), P(a)\}$. Notons que cet ensemble est un bien un modèle minimal du programme réduit par rapport à cet ensemble, et donné dans l'exemple 2.10.

Nous avons les relations suivantes entre les différentes sémantiques pour un NLP Π :

Propriété 2.1. Tout modèle supporté de Π est un modèle de Π.

Propriété 2.2. Tout modèle stable de Π est un modèle supporté de Π.

Enfin, nous avons les propriétés suivantes, qui lient la structure d'un NLP du premier ordre à l'ensemble de ses modèles stables ou supportés.

Soit Π un NLP du premier ordre.

Propriété 2.3. Si Π est stratifié, alors il admet un unique modèle stable.

Propriété 2.4. Si Π est fortement stratifié, alors il admet un unique modèle supporté.

Notons qu'un NLP fortement stratifié est stratifié, et qu'il admet donc un unique modèle supporté qui est également un modèle stable.

2.2.2 Règles de simplification et de transformation des NLP

Nous introduisons quatre règles de simplification et deux règles de transformation des NLP qui conservent les modèles supportés, et dont nous nous servirons au chapitre 5.

Règles de simplification : Étant donné un NLP Π et une règle R de ce programme logique, nous définissons les quatre règles de simplification suivantes, qui s'inspirent des transformations de Davis et Putnam pour un ensemble de clauses [DP60] :

(SR1) si $b \in body^+(R)$ et il existe une règle $R' \in \Pi$ telle que $R' = b \leftarrow$, alors supprimer b du corps de R ;

(SR2) si $b \in body^-(R)$ et il n'existe pas de règle $R' \in \Pi$ telle que $head(R) = b$, alors supprimer $\neg b$ du corps de R ;

(SR3) si $b \in body^+(R)$ et il n'existe pas de règle $R' \in \Pi$ telle que $head(R) = b$, alors supprimer R de Π ;

(SR4) si $b \in body^-(R)$ et il existe une règle $R' \in \Pi$ telle que $R' = b \leftarrow$, alors supprimer R de Π.

Ces règles conservent les modèles supportés :

Propriété 2.5. Soit Π un NLP et $R \in \Pi$ une règle de Π. Soit Π' le NLP obtenu en appliquant une des règles de simplification (SR1–4) à R. Π et Π' ont exactement les mêmes modèles supportés.

Cette propriété est établie dans l'annexe A.

Règle de transformation : Étant donné un programme logique Π et une règle R de ce programme, nous définissons les deux règles de transformation suivantes :

(TR1) si R est de la forme $h \leftarrow b \wedge C$, où C est une conjonction de littéraux, h et b sont des atomes tels que $h \neq b$ et il existe une règle S de Π telle que $head(S) = b$, alors remplacer R par l'ensemble de règles suivant :

$$\{h \leftarrow C \wedge \bigwedge_{l \in body(S)} l \mid S \in \Pi, head(S) = b\}$$

(TR2) si R est de la forme $h \leftarrow \neg b \wedge C$, où C est une conjonction de littéraux, h et b sont des atomes tels que $h \neq b$ et $\{S_1, \ldots, S_p\} = \{S \in \Pi \mid head(S) = b\} \neq \emptyset$, alors remplacer R par l'ensemble de règles suivant :

$$\{h \leftarrow C \wedge \bigwedge_{k=1}^{p} \neg l_k \mid l_1 \in body(S_1), \ldots, l_p \in body(S_p)\}$$

Ces règles conservent les modèles supportés :

> **Propriété 2.6.** Soit Π un NLP et $R \in \Pi$ une règle de Π. Soit Π' le NLP obtenu en appliquant une des règles de transformation (TR1–2) à un atome du corps de R. Π et Π' ont exactement les mêmes modèles supportés.

Cette propriété est établie dans l'annexe A.

2.2.3 Answer Set Programming

L'Answer Set Programming (ASP) est une forme purement déclarative de la programmation logique (là où Prolog avait une partie procédurale) basée sur la sémantique des modèles stables. Étant donné un programme logique encodé sous forme d'un programme ASP, un *ASP solveur* a pour but de trouver l'ensemble des modèles stables (aussi appelés *answer sets*) de ce programme.

Cette tâche est en fait divisée en deux étapes : étant donné un programme ASP, un *grounder* se charge d'abord d'instancier et de simplifier le programme logique ; puis le *solveur* se charge de trouver l'ensemble des modèles stables du programme instancié et simplifié.

La syntaxe des programmes ASP est de manière générale très proche de celle des programmes logiques. Les principales différences entre les deux syntaxes sont les suivantes :

- le symbole "$\leftarrow$" est remplacé par le symbole " :- " ;
- le symbole "$\wedge$" est remplacé par le symbole "; " ou le symbole "," ;
- le symbole "$\neg$" est remplacé par le symbole "not ".

Le Listing 2.1 donne le programme ASP correspondant au programme logique de l'exemple 2.1.

Listing 2.1 – Exemple de code ASP

```
1 P(a):-not Q(a).
2 Q(X):-Q(X);R(X).
3 P(b).
4 R(a).
```

Les grounders les plus courants, que sont gringo et DLV, acceptent également des éléments qui ne peuvent pas directement être écrits dans la syntaxe des programmes logiques. Par exemple, gringo accepte en plus les agrégateurs ou les "wild cards". Cependant, la sémantique des modèles stables a bien été définie pour ces éléments.

Nous donnons ci-après les éléments de syntaxe des programmes ASP n'appartenant pas à la syntaxe des programmes logiques classiques et que nous utilisons dans divers programmes ASP montrés dans ce manuscrit. La syntaxe que nous utilisons est celle du grounder gringo, et le solveur que nous utilisons est clasp. L'ensemble gringo + clasp est nommé clingo.

Littéral conditionnel. Un *littéral conditionnel* est de la forme $L_0 : L_1, \ldots, L_n$ où les L_i sont des littéraux. On dit que L_0 est conditionné à $L_1, \ldots, L_n$: un tel littéral est remplacé par L_0 si tous les littéraux $L_1, \ldots, L_n$ sont vrais, et par l'ensemble vide sinon.

Pour un littéral conditionnel, les littéraux L_i le composant peuvent ne pas être instanciés, i.e. ils peuvent comporter des variables. De tels littéraux permettent d'exprimer des collections de littéraux. Par exemple, le littéral conditionnel $a(X) : b(X)$ sera remplacé par la collection d'atomes $a(t_1); \ldots; a(t_n)$, où les termes $t_1, \ldots, t_n$ sont tels que les atomes $b(t_1), \ldots, b(t_n)$ sont toutes les instances vraies du prédicat b. Cette collection représente une conjonction si elle se trouve dans le corps d'une règle et une disjonction si elle se trouve dans la tête d'une règle.

Exemple 2.12 (Programme ASP avec un littéral conditionnel)**.** Le Listing 2.2 donne un exemple de programme ASP avec un littéral conditionnel. Le problème est le suivant : nous avons un ensemble de pommes, qui ont chacune une couleur, et nous voulons savoir si l'intégralité des pommes est de couleur rouge. Les lignes 1–6 définissent les pommes et leur couleur. La ligne 8 comporte une règle avec un littéral conditionnel dans son corps. Ce littéral conditionnel est remplacé par la conjonction $color(a1, red)$; $color(a2, red)$; $color(a3, red)$. Par conséquent, la variable propositionnelle $allRed$ ne sera vraie que si les trois pommes sont de couleur rouge. Comme ce n'est pas le cas, $allRed$ n'appartient pas à l'unique answer set de ce programme.

Listing 2.2 – Programme ASP avec un littéral conditionnel

```
1 apple(a1).
2 apple(a2).
3 apple(a3).
4 color(a1,red).
5 color(a2,green).
6 color(a3,red).
7
8 allRed:-color(A,red):apple(A).
```

Agrégats. Les agrégats peuvent se retrouver dans le corps ou dans la tête des règles ASP.

Dans le corps d'une règle, un agrégat a la forme générale suivante :

$$s_1 \prec_1 \alpha \{\boldsymbol{t_1} : \boldsymbol{L_1}; \ldots; \boldsymbol{t_n} : \boldsymbol{L_n}\} \prec_2 s_2$$

où s_1 et s_2 sont des termes, $\prec_1$ et $\prec_2$ des opérateurs de comparaison, α est un nom de fonction, les $\boldsymbol{t_i}$ des tuples de termes et les $\boldsymbol{L_i}$ des tuples de littéraux. Pour chaque tuple de terme $\boldsymbol{t_i}$, le premier de ses termes est le *poids* du terme.

Les opérateurs de comparaison $\prec_1$ et $\prec_2$ peuvent être les opérateurs suivants : $=$, $<$, $<=$, $>$, $>=$ et $!=$. Quant à la fonction α, elle peut être une des fonctions suivantes : $\#count$, $\#sum$, $\#sum+$, $\#min$ et $\#max$. L'élément de syntaxe $\boldsymbol{t_i} : \boldsymbol{L_i}$ est appelé un *élément de l'agrégat*. De façon analogue aux littéraux conditionnels, les termes des $\boldsymbol{t_i}$ peuvent être des variables et les littéraux des $\boldsymbol{L_i}$ des littéraux non-instanciés. Ainsi, un élément d'agrégat $\boldsymbol{t_i} : \boldsymbol{L_i}$ peut représenter une collection de tuples de termes instanciés. Par exemple, l'élément d'agrégat $1, X + Y : a(X, Y)$ sera remplacé par les tuples $1, t_{X_1} + t_{Y_1}; \ldots; 1, t_{X_n} + t_{Y_n}$, où les termes $t_{X_1}, t_{Y_1}, \ldots, t_{X_n}, t_{Y_n}$ sont tels que les atomes $a(t_{X_i}, t_{Y_i})$ sont toutes les instances vraies du prédicat a. Un agrégat sera vrai si la fonction α appliquée à l'ensemble de tuples représenté entre les accolades respecte les comparaisons données par $s_1 \prec_1$ et $\prec_2 s_2$. La fonction $\#count$ compte le nombre de tuples ; la fonction $\#sum$ fait la somme des poids de chaque tuple ; la fonction $\#sum+$ la somme des poids positifs seulement ; $\#min$ renvoie le poids minimal de l'ensemble des tuples ; et $\#max$ le poids maximal. Notons que la collection représentée entre accolades est un ensemble et non un multi-ensemble. Ainsi, si deux tuples sont égaux, ils ne seront considérés qu'une seule fois par les fonctions $\#count$, $\#sum$ et $\#sum+$. Les opérateurs de comparaison $\prec_1$ et $\prec_2$ peuvent être omis, et c'est dans ce cas l'opérateur $<=$ qui est considéré par défaut. De même, la fonction α peut être omise, et c'est alors la fonction $\#count$ qui est considérée.

Dans le reste du manuscrit, nous n'utiliserons que la fonction $\#count$.

Exemple 2.13 (Programme ASP avec un agrégateur dans le corps d'une règle)**.** Le Listing 2.3 donne un exemple de programme ASP avec un agrégat dans le corps d'une règle. Le problème est le suivant : comme pour l'exemple précédent, nous avons un ensemble de pommes qui ont chacune une couleur, et nous voulons vérifier qu'il y a bien deux pommes rouges, compter le nombre de pommes vertes, et, finalement, vérifier que le nombre de pommes vertes est égal au nombre de pommes rouges. Les lignes 1-8 définissent les pommes et leur couleur. La ligne 10 vérifie qu'il y a bien deux pommes rouges. Comme les opérateurs $\prec_1$ et $\prec_2$ sont omis, c'est l'opérateur $<=$ qui est utilisé dans les deux cas. La ligne 12 compte le nombre de pommes vertes. Pour cette règle, l'agrégat compte comme un atome positif, et $N =$ affecte le résultat de la fonction $\#count$ à N. Finalement, la ligne 14 vérifie que le nombre de pommes vertes est égal au nombre de pommes rouges. La première occurrence de $N =$ affecte le nombre de pommes vertes à N (de façon analogue au corps de la règle de la ligne 12), et la deuxième compare le nombre de pommes rouges à la valeur de N.

Comme il y a deux pommes rouges et deux pommes vertes, l'unique answer set de ce programme contient les atomes *twoRedApples* et *equalRedGreen*, ainsi que l'atome *numberOfGreenApples*(2).

Listing 2.3 – Programme ASP avec un agrégateur dans le corps d'une règle

```
1  apple(a1).
2  apple(a2).
3  apple(a3).
4  apple(a4).
5  color(a1,red).
6  color(a2,green).
7  color(a3,red).
8  color(a4,green).
9
10 twoRedApples:-2 #count {apple(A):color(A,red)} 2.
11
12 numberOfGreenApples(N):- N= #count {apple(A):color(A,green)}.
13
```

```
14 equalRedGreen:-#count {apple(A):color(A,red)} = N; N= #count {apple(A):color
       (A,green)}.
```

Dans la tête d'une règle, un agrégat à la forme générale suivante :

$$s_1 \prec_1 \alpha \ \{\boldsymbol{t_1} : L_1 : \boldsymbol{L_1}; \ldots; \boldsymbol{t_n} : L_n : \boldsymbol{L_n}\} \prec_2 s_2$$

où tous les éléments sont les mêmes que ceux définis pour les agrégats du corps des règles, et les L_i sont des littéraux. Ce sont ces littéraux qui seront dérivés par la règle comportant l'agrégat. Notons que pour ces types d'agrégats, les tuples $\boldsymbol{L_i}$ peuvent être omis, les littéraux conditionnels $L_i : \boldsymbol{L_i}$ devenant alors de simples littéraux. Les tuples de termes peuvent également être omis, et seront construits automatiquement par gringo.

Les éléments d'agrégats sont ici remplacés par des tuples de termes instanciés, chacun associé, par la syntaxe usant de ":", à un littéral également instancié.

Si le corps de la règle comportant l'agrégat est vrai, pour chaque sous-ensemble de l'ensemble représenté entre accolades qui respecte les comparaisons $s_1 \prec_1$ et $s_2 \prec_2$, tous les littéraux associés aux tuples de ces sous-ensembles sont dérivés, dans des modèles différents.

Exemple 2.14 (Programme ASP avec un agrégateur dans la tête d'une règle)**.** Le Listing 2.4 donne un exemple de programme ASP avec un agrégateur dans la tête d'une règle. Le problème est le suivant : nous avons toujours un ensemble de pommes, de différentes couleurs. Nous avons également Louis A., qui aime la couleur rouge. Une personne qui aime une certaine couleur ne peut manger des pommes que de la couleur qu'elle aime, et elle ne peut manger qu'une seule pomme à la fois. Étant donnée une personne, si plusieurs pommes de la couleur qu'elle aime sont disponibles, il faut qu'elle en choisisse une, pour la manger. Les lignes 1–6 définissent les pommes et leur couleur, et les lignes 7–8 définissent Louis et la couleur qu'il aime. La ligne 10 définit quelle pomme est choisie par une personne qui aime une certaine couleur. Notons qu'ici, le tuple de termes est omis : il sera donc construit automatiquement par gringo.

Ce programme a deux modèles stables, un pour chaque pomme rouge : le premier comporte l'atome $eat(a1)$, le deuxième l'atome $eat(a3)$. L'utilisation d'un agrégateur dans la tête d'une règle comportant la fonction *#count* (qui est ici omise) permet de réaliser un *choix* parmi différents atomes.

Listing 2.4 – Programme ASP avec un agrégateur dans la tête d'une règle

```
1  apple(a1).
2  apple(a2).
3  apple(a3).
4  color(a1,red).
5  color(a2,green).
6  color(a3,red).
7  person(louis).
8  loves(louis,red).
9
10 1 {eats(P,A):color(A,C)} 1:-loves(P,C);person(P).
```

Wild card. L'utilisation d'une wild card, de symbole "_", à la place d'une variable d'un atome permet d'omettre cette variable lors de l'instanciation de cet atome. La wild card s'utilise en particulier dans les littéraux négatifs. Par exemple, la règle $a(Y)$:– $b(Y)$; not $c(\_, Y)$ permet de dériver l'atome $a(t)$,

où t est un terme instancié, si $b(t)$ est établi et il n'existe pas de terme t' tel que $c(t', t)$ peut être déduit. La wild card est un raccourci qui permet d'omettre l'utilisation d'un prédicat auxiliaire et d'une règle auxiliaire.

Exemple 2.15 (Programme ASP avec une wild card)**.** Le Listing 2.6 donne un exemple de programme ASP utilisant une wild card, et le Listing 2.5 le programme équuivalent mais sans wild card. Le problème est le suivant : nous avons encore et toujours un ensemble de pommes, de différentes couleurs. Notons que cette fois-ci, nous pouvons ne pas connaître la couleur d'une pomme donnée, i.e. le prédicat *color* n'est pas défini pour cette pomme. Nous aimerions savoir quelles pommes n'ont pas de couleur définie. Les lignes 1–5 de chacun des deux programmes définissent nos pommes et leur couleur, la couleur de la pomme $a3$ n'étant pas définie. La ligne 7 du programme sans wild card permet de dériver quelles pommes ont une couleur définie, et la ligne 8 quelles pommes n'ont pas de couleur définie. Ces deux lignes sont résumées par une seule ligne dans le programme avec wild card, qui fait l'économie du prédicat intermédiaire *hasColor*. Les deux programmes ont un unique modèle stable comportant l'atome $unknownColor(a3)$, qui signifie que la pomme $a3$ n'a pas de couleur définie.

Listing 2.5 – Programme ASP sans wild card

```
1 apple(a1).
2 apple(a2).
3 apple(a3).
4 color(a1,red).
5 color(a2,green).
6
7 hasColor(A):-color(A,C);apple(A).
8 unknownColor(A):-not hasColor(A);apple(A).
```

Listing 2.6 – Programme ASP avec wild card

```
1 apple(a1).
2 apple(a2).
3 apple(a3).
4 color(a1,red).
5 color(a2,green).
6
7 unknownColor(A):-not color(A,_);apple(A).
```

Optimisation. Une déclaration d'optimisation est de la forme générale suivante :

$$op\ \{w_1@p_1, \boldsymbol{t_1} : \boldsymbol{L_1}, \ldots, w_n@p_n, \boldsymbol{t_n} : \boldsymbol{L_n}\}.$$

où op est soit $\#maximize$ soit $\#minimize$, les w_i sont des entiers représentant des poids, les p_i des priorités, les $\boldsymbol{t_i}$ des tuples de termes, et les $\boldsymbol{L_i}$ des tuples de littéraux. Les priorités peuvent être omises, et nous traiterons de ce cas-là ici.

De la même façon que pour les agrégateurs du corps des règles, les $\boldsymbol{t_i} : \boldsymbol{L_i}$ représentent des ensembles de tuples de termes. Chaque tuple de termes de l'ensemble représenté entre accolades participe à hauteur du poids qui lui est associé à une fonction de coût. Si $op = \#maximize$ (resp. $op = \#minimize$), un modèle stable sera optimal si le coût qui lui est associé est maximal (resp. minimal) parmi tous les coûts de tous les modèles stables.

Exemple 2.16 (Programme ASP avec optimisation)**.** Le Listing 2.7 donne un exemple de programme ASP avec optimisation. Le problème est le suivant : nous avons comme toujours un ensemble de pommes, qui sont chacune d'une couleur, et nous voulons choisir la couleur qui est la plus représentée parmi les pommes. Les lignes 1–6 définissent les pommes et leur couleur. La ligne 8 permet de choisir arbitrairement une couleur parmi toutes les couleurs représentées. Quant à la ligne 10, elle permet de ne sélectionner que les modèles stables où la couleur choisie est celle partagée par un maximum de pommes : chaque pomme de la couleur choisie se voit associer un poids de 1, et à chaque modèle stable, i.e. à chaque choix de couleur, est associé un poids qui est la somme des poids des pommes qui ont la couleur choisie. Ce programme a deux answer sets, un pour chaque couleur : le premier comporte l'atome $chosenColor(green)$, le deuxième l'atome $chosenColor(red)$. Seul le deuxième modèle stable est optimal, étant donné que nous avons deux pommes rouges et une seule pomme verte.

Listing 2.7 – Programme ASP avec optimisation

```
1  apple(a1).
2  apple(a2).
3  apple(a3).
4  color(a1,red).
5  color(a2,green).
6  color(a3,red).
7
8  1 {chosenColor(C):color(A,C)} 1.
9
10 #maximize {1,A:color(A,C),chosenColor(C)}.
```

Fonctions externes. Finalement, clingo permet de définir des fonctions Python, et de les intégrer aux programmes ASP. Par exemple, nous pouvons définir une fonction Python de concaténation des chaînes de caractères, et placer un appel de cette fonction comme argument d'un prédicat.

Deuxième partie

Représentation et construction des réseaux moléculaires en logique

Chapitre 3

SBGNLog : traduction des langages SBGN en logique

Sommaire

3.1 Introduction

Comme nous l'avons déjà expliqué dans la section 1.6, la Systems Biology Graphical Notation (SBGN) regroupe trois langages pour représenter les réseaux biologiques. Nous nous concentrons dans ce chapitre sur les langages SBGN Process Description (SBGN-PD) et SBGN Activity Flow (SBGN-AF). Ces langages permettent de représenter les réseaux de façon graphique : ils sont constitués de glyphes qui sont soit des noeuds, soit des arcs, et qui permettent chacun de représenter un concept particulier rencontré dans les réseaux.

La représentation graphique d'un réseau biologique permet une meilleure compréhension des relations qui unissent les différents éléments du réseau, et permet notamment d'avoir une vue holistique de ces relations. Ces représentations permettent également à ceux qui les visualisent de raisonner sur les réseaux représentés. Cependant le raisonnement automatique sur ces réseaux, réalisé par l'intermédiaire d'un programme sur ordinateur, nécessite d'en avoir une autre représentation, dite *représentation en machine*. Le format SBGN-ML [VI+12] et la librairire libSBGN [VI+12] permettent d'obtenir de telles représentations. En effet, le format SBGN-ML permet de stocker et d'échanger des cartes SBGN-ML dans un format facilement lisible par un ordinateur, tandis que la librairie libSBGN permet de lire une carte SBGN au format SBGN-ML et de créer une représentation machine de cette carte sous la forme d'instances JAVA ou Python, stockées en mémoire. Le raisonnement automatique peut alors se faire à partir de telles instances à l'aide d'algorithmes, et à chaque tâche de raisonnement correspond un algorithme différent.

Le raisonnement sur des réseaux moléculaires peut également être réalisé à l'aide de théories logiques et de moteurs d'inférence. De tels formalismes ont d'ailleurs été utilisés pour raisonner sur des réseaux moéculaires dans de nombreux travaux. Nous pouvons par exemple citer les travaux suivants :

dans [Vid+12], l'Answer Set Programming (ASP) est utilisé pour paramétrer des réseaux Booléens modélisant des graphes d'influences, à l'aide de résultats expérimentaux ; dans [IDN13], les auteurs utilisent la logique du premier ordre et le logiciel de consequence finding SOLAR [Nab+10] pour raisonner sur la causalité dans les graphes d'influences ; finalement, les auteurs de [RSI12] calculent les états stationnaires de réseaux de réaction à l'aide de programmes logiques, encodés sous forme de programmes ASP.

L'utilisation de formalismes logiques pour raisonner sur les réseaux moléculaires nécessite d'abord d'en avoir une représentation dans ces formalismes. Il faut que le langage utilisé permette à la fois d'exprimer tous les concepts biologiques contenus dans les réseaux, mais également les différentes règles de raisonnement. Les travaux que nous avons cités proposent chacun leur propre langage, qui est souvent assez peu détaillé : les graphes d'influences et les réseaux de réactions sont représentés sous forme de formules propositionnelles ou de prédicats comme seraient représentés de simples graphes dont les arêtes seraient étiquetées. Les représentations que nous avons recensées ne prennent par exemple pas en compte les modifications post-traductionnelles des protéines ou les stimulations nécessaires. De plus, ces représentations sont parfois spécifiques à la tâche de raisonnement considérée, et ne peuvent donc dans ce cas pas être réutilisées pour d'autres tâches.

Nous proposons dans ce chapitre deux langages de la logique du premier ordre basés sur SBGN qui permettent de formaliser les réseaux de réactions et les graphes d'influences. Ces deux langages permettent de formaliser en logique l'ensemble des concepts biologiques qui peuvent être représentés par les langages SBGN-AF et SBGN-PD. Ces langages, que nous appelons SBGNLog-AF et SBGNLog-PD, sont la correspondance, en logique du premier ordre, des langages SBGN-AF et SBGN-PD, respectivement. Il s'ensuit qu'un réseau peut directement être exprimé en SBGNLog-AF ou SBGNLog-PD (suivant sa nature) sous la forme d'atomes instanciés formés à partir du vocabulaire de ces langages, mais également que n'importe quelle carte SBGN-AF ou SBGN-PD peut être traduite dans le langage SBGNLog correspondant. Nous appelons une réprésentation d'un graphe d'influences en SBGNLog-AF une *carte SBGNLog-AF*, et la représentation d'un réseau de réactions en SBGNLog-PD une *carte SBGNLog-PD*.

Étant basés sur les concepts biologiques pris en compte par SBGN, les langages SBGNLog permettent d'exprimer de façon standard les représentations logiques des réseaux moléculaires ainsi que la formalisation des règles de raisonnement, de sorte qu'une même représentation en logique d'un réseau donné puisse être utilisée pour différentes tâches de raisonnement.

Nous introduisons dans la suite de ce chapitre les deux langages SBGNLog-AF et SBGNLog-PD, et le processus de traduction des cartes SBGN-AF et SBGN-PD dans ces deux langages. Nous présentons le vocabulaire des deux langages en même temps que le processus de traduction d'une carte dans l'un ou l'autre langage.

Le reste de ce chapitre est organisé comme suit. Nous donnons d'abord des considérations générales sur les deux langages SBGNLog, puis nous les présentons l'un après l'autre. Nous donnons ensuite un exemple de tâche de raisonnement réalisée à l'aide de ces deux langages : nous proposons une transformation des cartes SBGNLog-PD représentant des réseaux de signalisation vers des cartes SBGNLog-AF, qui sont plus simples et plus concises. Enfin, nous discutons des différences entre les langages SBGNLog-AF et SBGNLog-PD et d'autres langages de représentation des cartes SBGN, et comparons notre méthode de transformation des cartes SBGNLog-PD en cartes SBGNLog-AF avec la transformation des cartes SBGN-PD en cartes SBGN-AF introduite par les auteurs de [VCS13].

3.2 SBGNLog et traduction des cartes SBGN

3.2.1 Considérations générales

Sauf mention contraire, nous prenons en considération la version 1.3 de la spécification de SBGN-PD et la version 1.0 de celle de SBGN-AF.

3.2.1.1 Représentation des concepts plutôt que des glyphes

Les deux langages SBGNLog-AF et SBGNLog-PD visent à représenter en logique du premier ordre les concepts présents dans les cartes SBGN-AF et SBGN-PD, respectivement. Il ne s'agit donc pas de représenter en logique les glyphes d'une carte SBGN, ou encore leurs coordonnées dans la carte, mais seulement les concepts biologiques qui peuvent être représentés par les glyphes d'une carte. À la plupart des différents types de glyphe de chacun des deux langages correspondent des concepts de la biologie des systèmes liés à des termes de SBO. Par exemple, une protéine sera représentée en SBGN-PD par un rectangle au bord arrondi, et ce type de glyphe est associé au terme *macromolecule* de SBO. La protéine ERK non modifiée sera donc représentée dans une carte SBGN-PD par un glyphe de ce type, comportant l'étiquette "ERK". Cette protéine peut être représentée plusieurs fois dans la même carte, c'est-à-dire par plusieurs glyphes différents, portant chacun un *clone marker*. Celui-ci indique que des glyphes différents représentent le même concept, ici la *protéine ERK*. Comme nous voulons représenter ou traduire les concepts biologiques qui peuvent être représentés par des glyphes plutôt que des glyphes eux-mêmes, deux glyphes représentant par exemple la même protéine ERK seront traduits par un unique ensemble de prédicats instanciés.

La distinction formelle entre un glyphe d'une carte et le concept biologique qu'il représente n'est pas formalisée dans les spécifications actuelles des langages SBGN-PD et SBGN-AF. Cependant, cette différence est introduite et très largement discutée dans la version 2.0 de la spécification de SBGN-PD. Comme cette spécification est encore à l'état de version de travail, nous ne la prenons pas en considération directement, mais nous nous approprions la notion informelle de concept sous-jacent à un glyphe comme elle est présentée dans cette spécification.

Certains glyphes des deux langages ne permettent pas de représenter de concept biologique : par exemple, le glyphe *clone marker* de SBGN-PD permet juste d'indiquer qu'une certaine molécule est dupliquée dans la carte, et n'apporte pas d'information conceptuelle, mais seulement visuelle. Comme nous nous intéressons aux concepts biologiques représentés par les cartes, les glyphes comme le clone marker n'auront pas de correspondance dans nos langages. À l'opposé, certains concepts représentés par les cartes peuvent ne correspondre à aucun glyphe. Par exemple, dans SBGN-PD, un certain glyphe permet de représenter un pool de macromolécules ; un autre un certain compartiment cellulaire, comme le noyau ; mais aucun glyphe ne permet de représenter l'appartenance d'un pool à un compartiment. Le concept d'appartenance est par contre représenté par la localisation spatiale du glyphe représentant le pool par rapport au glyphe représentant le compartiment. Nous prendrons en compte ces concepts dans nos langages logiques.

3.2.1.2 Constantes identifiant les concepts biologiques d'une carte

Étant donnée une carte SBGN-AF ou SBGN-PD, nous attribuons à chaque concept représenté par un noeud dans la carte, un symbole de constante (ou plus généralement un terme instancié) qui identifie, en logique, ce concept. Notamment, chaque glyphe différent représentant une activité ou un pool d'entités, chaque compartiment, chaque noeud de processus et chaque glyphe représentant un opérateur logique se verront attribuer une constante. Par exemple, le concept représentant la protéine

ERK non modifiée pourra être associé au symbole de constante *erk*, et celui représentant la version phosphorylée de ERK au symbole de constante *perk*.

3.2.1.3 Règles de grammaires

Pour chacun des deux langages SBGN, un certain nombre de règles de grammaire sont définies. Elles permettent de s'assurer que les cartes SBGN construites ont un minimum de sens. Par exemple, ces règles de grammaires définissent les types d'objets qui peuvent être liés entre eux par une modulation.

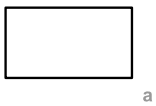
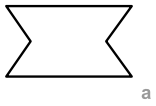
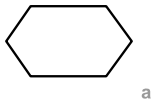
La vérification qu'une carte SBGN satisfait les règles de grammaire est appelée la *validation* de la carte. Une carte est valide *ssi* elle respecte toutes les règles de grammaire du langage avec lequel elle est représentée.

Les règles de grammaire de chacun des deux langages SBGN peuvent également être traduites en logique du premier ordre sous forme d'axiomes. Ainsi, pour chacun des langages SBGNLog-AF et SBGNLog-PD, nous donnons un ensemble d'axiomes logiques qui constituent la grammaire de ce langage, et qui permettent de valider une carte exprimée dans ce langage.

3.2.2 SBGNLog-AF et traduction des cartes SBGN-AF

3.2.2.1 Activités biologiques

Une activité biologique est traduite par un atome formé d'un symbole de prédicat unaire, dont l'argument est le symbole de constante attribué à l'activité.

Terme SBGN	Glyphe	SBGNLog
Biological Activity		$ba(a)$
Perturbation		$perturbation(a)$
Phenotype		$phenotype(a)$

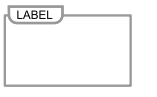
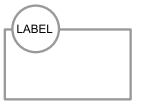
Nommage des activités. Les activités biologiques peuvent être nommées par des étiquettes. Nous attribuons à chaque étiquette une chaîne de caractères. Le nommage d'une activité par une étiquette est traduit par un atome formé d'un symbole de prédicat binaire, dont le premier argument est le symbole de constante attribué à l'activité, et le deuxième la chaîne de caractères attribuée à l'étiquette.

Terme SBGN	Glyphe	SBGNLog
Label		$label(a, \text{“}LABEL\text{”})$

3.2.2.2 Unités auxiliaires

Chaque activité biologique peut être pourvue d'une unité auxiliaire, qui est une unité d'information, et qui indique la provenance de l'activité, i.e. quelle molécule opère l'activité. Cette unité d'information est pourvue d'un type (qui est le type de la molécule qui opère l'activité) et d'une étiquette (qui est le nom de cette molécule). Nous attribuons à chacun des cinq types possibles un symbole de constante, et à chaque étiquette une chaîne de caractères.

Une unité d'information est traduite par un atome formé d'un symbole de prédicat tertiaire, dont le premier argument est le symbole de constante attribué à l'activité, le deuxième le symbole de constante attribué au type, et le troisième la chaîne de caractères attribuée à l'étiquette.

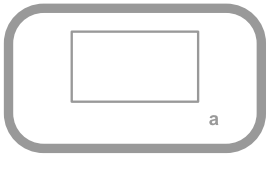
Terme SBGN	Glyphe	SBGNLog
Macromolecule		$uoi(a, macromolecule, \text{“}LABEL\text{”})$
Nucleic Acid Feature		$uoi(a, naf, \text{“}LABEL\text{”})$
Simple Chemical		$uoi(a, simplechemical, \text{“}LABEL\text{”})$
Unspecified Entity		$uoi(a, unspecifiedentity, \text{“}LABEL\text{”})$
Complex		$uoi(a, complex, \text{“}LABEL\text{”})$

3.2.2.3 Compartiments

Un compartiment est traduit par un atome formé d'un symbole de prédicat unaire, dont l'argument est le symbole de constante attribué au compartiment.

Terme SBGN	Glyphe	SBGNLog
Compartment		$compartment(c)$

Localisation. La localisation d'une activité dans un compartiment est traduite par un atome formé d'un symbole de prédicat binaire, dont le premier argument est le symbole de constante attribué à l'activité, et le deuxième le symbole de constante attribué au compartiment.

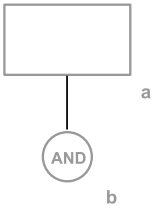
Terme SBGN	Glyphe	SBGNLog
		$localized(a, c)$

3.2.2.4 Opérateurs logiques

Un opérateur logique est traduit par un atome formé d'un symbole de prédicat unaire, dont l'argument est le symbole de constante attribué à l'opérateur.

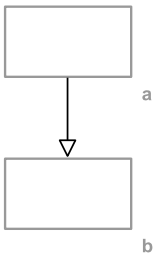
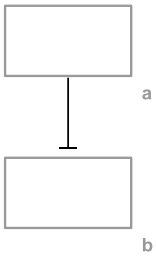
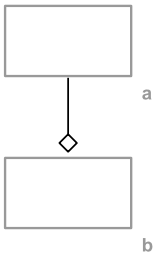
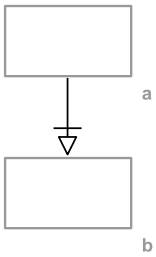
Terme SBGN	Glyphe	SBGNLog
And		$and(o)$
Or		$or(o)$
Not		$not(o)$
Delay		$delay(o)$

Arc logique. Les sources des opérateurs logiques sont liées à ces derniers par des arcs logiques. L'arc logique est traduit par un atome formé d'un symbole de prédicat binaire, dont le premier argument est le symbole de constante attribué à la source de l'arc, et le deuxième argument est le symbole de constante attribué à la cible de l'arc.

Terme SBGN	Glyphe	SBGNLog
Logic Arc		$input(a, b)$

3.2.2.5 Modulations

Une modulation est traduite par un atome formé d'un symbole de prédicat binaire, dont le premier argument est le symbole de constante attribué à la source de la modulation, et la deuxième le symbole de constante attribué à sa cible.

Terme SBGN	Glyphe	SBGNLog
Positive Influence		$stimulates(a, b)$
Negative Influence		$inhibits(a, b)$
Unknown Influence		$unknownInfluences(a, b)$
Necessary Stimulation		$necessarilyStimulates(a, b)$

3.2.2.6 Ontologies

La plupart des concepts biologiques généraux de SBGN-AF représentés par des types de glyphe sont organisés en trois ontologies : une pour les activités, une pour les opérateurs logiques, et une dernière pour les modulations.

La figure 3.1 montre l'ontologie des activités, la figure 3.2 celle des opérateurs logiques, et la figure 3.3 celle des modulations. Les rectangles représentent des classes, et les flèches des relations is_a. Par exemple, l'ontologie de la figure 3.1 comporte quatre classes, et la flèche entre les classes "Perturbation" et "Activity" signifie qu'une perturbation *est* une activité. Pour chaque ontologie, les classes ayant le même parent sont exclusives deux à deux : par exemple, une activité ne peut pas être à la fois une perturbation et une activité biologique.

Les contraintes et la structure de ces ontologies peuvent être exprimées par des règles et des contraintes logiques. Les relations is_a sont exprimées par des règles, alors que l'exclusivité des classes est exprimée par des contraintes. Nous supposons que le monde est clos, et la négation que nous employons ici est la négation par l'échec. Par exemple, si nous ne pouvons pas déduire l'information suivant laquelle une activité A module une activité B (exprimée par $modulates(A, B)$, alors nous te-

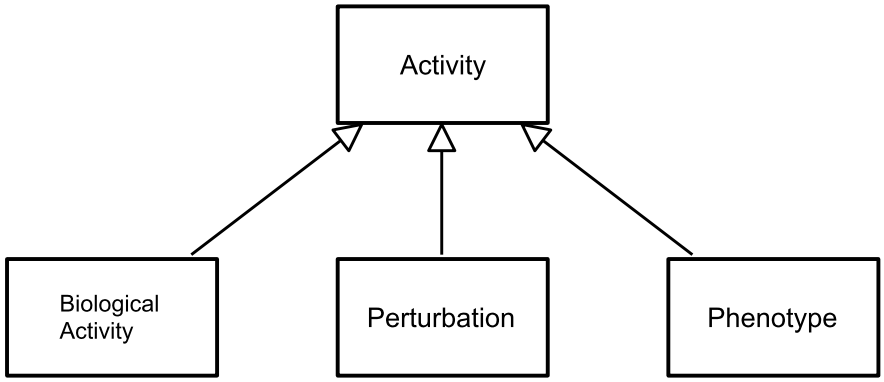


FIGURE 3.1 – **Ontologie des activités de SBGN-AF.** Les rectangles représentent les classes, et les flèches la relation is_a.

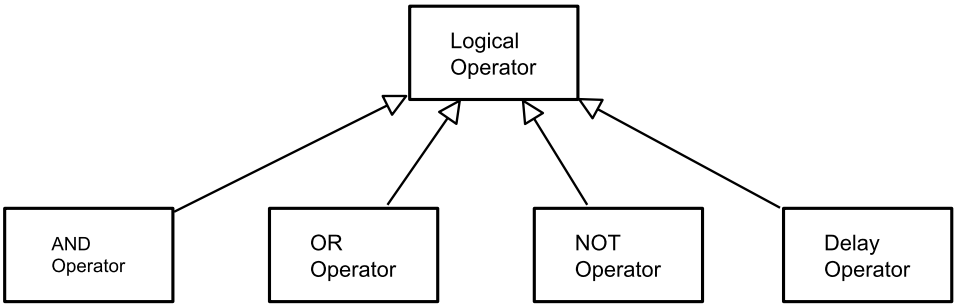


FIGURE 3.2 – **Ontologie des opérateurs logiques de SBGN-AF.** Les rectangles représentent les classes, et les flèches la relation is_a.

nons pour vrai que A ne module pas B (information exprimée par $\neg modulates(A, B)$. La constante $faux$ est symbolisée par $\bot$.

Chaque relation du type *classe1* is_a *classe2* est exprimée par la règle logique

$$classe2(E) \leftarrow classe1(E),$$

et chaque contrainte d'exclusion entre deux classes du genre $classe1 \cap classe2 = \emptyset$ par la contrainte

$$\bot \leftarrow classe1(E) \wedge classe2(E).$$

Par exemple, la structure et les contraintes sur l'ontologie des activités peuvent être exprimées par les règles et contraintes suivantes :

$$activity(A) \leftarrow biologicalActivity(A)$$

$$activity(A) \leftarrow perturbation(A)$$

$$activity(A) \leftarrow phenotype(A)$$

$$\bot \leftarrow perturbation(A) \wedge biologicalActivity(A)$$

$$\bot \leftarrow perturbation(A) \wedge phenotype(A)$$

$$\bot \leftarrow biologicalActivity(A) \wedge phenotype(A)$$

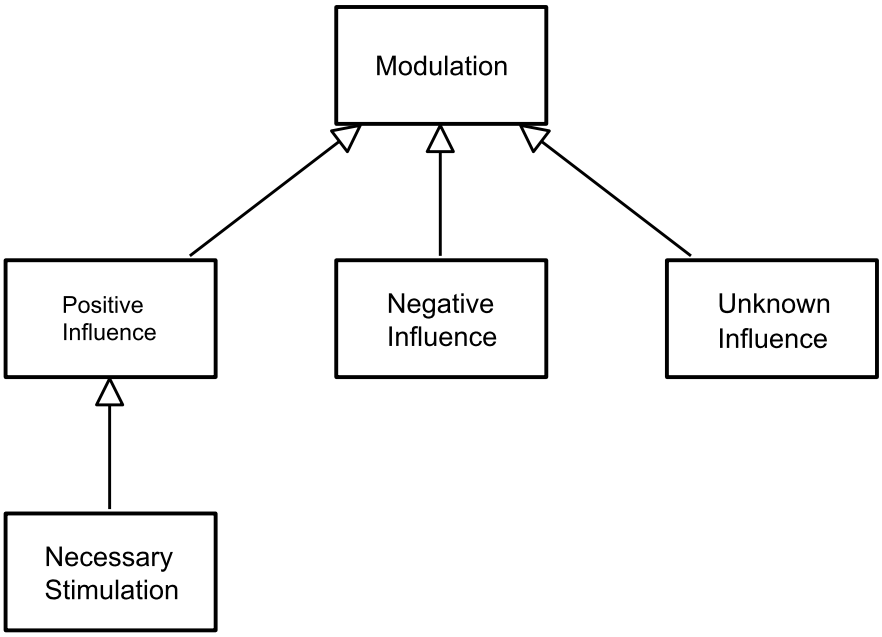


FIGURE 3.3 – **Ontologie des modulations de SBGN-AF.** Les rectangles représentent les classes, et les flèches la relation is_a.

3.2.2.7 Règles de grammaire

Nous traduisons la grammaire de SBGN-AF par des règles et contraintes logiques. Les règles de grammaire de SBGNLog-AF, traduites en partie de celles de SBGN-AF, reviennent à typer les prédicats non unaires.

Dans la suite, nous donnons des exemples de règles de grammaires exprimées sous formes de règles et de contraintes logiques.

Sources et cibles des modulations.

Chaque modulation doit avoir comme source, soit une activité qui n'est pas un phénotype, soit un opérateur logique. Donc, si la source d'une modulation est une activité biologique, sa source est correcte. C'est également le cas si sa source est une perturbation, ou un opérateur logique. Si nous ne pouvons pas déduire que la source d'une modulation est correcte, alors c'est que la grammaire n'est pas respectée. Les deux règles suivantes définissent si la source d'une modulation est correcte, et la contrainte suivante renvoie $faux$ si on n'a pas pu établir que la source d'une modulation est correcte :

$$goodSourceModulates(A, B) \leftarrow modulates(A, B) \wedge biologicalActivity(A)$$

$$goodSourceModulates(A, B) \leftarrow modulates(A, B) \wedge perturbation(A)$$

$$goodSourceModulates(A, B) \leftarrow modulates(A, B) \wedge logicalOperator(A)$$

$$\bot \leftarrow modulates(A, B) \wedge \neg goodSourceModulates(A, B)$$

Localisation dans un compartiment.

Seuls les activités et les compartiments peuvent être localisés dans un compartiment donné. Dans SBGN-AF, d'autres objets, comme les opérateurs logiques, peuvent être contenus dans un compartiment. Il faut donc bien distinguer le concept de *contenance* de celui de *localisation*. La contenance de SBGN-AF est purement graphique et n'a pas de sens biologique pour les objets qui ne sont pas des activités ou des compartiments.

Les règles de grammaire exprimées en logique pour le prédicat *localized*, qui exprime la localisation dans un compartiment, sont les suivantes :

$$goodFirstLocalized(A, C) \leftarrow localized(A, C) \wedge activity(A)$$

$$goodFirstLocalized(A, C) \leftarrow localized(A, C) \wedge compartment(A)$$

$$\bot \leftarrow \neg goodFirstLocalized(A, C) \wedge localized(A, C)$$

$$goodSecondLocalized(A, C) \leftarrow localized(A, C) \wedge compartment(C)$$

$$\bot \leftarrow \neg goodSecondLocalized(A, C) \wedge localized(A, C)$$

Étiquettes des activités.

Seules les activités peuvent avoir une étiquette (mises à part les unités d'informations qui ont une étiquette), ce qui peut être exprimé par des contraintes sur le premier argument du prédicat *label* :

$$goodLabel(A, L) \leftarrow label(A, L) \wedge activity(A)$$

$$\bot \leftarrow \neg goodLabel(A, L) \wedge label(A, L)$$

3.2.2.8 Exemple de traduction

La figure 3.4 montre un exemple de carte SBGN-AF. Cette carte représente la signalisation de TGF-β. La traduction de cette carte en SBGNLog-AF donne l'ensemble d'atomes instanciés donné ci-après. Les numéros de 1 à 14 font la correspondance entre les glyphes de la carte et leur traduction en SBGNLog-AF. Le symbole "," représente ici la conjonction.

Activités.

1. $biologicalActivity(ras)$; $label(ras, \text{"RAS"})$
3. $biologicalActivity(tgfbeta)$; $label(tgfbeta, \text{"TGF } \beta\text{"})$
4. $biologicalActivity(mutp53psmad)$; $label(mutp53psmad, \text{"Mutant p53/P-Smad"})$
5. $biologicalActivity(p63)$; $label(p63, \text{"p63"})$
6. $biologicalActivity(metasup)$; $label(metasup, \text{"Metastatic suppressor genes activity"})$
7. $biologicalActivity(meta)$,
   $label(meta, \text{"Pro-invasion migration metastasis gene expression platform"})$

Opérateurs logiques.

2. $and(lo1)$
8. $input(ras, lo1)$
9. $input(tgfbeta, lo1)$

Modulations.

10. $stimulates(lo1, mutp53psmad)$
11. $inhibits(mutp53psmad, p63)$
12. $necessarilyStimulates(p63, metasup)$
13. $inhibits(metasup, meta)$
14. $stimulates(tgfbeta, meta)$

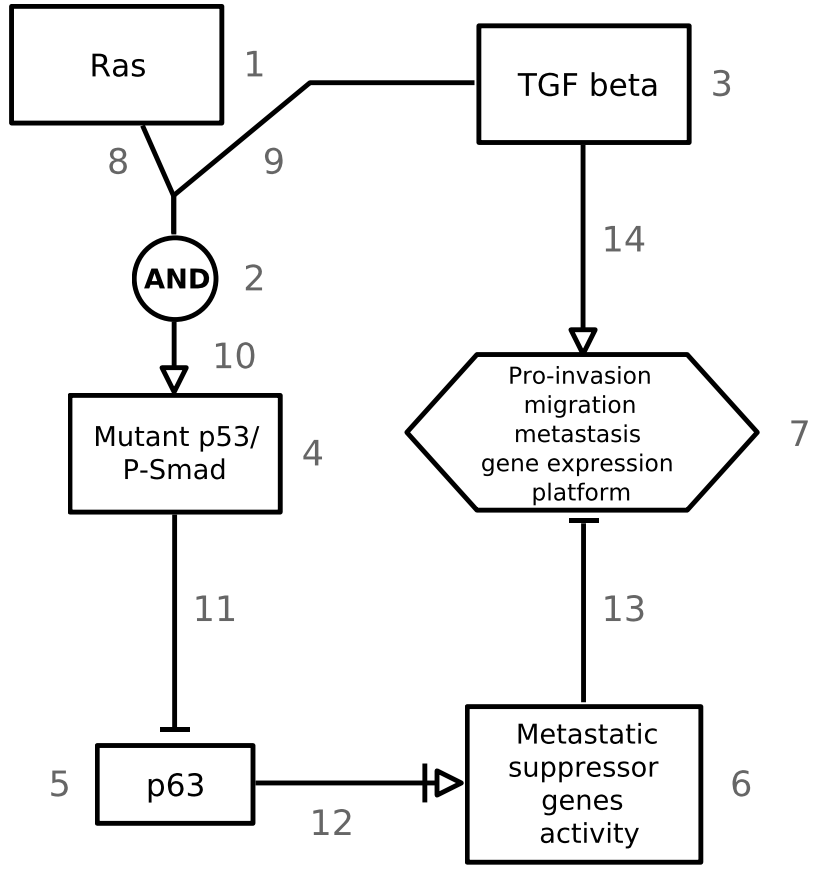


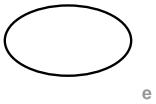
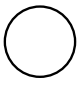
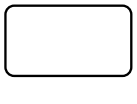
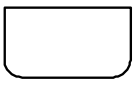
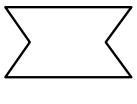
FIGURE 3.4 – **Carte SBGN-AF de la signalisation de TGF-β.** Cette carte comprend six activités (1,3-7), un opérateur logique AND (2), deux arcs logiques (8,9) et cinq arcs de modulation (10-14).

3.2.3 SBGNLog-PD et traduction des cartes SBGN-PD

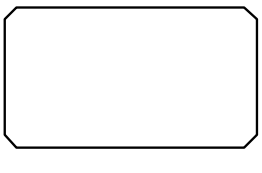
3.2.3.1 Pools d'entités

Un pool d'entités est traduit par atome formé d'un symbole de prédicat unaire, dont l'argument est le symbole de constante attribué au pool d'entités. Trois types principaux de pools d'entités peuvent être distingués : les monomères, les complexes, et les multimères.

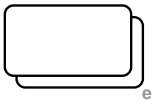
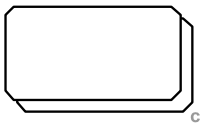
Monomères.

Terme SBGN	Glyphe	SBGNLog
Unspecified Entity		$unspecifiedEntity(e)$
Simple Chemical		$simpleChemical(e)$
Macromolecule		$macromolecule(e)$
Nucleic Acid Feature		$nucleicAcidFeature(e)$
Perturbation		$perturbation(e)$

Complexes.

Terme SBGN	Glyphe	SBGNLog
Complex		$complex(c)$

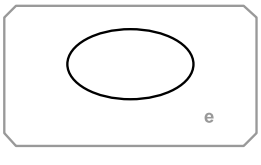
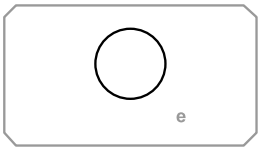
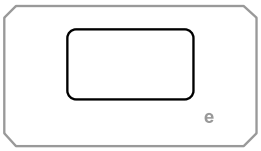
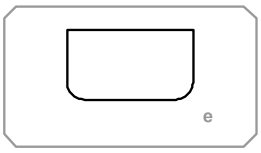
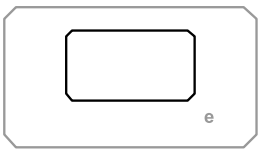
Multimères.

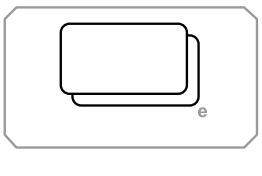
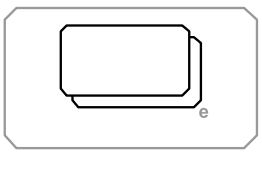
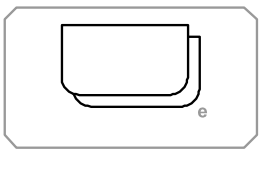
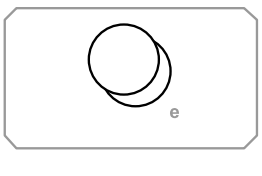
Terme SBGN	Glyphe	SBGNLog
Macromolecules Multimer		$multimerOfMacromolecules(e)$
Complexes Multimer		$multimerOfComplexes(c)$
Nucleic Acid Features Multimer		$multimerOfNucleicAcidFeatures(e)$
Simple Chemicals Multimer		$multimerOfSimpleChemicals(e)$

Sous-entités d'un complexe.

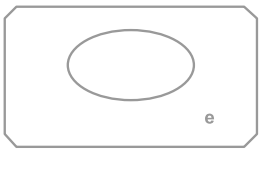
Nous traduisons différemment les sous-unités des pools d'entités monomères ou multimères correspondants. La distinction entre un pool d'entités et une sous-unité d'un complexe n'est pas faite dans la version 1.3 de la spécification, mais elle apparaît dans l'ébauche de la version 2.

Les sous-unités sont traduites exactement comme leurs pools d'entités correspondants sauf pour les symboles de prédicats auxquels le préfixe "sub" est ajouté.

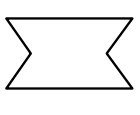
Terme SBGN	**Glyphe**	**SBGNLog**
Unspecified Entity		$subUnspecifiedEntity(e)$
Simple Chemical		$subSimpleChemical(e)$
Macromolecule		$subMacromolecule(e)$
Nucleic Acid Feature		$subNucleicAcidFeature(e)$
Complex		$subComplex(e)$

Terme SBGN	Glyphe	SBGNLog
Macromolecules Multimer		$subMultimerOfMacromolecules(e)$
Complexes Multimer		$subMultimerOfComplexes(e)$
Nucleic Acid Features Multimer		$subMultimerOfNucleicAcidFeatures(e)$
Simple Chemicals Multimer		$subMultimerOfSimpleChemicals(e)$

Appartenance à un complexe. L'appartenance d'une sous-unité à un complexe est traduite par un atome formé d'un symbole de prédicat binaire, dont le premier argument est le symbole de constante attribué à la sous-unité, et le deuxième le symbole de constante attribué au complexe qui contient la sous-unité.

Terme SBGN	Glyphe	SBGNLog
		$component(e, c)$

Perturbations. Une perturbation est traduite par un atome formé d'un symbole de prédicat unaire, dont l'argument est le symbole de constante attribué à la perturbation.

Terme SBGN	Glyphe	SBGNLog
Perturbation		$perturbation(e)$

Nommage des pools d'entités et des sous-unités. Les pools d'entités ainsi que les sous-unités présentés ci-dessus peuvent être nommés par des étiquettes. Nous attribuons à chaque étiquette une chaîne de caractères. Le nommage d'un pool d'entités ou d'une sous-unité par une étiquette est traduit par un atome formé d'un symbole de prédicat binaire, dont le premier argument est le symbole de constante attribué au pool d'entités ou à la sous-unité, et le deuxième la chaîne de caractères attribuée à l'étiquette.

Terme SBGN	Glyphe	SBGNLog
Label		$label(e, \text{“}LABEL\text{”})$

Sources et puits. Une source ou un puits est traduit par un atome formé d'un symbole de prédicat unaire, dont l'argument est le symbole de constante attribué à la source ou au puits.

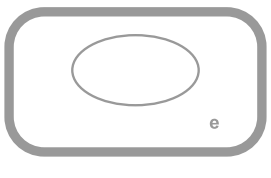
Terme SBGN	Glyphe	SBGNLog
Source or Sink		$emptySet(e)$

3.2.3.2 Compartiments

Un compartiment est traduit comme pour SBGN-AF.

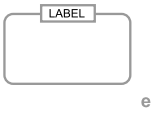
Terme SBGN	Glyphe	SBGNLog
Compartment		$compartment(c)$

Localisation. La localisation est traduite comme pour SBGN-AF.

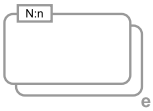
Terme SBGN	Glyphe	SBGNLog
		$localized(e, c)$

3.2.3.3 Unités auxiliaires

Unités d'information. À chaque unité d'information dont le contenu textuel est de la forme "pre :label" sont attibuées deux chaînes de caractères, l'une identifiant le préfixe, et l'autre l'étiquette. À chaque unité d'information dont le contenu textuel ne comporte pas de préfixe est associée une chaîne de caractères identifiant l'étiquette. L'unité d'information est traduite par un atome formé d'un symbole de prédicat tertiaire, dont le premier argument est le symbole de constante attribué au pool d'entités ou à la sous-unité contenant l'unité d'information, le deuxième argument la chaîne de caractères attribuée au préfixe ou la constante *void*, et le troisième la chaîne de caractères attribuée à l'étiquette.

Terme SBGN	Glyphe	SBGNLog
Unit of Information		$uoi(e, \text{“}pre\text{”}, \text{“}LABEL\text{”})$
Unit of Information		$uoi(e, void, \text{“}LABEL\text{”})$

Nous traduisons différemment le cas particulier où l'unité d'information représente la cardinalité d'un multimère. La cardinalité d'un multimère est traduite par un atome formé d'un symbole de prédicat binaire, dont le premier argument est le symbole de constante attribué au multimère, et le deuxième l'entier correspondant à la cardinalité.

Terme SBGN	Glyphe	SBGNLog
Cardinality		$cardinality(e, n)$

Variables d'état. À chaque variable d'état sont attribués deux termes, un pour la valeur, et un pour la variable. Pour la valeur, deux cas se présentent :

- si la valeur de la variable d'état est définie, i.e. la variable d'état est de la forme "$val@var$" ou "val", alors la chaîne de caractères "val" est attribuée à la valeur (voir cas (1) et (3) de la figure) ;
- si la valeur de la variable d'état n'est pas définie, i.e. la variable d'état est de la forme $@var$ ou est vide, alors le symbole de constante *unset* est attribué à la valeur (voir cas (2) et (4) de la figure).

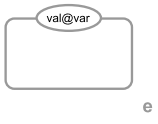
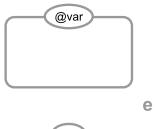
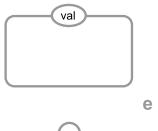
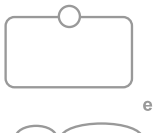
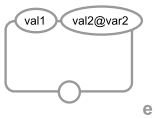
Deux cas se présentent également pour la variable :

- si la variable de la variable d'état est définie, i.e. la variable d'état est de la forme "$val@var$" ou "@var", alors la chaîne de caractères "var" est attribuée à la variable (voir cas (1) et (2) de la figure) ;

- si la variable de la variable d'état n'est pas définie, i.e. la variable d'état est de la forme "*val*" ou est vide, alors un terme fonctionnel formé du symbole de fonction binaire $undefined$, dont le premier argument est le symbole de constante attribué au pool d'entités ou à la sous-unité, et le deuxième un entier, est attribué à la valeur (voir cas (3) et (4) de la figure).

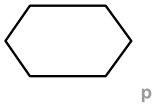
Les variables d'état d'un même pool d'entités ou d'une même sous-entité sont traduites l'une après l'autre en commencant par celle en haut à gauche et en suivant le sens horaire (ceci relève d'un choix arbitraire). À la première variable d'état pour laquelle la variable n'est pas définie est attribuée l'entier 1, et pour chaque autre variable d'état pour laquelle la variable n'est également pas définie, l'entier attribué à la variable est incrémenté de 1.

La variable d'état est traduite par une atome formé d'un symbole de prédicat tertiaire dont le premier argument est le symbole de constante attribué au pool d'entités ou à la sous-unité contenant la variable d'état, le deuxième argument est le terme attribué à la valeur, et le troisième le terme attribué à la variable.

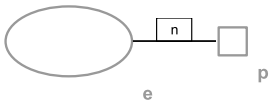
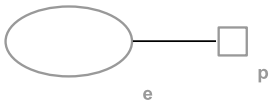
Terme SBGN	**Glyphe**	**SBGNLog**
State variable (1)		$stateVariable(e, \text{“}val\text{”}, \text{“}var\text{”})$
State variable (2)		$stateVariable(e, unset, \text{“}var\text{”})$
State variable (3)		$stateVariable(e, \text{“}val\text{”}, undefined(e, 1))$
State variable (4)		$stateVariable(e, unset, undefined(e, 1))$
State variable		$stateVariable(e, \text{“}val1\text{”}, undefined(e, 1))$ $stateVariable(e, \text{“}val2\text{”}, \text{“}var2\text{”})$ $stateVariable(e, unset, undefined(e, 2)$

3.2.3.4 Processus

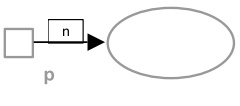
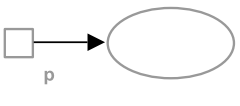
Un processus est traduit par un atome formé d'un symbole de prédicat unaire, dont l'argument est le symbole de constante attribué au processus.

Terme SBGN	Glyphe	SBGNLog
Process		$process(p)$
Omitted Process		$omittedProcess(p)$
Uncertain Process		$uncertainProcess(p)$
Association		$association(p)$
Dissociation		$dissociation(p)$
Phenotype		$phenotype(p)$

Consommation. La consommation d'un pool d'entités par un processus est traduite par un atome formé d'un symbole de prédicat tertiaire, dont le premier argument est le symbole de constante attribué au processus consommant, le deuxième le symbole de constante attribué au pool d'entités consommé, et le troisième l'entier correspondant à la stoechiométrie.

Terme SBGN	Glyphe	SBGNLog
Consumption		$consumes(p, e, n)$
Consumption		$consumes(p, e, 1)$

Production. La production d'un pool d'entités par un processus est traduite par un atome formé d'un symbole de prédicat tertiaire, dont le premier argument est le symbole de constante attribué au processus produisant, le deuxième le symbole de constante attribué au pool d'entités produit, et le troisième l'entier correspondant à la stoechiométrie.

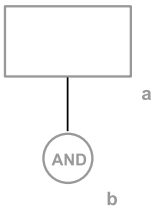
Terme SBGN	Glyphe	SBGNLog
Production		$produces(p, e, n)$
Production		$produces(p, e, 1)$

3.2.3.5 Opérateurs logiques

Un opérateur logique est traduit comme pour SBGN-AF (à l'opérateur de delai prêt, qui n'est pas présent dans SBGN-PD).

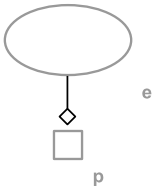
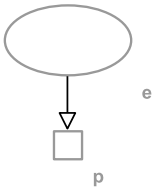
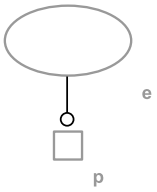
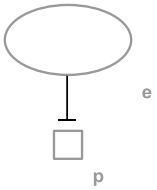
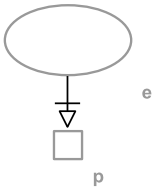
Terme SBGN	Glyphe	SBGNLog
And		$and(o)$
Or		$or(o)$
Not		$not(o)$

Arc logique. Les sources des opérateurs logiques sont liées à ces derniers par des arcs logiques. L'arc logique est traduit par un atome formé d'un symbole de prédicat binaire, dont le premier argument est le symbole de constante attribué à la source de l'arc, et le deuxième argument le symbole de constante attribué à la cible de l'arc.

Terme SBGN	Glyphe	SBGNLog
Logic Arc		$input(a, b)$

3.2.3.6 Modulations

Les modulations sont traduites par des prédicats binaires, dont le premier argument est le symbole de constante attribué à la source, et le deuxième le symbole de constante attribué à la cible.

Terme SBGN	Glyphe	SBGNLog
Modulation		$modulates(e, p)$
Stimulation		$stimulates(e, p)$
Catalysis		$catalyzes(e, p)$
Inhibition		$inhibits(e, p)$
Necessary Stimulation		$necessarilyStimulates(e, p)$

3.2.3.7 Ontologies

Comme pour le langage SBGN-AF, la plupart des concepts du langage SBGN-PD sont structurés en ontologies. Nous distinguons cinq ontologies : une pour les pool d'entités, une pour les sous-entités, une pour les opérateurs logiques, une pour les processus et une pour les modulations. Ces ontologies sont représentées dans les figures 3.5,3.6, 3.7, 3.8 et 3.9.

Comme pour le langage SBGNLog-AF, nous formalisons ces ontologies par l'intermédiaire de règles et de contraintes logiques.

3.2.3.8 Règles de grammaire

Comme SBGN-AF, SBGN-PD comporte un certain nombre de règles de grammaire, qui peuvent être traduites par des règles et des contraintes logiques.

3.2.3.9 Exemple de traduction

La figure 3.10 montre un exemple de carte SBGN-PD. Cette carte représente la signalisation d'IFN. La traduction de cette carte en SBGNLog-PD donne l'ensemble d'atomes instanciés donné ci-après. Les numéros de 1 à 26 font la correspondance entre les glyphes de la carte et leur traduction.

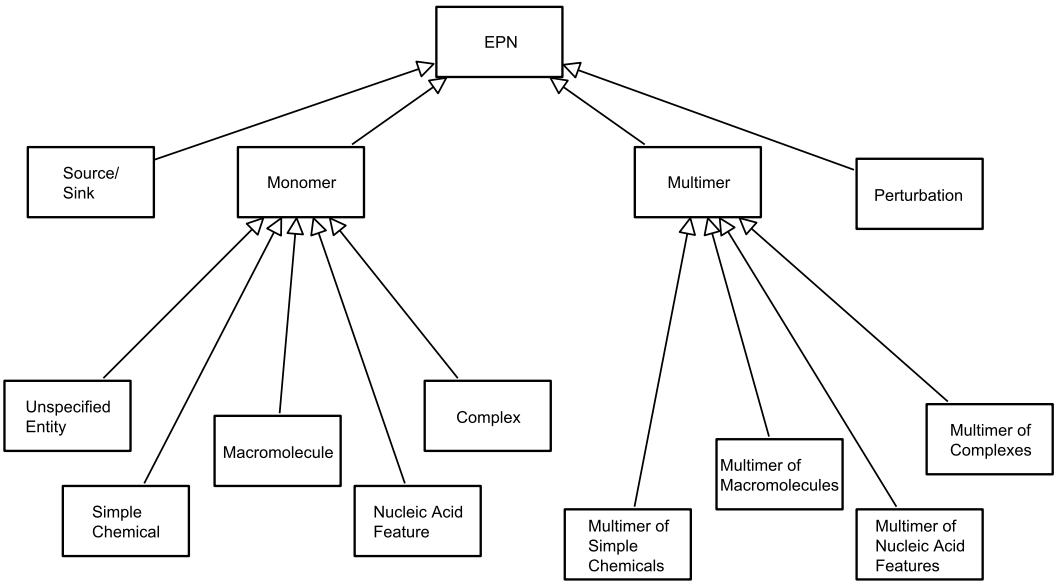


FIGURE 3.5 – **Ontologie des EPNs de SBGN-PD.** Les rectangles représentent les classes, et les flèches la relation is_a.

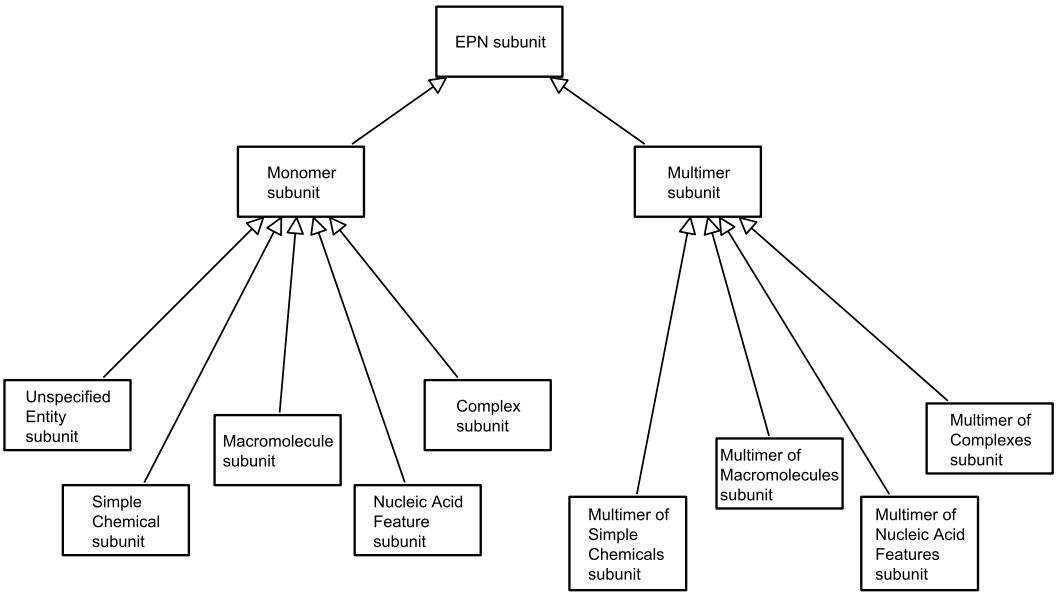


FIGURE 3.6 – **Ontologie des sous-unités de SBGN-PD.** Les rectangles représentent les classes, et les flèches la relation is_a.

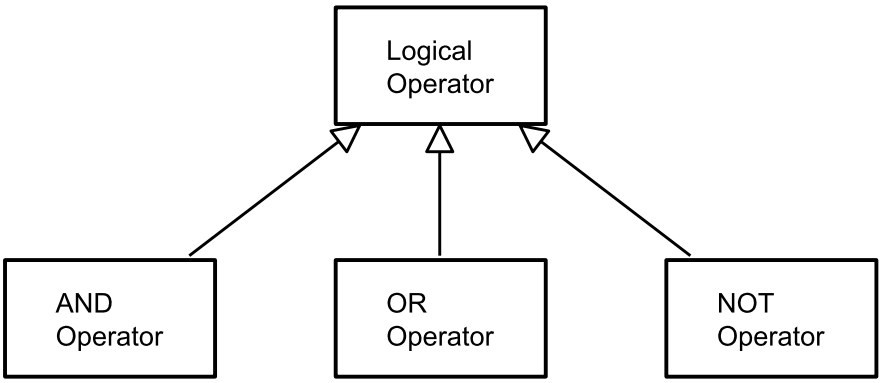


FIGURE 3.7 – **Ontologie des opérateurs logiques de SBGN-PD.** Les rectangles représentent les classes, et les flèches la relation is_a.

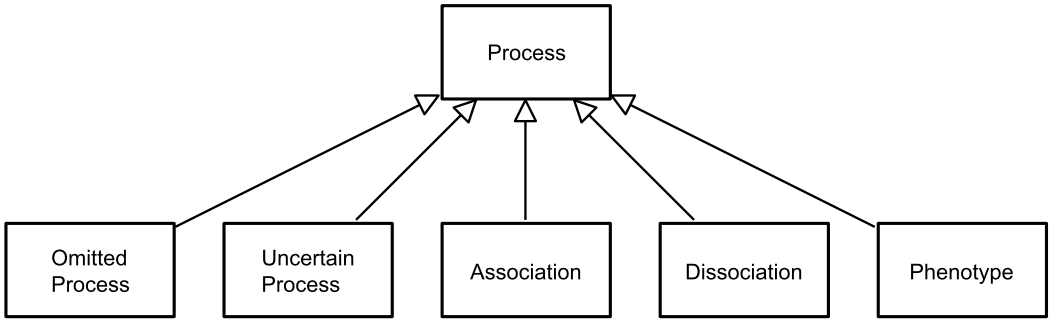


FIGURE 3.8 – **Ontologie des processus de SBGN-PD.** Les rectangles représentent les classes, et les flèches la relation is_a.

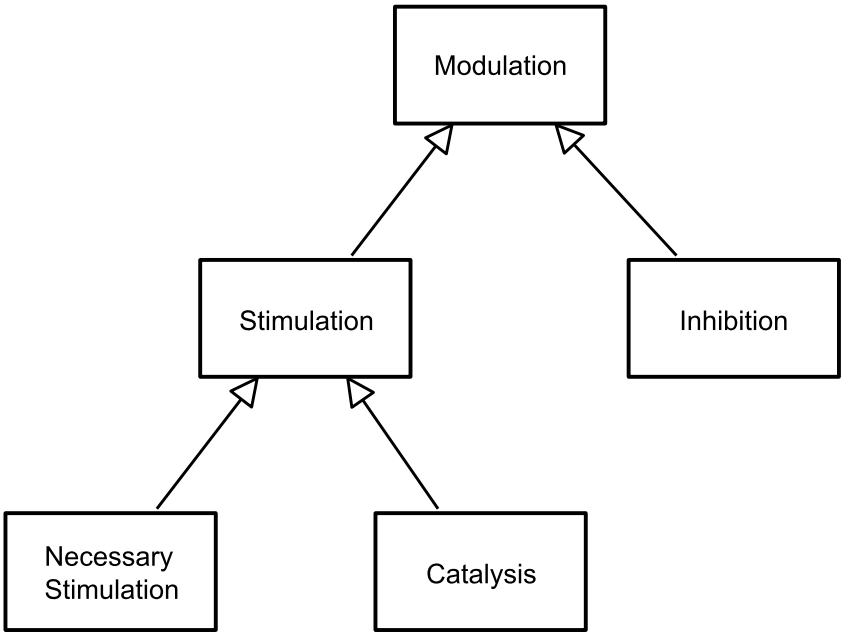


FIGURE 3.9 – **Ontologie des modulations de SBGN-PD.** Les rectangles représentent les classes, et les flèches la relation is_a.

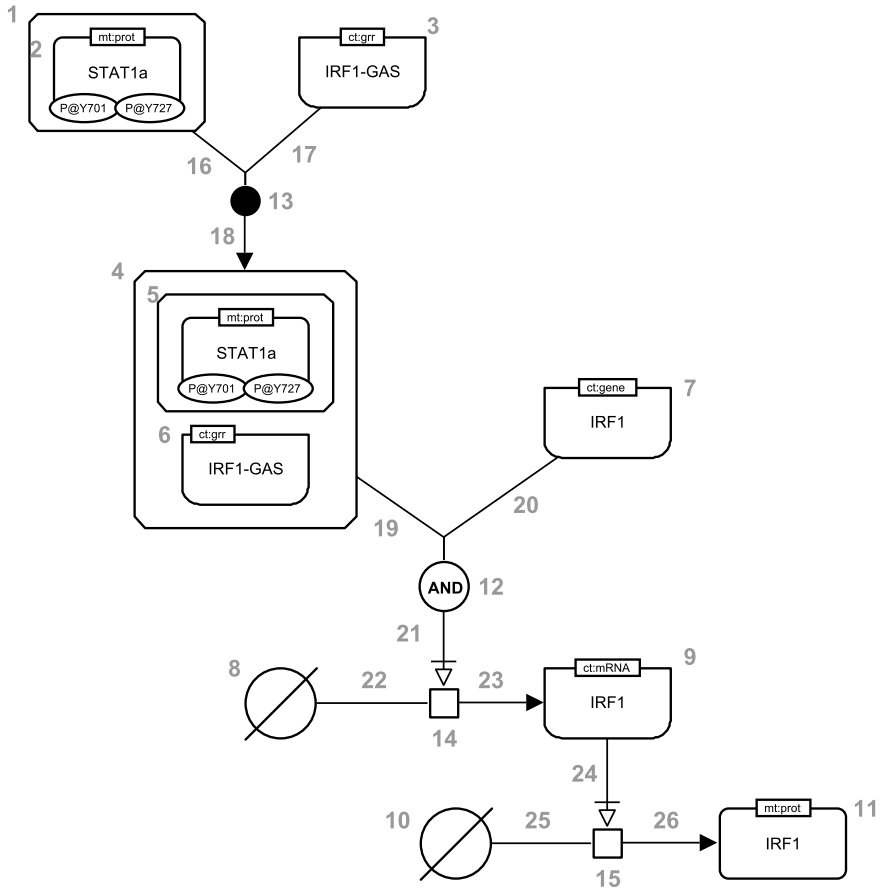


FIGURE 3.10 – **Carte SBGN-PD de la signalisation de STAT1.** Cette carte comprend huit EPNs (1,3,4,7-11), trois sous-unités (2,5,6), un opérateur logique AND (12), trois processus (13-15), deux arcs logiques (19,20), quatre arcs de consommation (16,17,22,25), trois arcs de production (18,23,26) et deux arcs de modulation (21,24). Comme les deux glyphes qui sont respectivement des composants du complexe numéroté 1 et du sous-complexe numéroté 5 représentent le même concept (i.e. la sous-unité STAT1a phopshorylée sur ses positions Y701 et Y727), ils sont traduits par le même ensemble de prédicats. Notons que les glyphes numérotés 1 et 5, mêmes s'ils sont les mêmes, ne représentent pas les mêmes concepts : le glyphe 1 représente un pool d'entités alors que le deuxième représente une sous-unité.

Pools d'entités et sous entités.

1. $complex(c1)$
2. $subMacromolecule(substat1a)$, $label(substat1a, "STAT1a")$, $stateVariable(substat1a, "P", "Y701")$; $stateVariable(substat1a, "P", "Y727")$, $unitOfInformation(substat1a, "mt", "prot")$; $component(substat1a, c1)$
3. $nucleicAcidFeature(irf1gas)$; $label(irf1gas, "IRF1-GAS")$, $unitOfInformation(irf1gas, "ct", "grr")$
4. $complex(c2)$
5. $subComplex(c3)$; $component(c3, c2)$
6. $subNucleicAcidFeature(subirf1gas)$; $label(subirf1gas, "IRF1-GAS")$, $unitOfInformation(subirf1gas, "ct", "grr")$; $component(subirf1gas, c2)$
7. $nucleicAcidFeature(irf1gene)$; $label(irf1gene, "IRF1")$, $unitOfInformation(irf1gene, "ct", "gene")$
8. $emptyset(es1)$
9. $nucleicAcidFeature(irf1mrna)$; $label(irf1mrna, "IRF1")$, $unitOfInformation(irf1mrna, "ct", "mRNA")$
10. $emptyset(es2)$
11. $macromolecule(irf1prot)$; $label(irf1prot, "IRF1")$, $unitOfInformation(irf1prot, "mt", "prot")$

Opérateurs logiques.

12. $and(lo1)$
19. $input(c2, lo1)$
20. $input(irf1gene, lo1)$

Processus.

13. $association(p1)$
16. $consumes(p1, c1, 1)$
17. $consumes(p1, irf1gas, 1)$
18. $produces(p1, c2, 1)$
14. $process(p2)$
22. $consumes(p2, es1, 1)$
23. $produces(p2, irf1mrna, 1)$
15. $process(p3)$
25. $consumes(p3, es2, 1)$
26. $produces(p3, irf1prot, 1)$

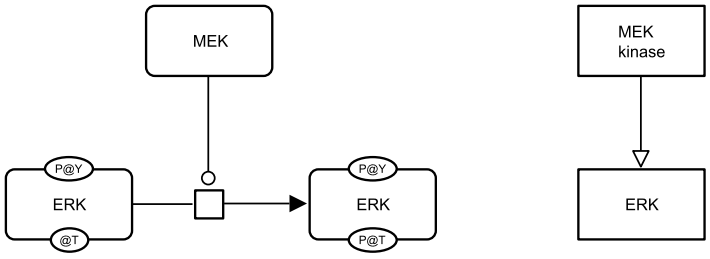
Modulations.

21. $necessarilyStimulates(lo1, p2)$
24. $necessarilyStimulates(irf1mrna, p3)$

3.3 Application : transformation des cartes SBGNLog-PD vers SBGNLog-AF

Nous avons proposé, dans les sections précédentes, deux langages logiques pour la représentation des réseaux de réactions et des graphes d'influences. Nous donnons dans cette section un exemple d'utilisation de ces langages : la transformation des réseaux de réactions en graphes d'influences. Les réseaux de réactions étant souvent grands et détaillés, la transformation de ces réseaux en graphes d'influences permet d'obtenir des réseaux de moins grande taille décrivant les mêmes processus biologiques.

Cette transformation est une tâche de raisonnement complexe, et ne doit pas être confondue avec une simple traduction. En effet, les concepts sous-jacents aux deux types de réseaux ne sont pas les mêmes. Il s'agit donc de transformer des objets biologiques (par exemple, des pools d'entités) en d'autres objets (par exemple, des activités) qui n'ont pas le même sens biologique. Cette transformation nécessite donc de faire une correspondance entre les concepts rencontrés dans les deux types de réseaux. Il faudra par exemple se poser la question du lien entre un pool d'entités (EPN) et une activité, ou plus concrètement, étant donné un réseau de réactions, déterminer à quels EPNs du réseau correspond une activité. Un premier élément de réponse est le suivant : si un EPN est la source d'une modulation, alors cet EPN a une activité biologique qui consiste précisément en cette modulation. Prenons l'exemple donné dans le schéma suivant :



La carte de gauche est une carte SBGN-PD représentant la phosphorylation d'ERK par MEK sur un résidu thréonine (et est un fragment de la carte introduite dans [LN+09]). Comme la molécule MEK catalyse un processus, elle a une activité, qui consiste précisément à catalyser ce processus. Et comme ce processus est une phosphorylation, cette activité est une activité kinase. La carte SBGN-AF obtenue à partir de la carte SBGN-PD de gauche, et représentée à droite, comportera donc une activité kinase provenant de MEK. De plus, en supposant que la molécule ERK phosphorylée sur ses deux résidus a elle-même une activité, nous pouvons ajouter à cette carte une activité pour ERK. Et comme MEK catalyse la phosphorylation d'ERK sur ses deux résidus, l'activité kinase de MEK aura une influence positive sur celle d'ERK.

Nous voyons apparaître que la transformation des réseaux de réactions en graphes d'influences nécessite de rechercher des motifs dans le réseau (i.e. des assemblages de glyphes), et de faire correspondre à ces motifs des motifs de graphes d'influences. Pour reprendre notre exemple, nous avons par exemple d'abord identifié un motif consistant en un EPN, un processus, et un arc de modulation partant de cet EPN et ciblant ce processus, et nous avons transformé ce motif en une activité dans le graphe d'influences obtenu. Ainsi, la correspondance entre les concepts des deux types de réseaux peut se ramener à une correspondance de motifs, et la transformation à effectivement faire correspondre des motifs du réseau de réactions avec des motifs d'un graphe d'influences.

Or, avec un langage logique comme SBGNLog-PD ou SBGNLog-AF, les motifs peuvent naturellement s'écrire sous la formes de conjonctions d'atomes ou de règles logiques. Reprenons nos exemples de motifs : un EPN source d'une modulation ciblant un processus d'une part, et une activité biologique d'autre part. Le premier motif s'écrit par une conjonction d'atomes en SBGNLog-PD : $epn(E) \wedge$

$process(P) \wedge modulates(E, P)$. Quant au deuxième, il s'écrit sous la forme d'un simple atome en SBGNLog-AF : $biologicalActivity(A)$. Ces deux motifs ont un ensemble d'éléments fixe, ils peuvent donc s'écrire sous la forme de conjonctions d'atomes. Certains motifs d'intérêt peuvent avoir une taille variable, et pourront être représentés par des règles logiques. Par exemple, une chaîne de réactions peut comporter un nombre quelconque de réactions, et pourra être recherchée dans le réseau de réactions par l'intermédiaire d'une règle de transitivité. Nous donnons dans la suite la définition de différents motifs des cartes SBGNLog-PD qui seront transformés en motifs des cartes SBGNLog-AF. Nous exprimons ces motifs, ainsi que leur correspondance, à l'aide de règles logiques écrites en ASP. L'ensemble de ces règles constitue un programme ASP qui, étant donné un réseau de réactions formalisé en SBGNLog-PD, le transforme en graphe d'influences, exprimé en SBGNLog-AF.

Définition des EPNs actifs. Les motifs des réseaux de réactions définissant les EPNs actifs sont représentés dans la figure 3.11.

La formalisation de ces motifs en règles logiques est donnée ci-après. Nous introduisons un symbole de prédicat binaire *hasActivity* dont le premier argument est une variable, et le deuxième un terme. L'atome $hasActivity(E, R)$ signifie que l'EPN E est actif selon le motif défini par le terme R. Notons que le symbole de prédicat *hasActivity* n'appartient ni à SBGNLog-PD, ni à SBGNLog-AF. Il est défini seulement pour cette transformation spécifique.

① Un EPN E qui module un processus P opère une activité.

$$hasActivity(E, mod(P)) \text{ :- } epn_{PD}(E);\ process_{PD}(P);\ modulates_{PD}(E, P).$$

Nous mémorisons le motif par un terme fonctionnel formé d'un symbole de fonction *mod* unaire, dont l'argument représente le processus modulé.

② Un EPN E qui est lié à un opérateur logique O opère une activité :

$$hasActivity(E, inp(O)) \text{ :- } epn_{PD}(E);\ logicalOperator_{PD}(O);\ input_{PD}(E, O).$$

Nous mémorisons le motif par un terme fonctionnel formé d'un symbole de fonction *inp* unaire, dont l'argument représente l'opérateur logique.

③ Un EPN E qui peut s'associer à un autre EPN E' qui opère une activité définie par le motif ① opère une activité. En effet, cet EPN E, en s'associant à l'EPN actif E', l'empêche d'opérer son activité, et donc l'inhibe :

$$hasActivity(E, inh(E')) \text{ :- } epn_{PD}(E);\ epn_{PD}(E');\ hasActivity(E', mod);\ association_{PD}(P);\ consumes_{PD}(P, E, Ne);\ consumes_{PD}(P, E', Ne').$$

Nous mémorisons le motif par un terme fonctionnel formé d'un symbole de fonction *inh* unaire, dont l'argument représente l'EPN actif E'.

④ Un EPN E qui peut s'associer à d'autres EPNs pour donner un complexe C qui opère une activité opère une activité :

$$hasActivity(E, bind(P, C)) \text{ :- } epn_{PD}(E);\ complex_{PD}(C);\ hasActivity(C, \_);\ association_{PD}(P);\ consumes_{PD}(P, E, Ne);\ produces_{PD}(P, C, Nc).$$

Nous mémorisons le motif par un terme fonctionnel formé d'un symbole de fonction *bind* binaire, dont le premier argument représente le processus d'association, et le deuxième le complexe.

⑤ Un processus P qui est un phénotype opère une activité :

$$hasActivity(P, pheno) \text{ :- } phenotype_{PD}(P).$$

Nous mémorisons le motif par un symbole de constante *pheno*.

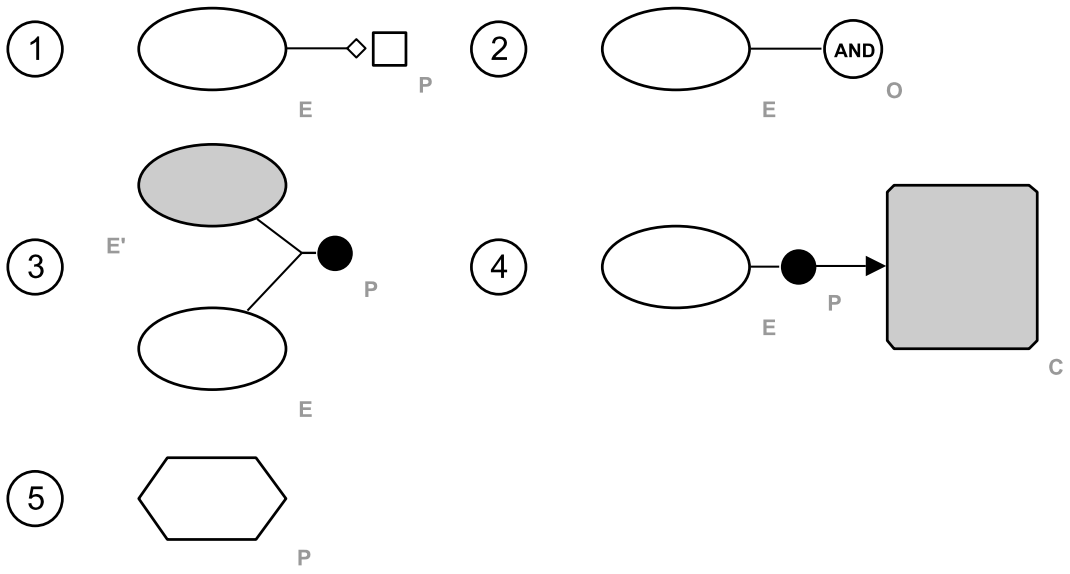


FIGURE 3.11 – **Motifs définissant les EPNs actifs.** Pour chacun des cinq motifs, si l'EPN E appartient à ce motif, alors il opère une activité. Les EPNs colorés en gris sont des EPNs qui ont préalablement été définis comme actifs, i.e. qui appartiennent également à l'un des six motifs représentés.

Transformation des EPNs actifs en activités. Chaque EPN actif E du réseau de réactions est transformé en une activité dans le graphe d'influences. À chaque activité obtenue est attribué un terme fonctionnel formé d'un symbole de fonction binaire a, dont le premier argument est E et le deuxième un symbole de constante associé au type de l'activité. Nous distinguons trois types d'activités : modulatrice, auquel est associé le symbole de constante m ; d'association, auquel est associé le symbole de constante b ; et phénotypique, auquel est associé le symbole de constante p. Le terme $a(E, m)$ (resp. $a(E, b)$, $a(E, p)$) identifie de façon unique cette nouvelle activité.

Les motifs définissant les EPNs actifs dans le réseau de réactions sont transformés en activités, et les unités d'informations et étiquettes des activités sont construites à partir du type et de l'étiquette des EPNs dont elles proviennent.

Chaque motif est transformé en une activité de la manière suivante :

- Les motifs ① et ② sont chacun transformés en une activité modulatrice. Nous distinguons le cas où l'EPN actif est une perturbation :

$$biologicalActivity_{AF}(a(E,m)) \text{ :- } epn_{PD}(E);\ \text{not } perturbation_{PD}(E);\ hasActivity(E, mod(\_)).$$
$$biologicalActivity_{AF}(a(E,m)) \text{ :- } epn_{PD}(E);\ \text{not } perturbation_{PD}(E);\ hasActivity(E, inp(\_)).$$
$$perturbation_{AF}(a(E,m)) \text{ :- } ;\ perturbation_{PD}(E);\ hasActivity(E, mod(\_)).$$
$$perturbation_{AF}(a(E,m)) \text{ :- } ;\ perturbation_{PD}(E);\ hasActivity(E, inp(\_)).$$

- Les motifs ③ et ④ sont chacun transformés en une activité biologique d'association. Encore une

fois, nous distinguons le cas où l'EPN actif est une perturbation :

$$biologicalActivity_{AF}(a(E,b)) \text{ :- } epn_{PD}(E);\ \text{not } perturbation_{PD}(E);\ hasActivity(E, inh(\_)).$$
$$biologicalActivity_{AF}(a(E,b)) \text{ :- } epn_{PD}(E);\ \text{not } perturbation_{PD}(E);\ hasActivity(E, bind(\_,\_)).$$
$$perturbation_{AF}(a(E,b)) \text{ :- } perturbation_{PD}(E);\ hasActivity(E, inh(\_)).$$
$$perturbation_{AF}(a(E,b)) \text{ :- } perturbation_{PD}(E);\ hasActivity(E, bind(\_,\_)).$$

- Finalement, le motif ⑤ est transformé en un phénotype :

$$phenotype_{AF}(a(P,p)) \text{ :- } phenotype_{PD}(P);\ hasActivity(P, pheno).$$

Notons que pour cette dernière transformation, le phénotype du réseau de réactions aurait directement pu être transformé en un phénotype du graphe d'influences, sans utiliser le prédicat *hasActivity*. Ce dernier est utilisé pour cette transformation par souci de cohérence avec les autres transformations présentés ci-dessus.

L'étiquette d'une activité est créée directement à partir du type de l'activité et de l'étiquette de l'EPN dont elle provient. Afin de distinguer les activités d'association des activités de modulation et des phénotypes, nous concaténons l'étiquette associée à ce type d'activités avec la chaîne de caractères "binding". Cette concaténation est réalisée par l'intermédiaire d'une fonction python *concat* qui concatène les chaînes de caractères qui lui sont passées en argument.

$$label_{AF}(a(E,m),L) \text{ :- } biologicalActivity_{AF}(a(E,m));\ epn_{PD}(E);\ label_{PD}(E,L).$$
$$label_{AF}(a(P,p),L) \text{ :- } phenotype_{AF}(a(P,p));\ phenotype_{PD}(P);\ label_{PD}(P,L).$$
$$label_{AF}(a(E,b), @concat(L, \text{"}binding\text{"})) \text{ :- } biologicalActivity_{AF}(a(E,b));\ epn_{PD}(E);\ label_{PD}(E,L).$$

L'unité d'information associée à une activité est créée à partir de l'étiquette, du type et des variables d'état de l'EPN dont provient l'activité de façon à ce que deux activités provenant de deux EPNs différents aient des unités d'informations différentes. Le détail de la création des unités d'information est donné dans l'annexe B.

Création des opérateurs et arcs logiques. Dans un réseau de réactions, tout ensemble, possiblement singleton, d'opérateurs logiques liés entre eux par des arcs logiques contient exactement un opérateur logique qui est la source d'une modulation ; et tout opérateur de cet ensemble est forcément la cible d'un arc logique dont la source est soit un autre opérateur logique de cet ensemble, soit un EPN, qui est actif (d'après le motif ②). En conséquence, tout opérateur logique et tout arc logique du réseau de réactions participent à une activité, et ils sont transformés en un opérateur logique ou arc logique dans le graphe d'influences :

$$and_{AF}(O) \text{ :- } and_{PD}(O).$$
$$or_{AF}(O) \text{ :- } or_{PD}(O).$$
$$not_{AF}(O) \text{ :- } not_{PD}(O).$$
$$input_{AF}(O,O') \text{ :- } logicalOperator_{PD}(O);\ logicalOperator_{PD}(O');\ input_{PD}(O,O').$$
$$input_{AF}(a(E,m),O) \text{ :- } biologicalActivity(a(E,m));\ logicalOperator_{PD}(O);\ input_{PD}(E,O).$$

Un opérateur logique AND est aussi créé pour chaque processus d'association P qui produit un complexe actif C. À chaque opérateur ainsi créé est attribué un terme fonctionnel formé d'un symbole de fonction binaire o, dont le premier argument est P et le deuxième C. Le terme $o(P, C)$ identifie de façon unique ce nouvel opérateur logique. Nous créons un arc logique ciblant cet opérateur pour chaque EPN consommé par P, c'est-à-dire chaque EPN appartenant à un motif ④ auquel appartiennent également P et C :

$$\begin{aligned} and_{AF}(o(P,C)) \text{ :- } & complex_{PD}(C);\ hasActivity(C,\_);\\ & association_{PD}(P);\ produces(P,C,Nc).\\ input_{AF}(a(E,b),o(P,C)) \text{ :- } & epn_{PD}(E);\ hasActivity(E,bind(P,C));\ complex_{PD}(C);\\ & association_{PD}(P);\ produces(P,C,Nc). \end{aligned}$$

Création des compartiments. Chaque compartiment du réseau de réactions est traduit tel quel en un compartiment du réseau d'influences :

$$compartment_{AF}(C) \text{ :- } compartment_{PD}(C).$$

La localisation d'un EPN E qui opère une activité de type T est transformée en localisation de l'activité correspondante de la manière suivante :

$$localized_{AF}(a(E,T),C) \text{ :- } activity_{AF}(a(E,T));\ localized_{PD}(E,C);\ epn_{PD}(E).$$

Maintenant que nous avons défini les motifs permettant de reconnaître les EPNs actifs d'un réseau de réactions, et de transformer ces EPNs en activités, il reste encore à créer les influences entre ces activités, à partir du réseau de réactions et des activités du graphe d'influences.

Les influences sont créées à partir des motifs des EPNs actifs déjà définis ainsi que des motifs représentant des chemins dans le réseau de réactions. Les motifs des réseaux de réactions définissant des chemins sont représentés dans la figure 3.12. La formalisation de ces motifs en règles logiques est donnée ci-après.

⑥ Il y a un chemin *réactionnel* entre un EPN A et un EPN B si B peut être produit à partir de A par une succession de processus, et A et B partagent une même étiquette. Un tel motif est défini par récursivité, avec les deux règles suivantes :

$$\begin{aligned} reacPath(A,B) \text{ :- } & epn_{PD}(A);\ epn_{PD}(B);\ sameLabel(A,B);\\ & process_{PD}(P);\ consumes(P,A,\_);\ produces(P,B,\_).\\ reacPath(A,B) \text{ :- } & epn_{PD}(A);\ epn_{PD}(B);\ epn_{PD}(C);\ reacPath(A,C);\ sameLabel(C,B);\\ & process_{PD}(P);\ consumes(P,C,\_);\ produces(P,B,\_). \end{aligned}$$

⑦ Il y a un chemin *positif* entre un EPN ou opérateur logique A et un EPN B si B est actif et : A stimule un processus P qui produit B ; ou A inhibe un processus P qui consomme B ; ou A stimule un processus P qui produit C, il y a un chemin réactionnel entre C et B, et B n'est pas

consommé par le processus P. Nous ne donnons ici que les axiomes pour le cas où A est un EPN. Les mêmes axiomes doivent être écrits pour le cas où A est un opérateur logique.

$$\begin{aligned}
posPath_{PD}(A,B) \text{ :- } & epn_{PD}(A);\ epn_{PD}(B);\ hasActivity(B,_);\\
& process_{PD}(P);\ produces_{PD}(P,B,_);\ stimulates_{PD}(A,P).\\
posPath_{PD}(A,B) \text{ :- } & epn_{PD}(A);\ epn_{PD}(B);\ hasActivity(B,_);\\
& process_{PD}(P);\ consumes_{PD}(P,B,_);\ inhibits_{PD}(A,P).\\
posPath_{PD}(A,B) \text{ :- } & epn_{PD}(A);\ epn_{PD}(B);\ epn_{PD}(C);\ hasActivity(B,_);\\
& reacPath(C,B);\ process_{PD}(P);\ produces_{PD}(P,C,_);\\
& \text{not } consumes_{PD}(P,B,_);\ stimulates_{PD}(A,P).
\end{aligned}$$

Intuitivement, il y a un chemin positif entre A et B si on peut trouver un lien causal entre la production de B et une stimulation d'un processus par A ou entre la non consommation de B et une inhibition d'un processus par A.

⑧ Les chemins négatifs sont définis de façon analogue aux chemins positifs, en remplaçant dans les règles les inhibitions par des stimulations, et vice-versa.

Transformation des différents motifs en influences. Les motifs définissant les EPNs actifs et les chemins, ainsi que les activités obtenues par transformations de ces premiers motifs, permettent la création des relations d'influence qui ont lieu entre ces activités.

- Le motif ③ est transformé en une influence : l'activité opérée par un EPN E' qui peut s'associer à un autre EPN E qui opère lui-même une activité modulatrice, a une influence négative sur cette activité modulatrice.

$$\begin{aligned}
inhibits_{AF}(a(E',b),a(E,m)) \text{ :- } & epn_{PD}(E');\ epn_{PD}(E);\\
& hasActivity(E,inh(E'));\ activity_{AF}(a(E,m)).
\end{aligned}$$

- Le motif ④ est transformé en une influence : si un processus d'association P produit un complexe C qui opère une activité du type représenté par T, alors l'opérateur logique associé au couple (P,C) est la source d'une stimulation nécessaire ciblant cette activité :

$$\begin{aligned}
necessarilyStimulates_{AF}(o(P,C),a(C,T)) \text{ :- } & epn_{PD}(E);\ complex_{PD}(C);\\
& hasActivity(E,bind(P,C));\ activity_{AF}(a(C,T)).
\end{aligned}$$

- Le motif ⑦ est transformé en une influence : s'il y a un chemin positif entre un EPN ou opérateur logique A et un EPN B, et que B opère une activité du type représenté par T, alors l'activité modulatrice opérée par A a une influence positive sur l'activité opérée par B.

$$\begin{aligned}
stimulates_{AF}(a(A,mod),a(B,T)) \text{ :- } & epn_{PD}(A);\ epn_{PD}(B);\ activity_{AF}(a(B,T));\ activity_{AF}(a(A,mod));\\
& posPath(A,B);\ \text{not } transPos(A,B).
\end{aligned}$$

Ici, not $transPos(A,B)$ signifie qu'il n'existe pas de chemin positif constitué d'influences et d'arcs logiques entre A et B qui puisse être construit par transitivité d'au moins deux influences. L'ajout de cette négation dans le corps de la règle permet de ne pas construire d'influence positive qui puisse s'exprimer par transitivité d'au moins deux autres influences. Pour plus de simplicité de lecture, nous ne donnons pas ici le détail ici des axiomes définissant ce prédicat.

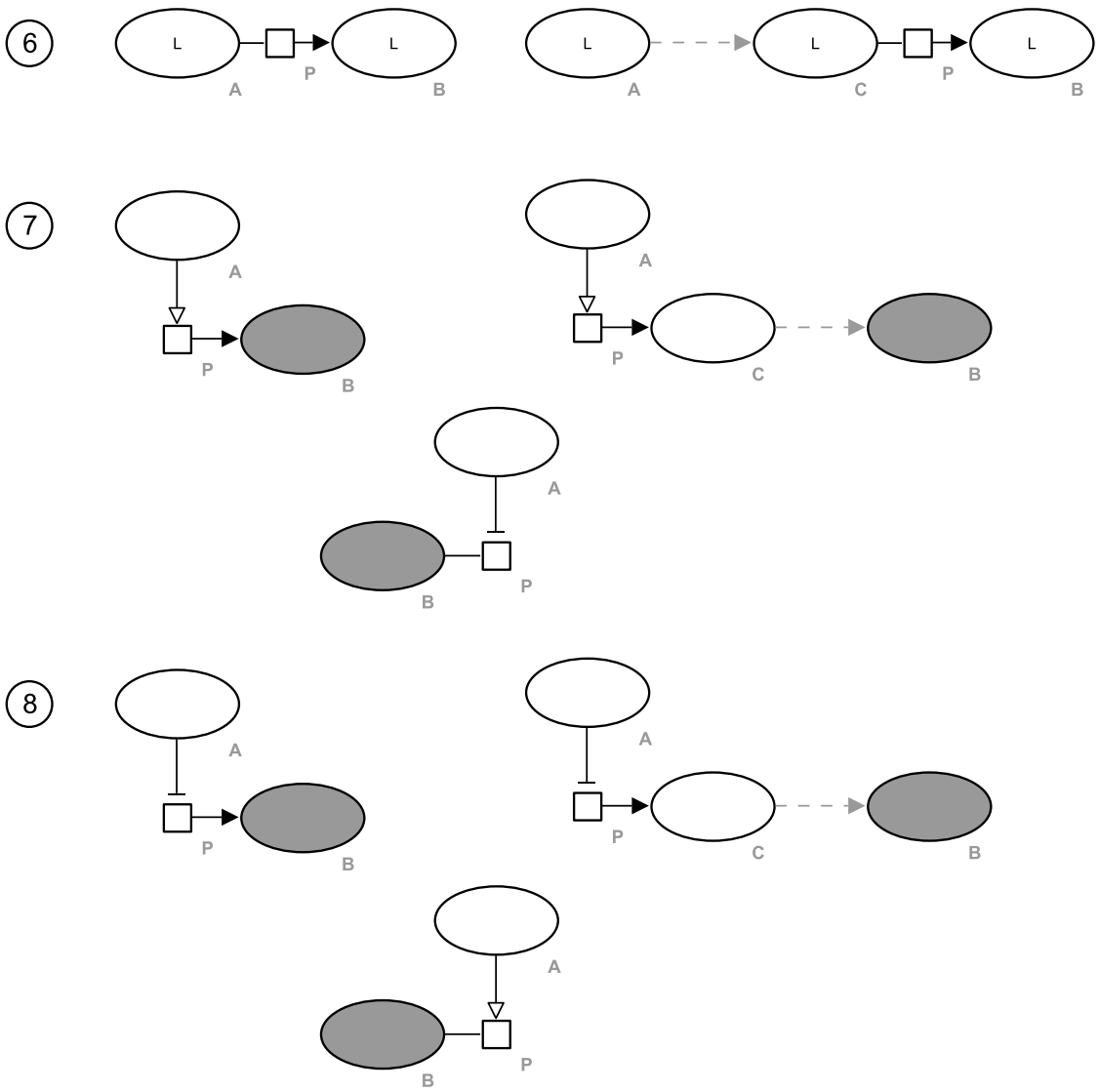


FIGURE 3.12 – **Motifs définissant les chemins.** ⑥ : motifs définissant les chemins réactionnels. ⑦ : motifs définissant les chemins positifs. ⑧ : motifs définissant les chemins négatifs. Les EPNs colorés en gris sont des EPNs actifs. Les flèches en pointillés et colorées en gris sont des chemins réactionnels. Pour les motifs ⑦ et ⑧, la source de la modulation, dénotée par A, est représentée sous la forme d'un EPN, mais peut également être un opérateur logique.

- Finalement, le motif ⑧ est transformé en une influence : s'il y a un chemin négatif entre un EPN ou opérateur logique A et un EPN B, et que B opère une activité du type représenté par T, alors l'activité modulatrice opérée par A a une influence négative sur l'activité opérée par B.

$$inhibits_{AF}(a(A, mod), a(B, T)) \text{ :- } epn_{PD}(A);\ epn_{PD}(B);\ activity_{AF}(a(B, T));\ activity_{AF}(a(A, mod));\ negPath(A, B);\ \text{not } transNeg(A, B).$$

Notons qu'il est *a priori* possible qu'un EPN opère une activité modulatrice du fait qu'il appartient au motif ① ou au motif ② sans qu'il y ait pour autant de chemin positif partant de cet EPN ou de l'opérateur logique auquel il est lié. Par conséquent, l'activité formée à partir de cet EPN, ou l'opérateur logique formé à partir de l'opérateur lié à cet EPN, pourrait n'être la source d'aucune influence dans la carte SBGNLog-AF obtenue. Cette carte pourrait donc contenir une activité liée à aucune autre, ou un opérateur logique n'étant la source d'aucune influence, ce qui est contraire à la grammaire du langage. C'est pourquoi nous laissons le soin à l'utilisateur de vérifier que ce cas n'arrive pas : pour s'en assurer, il lui faudra, pour chaque EPN appartenant au motif ① ou chaque opérateur logique étant la source d'une modulation, vérifier qu'au moins un chemin réactionnel lié à cet EPN ou opérateur logique suivant la définition des chemins positifs et négatifs que nous avons donnée, contienne au moins un EPN actif ; et, si ce n'est pas le cas, en définir un manuellement, à l'aide du prédicat *hasActivity*.

De plus, avec la transformation que nous avons présentée, un même opérateur logique de la carte SBGNLog-AF obtenue pourra être la source de plusieurs modulations, ce qui est contraire à la grammaire de SBGNLog-AF. Afin d'éviter de tels cas, il est donc nécessaire, pour chaque chemin positif ou négatif dont la source est un opérateur logique dans la carte SBGNLog-PD à transformer, de transformer cet opérateur logique, ainsi que tous les opérateurs liés à celui-ci récursivement par des arcs logiques, en de nouveaux opérateurs logiques. Pour des raisons de simplicité, nous n'avons pas pris en compte la création des ces nouveaux opérateurs logiques dans les axiomes que nous avons présentés.

Exemple de transformation. La figure 3.13 donne un exemple de réseau de réactions, représenté en SBGN-PD. Cette carte est adaptée de la carte donnée dans [Hei+12]. Elle montre les deux voies de signalisation induites par $AT_{1A}R$, et plus généralement par les récepteurs 7TMR, menant à l'activation d'ERK. Le récepteur $AT_{1A}R$ active la voie G dont l'action sur ERK est bien connue, mais aussi la voie β-arrestine, elle-même découverte plus récemment. Ces deux voies sont finement régulées par la présence de molécules appelées kinases des récepteurs couplés à la protéine G (GRK2/3 et GRK5/6), qui agissent directement sur la phosphorylation du récepteur.

Nous avons transformé ce réseau en un graphe d'influences avec notre méthode. Nous avons d'abord traduit la carte de la figure 3.13 en une carte SBGNLog-PD, que nous avons ensuite transformée en une carte SBGNLog-AF à l'aide de notre programme ASP. Nous avons ensuite construit, à la main, la carte SBGN-AF représentant le graphes d'influences obtenu, à partir de la carte SBGNLog-AF. La carte SBGN-AF reconstruite est donnée dans la figure 3.14.

Pour cette transformation, nous avons considéré que les EPNs représentant les états G et β de ERK, induits respectivement par les voies G et βarrestine, étaient actifs.

Nous retrouvons bien deux voies de signalisation distinctes (la voie G et la voie des β-arrestines), chacune étant induite par l'un des couples de GRKs : la voie G est induite par GRK2/3 qui a une influence positive sur l'activité du récepteur HR phosphorylé sur sa première position, alors que la voie des β-arrestines est induite par GRK5/6, dont l'activité a une influence positive sur l'activité d'association du récepteur HR phosphorylé sur sa deuxième position. Le récepteur HR non modifié opère deux activités : une activité d'association provenant de son association avec la β-arrestine 2, et une activité de modulation, provenant de la modulation du processus qui active la protéine G. Cette dernière activité induit la voie G, qui résulte par l'induction de l'activité d'ERK dans son état G.

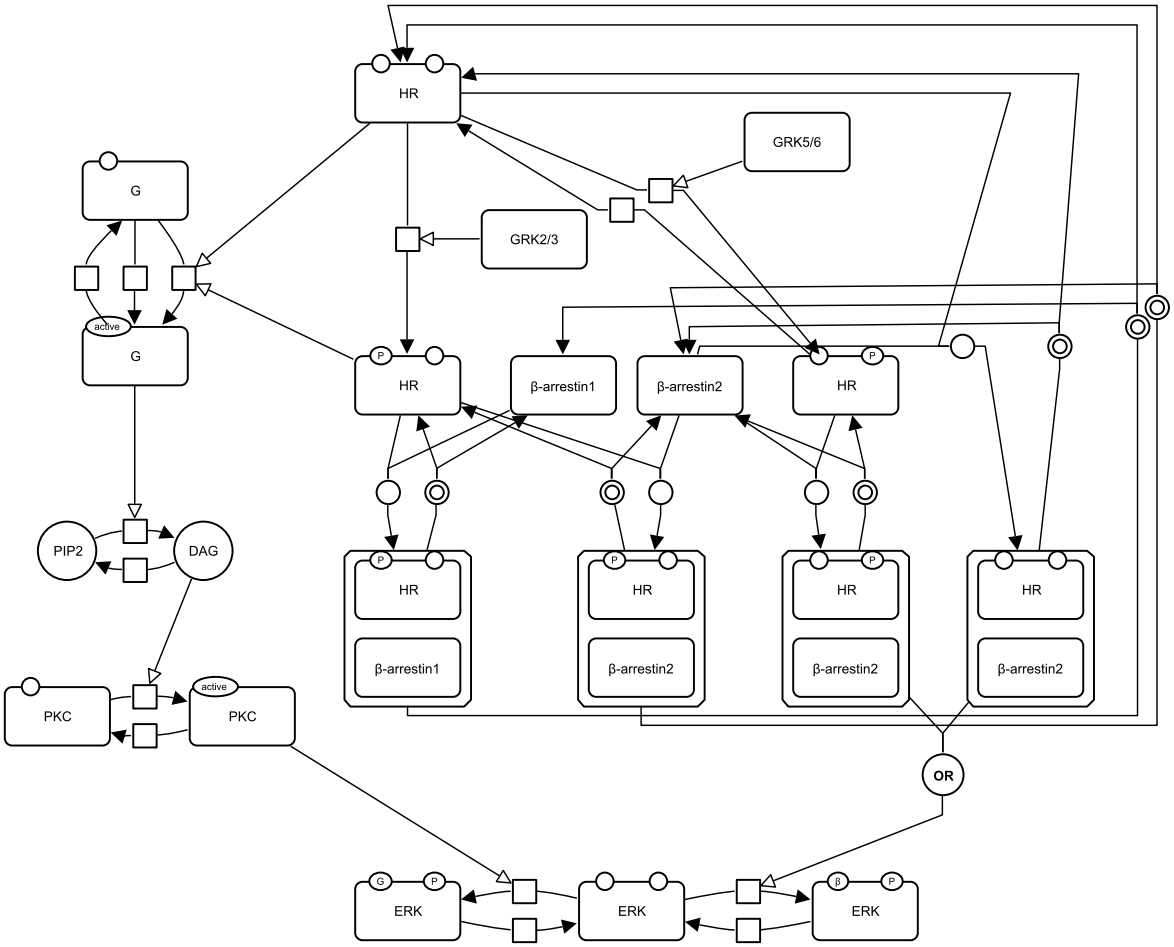


FIGURE 3.13 – **Carte SBGN-PD de l'activation d'ERK induite par le récepteur $AT_{1A}R$.** Cette carte est adaptée de la carte CellDesigner introduite dans [Hei+12]. Elle représente les deux voies principales activant ERK induites par le récepteur $AT_{1A}R$ (récepteur de l'angiotensine). Ce récepteur active, d'une part, la voie de la protéine G qui active ERK, et d'autre part la voie des β-arrestines. Ces deux voies sont finement régulées par les kinases des récepteurs couplés aux protéines G (GRK) (GRK2/3 et GRK5/6), qui agissent directement sur la phosphorylation du récepteur. Afin de différencier les deux états d'ERK induits par chacune des deux voies de signalisation mentionnées plus haut, des variables d'état avec comme valeur "G" et "β" ont été ajoutées aux deux EPNs représentant ces états.

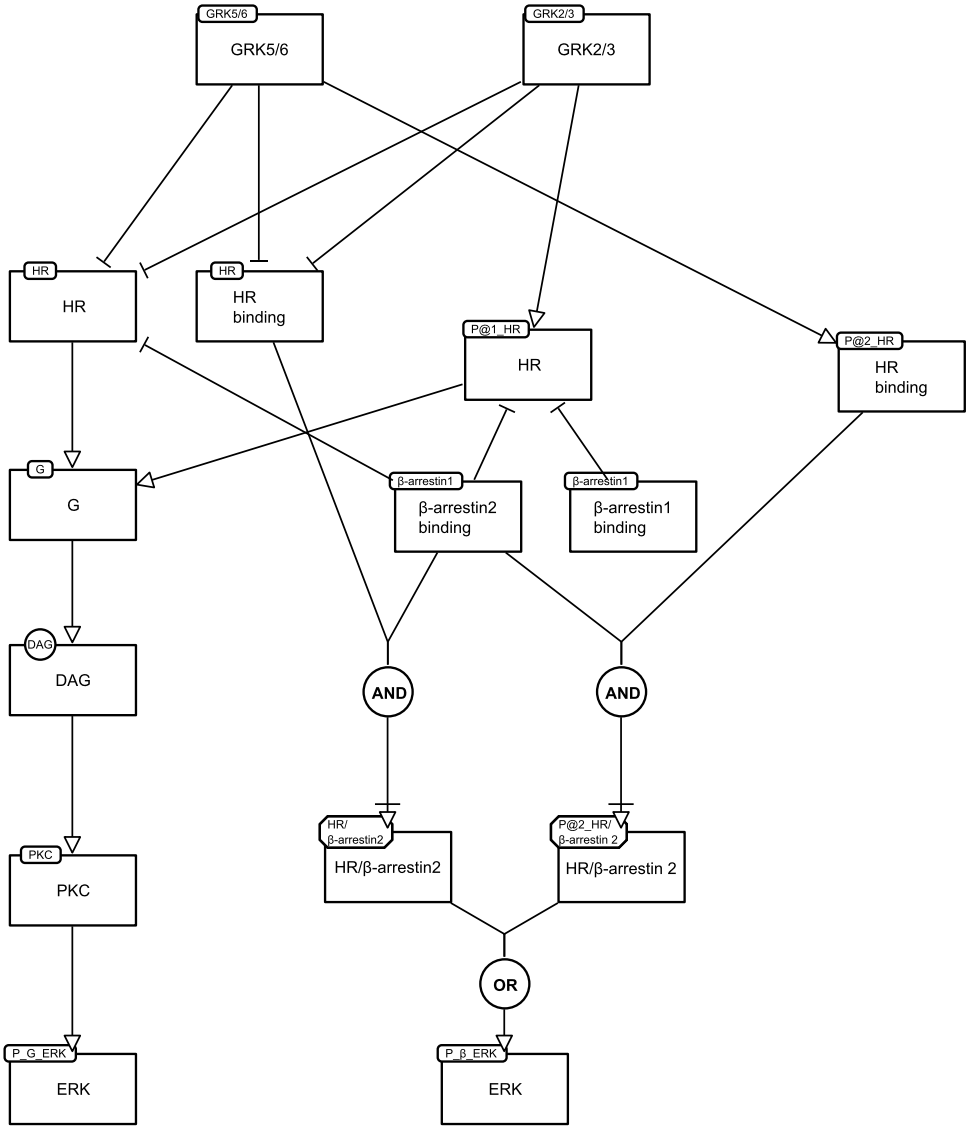


FIGURE 3.14 – **Carte SBGN-AF de l'activation d'ERK induite par le récepteur $AT_{1A}R$ obtenue semi-automatiquement à partir de la carte SBGN-PD correspondante.** Cette carte a été obtenue semi-automatiquement par transformation de la carte SBGN-PD de la figure 3.13 avec notre méthode. La carte SBGN-PD a d'abord été traduite en SBGNLog-PD. Puis une carte SBGNLog-AF a été obtenue par transformation automatique de la carte SBGNLog-PD. Enfin, la carte SBGN-AF a été construite manuellement à partir de la carte SBGNLog-AF obtenue.

Quant aux β-arrestines, chacune d'entre elles opère une activité d'association. Ces deux activités sont, conjointement aux activités d'association du récepteur non phosphorylé ou phosphorylé sur sa position deux, nécessaires aux activités de modulations des complexes HR-β-arrestines. À leur tour, ces activités induisent l'activité d'ERK dans son état β.

3.4 Travaux connexes

Si de nombreux articles décrivent des travaux où le raisonnement sur des réseaux biologiques s'effectue avec des méthodes purement logiques, et proposent, de ce fait, une représentation en logique de ces réseaux, il n'existe, à notre connaissance, aucune traduction des langages SBGN-PD et SBGN-AF en logique. Cependant, il existe des formalisations de ces deux langages dans des formalismes autres que celui de la logique, que nous présentons ci-après.

La plus évidente de ces formalisations est SBGN-ML [VI+12], dont nous avons déjà parlé. SBGN-ML est le format standard de stockage et d'échange des réseaux représentés en SBGN. C'est un format XML : les *éléments XML* permettent de représenter les différents glyphes d'un réseau ; les *balises XML* comprises dans ces éléments permettent de distinguer les différents concepts (un glyphe, une variable d'état, un arc, etc.) ; finalement, les *attributs XML* associés aux éléments permettent de représenter les attributs associés aux glyphes (l'étiquette d'un pool d'entités, sa position dans la carte, la source d'un arc, etc.).

La seconde de ces formalisations, SBGNtext, a été introduite dans [LMH09]. SBGNtext est une représentation textuelle d'une abstraction des cartes SBGN-PD appelée "Abstraction des Flux de Processus" ("Process Flow Abstraction, ou PFA, dans le texte) par ses auteurs. Des modèles quantitatifs peuvent facilement être construits à partir d'une carte SBGN-PD abstraite en SBGNtext, ce qui n'est pas le cas, selon les auteurs, à partir des cartes SBGN-PD elles-mêmes. Ils donnent comme exemple la création d'un modèle Bio-PEPA à partir d'une carte représentée en SBGNtext. Au contraire de SBGN-PD ou de SBGNLog-PD, SBGNtext s'intéresse à formaliser la sémantique PFA des cartes plutôt que les glyphes eux-mêmes ou les concepts représentés par les glyphes. Ils ajoutent donc à la représentation des différents éléments d'une carte sous forme textuelle des notions appartenant à la sémantique PFA. Par exemple, à chaque pool d'entités est associé, en SBGNtext, un attribut représentant la quantité initiale de ce pool. Ce sont ces types d'attributs qui permettront la création de modèles quantitatifs. SBGNtext n'est donc pas à proprement parler une traduction de SBGN-PD en texte, mais permet plutôt de représenter une interprétation des cartes SBGN-PD suivant la sémantique PFA en texte, et est présenté par les auteurs comme un intermédiaire pratique entre les cartes SBGN-PD et des modèles quantitatifs modélisant ces cartes.

En ce qui concerne la transformation des cartes SBGN-PD en cartes SBGN-AF, une unique méthode a été publiée, à notre connaissance. Les auteurs de [VCS13] proposent une "traduction" semi-automatique des cartes SBGN-PD vers SBGN-AF, implémentée dans SBGN-ED [CKS10], qui est un logiciel d'édition et de visualisation de cartes SBGN. Comme dans notre approche, leur méthode est basée sur la correspondance de motifs. Le logiciel donne le choix entre plusieurs correspondances prédéfinies, ou laisse à l'utilisateur lui-même le soin de définir la correspondance. Les motifs SBGN-PD considérés sont les suivants : les EPNs, les chemins réactionnels faisant intervenir un unique processus, et les chemins positifs et négatifs ne prenant pas en compte les chemins réactionnels (i.e. faisant intervenir un unique processus, voir les premiers et derniers motifs des motifs ⑦ et ⑧). À chaque type d'EPN correspond une activité avec une unité d'information différente ; à chaque autre motif une modulation, dont la nature peut être définie par l'utilisateur, ou aucun motif. Notons que le motif ② est pris en compte automatiquement par leur méthode.

Les différences principales, du point de vue de la transformation, d'avec notre méthode, sont les suivantes. Leur méthode ne considère pas certains motifs pour les EPNs actifs que nous avons considérés, et en particulier le motif ③, qui peut être important pour les réseaux de signalisation. De plus, avec leur méthode, chaque EPN, qu'il soit actif ou non, est transformé en une activité, qui sera représentée dans la carte SBGN-AF obtenue, à condition qu'elle soit la cible ou la source d'au moins un arc. Or, ceci semble contradictoire avec le sens que nous donnons au concept d'activité.

3.5 Conclusion et perspectives

Nous avons proposé deux langages de la logique du premir ordre, nommément SBGNLog-AF et SBGNLog-PD, qui permettent respectivement de représenter des graphes d'influences et des réseaux de réactions par des ensembles d'atomes. Nous avons dans le même temps montré comment traduire des cartes SBGN-AF et SBGN-PD dans ces deux langages. Nous avons ensuite proposé, comme illustration de l'utilisation pratique de ces langages, une méthode pour transformer des réseaux de signalisation représentés sous forme de cartes SBGNLog-PD en graphes d'influences repésentés sous forme de cartes SBGNLog-AF.

Nos deux langages n'ont pas été construits en vue d'une tâche de raisonnement particulière. Par conséquent, ils sont génériques, et pourraient être utilisés pour une variété de tâches de raisonnement nécessitant d'avoir une représentation logique d'un réseau de réactions ou d'un graphe d'influences. De plus, comme ils sont construits directement à partir des langages SBGN qui sont maintenant admis comme des standards pour la représentation graphique des réseaux moléculaires, ils sont de bons candidats pour servir de base commune aux méthodes formalisant, en logique du premier ordre, des tâches de raisonnement sur ces réseaux.

Les langages SBGNLog devront prendre en compte les évolutions futures des différents langages SBGN, et notamment l'ajout de nouveaux glyphes. Un logiciel permettant d'une part la traduction des cartes SBGN-AF en cartes SBGNLog-AF et d'autre part la traduction des cartes SBGNLog-AF en cartes SBGN-AF, et la mise en forme automatique de ces dernières est en cours de développement. Le développement de ce logiciel est actuellement réalisé par Eléa Greugny, stagiaire de troisième année de l'INSA de Lyon que nous encadrons.

Chapitre 4

Automatisation de l'interprétation d'expériences pour la construction de réseaux de signalisation et la proposition de plans expérimentaux

Sommaire

4.1 Introduction

Le comportement des cellules des organismes multicellulaires est contrôlé par l'intermédiaire d'hormones qui se lient à leur récepteur. La liaison d'une hormone à son récepteur induit un certain nombre

de processus cellulaires prédéfinis qui traitent le signal et produisent des effets adaptés (comme la transcription de certains gènes). L'ensemble des molécules et des processus moléculaires participant à l'induction de la réponse cellulaire constitue un *réseau de signalisation*. Découvrir ces réseaux de signalisation nécessite la réalisation de nombreuses expériences en laboratoire, et notamment des expériences faisant entrer en jeu des protéines phosphorylées, qui sont des acteurs majeurs de la signalisation. À mesure que notre connaissance des différentes voies de signalisation s'est enrichie, les réseaux de signalisation se sont complexifiés et ont augmenté en taille.

Durant la dernière décennie, des reconstructions exhaustives de réseaux de signalisation ont été entreprises, en rassemblant manuellement un ensemble de connaissances de la littérature (p. ex. [Oda+05 ; Glo+11 ; Cal+08] que nous avons déjà cités, ou encore [Car+10]). Ces réseaux ont deux particularités, comparé aux réseaux que l'on pouvait auparavant retrouver dans la littérature. Premièrement, plutôt que de représenter de simples modulations entre molécules ou activités, ils représentent des mécanismes moléculaires précis sous forme de processus moléculaires concrets. Deuxièmement, ces réseaux sont publiés sous la forme de schémas, qui utilisent des standards de la biologie des systèmes. Notamment, l'ensemble des réseaux cités précedemment sont représentés, dans leur version publiée, sous formes de cartes SBGN-PD.

La reconstruction, à la main, de tels réseaux de signalisation exhaustifs est devenue complexe, au fur et à mesure que le nombre de données expérimentales a augmenté. De plus, les réseaux contruits ne sont généralement pas mis à jour avec les nouvelles connaissances obtenues expérimentalement. Par conséquent, des méthodes automatiques de construction des réseaux de signalisation sont devenues nécessaires. Nous avons déjà introduit un certain nombre de méthodes existantes permettant de construire des réseaux moléculaires. Ces méthodes ne permettent généralement pas de construire des réseaux constitués de mécanismes moléculaires précis, mais plutôt des graphes d'influences.

C'est pourquoi nous proposons une méthode de construction des réseaux de signalisation à l'échelle des processus moléculaires, à partir de résultats expérimentaux, en collaboration avec l'équipe BIOS (INRA Centre Val de Loire), et notamment avec Pauline Gloaguen et Anne Poupon. Nous avons construit un ensemble de règles interprétatives formalisées à l'aide de la logique du permier ordre, qui permettent d'imiter le raisonnement entrepris par les biologistes lorsqu'ils interprètent des résultats expérimentaux. Nos règles interprétatives permettent de reconstruire des mécanismes moléculaires précis à partir d'une grande variété de types d'expérience, et en particulier à partir d'expériences de perturbation impliquant par exemple des inhibiteurs ou des petits ARNs interférents (siRNA). Ces mécanismes peuvent ensuite être regroupés sous forme de réseaux moléculaires, et la dynamique de ces derniers peut être analysée à l'aide de modèles dynamiques. Nos règles permettent également d'émettre des hypothèses qui peuvent ensuite être testées expérimentalement.

En utilisant des règles explicites, les processus moléculaires constituant le réseau de signalisation reconstruit peuvent être liés aux résultats expérimentaux dont ils proviennent. De cette manière, la provenance d'un processus est explicite, et peut être vérifiée. Ceci est un avantage majeur de notre méthode étant donné qu'il n'est pas rare que deux résultats expérimentaux réalisés par deux équipes différentes semblent contradictoires, et donnent lieu à deux processus opposés.

Finalement, le raisonnement utilisé pour la construction des réseaux à l'aide des règles interprétatives peut être inversé afin de proposer des plans expérimentaux pour tester une hypothèse biologique donnée.

L'ensemble de nos règles interprétatives sont formalisées en logique du premier ordre, qui est suffisamment expressive pour formaliser le raisonnement expert. La construction automatique des réseaux à partir de résultats expérimentaux formalisés est réalisée à l'aide du logiciel ASP clingo [Geb+08], et la proposition des plans expérimentaux est réalisée à l'aide du logiciel de *consequence finding* SOLAR [Nab+10].

Le reste de ce chapitre est organisé comme suit. Nous introduisons d'abord nos règles interprétatives

et donnons quelques exemples de règles supplémentaires de raisonnement. Puis nous montrons comment l'ensemble de nos règles peut être utilisé pour la construction des réseaux et la proposition de plans expérimentaux. Ensuite, nous donnons un exemple d'application de notre méthode. Nous montrons notamment comment ces règles ont permis de reconstruire le réseau induit par le récepteur de la FSH, et comment la reconstruction de ce réseau conjointement à celui induit par le récepteur de l'EGF nous a permis d'émettre un nouvelle hypothèse, que nous avons validée expérimentalement. Enfin, nous discutons des différences entre notre méthode et d'autres approches déductives trouvées dans la littérature, de la représentation graphique des réseaux construits et de la possibilité d'utiliser nos règles pour interpréter les expériences haut-débit.

4.2 Règles d'interprétation explicites

La biologie des systèmes s'appuie en partie sur la découverte et l'étude des mécanismes moléculaires sous-jacents aux processus biologiques. Ces mécanismes sont le plus souvent des réactions (p. ex. des modification post-traductionnelles ou des complexations) et des modulations de ces réactions (p. ex. des catalyses, des inhibitions).

La biologie des systèmes utilise des règles de déduction implicites pour interpréter des résultats expérimentaux par de nouvelles connaissances. Ces règles, rendues explicites, suivent le schéma général suivant :

$$\begin{gathered}\text{SI un } \textit{résultat expérimental} \text{ est observé} \\ \text{ET des } \textit{connaissances biologiques} \text{ sont établies} \\ \text{ALORS de nouvelles } \textit{connaissances biologiques} \text{ sont établies} \\ \text{OU de nouvelles } \textit{hypothèses biologiques} \text{ sont émises}\end{gathered} \tag{R1}$$

Les règles d'interprétation, qui suivent ce schéma, sont à la fois implicites et générales. Bien entendu, un résultat expérimental particulier mènera à une interprétation spécifique, i.e. un fait biologique instancié ; cependant, la manière dont ce résultat est interprété ne dépend pas de ce résultat en particulier, mais plutôt du type de l'expérience qui a été réalisée pour l'obtenir. Par conséquent, les règles d'interprétation suivant le schéma (R1) ne s'appliquent pas seulement à la réalisation particulière d'une expérience mais plutôt à un type particulier d'expérience. En effet, la conclusion d'une telle règle ne dépend pas des espèces moléculaires qui entrent en jeu dans l'expérience, mais du type des réactions qui entrent en jeu dans le type d'expérience dont l'expérience est une réalisation.

En conséquence, il est possible d'expliciter les règles utilisées pour interpréter les résultats obtenus à partir d'un certain nombre de types d'expérience, et d'utiliser ces règles pour automatiser l'interprétation de résultats expérimentaux relatifs à la signalisation, et ce faisant, d'imiter le raisonnement des biologistes.

4.2.1 La construction des règles

Pour un type d'expérience donné, nous avons construit une ou plusieurs règles d'interprétations en analysant les processus moléculaires qui entrent en jeu lors de la réalisation d'une expérience de ce type.

Considérons une expérience de phosphorylation simple. Une expérience de ce type consiste à comparer la quantité d'une certaine molécule phosphorylée d'intérêt (p. ex. phospho-ERK) dans deux lysats cellulaires, le premier de ces lysats étant obtenu à partir de cellules non traitées (le contrôle), et le deuxième obtenu à partir de cellules préalablement traitées avec un signal (p. ex. une hormone

comme la FSH). La mesure de la quantité de la cellule d'intérêt dans l'un ou l'autre des lysats est le plus souvent réalisée par l'intermédiaire d'un anticorps spécifique de la molécule phosphorylée ou par l'intermédiaire de techniques de radioactivité. L'unique processus moléculaire qui entre en jeu dans une expérience de ce type est celui qui produit la molécule d'intérêt à partir d'une autre forme de cette même entité moléculaire (p. ex. phospho-ERK est produit par phosphorylation d'ERK), qui est connue par l'expérimentateur. Ainsi, si la quantité de la molécule d'intérêt est plus élevée dans le lysat des cellules qui ont été traitées par le signal que dans le contrôle, nous pouvons conclure que le signal stimule le processus qui produit la molécule d'intérêt. À l'inverse, si la quantité est plus élevée dans le contrôle que dans le lysat des cellules traitées, alors nous pouvons conclure que le signal inhibe ce même processus.

Un résultat expérimental obtenu en réalisant ce type d'expérience peut être interprété par la règle ci-dessous, écrite en langage naturel, en considérant que le détecteur utilisé est un anticorps, et qu'une plus grande quantité de la molécule d'intérêt est mesurée dans les cellules traitées :

SI une expérience de phosphorylation pour laquelle le signal est la molécule X, la molécule d'intérêt la molécule Y' qui est détectée par l'anticorps A et obtenue par transformation de la molécule Y, amène à observer une plus grande quantité de Y' dans des cellules traitées par X que dans le contrôle,
ALORS X stimule le processus qui transforme Y en Y'.

Cette règle d'interprétation des expériences de phosphorylation simples peut être formalisée en logique du premier ordre par la règle déductive suivante :

$$\begin{aligned} & pa(X, Y, A, 1) \\ \wedge\ & antibodyAgainst(A, Y') \wedge\ modifiedForm(Y', Y) \\ \Rightarrow\ & modulates(X, Y, Y', 1, unknown, confirmed) \end{aligned} \tag{R2}$$

La partie gauche de la règle (avant le symbole "⇒ ") est appelée ses prémisses, tandis que la partie droite est appelée sa conclusion. Le résultat expérimental apparaît dans les prémisses, tandis que l'interprétation de ce résultat apparaît dans la conclusion. Une telle règle de déduction doit être lue de la manière suivante : si les prémisses sont établies, alors la conclusion l'est également.

Les symboles en lettres minuscules ou chiffres romains contenus entre les parenthèses (p. ex. 1, $confirmed$) sont des constantes qui renvoient à des molécules ou à des paramètres. Les symboles en lettres majuscules eux aussi contenus entre les parenthèses (p. ex. X, A) sont des variables, qui peuvent être instanciées par des constantes. Les symboles contenus en dehors des parenthèses (p. ex. pa, $modulates$) sont des symboles de prédicats, dont les arguments sont les termes contenus à l'intérieur des parenthèses. Un prédicat complètement instancié (p. ex. $modulates(fsh, erk, perk, 1, unknown, confirmed)$) est appelé un *fait*.

Le prédicat $modulates$ prend six arguments, et l'énoncé $modulates(X, Y, Y', E, D, S)$ qualifie la modulation par X du processus moléculaire qui transforme Y en Y'. Les trois arguments supplémentaires (ici, E, D et S) de ce prédicat sont les suivants :

- E est le paramètre d'*effet*. Il représente l'effet de la modulation sur le processus. Il peut prendre la valeur 1 ou -1, selon que la modulation est une stimulation ou une inhibition, respectivement. Ce paramètre est également un argument pour certains prédicats d'expérience comme le prédicat pa. Pour ces prédicats, les valeurs 1 et -1 signifient respectivement une plus grande quantité et une plus faible quantité de la molécule d'intérêt dans le lysat des cellules traitées que dans le contrôle.

- D est le paramètre de *distance*. Il représente la distance avec laquelle la modulation du processus a lieu : une modulation sans intermédiaires est représentée par la constante *direct*, et une modulation avec intermédiaires par la constante *indirect*. Finalement, la constante *unknown* permet de spécifier que la modulation a lieu avec une ditance inconnue.
- S est le paramètre de *statut*. Il peut prendre la valeur $confirmed$ ou *hypothesis*. La constante $confirmed$ signifie que la modulation a été établie, et la constante *hypothesis* que la modulation est hypothétique, et devrait être établie par des expériences supplémentaires.

Les résultats de certains types d'expérience peuvent être interprétés par plusieurs règles d'interprétation, une pour chaque valeur possible du paramètre d'effet. La règle (R2) peut être réécrite pour une expérience de phosphorylation où une plus faible quantité de la molécule Y' est observée dans le lysat des cellules traitées que dans le contrôle (i.e. où $pa(X, Y, A, -1)$ est vrai). Cette expérience de phosphorylation mènerait à la conclusion selon laquelle le signal inhibe le processus qui transforme Y en Y' (i.e. la conclusion $modulates(X, Y, Y', -1, unknown, confirmed)$). Pour une telle expérience, l'effet du prédicat d'expérience est donc le même que celui du prédicat *modulates*. Par conséquent, les deux règles d'interprétation chacune prenant en compte une des deux valeurs 1 et -1 peuvent être généralisées en une seule règle d'interprétation où les valeurs 1 et -1 sont remplacées par une variable E :

$$\begin{aligned} & pa(X, Y, A, E) \\ \wedge\ & antibodyAgainst(A, Y') \wedge\ modifiedForm(Y', Y) \\ \Rightarrow\ & modulates(X, Y, Y', E, unknown, confirmed) \end{aligned} \tag{R3}$$

Nous avons construit un ensemble de règles d'interprétation en considérant tous les types d'expérience communément utilisés en biologie de la signalisation, et en analysant les processus moléculaires entrant en jeu dans ces expériences. Nous présentons les différents types de règles provenant des différents types d'expérience dans la section suivante.

4.2.2 Des règles d'interprétation différentes pour des types d'expérience différents

Les types d'expériences utilisés en biologie de la signalisation cellulaire ne sont pas très nombreux. Nous avons considéré 28 types d'expérience différents, qui ont été utilisés pour construire le réseau induit par le récepteur de la FSH (R-FSH) donné dans [Glo+11]. Le tableau 4.1 montre l'ensemble des types d'expérience pris en compte. À chaque type d'expérience est associé un prédicat qui formalise un résultat expérimental obtenu en réalisant une expérience de ce type.

L'ensemble de ces types d'expérience peut être classé en trois groupes selon le type de connaissances qu'ils permettent de déduire. Les trois différents types de conclusion sont les suivants :

- la qualification d'une modulation, par une certaine molécule, d'un processus moléculaire ;
- l'existence d'un processus de complexation ;
- la localisation d'une molécule dans un compartiment.

Pour chaque type d'expérience simple (comme un test enzymatique ou une expérience de phosphorylation) il existe un certain nombre de types d'expérience complexes correspondants qui font intervenir des perturbateurs (comme un inhibiteur ou un antagoniste).

Nous distinguons deux classes de perturbateurs : les perturbateurs *spécifiques*, comme les inhibiteurs et les antagonistes, qui ciblent une forme particulière (p. ex. phosphorylée) d'une certaine entité moléculaire, et les perturbateurs *non-spécifiques*, comme les petits ARNs interférents (siRNA)

ou les Knock-Down (KO), qui ciblent, *in fine*, toutes les formes d'une certaine entité moléculaire, par exemple toutes les formes provenant des modifications post-traductionnelles d'une protéine donnée. Ces deux classes de perturbateurs peuvent être formalisées par deux prédicats $specificDisruptor$ et $unspecificDisruptor$, et la relation ontologique entre un perturbateur particulier et la classe de perturbateurs à laquelle il appartient par une règle logique. Par exemple, la règle suivante spécifie qu'un inhibiteur est un perturbateur spécifique :

$$inhibitor(I, X) \Rightarrow specificDisruptor(I, X) \tag{R4}$$

Pour chaque type d'expérience, la mesure de la quantité de la molécule d'intérêt est soit réalisée directement soit par l'intermédiaire d'un détecteur (p. ex. un anticorps ou une forme radioactive). De la même façon que pour les perturbateurs, deux classes de détecteurs peuvent être distinguées, également en se basant sur leur spécificité : les détecteurs spécifiques, comme les anticorps, ciblent une forme précise d'une certaine entité moléculaire, alors que les détecteurs non-spécifiques, comme les pan-anticorps, ciblent l'ensemble des formes d'une entité moléculaire donnée. Comme pour la classification des perturbateurs, celle des détecteurs peut être formalisée par deux prédicats $specificDetector$ et $unspecificDetector$, et les règles ontologiques régissant la relation entre ces deux classes et leurs membres par des règles logiques analogues à la règle (R4).

La classification des perturbateurs et détecteurs en deux classes (spécifiques et non-spécficiques) peut être prise en compte pour généraliser, une fois de plus, les règles d'interprétation. Cette généralisation est réalisée en remplaçant dans les règles toute occurrence d'un prédicat formalisant un perturbateur ou un détecteur par le prédicat plus général correspondant à la classe à laquelle il appartient.

En utilisant les classes des perturbateurs et de détecteurs qu'ils font intervenir, et en analysant la forme des prédicats qui les formalisent, les types d'expérience peuvent être classés et regroupés en clusters (voir les couleurs dans le tableau 4.1). Tous les types d'expérience appartenant au même cluster sont formalisés par des prédicats spécifiques qui ont exactement les mêmes arguments, utilisent des perturbateurs et détecteurs appartenant à la même classe, et peuvent être interprétés de la même manière. Par conséquent, tous les types d'expérience appartenant au même cluster peuvent être formalisés par un même prédicat plus général, et les règles d'interprétation qui leur sont associées peuvent être généralisées en une ou plusieurs règles d'interprétation qui utilisent ce prédicat plus général.

Par exemple, une PA (expérience de phosphorylation simple avec un anticorps pour détecteur) et une PRA (expérience de phosphorylation simple avec une forme radioactive pour détecteur) sont deux types d'expérience simples, qui utilisent tous deux un détecteur spécifique : un anticorps pour la PA, et une forme radioactive pour la PRA. De plus, les prédicats qui les formalisent ont exactement les mêmes arguments. Par conséquent, ces deux types d'expérience appartiennent au même cluster, et des résultats provenant de ces deux types peuvent être interprétés par la même règle générale :

$$\begin{aligned} & exp1(X, Y, D, E) \\ \wedge\ & specificDetector(D, Y') \wedge\ modifiedForm(Y', Y) \\ \Rightarrow\ & modulates(X, Y, Y', E, unknown, confirmed) \end{aligned} \tag{R5}$$

Le prédicat $exp1$ formalise n'importe quel résultat provenant d'un type d'expérience appartenant au cluster 1 (représenté en bleu clair dans la tableau 4.1, nommément PA et PRA). La règle d'interpé-

tation (R5) est complétée par deux règles de classification ontologique qui formalisent l'appartenance de PA et PRA au cluster 1 :

$$pa(X, Y, D, E) \Rightarrow \ exp1(X, Y, D, E) \tag{R6}$$

$$pra(X, Y, D, E) \Rightarrow \ exp1(X, Y, D, E) \tag{R7}$$

Tous les arguments des différents prédicats sont *typés*. Par exemple, les arguments du prédicat $pra(X, Y, D, E)$, qui formalise un résultat expérimental de type PRA, doivent être des types suivants : X doit être une molécule ; Y doit être une protéine ; D doit être une molécule radiocative ; E, qui est le paramètre d'effet, doit prendre la valeur 1 ou -1.

Ces contraintes de typage peuvent être exprimées par des contraintes logiques. Par exemple, le typage du premier argument du prédicat *pra* peut être exprimé par la formule suivante :

$$pra(X, Y, D, E) \wedge \ \neg molecule(X) \Rightarrow \ \bot \tag{R8}$$

Ces contraintes de typage permettent en particulier de vérifier que les faits expérimentaux sont bien écrits et qu'ils respectent la sémantique associée à chaque type d'expérience. Par conséquent, étant donné un ensemble de résultats expérimentaux formalisés par des prédicats comme *pa* ou *pra*, il faut vérifier que tous ces résultats expérimentaux respectent les contraintes de typage avant qu'ils puissent être interprétés en de nouvelles connaissances par l'intermédiaire des règles d'interprétation.

4.2.3 Règles d'interprétation simples et complexes

Dans les règles comme la règle (R5), les prémisses comportent un prédicat formalisant un résultat expérimental, et d'autres prédicats qui ne peuvent pas être déduits par une règle d'interprétation (comme le prédicat $modifiedForm$). Nous appelons ce dernier type de prédicats *prédicats du domaine*. Quant à la conclusion, elle est formée d'un unique prédicat, qui formalise, par définition des règles, une connaissance déduite : nous appelons ces prédicats des *prédicats déduits*. Les règles comme (R5) n'ont que des prédicats formalisant des résultats expérimentaux ou des prédicats du domaine dans leurs prémisses, et ne reposent pas sur des prédicats déduits par d'autres règles. Nous appelons de telles règles des *règles simples*.

D'autres règles ont dans leurs prémisses des prédicats déduits, et ne peuvent être utilisées qu'une fois que des règles tierces ont déjà permis de déduire ces prédicats. Nous appelons ces règles *règles complexes*. Toutes les règles contruites pour interpréter des types d'expérience complexes, qui impliquent la présence d'un perturbateur, sont des règles complexes. En effet, les expériences utilisant, par exemple, des inhibiteurs, permettent d'émettre des hypothèses sur des intermédiaires de modulations, en se reposant sur le résultat de la même expérience mais réalisée sans inhibiteur.

Par exemple, l'expérience consistant à comparer la phosphorylation d'ERK induite par la FSH dans des cellules non traitées et dans des cellules traitées avec un inhibiteur de la PKA (par exemple, avec l'inhibiteur H89, voir [Cré+01], figure 4) montre que la phosphorylation d'ERK est entravée par l'inhibiteur, et par conséquent que PKA est peut-être un intermédiaire de la stimulation d'ERK par la FSH. La réalisation de cette expérience, et donc *a fortiori* son interprétation, nécessitent au préalable la connaissance de la stimulation de la phosphorylation d'ERK par FSH.

Ce type d'expérience est appelé $ICPPA$ (expérience de phosphorylation en présence d'un inhibiteur avec un anticorps pour détecteur), et appartient au cluster 2 (représenté en bleu foncé dans le tableau 4.1), ce qui est formalisé par la règle de classification ontologique suivante :

$$icppa(X', Y, A, I, E) \Rightarrow exp2(X', Y, A, I, E) \tag{R9}$$

La règle interprétant ce type d'expérience, quand une plus faible quantité de la molécule d'intérêt est mesurée dans le lysat des cellules traitées par l'inhibiteur que dans le contrôle, est la suivante :

$$\begin{aligned} & exp2(X', Z, D, P, -1) \\ \wedge\ & specificDetector(D, Z') \wedge specificDisruptor(P, Y') \\ \wedge\ & modifiedForm(Y', Y) \\ \wedge\ & notModified(X') \\ \wedge\ & modulates(X', Z, Z', 1, unknown, confirmed) \\ \Rightarrow\ & modulates(Y', Z, Z', 1, unknown, confirmed) \\ \wedge\ & modulates(X', Y, Y', 1, unknown, hypothesis) \end{aligned} \tag{R10}$$

Dans ce type d'expérience, le signal est la molécule X', et la règle ci-dessus formalise le cas où X' n'a pas de forme modifiée intra-cellulaire, d'où la présence du prédicat $notModified(X')$ dans les prémisses de la règle. Le résultat expérimental précédent, portant sur la découverte d'un intermédiaire à la stimulation de la phosphorylation d'ERK par la FSH, sera obtenu par cette règle. En effet, la FSH n'a pas de forme modifiée qui puisse être produite par une réaction à l'intérieur de la cellule.

La figure 4.1 (partie en haut) montre l'interprétation de ce résultat expérimental pour ce cas-là. De ce type d'expérience et sachant que X' n'a pas de forme modifiée, nous pouvons conclure que la molécule Y' stimule la transformation de la molécule Z en la molécule Z', puisque la quantité de Z' est diminuée par la présence d'un perturbateur P de Y'. Cette conclusion est montrée en vert dans la figure 4.1. À partir de cette conclusion, deux alternatives apparaissent : soit Y' peut influencer la transformation de Z en Z' indépendamment de X', soit Y' est un intermédiaire de la stimulation par X' de la transformation de Z en Z', et X' stimule la transformation de Y en Y'. Comme il n'est pas possible de choisir entre ces deux alternatives avec seulement cette expérience, ces deux alternatives sont considérées comme hypothétiques tant qu'une des deux n'a pas été confirmée. Seule la deuxième alternative amène à conclure sur un nouveau fait, qui est ici hypothétique, et représenté en rouge dans la figure 4.1.

Pour reprendre notre exemple, nous pouvons conclure que la PKA stimule la phosphorylation d'ERK, mais nous ne pouvons qu'émettre l'hypothèse que la PKA est un intermédiaire de la stimulation par la FSH de cette phosphorylation.

La règle suivante permet d'interpréter le même résultat expérimental, mais sachant que le signal

X' a une forme modifiée X :

$$\begin{aligned} & exp2(X', Y, D, P, -1) \\ \wedge\ & specificDetector(D, Z') \wedge\ specificDisruptor(P, Y') \\ \wedge\ & modifiedForm(Y', Y) \\ \wedge\ & modfiedForm(X', X) \\ \wedge\ & modulates(X', Z, Z', 1, unknown, confirmed) \\ \Rightarrow\ & modulates(Y', Z, Z', 1, unknown, confirmed) \\ \wedge\ & modulates(X', Y, Y', 1, unknown, hypothesis) \\ \wedge\ & modulates(Y', X, X', 1, unknown, hypothesis) \end{aligned} \tag{R11}$$

L'interprétation du résultat expérimental pour ce cas-là est montrée dans la figure 4.1 (partie basse). Cette règle a une conclusion supplémentaire, comparé à la règle (R10) :

$$modulates(Y', X, X', i, unknown, hypothesis)$$

Les deux alternatives données pour le cas où X' n'a pas de forme modifiée restent valables, et sont représentées dans la partie gauche de la figure. Il y a cependant une alternative supplémentaire pour ce cas-ci, représentée cette fois-ci dans la partie droite : il est possible que X' soit un intermédiaire de la stimulation par Y' de la transformation de Z en Z', et ainsi que Y' stimule la transformation de X en X'. Comme pour les cas précédents, cette alternative reste une hypothèse, qui ne peut pas être exclue sans expérience supplémentaire.

4.2.4 Des règles d'interprétation explicites pour une interprétation prudente

Les différentes alternatives qui apparaissent lors d'une interprétation d'une $ICPPA$ ne sont généralement pas prises en compte par les biologistes. Pour des raisons de simplicité, ces derniers concluent généralement que Y' est un intermédiaire de la modulation par X' de la transformation de Z en Z'.

D'un point de vue général, la prise en compte de ces différentes alternatives amenant à émettre des hypothèses plutôt qu'à établir des faits est permise par l'utilisation de règles explicites, et est un des principaux atouts de l'utilisation de méthodes de raisonnement automatique. Lorsque les biologistes interprètent des résultats expérimentaux, ils ne retiennent le plus souvent que les hypothèses qu'ils considèrent les plus probables, produisant des modèles incomplets et biaisés. Au contraire, l'utilisation de méthodes de raisonnement automatiques permet de garder toutes les hypothèses sans éliminer les moins probables d'entre elles, jusqu'à ce que certaines soient confirmées ou invalidées par l'expérience. De plus, le choix du biologiste entre différentes alternatives est souvent basé sur de la connaissance *a priori* le plus souvent implicite, et qui n'est pas tout le temps établie à partir de bases expérimentales solides.

4.3 Règles de raisonnement

En plus des règles d'interprétation, nous avons construit un ensemble de règles de raisonnement qui ne permettent pas d'interpréter des résultats expérimentaux, mais plutôt de compléter les connaissances du domaine, de déduire de nouvelles connaissances relatives aux réseaux, ou encore d'émettre de nouvelles hypothèses, à partir du seul raisonnement.

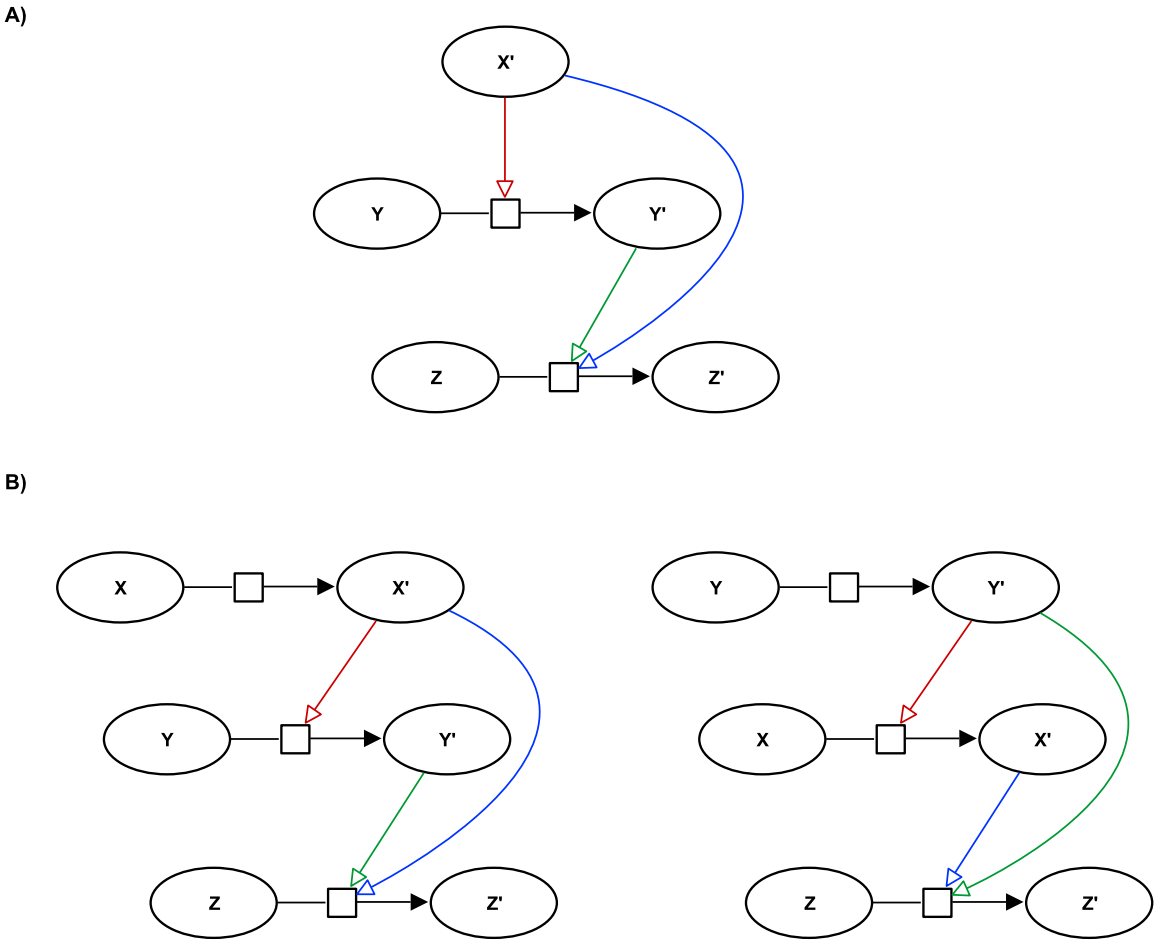


FIGURE 4.1 – **Interprétation schématisée sous forme de réseau d'une expérience de phosphorylation avec un inhibiteur.** A) Cas où le signal X' n'a pas de forme modifiée, interprété par la règle (R10). B) Cas où le signal X' a une forme modifiée X, interprété par la règle (R11). La partie gauche représente l'alternative où Y' est un intermédiaire de la phosphorylation de Z par X', et celle de droite où X' est un intermédiaire de la phosphorylation de Z par Y'.
Les modulations en vert représentent les conclusions des règles (R10) et (R11) qui sont confirmées, celles en rouge les conclusions qui restent des hypothèses. Quant à celles qui sont en bleu, elles représentent des prémisses de ces mêmes règles.

Par exemple, la règle (R12) permet de déduire une nouvelle relation antigène/anticorps, tandis que la règle (R13) permet de déduire une nouvelle modulation sachant que deux molécules ont les même fonctions moléculaires :

La règle suivante formalise le fait que si A est un pan-anticorps d'une molécule X et que X' est une forme modifiée de X, alors A est un pan-anticorps de X' :

$$\begin{aligned} & panantibodyAgainst(A, X) \wedge\ modifiedForm(X', X) \\ \Rightarrow\ & panantibodyAgainst(A, X') \end{aligned} \tag{R12}$$

Quant à la règle suivante, elle concerne les équivalents fonctionnels. Un équivalent fonctionnel est une molécule qui le plus souvent n'est pas présente dans les systèmes biologiques étudiés mais qui présente les mêmes propriétés fonctionnelles qu'une des molécules du système. Par exemple la molécule 8-CPT-2Me-cAMP est un équivalent fonctionnel de la cAMP, qui permet d'activer sélectivement la molécule EPAC. Ces équivalents fonctionnels sont utilisés dans les expériences pour "mimer" l'effet d'un molécule d'intérêt donnée. Cette règle formalise le fait que si X' et X sont deux équivalents fonctionnels, et X module un processus transformant Y en Y' avec un certain effet E, alors X' module ce même processus avec ce même effet.

$$\begin{aligned} & modulates(X, Y, Y', confirmed, E) \wedge\ functionalEquivalent(X', X) \\ \Rightarrow\ & modulates(X', Y, Y', confirmed, E) \end{aligned} \tag{R13}$$

Finalement, la règle (R14), donnée ci-après, permet d'émettre une hypothèse sur une modulation, en se basant sur des modulations qui ont été préalablement établies. Cette règle formalise le fait que si X' stimule la transformation de Z en Z', et Y' stimule ce même processus mais de façon directe (i.e. sans qu'un intermédiaire n'entre en jeu), alors il est possible que X' stimule la transformation de Y en Y' :

$$\begin{aligned} & modulates(X', Z, Z', 1, unknown, confirmed) \\ \wedge\ & modulates(Y', Z, Z', 1, direct, confirmed) \\ \wedge\ & modifiedForm(Y', Y) \\ \Rightarrow\ & modulates(X', Y, Y', 1, unknown, hypothesis) \end{aligned} \tag{R14}$$

4.4 Inférence de réseaux de signalisation et de plans expérimentaux

Nous montrons dans la suite comment nos règles d'interprétation et de raisonnement permettent à la fois d'inférer des réseaux de signalisation à l'échelle des processus moléculaires mais aussi des plans expérimentaux visant à tester une hypothèse biologique donnée.

4.4.1 Inférence automatique de réseaux de signalisation par déduction

Nos règles d'interprétation et de raisonnement permettent d'interpréter automatiquement un ensemble de résultats expérimentaux donné par de nouvelles connaissances ou hypothèses.

L'interprétation des résultats utilise une méthode de raisonnement dite *déductive*. Ce mode de raisonnement permet d'inférer une conclusion à partir de prémisses. Plus formellement, considérant un ensemble de règles logiques, dénoté par B_R, et un ensemble de faits dénoté par B_F, le raisonnement déductif permet d'inférer tous les faits f tels que $B_R \cup B_F \models f$, où $\models$ désigne la notion classique de conséquence logique.

Pour l'interprétation d'un ensemble donné de résultats expérimentaux, l'ensemble B_F de faits considéré est constitué des faits formalisant ces résultats et d'un certain nombre de faits du domaine, tandis que l'ensemble B_R est constitué des règles d'interprétation et de raisonnement que nous avons présentées plus haut.

Exemple. Nous illustrons cette tâche de raisonnement en interprétant automatiquement trois résultats expérimentaux, qui peuvent être formalisés par les faits donnés ci-après. Pour chacun de ces faits, nous donnons également sa signification en langage naturel :

- $pa(fsh, erk, a\_perk, 1)$: la quantité de la molécule d'intérêt détectée par l'anticorps a_perk est plus grande dans des cellules stimulées par la FSH que dans le contrôle ;
- $pa(fsh, mek, a\_pmek, 1)$: même chose mais pour la molécule détectée par l'anticorps a_pmek ;
- $icppa(fsh, erk, a\_perk, pd98059, -1)$: la quantité de la molécule d'intérêt détectée par l'anticorps a_perk est plus faible dans des cellules stimulées par la FSH et préalablement traitées par l'inhibiteur $PD98059$ que dans des cellules stimulées par la FSH non-traitées (le contrôle).

Considérons également l'ensemble de faits du domaine suivant :

- $modifiedForm(perk, erk)$: phospho-ERK est une forme modifiée de ERK ;
- $modifiedForm(pmek, mek)$: phospho-MEK est une forme modifiée de MEK ;
- $notModified(fsh)$: la FSH n'a pas de forme modifiée ;
- $antibodyAgainst(a\_perk, perk)$: a_perk est un anticorps spécifique de phospho-ERK ;
- $antibodyAgainst(a\_pmek, pmek)$: a_pmek est un anticorps spécifique de phospho-MEK ;
- $inhibitor(pd98059, pmek)$: la PD98059 est un inhibiteur de phospho-MEK.

Trois étapes successives de déduction permettent de déduire de nouvelles modulations à partir de ces trois résultats expérimentaux.

Premièrement, en utilisant les règles de classification ontologique des prédicats d'expérience (p. ex. la règle (R6)) et les règles ontologiques pour les détecteurs et les perturbateurs (p. ex. la règle (R4)), nous déduisons les faits suivants :

- $exp1(fsh, mek, a\_pmek, 1)$;
- $exp1(fsh, erk, a\_perk, 1)$;
- $exp2(fsh, erk, a\_perk, pd98059, 1)$;
- $specificDetector(a\_perk, perk)$;
- $specificDetector(a\_pmek, pmek)$;
- $specificDisruptor(pd98059, pmek)$.

Deuxièmement, nous utilisons la règle simple (R5) pour interpréter les deux PA, et ainsi déduire deux nouvelles modulations :

- $modulates(fsh, erk, perk, 1, unknown, confirmed)$: la FSH stimule la transformation de ERK en phospho-ERK ;
- $modulates(fsh, mek, pmek, 1, unknown, confirmed)$: la FSH stimule la transformation de MEK en phospho-MEK.

Finalement, en utilisant ce dernier fait et la règle complexe (R10), nous interprétons le résultat expérimental provenant d'une $ICPPA$, et déduisons les modulations suivantes :

- $modulates(pmek, erk, perk, 1, unknown, confirmed)$
- $modulates(fsh, mek, pmek, 1, unknown, hypothesis)$

4.4.1.1 Déduction par ASP

La déduction des nouveaux faits est réalisée en ASP, avec le logiciel clingo.

Étant donné un ensemble de résultats expérimentaux que nous voulons interpréter, soit B_F l'ensemble de faits rassemblant des faits formalisant ces résultats, et des faits du domaine. Soit B_R l'ensemble de nos règles d'interprétation et de raisonnement transformées en règles normales. Cette transformation est rendue possible par le fait que chacune de nos règles comporte une conclusion qui est soit un unique littéral, soit une conjonction de littéraux. Une règle peut ainsi être transformée en un ensemble de règles normales dont le corps est celui de la règle d'origine, et la tête l'un des littéraux de la conclusion de la règle d'origine.

Nous dénotons par $B = B_F \cup B_R$ la théorie logique formée des faits de B_F et des règles de B_R. Nous transformons la théorie B en un programme ASP Π, en remplaçant le symbole de l'implication par le symbole ASP correspondant, c'est-à-dire le symbole " :– ".

Comme la théorie B ne comporte aucun symbole de fonction, Π n'en comporte pas non plus, et est finiement instanciable. De plus, étant donné que B ne comporte pas de négation, Π n'en comporte pas non plus. Par conséquent, Π est stratifié, et admet un unique modèle stable. Les atomes de cet unique modèle stable sont alors exactement les faits qui peuvent être déduits de B.

Le Listing 4.1 donne le programme ASP pour l'interprétation des résultats expérimentaux de l'exemple précédent, ainsi que le résultat de cette interprétation donné par clingo.

Listing 4.1 – Exemple de programme ASP pour l'interprétation automatique de résultats expérimentaux

```
1  pa(fsh,erk,a_perk,1).
2  pa(fsh,mek,a_pmek,1).
3  icppa(fsh,erk,a_perk,pd98059,-1).
4
5  modifiedForm(perk,erk).
6  modifiedForm(pmek,mek).
7  notModified(fsh).
8  antibodyAgainst(a_perk,perk).
9  antibodyAgainst(a_pmek,pmek).
10 inhibitor(pd98059,pmek).
11
12 specificDisruptor(I,X):-inhibitor(I,X).
13 specificDetector(A,X):-antibodyAgainst(A,X).
14 specificDetector(R,X):-radioLabeledForm(R,X).
15
16 exp1(X,D,P,E):-pa(X,D,P,E).
17 exp1(X,D,P,E):-pra(X,D,P,E).
18 exp2(X',Y,D,P,E):-icppa(X',Y,D,P,E).
19
20 modulates(X,Y,Y',E,unknown,confirmed):-exp1(X,Y,D,E);specificDetector(D,Y');
       modifiedForm(Y',Y).
21
22 modulates(Y',Z,Z',1,unknown,confirmed):-exp2(X',Z,D,P,-1);specificDetector(D
       ,Z');specificDisruptor(P,Y');modifiedForm(Y',Y);notModified(X');modulates
       (X',Z,Z',1,unknown,confirmed).
23 modulates(X',Y,Y',1,unknown,hypothesis):-exp2(X',Z,D,P,-1);specificDetector(
       D,Z');specificDisruptor(P,Y');modifiedForm(Y',Y);notModified(X');
       modulates(X',Z,Z',1,unknown,confirmed).
24
25 Solving...
26 Answer: 1
```

```
27 pa(fsh,erk,a_perk,1) pa(fsh,mek,a_pmek,1) modifiedForm(perk,erk)
       modifiedForm(pmek,mek) notModified(fsh) antibodyAgainst(a_perk,perk)
       antibodyAgainst(a_pmek,pmek) inhibitor(pd98059,pmek) icppa(fsh,erk,a_perk
       ,pd98059,-1) specificDisruptor(pd98059,pmek) specificDetector(a_perk,perk
       ) specificDetector(a_pmek,pmek) exp1(fsh,erk,a_perk,1) exp1(fsh,mek,
       a_pmek,1) exp2(fsh,erk,a_perk,pd98059,-1) modulates(fsh,erk,perk,1,
       unknown,confirmed) modulates(fsh,mek,pmek,1,unknown,confirmed) modulates(
       pmek,erk,perk,1,unknown,confirmed) modulates(fsh,mek,pmek,1,unknown,
       hypothesis)
28 SATISFIABLE
29
30 Models       : 1
31 Calls        : 1
32 Time         : 0.003s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
33 CPU Time     : 0.000s
```

Les lignes 1–23 définissent le programme ASP tandis que les lignes 25–27 donnent la sortie de clingo pour ce programme.

Les lignes 1–3 donnent les faits formalisant les résultats expérimentaux, les lignes 5–10 les faits du domaine.

Les lignes 12–14 donnent le règles de classification ontologique des disrupteurs et des détecteurs, et les lignes 16–18 celles des expériences. La ligne 20 donne la règle d'interprétation pour les types d'expérience appartenant au cluster 1 (la règle (R5)), et les lignes 22–23 donnent la règle d'interprétation pour les types d'expérience appartenant au cluster 2 pour le cas où le signal X' n'a pas de forme modifiée (la règle (R10)).

Finalement, la ligne 27 donne l'ensemble des faits déduits à partir des résultats expérimentaux, des faits du domaine et des règles. Nous retrouvons bien l'ensemble des faits que nous avons déduits à la sous-section 4.4.1. Notons que les faits déduits comprennent aussi les faits formalisant les résultats expérimentaux et les faits du domaine.

4.4.1.2 Traces des faits déduits

Nous obtenons l'ensemble des faits déduits à partir d'un programme ASP Π, et nous voulons garder une trace du raisonnement qui a permis de les obtenir.

Pour un fait déduit, nous représentons une trace du raisonnement qui a permis de le déduire par un arbre.

Soit B_R un ensemble de règles normales, B_F un ensemble de faits, et f un fait tel que $B_R \cup B_F \models f$. Une trace de f est définie de la manière suivante :

Définition 4.1 (Trace d'un fait)**.** Une *trace* de f est un arbre avec la structure récursive suivante :

- si $f \in B_F$, une racine étiquetée par f ;
- si $f \notin B_F$, une racine étiquetée par f dont les sous-arbres sont des traces des faits $f_1, \ldots, f_n$, respectivement, tels qu'il existe une règle R de B_R et une substitution θ pour lesquelles $R\theta$ soit égal à $f_1 \wedge \cdots \wedge f_n \Rightarrow f$.

Ces traces peuvent être reconstruites en associant à chaque règle du programme Π qui n'est pas un fait un nom, et en réécrivant cette règle en deux nouvelles règles.

Soit R une règle de Π qui n'est pas un fait. Comme Π est un programme ASP correspondant à un programme logique normal, R est de la forme :

$$H_R \text{ :- } B_1;\ \ldots;\ B_{n_R}.$$

Nous associons à R un prédicat dénoté par $name_R$, et nous dénotons par $vars(R)$ le tuple des variables apparaissant dans le corps de R, lu de gauche à droite. La règle R est réécrite sous la forme des deux règles suivantes :

$$\begin{aligned} name_R(X_1, \ldots, X_{m_R}) &\text{ :- } B_1;\ \ldots;\ B_{n_R}. \\ H_R &\text{ :- } name_R(X_1, \ldots, X_{m_R}). \end{aligned}$$

où $(X_1, \ldots, X_{m_R}) = vars(R)$.

En réécrivant chaque règle de Π qui n'est pas un fait sous la forme de deux règles, nous obtenons un programme Π^{Tr}. Ce programme permet non seulement de déduire tous les faits qui sont déduits par Π, mais aussi de garder une trace des règles qui ont servi à déduire ces faits, sous la forme des atomes $name_R(X_1, \ldots, X_{m_R})$. En effet, le programme Π^{Tr} admet également un unique modèle stable, dénoté par M^{Tr}, qui est un super-ensemble du modèle stable de Π. Plus précisément, M^{Tr} est l'union de M et d'un ensemble d'atomes qui sont des instances des prédicats de la forme $name_R$.

Les traces des faits déduits peuvent être obtenues à partir du modèle stable M^{Tr}.

Pour chaque atome $name_R(t_1, \ldots, t_{m_R})$ de M^{Tr} qui est une instance d'un prédicat de la forme $name_R(X_1, \ldots, X_{m_R})$, nous construisons une règle instanciée qui est la règle R instanciée par les termes $t_1, \ldots, t_{m_R}$. Nous obtenons alors un ensemble de règles instanciées qui vérifie les deux propriétés suivantes :

- la tête d'une règle de cet ensemble est un atome de M ;
- pour chaque atome de M qui n'est pas un fait de Π, il existe une règle de cet ensemble telle que sa tête soit exactement cet atome.

Ces deux propriétés découlent directement de la façon dont nous avons contruit Π^{Tr} à partir de Π.

Ainsi, pour chaque atome de M, soit il existe au moins une règle instanciée de cet ensemble telle que sa tête soit exactement cet atome, soit cet atome est un fait de Π. Par conséquent, une trace d'un atome a de M peut être construite récursivement : si a est un fait de M, alors sa trace est un unique noeud étiqueté par a ; si a n'est pas un fait de Π, alors sa trace comprend une racine étiquetée par f, qui a pour fils les sous-arbres qui correspondent à des traces des atomes qui apparaissent dans le corps d'une règle instanciée dont la tête est a.

Nous illustrons l'obtention des traces sur un exemple. Soit le programme suivant ASP Π suivant :

$$\begin{aligned} &b(1,2). \\ &c(2,3). \\ &a(X,Z) \text{ :- } b(X,Y);\ c(Y,Z). \\ &a(Y,X) \text{ :- } a(X,Y). \end{aligned}$$

Le programme Π admet l'unique modèle stable M suivant :

$$M = \{b(1,2), c(2,3), a(1,3), a(3,1)\}$$

Nous associons aux deux dernières règles de ce programme les prédicats $r1$ et $r2$, respectivement, et réécrivons ces deux dernières règles pour obtenir le programme Π^{Tr} suivant :

$$
\begin{aligned}
&b(1,2).\\
&c(2,3).\\
&r1(X,Y,Z) \text{ :- } b(X,Y);\ c(Y,Z).\\
&a(X,Z) \text{ :- } r1(X,Y,Z).\\
&r2(X,Y) \text{ :- } a(X,Y).\\
&a(Y,X) \text{ :- } r2(X,Y).
\end{aligned}
$$

Ce programme admet un unique modèle stable M^{Tr}, comportant les atomes suivants :

$$M^{Tr} = \{b(1,2), c(2,3), a(1,3), a(3,1), r1(1,2,3), r2(1,3), r2(3,1)\}$$

Le modèle M^{Tr} comporte trois atomes construits à partir des prédicats $r1$ et $r2$: $r1(1,2,3)$, $r2(1,3)$ et $r2(3,1)$. Nous construisons donc un ensemble de trois règles instanciées, en instanciant la règle associée à $r1$ avec les termes $(1,2,3)$ et celle associée à $r2$ avec les termes $(1,3)$ ou les termes $(3,1)$. Les trois règles instanciées de cet ensemble sont les suivantes :

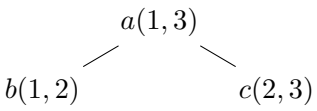
$$
\begin{aligned}
&a(1,3) \text{ :- } b(1,2);\ c(2,3).\\
&a(3,1) \text{ :- } a(1,3).\\
&a(1,3) \text{ :- } a(3,1).
\end{aligned}
$$

L'atome $b(1,2)$ étant un fait de Π, sa trace est un unique noeud étiqueté par cet atome. Il en est de même pour l'atome $c(2,3)$.

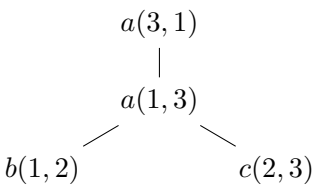
Quant aux traces des atomes $a(1,3)$ et $a(3,1)$, elles peuvent être représentées par des arbres construits à l'aide de l'ensemble de règles instanciées donné précédemment.

Pour chacun des ces atomes, il existe une infinité de traces. En effet, la règle associée au prédicat $r2$ peut être répétée un nombre illimité de fois. Nous donnons ici, pour chacun des deux atomes, l'unique trace de cet atome qui ne contient pas deux fois le même noeud.

La trace de $a(1,3)$ est l'arbre suivant :



Finalement, celle de l'atome $a(3,1)$ est l'arbre suivant :



4.4.2 Inférence automatique de plans d'expériences par abduction

Notre ensemble de règles peut aussi être utilisé afin de proposer des plans expérimentaux pour confirmer ou invalider une hypothèse donnée. La proposition des plans expérimentaux utilise une méthode de raisonnement dite *abductive*.

De façon générale, le raisonnement abductif permet d'inférer des ensembles minimaux de faits d'une forme particulière qui sont suffisants pour expliquer une certaine observation étant donnée une théorie logique. La forme des faits inférés est généralement contrainte par un ensemble de prédicats appelés *abducibles*. : tout fait inféré doit être une instance d'un abducible.

Plus formellement : Soit B une théorie logique, A un ensemble d'abducibles, et O une observation telle que $B \not\models O$. Le raisonnement abductif consiste à trouver tous les ensembles minimaux H de littéraux tels que :

- $B \wedge H \models O$;
- $B \wedge H \not\models \bot$;
- tout littéral de H est une instance d'un littéral de A.

Pour l'inférence de plans expérimentaux, nous spécifions l'observation, la théorie logique et l'ensemble des abducibles de la manière suivante :

- l'observation O est un fait qui formalise l'hypothèse biologique pour laquelle nous voulons inférer des plans expérimentaux ;
- la théorie logique B est formée de notre ensemble de règles d'interprétation, de raisonnement, et d'un ensemble de faits instanciés qui représentent la connaissance du domaine ou des connaissances déjà déduites ;
- l'ensemble des abducibles A est l'ensemble des prédicats non-instanciés formalisant les différents types d'expérience.

Afin d'inférer les plans expérimentaux qui permettraient de tester une hypothèse biologique donnée, nous supposons que cette hypothèse est vérifiée, et nous formalisons cette supposition par un fait avec le statut $confirmed$. Le raisonnement abductif permet alors d'obtenir des ensembles alternatifs de faits formalisant des résultats expérimentaux, chacun de ces ensembles étant suffisant pour expliquer le fait biologique d'intérêt.

Étant donné un ensemble de faits inféré par notre méthode abductive, la ou les expériences à mener afin de tester l'hypothèse que nous avons supposée vérifiée peuvent être obtenues directement à partir de ces faits en en supprimant le paramètre d'effet. L'ensemble des faits sans le paramètre d'effet formalisent alors un ensemble d'expériences à mener, qui constitue un plan expérimental. Si les résultats obtenus en laboratoire en réalisant ces expériences concordent avec les résultats expérimentaux inférés par abduction, alors l'hypothèse est en effet vérifiée, et elle est invalidée sinon.

Exemple. Nous illustrons la proposition de plans expérimentaux sur un exemple. Nous faisons l'hypothèse que phospho-MEK stimule la phosphorylation d'ERK, et nous voulons obtenir des plans expérimentaux qui permettent de tester cette hypothèse, qui soient exclusivement composés d'expériences de phosphorylation.

L'observation abductive que nous contruisons à partir de cette hypothèse est le fait suivant :

$$modulates(pmek, erk, perk, 1, unknown, confirmed) \quad \text{(F1)}$$

Notons que ce fait n'a pas le statut *hypothesis*, mais $confirmed$. Nous supposons en effet que l'hypothèse est vérifiée.

Pour cet exemple, nous considérons la théorie formée des règles (R5), (R6), (R7), (R9) et (R10), et des faits suivants :

- $modifiedForm(pmek, mek)$
- $modifiedForm(perk, erk)$
- $notModified(fsh)$
- $antibodyAgainst(a\_perk, perk)$
- $antibodyAgainst(a\_pmek, pmek)$
- $radioLabelledForm(r\_erk, erk)$
- $inhibitor(pd98059, pmek)$

Enfin, nous limitons l'ensemble des abducibles aux prédicats qui formalisent les résultats provenant des expériences de phosphorylation suivantes PA, PRA et $ICPPA$, c'est-à-dire aux prédicats suivants : $pa(X, Y, A, E)$, $pra(X, Y, A, E)$, $icppa(X, Y, A, I, E)$.

Il y a deux explications principales pour le fait (F1) :

(1) Le fait (F1) pourrait être une conséquence de la règle (R5), à condition que

$$exp1(pmek1, erk, a\_perk, 1) \tag{F2}$$

ou

$$exp1(pmek1, erk, r\_perk, 1) \tag{F3}$$

soit vrai.

À leur tour, chacun de ces deux faits pourrait être une conséquence de la règle (R6) ou de la règle (R7), menant à quatre résultats expérimentaux possibles (deux provenant d'une PA, l'une utilisant un anticorps de phospho-ERK, et l'autre utilisant une forme radioactive de cette même molécule ; et deux d'une PRA, pour les mêmes raisons). Seules la PA utilisant l'anticorps et la PRA utilisant la forme radioactive respectent les contraintes de typage des arguments des prédicats qui leur sont associés. Ainsi, parmi les expériences simples, seuls les deux résultats expérimentaux suivants pourraient indépendamment expliquer le fait (F1) :

$$pa(pmek1, erk, a\_perk, 1) \tag{F4}$$

et

$$pra(pmek1, erk, r\_perk, 1) \tag{F5}$$

(2) Le fait (F1) pourrait être une conséquence de la règle (R10), à condition que le fait

$$modulates(fsh, erk, perk, 1, unknown, confirmed) \tag{F6}$$

soit vrai et au moins l'un de ces deux faits soit vrai :

$$exp2(fsh, erk, a\_perk, pd98059, -1) \tag{F7}$$

$$exp2(fsh, erk, r\_perk, pd98059, -1) \tag{F8}$$

En raisonnant comme précédemment, la modulation formalisée par le fait (F6) pourrait indépendamment être expliquée par deux résultats provenant d'expériences de phosphorylation simples :

$$pa(fsh, erk, a\_perk, 1) \tag{F9}$$

et

$$pra(fsh, erk, r\_perk, 1) \tag{F10}$$

Deux explications restent possibles, une pour chacun des faits (F7) et (F8). Chacun de ces deux faits pourrait être une conséquence de la règle (R9). Par conséquent, le fait (F7) pourrait être expliqué par le résultat expérimental formalisé par le fait

$$icppa(fsh, erk, a\_perk, pd98059, -1) \tag{F11}$$

et le fait (F8) par le résultat expérimental formalisé par le fait

$$icppa(X, mek1, r\_perk, pd98059, -1) \tag{F12}$$

Ce dernier fait ne respecte pas les contraintes de typage des arguments du prédicat *icppa* (le détecteur devant être un anticorps et non une forme radioactive), et n'est donc pas une explication satisfaisante pour le fait (F8).

Pour résumer ce point (2), notre hypothèse supposée vraie et formalisée par le fait (F1) pourrait être expliquée par le fait (F6) et l'un des deux faits (F7) ou (F8). Le fait (F6) peut à son tour être expliqué par un résultat d'expérience simple parmi les deux résultats formalisés par les faits (F9) et (F10). Quant aux faits (F7) et (F8), le premier peut être expliqué par le résultat d'une expérience de phosphorylation complexe formalisé par le fait (F11), et le deuxième n'a pas d'explication possible.

Le résultat des inférences faites aux points (1) et (2) est la collection d'ensembles de résultats expérimentaux suivante :

- $\{pa(pmek1, erk, a\_perk, 1)\}$
- $\{pra(pmek1, erk, r\_perk, 1)\}$
- $\{pa(fsh, erk, a\_perk, 1), icppa(fsh, erk, a\_perk, pd98059, -1)\}$
- $\{pra(fsh, erk, r\_perk, 1), icppa(fsh, erk, a\_perk, pd98059, -1)\}$

Chacun de ces ensembles est une explication possible du fait (F1). Des plans expérimentaux peuvent être obtenus à partir de ces résultats expérimentaux en ne prenant pas en compte le paramètre d'effet.

Les deux premiers plans expérimentaux consistent à montrer que la phosphorylation d'ERK est stimulée en présence de phospho-MEK à l'aide d'expériences de phosphorylation simples où phospho-MEK serait le signal. Ces deux expériences semblent difficiles à mettre en place. Les deux derniers plans

expérimentaux consistent d'abord à montrer qu'en présence de la FSH, la phosphorylation d'ERK est stimulée. Ceci peut être établi à l'aide d'expériences de phosphorylation simples, avec la FSH comme signal. Puis il faut montrer que cette stimulation est moins forte en présence de l'inhibiteur PD98059, à l'aide d'une expérience de phosphorylation complexe où le signal serait encore une fois la FSH.

Le troisième de ces plans expérimentaux a été mis en place par les auteurs de [Ala+09] (voir la figure 6-B de cet article). Ils ont ainsi pu valider le fait (F1), confirmant ainsi l'hypothèse selon laquelle phopsho-MEK stimule la phosphorylation d'ERK.

4.4.2.1 Raisonnement abductif avec le logiciel SOLAR

Le raisonnement par abduction est réalisé à l'aide du logiciel SOLAR [Nab+10]. SOLAR est un logiciel de *consequence finding* qui implémente la résolution SOL [Ino92] à l'aide de tableaux, pour des théories propositionnelles ou du premier ordre.

Étant donnés une théorie logique et un ensemble de littéraux appelé *champs de production* et dénoté par $\mathcal{P}$, SOLAR calcule toutes les conséquences *θ-subsomption minimales* qui sont des instances des littéraux du champ de production. Ces conséquences sont appelées *clauses caractéristiques* de cette théorie par rapport au champ de production $\mathcal{P}$, et peuvent être vues comme les conséquences "intéressantes" de la théorie.

Le raisonnement abductif peut être vu comme une tâche de consequence finding, en considérant le principe d'*inverse entailment*. Comme nous l'avons déjà écrit, le raisonnement abductif consiste à trouver tous les ensembles minimaux H de littéraux tels que :

- $B \wedge H \models O$;
- $B \wedge H \not\models \bot$;
- tout littéral de H est une instance d'un littéral de A.

Par le principe de l'implication inverse, si $B \wedge H \models O$, alors $B \wedge \neg O \models \neg H$. Trouver l'ensemble des explications de O revient alors à trouver l'ensemble des conséquences de $B \wedge \neg O$ qui ne sont pas conséquences de B seule, et qui sont des instances des littéraux de l'ensemble $\{\neg a \mid a \in A\}$.

En fixant le champ de production $\mathcal{P}$ à l'ensemble $\{\neg a \mid a \in A\}$, l'ensemble des explications minimales de O est donc l'ensemble des clauses caractéristiques de $B \wedge \neg O$ par rapport à $\mathcal{P}$ qui ne sont pas des clauses caractéristiques de B seule, toujours par rapport à $\mathcal{P}$ (voir le théorème 2 de [Nab+10]).

Le Listing 4.2 donne le programme SOLAR pour l'inférence des plans expérimentaux de l'exemple précédent, ainsi que le résultat de cette inférence donné par SOLAR.

Listing 4.2 – Exemple de fichier SOLAR pour la proposition automatique de plans expérimentaux

```
1  cnf(f,axiom,[modifiedForm(pmek,mek)]).
2  cnf(f,axiom,[modifiedForm(perk,erk)]).
3  cnf(f,axiom,[notModified(fsh)]).
4  cnf(f,axiom,[antibodyAgainst(a_perk,perk)]).
5  cnf(f,axiom,[antibodyAgainst(a_pmek,pmek)]).
6  cnf(f,axiom,[radioLabelledForm(r_perk,perk)]).
7  cnf(f,axiom,[inhibitor(pd98059,pmek)]).
8
9  cnf(r,axiom,[-antibodyAgainst(A,X),+specificDetector(A,X)]).
10 cnf(r,axiom,[-radioLabelledForm(R,X),+specificDetector(R,X)]).
11 cnf(r,axiom,[-inhibitor(I,X),+specificDisruptor(I,X)]).
12
13 cnf(r,axiom,[-pa(X,Y,A,E),+exp1(X,Y,A,E)]).
14 cnf(r,axiom,[-pra(X,Y,A,E),+exp1(X,Y,A,E)]).
15 cnf(r,axiom,[-icppa(X,Y,A,I,E),+exp2(X,Y,A,I,E)]).
```

```
16
17 cnf(r,axiom,[-exp1(X,Y,A,E),-specificDetector(A,YY),-modifiedForm(YY,Y),+
       modulates(X,Y,YY,E,unknown,confirmed)]).
18
19 cnf(r,axiom,[-exp2(XX,Y,A,I,-1),-specificDetector(A,ZZ),-specificDisruptor(I
       ,YY),-modifiedForm(YY,Y),-notModified(XX),-modulates(XX,Z,ZZ,1,unknown,
       confirmed),+modulates(YY,Z,ZZ,1,unknown,confirmed)]).
20 cnf(r,axiom,[-exp2(XX,Y,A,I,-1),-specificDetector(A,ZZ),-specificDisruptor(I
       ,YY),-modifiedForm(YY,Y),-notModified(XX),-modulates(XX,Z,ZZ,1,unknown,
       confirmed),+modulates(XX,Y,YY,1,unknown,hypothesis)]).
21
22 cnf(top,top_clause,[-modulates(pmek,erk,perk,1,unknown,confirmed)]).
23
24 pf([-pa(_,_,_,_),-pra(_,_,_,_),-icppa(_,_,_,_,_)]).
25
26 SOLAR (SOL for Advanced Reasoning) 2.0 alpha (build 314)
27 SATISFIABLE
28
29 12 FOUND CONSEQUENCES
30 [-pa(pmek,erk,r_perk,1)]
31 [-icppa(fsh,mek,a_perk,pd98059,-1), -pa(fsh,erk,a_perk,1)]
32 [-icppa(fsh,mek,a_perk,pd98059,-1), -pra(fsh,erk,r_perk,1)]
33 [-icppa(fsh,mek,r_perk,pd98059,-1), -pa(fsh,erk,r_perk,1)]
34 [-pa(pmek,erk,a_perk,1)]
35 [-pra(pmek,erk,r_perk,1)]
36 [-icppa(fsh,mek,a_perk,pd98059,-1), -pa(fsh,erk,r_perk,1)]
37 [-icppa(fsh,mek,r_perk,pd98059,-1), -pra(fsh,erk,r_perk,1)]
38 [-icppa(fsh,mek,a_perk,pd98059,-1), -pra(fsh,erk,a_perk,1)]
39 [-icppa(fsh,mek,r_perk,pd98059,-1), -pa(fsh,erk,a_perk,1)]
40 [-icppa(fsh,mek,r_perk,pd98059,-1), -pra(fsh,erk,a_perk,1)]
41 [-pra(pmek,erk,a_perk,1)]
```

Les lignes 1–24 définissent le fichier d'entrée tandis que les lignes 26–41 donnent la sortie de SOLAR pour ce fichier d'entrée.

SOLAR prend en entrée une théorie écrite sous forme de clauses, le mot-clé *cnf* permettant de définir une clause, sous la forme d'un ensemble de littéraux, séparés par des virgules. Les lignes 1–7 donnent les faits du domaine, et les lignes 9–20 donnent l'ensemble des règles considérées dans notre exemple.

La ligne 22 donne le fait (F1) que nous cherchons à expliquer à l'aide de faits formalisant des résultats expérimentaux. Le mot-clé *top_clause* employé dans la définition de ce fait permet de le définir comme point de départ de toutes les résolutions réalisées par SOLAR. Finalement, la ligne 24 définit le champ de production, qui correspond ici à notre ensemble d'abducibles, c'est-à-dire, pour notre exemple, aux prédicats formalisant des expériences de phosphorylation.

Les lignes 30–41 donnent l'ensemble des explications possibles pour le fait (F1), en ne considérant pas les contraintes de typage des arguments des prédicats *pa*, *pra* et *icppa*. En considérant ces contraintes de typage, seules les explications des lignes 31, 32, 34 et 35 sont possibles.

4.5 Application aux réseaux de signalisation induits par le récepteur de la FSH et le récepteur de l'EGF

Nous avons appliqué notre méthode à deux réseaux de signalisation, nommément le réseau induit par le récepteur de l'hormone folliculo-stimulante (R-FSH) et le réseau induit par le récepteur du facteur de croissance épidermique (R-EGF).

L'hormone folliculo stimulante (FSH) est une hormone hypophysaire qui intervient dans le contrôle de la reproduction, aussi bien chez la femme que chez l'homme. Cette hormone agit via son récepteur (le R-FSH), qui fait partie de la famille des récepteurs couplés à la protéine G (R-CPG). Chez la femme, la FSH participe entre autres à l'induction de la croissance et de la maturation des follicules ovariens, tandis que chez l'homme, elle participe à l'induction de la spermatogénèse en stimulant les cellules de Sertoli. Lorsque la FSH se fixe sur son récepteur, ce dernier induit un ensemble de voies de signalisation. La plus connue de ces voies est sans conteste la voie G, qui a pour second messager la protéine G. D'autres voies, moins connues, mais de plus en plus étudiées, sont également induites par ce récepteur, et plus généralement par les R-CPG : nous pouvons par exemple citer la voie Phosphoinositide 3-kinase (PI3K), ainsi que la voie des β-arrestines.

Quant au facteur de croissance épidermique (EGF), il participe à la croissance tissulaire. L'EGF agit également via son récepteur (le R-EGF), qui lui fait partie de la famille des récepteurs tirosine kinase (RTK). Comme leur nom l'indique, ces récepteurs sont également des kinases, et ont ainsi une double activité. La fixation de l'EGF à son récepteur induit également un ensemble de voies de signalisation comme la voie des mitogen-activated protein kinases (MAPK) ou des c-Jun N-terminal kinases (JNK).

Ces deux réseaux ne sont pas indépendants. Le réseau induit par le R-EGF est transactivé par le réseau induit par le R-FSH, par l'intermédiaire de l'oncogène du sarcoma de Rous (SRC). Cette protéine est indirectement activée par le R-FSH, et active le R-EGF en le phosphorylant.

Nous avons d'abord reconstruit automatiquement le réseau de signalisation induit par le R-FSH, en interprétant automatiquement un ensemble de résultats expérimentaux relatifs à ce réseau. Puis nous avons interprété automatiquement ce même ensemble de résultats expérimentaux couplé à un autre ensemble de résultats, cette fois-ci relatifs au réseau induit par le R-EGF. Cette dernière interprétation nous a permis d'émettre, toujours automatiquement, une hypothèse, que nous avons ensuite validée expérimentalement.

4.5.1 Extraction des faits expérimentaux et des faits du domaine

L'ensemble des résultats expérimentaux et des faits du domaine relatifs au réseau induit par le R-FSH ou le R-EGF a été extrait de la littérature par une lecture minutieuse des articles relatifs à ces réseaux. Le plus souvent, l'extraction d'un résultat de la littérature nécessite l'analyse de figures représentant des bandes obtenues par électrophorèse et la lecture de protocoles expérimentaux.

Pour le réseau R-FSH, 240 résultats d'expérience ont été extraits, et ce nombre s'élève à 323 pour le réseau R-EGF. Quant aux faits du domaine, leur nombre s'élève à 542 pour le réseau R-FSH et à 572 pour le réseau R-EGF. Notons que, contrairement aux résultats expérimentaux, les faits du domaine ne sont pas spécifiques au réseau étudié, mais sont des connaissances générales de la biologie des systèmes.

L'extraction de l'ensemble des résultats expérimentaux et des faits du domaine a été réalisée par Pauline Gloaguen durant sa thèse.

4.5.2 Inférence automatique du réseau induit par le récepteur de la FSH

Nous avons interprété automatiquement les 240 résultats expérimentaux relatifs au réseau induit par le R-FSH à l'aide de nos règles d'interprétation et de raisonnement et de l'ensemble des faits du domaine extraits de la littérature. Nous avons obtenu 435 nouveaux faits déduits, parmi lesquels 54 représentent des modulations faisant entrer en jeu des analogues fonctionnels (i.e. des molécules ayant les mêmes fonctions que des molécules du réseau) utilisés pour les expériences, mais ne faisant pas à proprement parler partie du réseau induit par le R-FSH.

L'ensemble de ces faits déduits constitue un réseau que nous avons comparé au réseau induit par le R-FSH de la littérature. Le réseau de la littérature que nous avons considéré comme référence est le réseau présenté dans [Glo+11]. Nous montrons, sans entrer dans le détail, le résultat de cette comparaison pour la voie G.

Le réseau de la littérature représentant la voie G induite par le R-FSH est donné dans la figure 4.2. Il est adapté du réseau introduit dans [Glo+11], qui est représenté sous la forme d'une carte CellDesigner. Nous avons procédé à quelques changements ponctuels afin de le rendre intégralement conforme au standard SBGN-PD. Nous avons notamment remplacé le glyphe CellDesigner signifiant qu'un EPN est actif par une variable d'état. Aussi, les stimulations de ce réseau sont toutes représentées par des catalyses (qui sont elles-mêmes des stimulations). Or certaines des stimulations représentées ne sont en réalité pas des catalyses. Nous les avons donc représentées avec des arcs de stimulation dans la carte de la figure 4.2.

Nous avons déduit 162 faits relatifs à la voie G. À partir de cet ensemble de faits, nous avons construit un nouvel ensemble ne contenant plus que 67 faits, en deux phases. Nous avons d'abord élagué l'ensemble de faits de départ en supprimant les faits jugés non pertinents. Un fait de cet ensemble est jugé non pertinent s'il existe un autre fait de cet ensemble qui est plus précis ; ou si ce fait peut être déduit par transitivité à partir d'autres faits de cet ensemble ; ou encore si ce fait mentionne un équivalent fonctionnel qui ne fait pas partie du réseau. Plus précisément, un fait de l'ensemble de départ est élagué *ssi* :

- il représente une modulation avec un statut *hypothesis*, et il existe dans l'ensemble de faits le même fait avec le statut $confirmed$;
- il représente une modulation avec une distance *unknown*, et il existe dans l'ensemble de faits le même fait avec le statut *direct* ou *indirect* ;
- il représente une modulation qui peut être déduite par transitivité à partir d'autres faits de l'ensemble représentant également des modulations ;
- il représente une modulation par un équivalent fonctionnel qui ne fait pas partie du réseau.

Nous avons ensuite étendu le nouvel ensemble de faits obtenus de la manière suivante. Pour chaque fait mentionnant une molécule phosphorylée sur un site particulier de phosphorylation, nous avons ajouté à l'ensemble de faits le même fait mentionnant la même molécule phosphorylée sans préciser sur quel site. Par exemple, le fait

$$modulates(pmek2, p38mapk, pp38mapkt180\_y182, 1, unknown, confirmed)$$

appartenait à l'ensemble de faits élagué. La constante $pp38mapkt180_y182$ est associée à la molécule p38MAPK phosphorylée sur les sites de phosphorylation T180 et Y182. Nous avons donc ajouté à l'ensemble le fait

$$modulates(pmek2, p38mapk, pp38mapk, 1, unknown, confirmed),$$

où $pp38MAPK$ est associée à phospho-p38MAPK, sans précision du site de phosphorylation. Cette extension de l'ensemble élagué est nécessaire pour la comparaison avec le réseau de référence, ce dernier ne contenant en effet pas le détail des sites de phosphorylation.

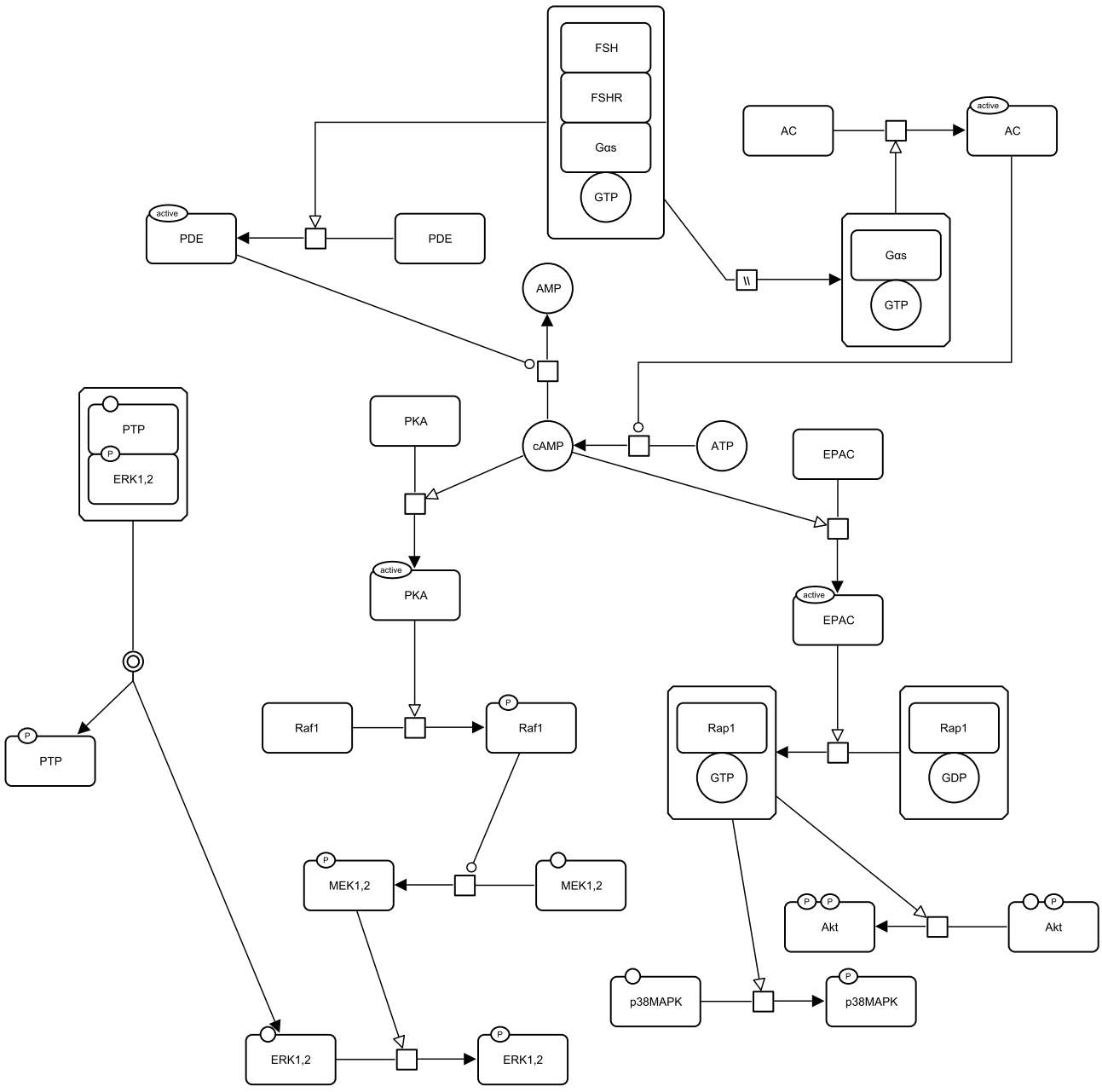


FIGURE 4.2 – **Carte SBGN-PD de la littérature représentant la voie G induite par le R-FSH.**

Ces deux phases de transformation nous ont permis d'obtenir un ensemble de 67 faits, 37 de ces faits étant établis (i.e. ayant un statut *confirmed*), et le reste étant des hypothèses (i.e. ayant un statut *hypothesis*). Nous dénotons l'ensemble de ces faits par F_{DED}.

À des fins de comparaison, nous avons traduit le réseau de référence de la voie G de la figure 4.2 en faits logiques, exprimés avec les mêmes prédicats que ceux avec lesquels nos faits déduits sont exprimés. Nous dénotons par F_{LIT} l'ensemble des faits obtenus.

Sept faits appartiennent à la fois à F_{DED} et à F_{LIT}. Ces faits formalisent la modulation de la phosphorylation d'ERK1 ou ERK2 par phospho-MEK1 ou phospho-MEK2, la catalyse de l'ATP en AMP cyclique (cAMP) par l'adenylate cyclase, la catalyse de cAMP en AMP par PDE, et la modulation de la phosphorylation de p38MAPK par la protéine kinase A (PKA).

Afin d'expliquer la présence de faits de F_{DED} qui ne sont pas dans F_{LIT}, nous distinguons les cas généraux suivants :

- Certains faits de F_{DED} ne sont pas présents en tant que tels dans F_{LIT}, mais sont *implicites* dans le réseau de référence. Par exemple, un fait de F_{LIT} formalise la complexation de la GTP avec la protéine Gαs. Ce processus de complexation n'est pas représenté dans le réseau de référence, et n'est donc par formalisé par un fait de F_{LIT}. Cependant, le complexe GTP-Gαs est présent dans le réseau. Le processus de complexation y est donc présent implicitetement.
- Certains faits de F_{DED} peuvent être déduits par *transitivité* à partir des faits de F_{LIT}. Par exemple, un fait de F_{DED} formalise la stimulation par le complexe Gαs-GTP de la phosphorylation d'ERK1,2. Or, cette stimulation n'est pas présente dans le réseau de la littérature. Cependant, ce complexe agit bien sur la phosphorylation d'ERK1,2 par l'intermédiaire de la cAMP, de la PKA, de Raf1 et de MEK1,2, dans l'ordre. Par conséquent, le fait formalisant cette stimulation pourrait être obtenu par transitivité à partir des faits de F_{LIT}.
- Certains faits de F_{DED} formalisent des relations qui sont *plus précises* que celles présentes dans le réseau. Par exemple, un fait de F_{DED} formalise la stimulation de la complexation de Rap1 et GTP par le complexe cAMP-EPAC. Cette stimulation n'est pas présente telle quelle dans le réseau, où c'est la forme active d'EPAC qui stimule cette complexation. Cependant, nous savons que la forme active d'EPAC est en fait la forme d'EPAC complexée avec la cAMP. Ainsi, dans le réseau de la littérature, le mécanisme précis de l'activation d'EPAC n'est pas montré, alors qu'il est présent dans certains faits de F_{DED}.
- Certains faits de F_{DED} ont le statut d'*hypothèse*, alors qu'ils sont confirmés dans F_{LIT}. C'est par exemple le cas du fait formalisant la stimulation du processus activant la PKA par cAMP.
- Certains faits de F_{DED} proviennent d'expériences qui n'ont tout simplement pas été prises en compte pour la construction du réseau de la littérature, car jugées peu fiables, mais que nous avons tout de même pris en compte pour l'inférence automatique afin de déduire un maximum de faits et d'hypothèses. C'est par exemple le cas de la stimulation de la phosphorylation de p38MAPK par MEK2.

Quant aux faits de F_{LIT} qui ne sont pas dans F_{DED}, ils relèvent des cas suivants :

- Certains faits de F_{LIT} formalisent des relations plus précises que des faits de F_{DED}. Par exemple, un fait de F_{LIT} formalise la stimulation du processus activant la PDE par le complexe FSH-FSHR-Gαs-GTP. Or dans F_{LIT}, un fait formalise cette même stimulation mais par le complexe FSH-FSHR seul.
- Certains faits de F_{LIT} formalisent des modulations qui n'ont pas été prouvées expérimentalement en tant que telles, mais qui sont généralement admises par la communauté. Par exemple, dans le réseau de référence, le complexe Rap1-GTP stimule la phosphorylation de p38MAPK. Or cette

relation n'a pas été établie expérimentalement. Une expérience montre en effet que le complexe cAMP-EPAC (qui constitue la forme active d'EPAC) stimule ce processus ; et il est seulement admis que c'est par l'intermédiaire du complexe Rap1-GTP. Par conséquent, il est normal que ces modulations ne puissent pas être obtenues par déduction, n'étant liées à aucun résultat expérimental pris en compte.

Nous avons donc observé un certain nombre de différences entre le réseau que nous avons déduit et le réseau de référence. La plupart de ces différences peuvent être expliquées par la façon dont les réseaux sont représentés de manière graphique d'une part, et par la présence de certaines connaissances représentées dans le réseau de référence qui n'ont pas de preuve expérimentale d'autre part. Nous revenons plus en détail sur ces deux points dans la discussion.

4.5.3 Une nouvelle hypothèse : la phosphorylation de MEK par p38MAPK

La signalisation cellulaire est généralement étudiée pour un type de cellules particulier (p. ex. des cellules de Sertoli, des cellules HEK) et en réponse à un certain signal (p. ex. la FSH ou l'EGF). Le résultat de ces études est donc obtenu pour un système biologique particulier, et n'est *a priori* pas généralisable à d'autres systèmes.

Cependant, mettre en relation des connaissances obtenues par l'étude de différents systèmes pourrait conduire à la découverte de nouvelles connaissances. Nous avons construit nos règles d'interprétation dans une perspective générale. Ces règles ne tiennent notamment pas compte du type cellulaire. Par conséquent, ces règles devraient permettre d'interpréter un ensemble de résultats expérimentaux obtenus par l'étude de différents systèmes, et de nouvelles connaissances pourraient émerger de cette interprétation.

Afin d'illustrer cette possibilité, nous avons réuni les ensembles de résultats expérimentaux et de faits du domaine relatifs au réseau induit par le R-FSH et ceux relatifs au réseau induit par le R-EGF, et interprété cet ensemble avec nos règles.

Aucune des règles d'interprétation n'a permis de déduire de fait qui n'aurait pas pu être déduit avec les résultats relatifs au réseau induit par le R-FSH seuls ou relatifs au réseau induit par le R-EGF seul. Ce résultat était attendu : l'interprétation d'un résultat expérimental relatif exclusivement au réseau R-FSH ne nécessite pas de connaissance sur le réseau EGF, et vice-versa. Par contre, les règles de raisonnement ont permis de déduire un ensemble de nouvelles hypothèses. Les hypothèses jugées intéressantes ont été déduites par la règle (R14), qui permet de raisonner sur la transitivité des modulations. Un exemple d'une telle hypothèse est le fait suivant :

$$modulates(pp38mapk, mek, pmek, 1, unknwown, hypothesis) \tag{F13}$$

Ce fait signifie qu'en se basant sur les données expérimentales que nous détenons, nous pouvons émettre l'hypothèse que phospho-p38MAPK stimule la phosphorylation de MEK. Cette hypothèse a été obtenue en appliquant la règle (R14) sur les deux faits suivants :

$$modulates(pmek, erk, perk, 1, direct, confirmed) \tag{F14}$$

et

$$modulates(pp38mapk, erk, perk, 1, confirmed, unknown) \tag{F15}$$

Le fait (F14) a été obtenu en interprétant une expérience de test enzymatique relative au réseau R-EGF, et le fait (F15) en interprétant une expérience de phosphorylation complexe relative au réseau R-FSH. Le fait (F14) peut être déduit par l'interprétation d'un expérience relative au réseau R-FSH, mais avec une distance inconnue (i.e. pour un paramètre de distance valant *unknown*), et ne peut donc pas être utilisé pour vérifier une prémisse de la règle (R14).

Cette nouvelle hypothèse est intéressante pour deux raisons. D'abord, c'est la stimulation inverse (i.e. phospho-MEK stimule la phosphorylation de p38MAPK) qui a été établie pour des cellules stimulées par la FSH. Cette stimulation n'a pas été considérée par les auteurs de [Glo+11] qui ont construit le réseau R-FSH que nous avons pris comme référence. En effet, le résultat expérimental qui a permis de déduire cette modulation n'est, pour eux, pas assez fiable. Il n'a notamment, à notre connaissance, pas été reproduit par une autre équipe.

Deuxièmement, la validation de cette hypothèse permettrait en même temps de valider un certain nombre d'autres hypothèses aussi déduites par la règle (R14), et qui sont relatives à des modulations de la phosphorylation de MEK par des molécules se trouvant en amont de p38MAPK dans la voie G. Ce dernier point donne encore plus de poids à l'hypothèse formalisée par le fait (F13), et illustre un avantage certain de l'interprétation automatique de résultats expérimentaux : après un changement ponctuel de la théorie (comprise comme un ensemble de connaissances à un moment donné), toutes les nouvelles connaissances qui résultent de ce changement peuvent facilement être déduites, même si ces nouvelles connaissances sont éloignées de ce changement dans la théorie. Ainsi, l'interprétation automatique permet la mise en relation de faits qui semblent n'avoir aucun rapport *a priori*.

4.5.4 Plans expérimentaux pour établir l'hypothèse

Nous avons inféré par abduction l'ensemble des plans expérimentaux permettant de tester l'hypothèse selon laquelle phospho-p38MAPK stimulerait la phosphorylation de MEK, hypothèse formalisée par le fait (F13).

Pour ce faire, nous avons abduit l'ensemble des faits formalisant des résultats expérimentaux suffisants pour expliquer le fait suivant, obtenu du fait (F13) en remplaçant le statut *hypothesis* par le statut $confirmed$:

$$modulates(pp38mapk, mek, pmek, 1, unknwown, confirmed) \tag{F16}$$

Nous avons considéré la théorie formée de l'ensemble de nos règles d'interprétation et de raisonnement, de l'ensemble des faits du domaine relatifs aux réseaux R-FSH et R-EGF, ainsi que de l'ensemble des faits déduits pour ces deux réseaux. Finalement, nous avons construit un ensemble d'abductibles contenant tous les prédicats formalisant les différents types d'expérience considérés.

Six plans expérimentaux ont pu être inférés, chacun d'entre eux n'étant constitué que d'une seule expérience à réaliser. Deux de ces plans contenaient une $ICPEA$ (test enzymatique en présence d'un inhibiteur), un une $ICRIA$ (expérience de radio-immunologie en présence d'un inhibiteur), un une $ICELISA$ (ELISA en présence d'un inhibiteur), un une PA (expérience de phosphorylation simple) et un une $ICPPA$ (expérience de phosphorylation en présence d'un inhibiteur). Les plans d'expérience formés de l'$ICRIA$, de l'$ICELISA$ er de l'$ICPPA$ proposaient tous d'étudier l'effet de l'inhibiteur de p38MAPK (SB203580) sur la stimulation de la phosphorylation de MEK par la FSH. Les deux plans formés d'une $ICPEA$ proposaient quant à eux d'étudier l'effet de ce même inhibiteur sur l'activité de MEK, à travers l'étude *in vitro* de la réaction de phosphorylation d'ERK. Finalement, le plan formé d'une PA proposait d'étudier directement l'effet de phospho-p38MAPK sur la phosphorylation de MEK, en utilisant comme signal phospho-p38MAPK.

Afin de confirmer notre hypothèse, nous avons choisi de réaliser, en laboratoire, le plan d'expérience formé d'une $ICPPA$. Elle montre qu'en effet, l'inhibition de p38MAPK induit une diminution de la phosphorylation de MEK. À des fins de contrôle, nous avons également montré expérimentalement que cette inhibition conduit à une diminution de la phosphorylation d'ERK (réaction qui est catalysée par phospho-MEK). Ces expériences ont été réalisées par Nathalie Langonné (équipe BIOSS, INRA-Centre Val de Loire).

4.6 Discussion

4.6.1 Travaux connexes

Dans [Zup+03], les auteurs présentent une méthode automatique d'inférence de réseaux de régulation génétique. Leur méthode permet d'interpréter automatiquement des expériences de mutants, à l'aide de règles expertes. Ces règles permettent d'inférer des relations causales (telles que des stimulations ou des inhibitions) entre des gènes impliqués dans les expériences interprétées. Comme les règles ne sont définies que pour l'interprétation d'expériences génétiques, les relations inférées restent causales, et ne sont pas à l'échelle du processus.

Les travaux se rapprochant le plus des nôtres ont été publiés récemment dans [Nig+15]. Dans cette étude, les auteurs infèrent des modèles à partir de résultats expérimentaux formalisés appelés *datums*. Ces datums formalisent des résultats provenant de divers types d'expérience classiques de la biologie des systèmes, comme des expériences de phosphorylation. Comme dans notre méthode, des règles explicites, formalisées en logique, permettent de déduire des relations causales entre les molécules qui entre en jeu dans les divers datums interprétés. Ces relations causales sont ensuites rassemblées pour former des modèles qui peuvent être exécutés par le logiciel Pathway Logic [Eke+02]. Les modèles exécutables obtenus ressemblent à des réseaux de réactions, étant donné qu'ils permettent de modéliser différents types de processus moléculaires et de modulations. Cependant, la connaissance biologique sous-jacente aux modèles reste implicite, celle-ci n'étant pas nécessaire à l'exécution des modèles. Par conséquent, la méthode d'inférence proposée dans [Nig+15] ne vise pas à construire des réseaux moléculaires qui pourraient ensuite être modélisés en utilisant différentes sémantiques (p. ex. des équations différentielles), mais plutôt à construire des modèles qui sont déjà une interprétation des processus moléculaires sous-jacents. Une autre limitation de leur méthode est le manque de règles d'interprétation pour les expériences faisant entrer en jeu des perturbateurs (comme les inhibiteurs), qui sont largement utilisées en biologie de la signalisation.

4.6.2 Provenance et qualité des faits inférés.

Avec notre méthode, il est possible de garder une trace du raisonnement qui a permi de déduire un fait donné, c'est-à-dire de lier explicitement un fait déduit au résultat expérimental dont il provient. En connaissant la fiabilité de ce résultat (par exemple, s'il a été confirmé par plusieurs équipes où non), il est alors possible de connaître la qualité de ce fait déduit. Aussi, des résultats expérimentaux différents peuvent être interprétés par un même fait déduit, ce qui augmentera sa qualité. La qualité d'un fait dépendra donc à la fois de la fiabilité des résultats expérimentaux qui ont permi de l'obtenir, mais aussi de leur nombre.

4.6.3 Utilisation du langage SBGNLog-PD pour les règles d'interprétation

Nous n'avons pas utilisé le langage SBGNLog-PD pour formaliser les faits qui peuvent être déduits en interprétant les résultats expérimentaux que nous considérons, mais un ensemble de prédicats différents propres à l'interprétation de résultats expérimentaux. Par exemple, en SBGNLog-PD, le prédicat

modulates prend deux arguments, le premier étant une constante associée à l'EPN qui module, et le deuxième une constante associée au processus modulé. Dans cette étude, nous avons utilisé un prédicat *modulates* différent. Le premier argument de ce prédicat est associé, comme en SBGNLog-PD, à la molécule qui module, mais les deux suivants sont associés à un des réactifs et à un des produits du processus modulé.

En effet, dans la biologie de signalisation, une partie des réactifs et des produits des processus moléculaires est souvent omise, et seules les molécules principales de ces processus sont considérées. Par exemple, dans la représentation des processus de phosphorylation, la consommation d'ATP et la production d'ADP restent souvent *implicites*.

Ainsi, la description de ces processus se limite le plus souvent à spécifier la molécule qui va être phosphorylée et sa version phosphorylée, et le raisonnement utilisé pour l'interprétation de résultats faisant entrer en jeu ce type de processus se focalise également sur ces deux seules molécules. C'est pourquoi, dans la représentation que nous avons choisie, les processus ne sont pas représentés comme en SBGNLog-PD, mais plutôt par un couple molécule à phosphoryler/molécule phosphorylée. Cependant, lorsque le raisonnement nécessite de se focaliser sur un ensemble de réactifs et de produits, comme par exemple pour interpréter des tests enzymatiques, nous avons introduit une notion d'ensemble qui est proche de la représentation SBGNLog-PD. Par exemple, le prédicat $catalyze(X, S1, S2, S)$ formalise la catalyse par X du processus qui transforme les réactifs de l'ensemble $S1$ en produits, donnés par l'ensemble $S2$, et avec un statut S.

Notons cependant que si nous n'avons pas utilisé le langage SBGNLog-PD pour écrire nos règles d'inférences, cela reste possible. Les règles obtenues comporteraient néanmoins plus de prédicats dans leurs prémisses, et seraient par conséquent sans doute moins facilement compréhensibles.

4.6.4 Des faits déduits aux cartes SBGN-PD

L'interprétation automatique de résultats expérimentaux permet d'obtenir un ensemble de faits déduits qui formalisent de nouvelles connaissances. Ces faits représentent soit des processus de complexation, soit des modulations de processus moléculaires déjà connus. Ces complexations, modulations, et processus constituent un réseau moléculaire, qui peut être représenté en SBGN-PD.

Nous avons commencé le développement d'une méthode de transformation automatique des faits déduits en réseau SBGN-PD. Étant donné un ensemble de faits déduits avec la méthode présentée dans ce chapitre, leur représentation en SBGN-PD se fait en trois temps. D'abord, l'ensemble de ces faits est traduit en SBGNLog-PD, pour obtenir un nouvel ensemble de faits qui constitue une carte SBGNLog-PD. Puis cette carte est élaguée, c'est-à-dire que certains faits représentant des modulations qui ne sont pas utiles pour la représentation graphique sont éliminés. C'est par exemple le cas des modulations qui peuvent être déduites par des lois de transitivité. Enfin, l'ensemble de ces faits est traduit en SBGN-PD via le format SBGN-ML, en laissant la position de chaque glyphe indéterminée. L'ensemble des faits est donc traduit sous la forme d'un fichier SBGN-ML, qui peut ensuite être automatiquement mis en forme par des logiciels de mise en forme disponibles publiquement.

Notons que lors de la traduction de nos faits déduits en SBGNLog-PD, nous perdons certains détails de la représentation que nous avons choisie pour l'interprétation automatique. Par exemple, le paramètre de distance des modulations ne peut pas être représenté en SBGNLog-PD, étant donné qu'il ne peut pas être représenté en SBGN-PD.

4.6.5 Faits inférés et représentations graphiques des réseaux

La comparaison des faits que nous avons inférés en interprétant automatiquement un ensemble de résultats expérimentaux relatifs au réseau R-FSH avec les faits traduits à partir du réseau de référence

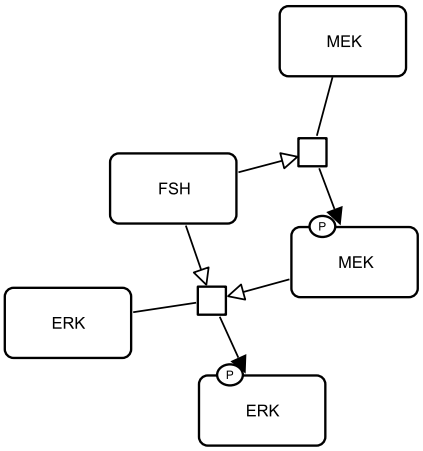


FIGURE 4.3 – **Exemple de réseau SBGN-PD construit automatiquement à partir de faits déduits.** Réseau construit automatiquement à partir des faits déduits à l'exemple de la sous-section 4.4.1. Le fichier SBGN-ML automatiquement créé à partir des faits déduits a été mis en forme avec automatiquement à l'aide d'SBGNViz [Sar+15].

de la voie G, représenté sous forme de carte SBGN-PD et tiré de la littérature, illustre deux points concernant l'interprétation des résultats expérimentaux et la représentation de ces interprétations.

D'abord, les relations des réseaux représentés sous forme graphique sont souvent moins précises que les relations qui peuvent être conclues par l'expérience. Cela est sans doute dû au désir de clarté allant de pair avec la représentation graphique d'un réseau. L'objectif d'une représentation graphique d'un réseau moléculaire, et même d'un réseau de réactions, n'est pas de détailler à tout prix l'ensemble des mécanismes connus, mais plutôt de donner une représentation simple et compréhensible de ce réseau. Par exemple, le réseau de référence que nous avons montré ne donne pas les détails des sites de phosphorylation, qui pourraient gêner la lecture du réseau.

Ensuite, les réseaux de la littérature, qui sont le plus souvent représentés sous forme graphique, comprennent un certain nombre de relations qui n'ont pas été directement montrées par l'expérience, mais qui sont supposées. C'était par exemple le cas de l'action de Rap1 sur p38MAPK, qui n'a pas été prouvée pour le réseau R-FSH.

Par conséquent, l'ensemble de nos faits inférés et les réseaux représentés de manière graphique ne représentent pas les connaissances relatives à un réseau de signalisation exactement de la même manière. Nos faits sont tous déduits à partir de résultats expérimentaux auxquels ils peuvent être rattachés et décrivent des processus et des modulations le plus précisément possible. Ils représentent, d'un certain point de vue, ces connaissances "brutes". À l'inverse, les réseaux représentés de manière graphique dans la littérature sont une synthèse de connaissances certes le plus souvent inférées à partir de résultats expérimentaux, mais enrichie de connaissances ne pouvant pas directement être liées à des résultats expérimentaux. Ils sont donc davantage une image travaillée et plus synthétique d'un ensemble de connaissances scientifiques relatives à un objet d'études, comme une voie de signalisation.

4.6.6 Interprétation automatique des expériences haut-débit

Avec l'évolution récente de la technologie de spectrométrie de masse, des expériences haut-débit visant à étudier le protéome ont vu le jour. Notamment, il est maintenant possible de mesurer les quantités de centaines de protéines, et en particulier de phospho-protéines, en une seule étude. Quand, dans une $ICPPA$ par exemple, une seule phospho-protéine d'intérêt est étudiée, il est mainenant possible

de répéter cette même expérience, avec le même signal et le même inhibiteur, mais en considérant un grand nombre de phospho-protéines d'intérêt en parallèle. Par conséquent, le résultat provenant d'une telle expérience haut-débit peut être formalisé par un ensemble de faits instanciant le prédicat *icppa*, et peut être interprété automatiquement grâce à notre méthode.

4.7 Conclusion et perspectives

Nous avons proposé une méthode automatisant la construction des réseaux de signalisation. Notre méthode est basée sur l'interprétation automatique de résultats expérimentaux à l'aide de règles déductives, formalisée en logique du premier ordre. Nous avons montré comment nos règles permettent d'inférer de nouvelles connaissances ou d'émettre de nouvelles hypothèses, liées à des mécanismes moléculaires précis, à partir de données expérimentales provenant d'une grande variété de types d'expériences. Nous avons également montré que le raisonnement déductif peut être inversé afin de proposer des plans expérimentaux permettant de tester une hypothèse donnée. Nous avons illustré notre méthode en reconstruisant le réseau induit par le récepteur de la FSH, et comparé ce réseau à un réseau de référence extrait de la littérature. Notre méthode nous a également permis d'émettre une nouvelle hypothèse sur la phosphorylation de MEK par p38MAPK, que nous avons ensuite validée expérimentalement.

Nous avons vu que pour certains résultats expérimentaux, plusieurs interprétations alternatives sont possibles. Notre méthode permet de prendre en compte ces différentes alternatives sous forme de faits hypothétiques, là où les biologistes considèrent le plus souvent uniquement l'alternative la plus plausible selon eux, pour des raisons de simplicité. Aussi, l'utilisation de règles explicites permet de lier les connaissances ou hypothèses déduites aux résultats expérimentaux dont elles proviennent. Les réseaux construits avec notre méthode sont ainsi plus proches des expériences que les réseaux pouvant être trouvés dans la littérature, qui représentent parfois des connaissances qui n'ont pas été prouvées expérimentalement, et qui sont chargées de savoir implicite.

D'un point de vue théorique, il nous reste à justifier l'ensemble de nos règles d'interprétation, c'est-à-dire à les ancrer dans un cadre théorique de plus bas niveau. Nous avons initié ce travail, qui fait suite au regroupement des types d'expériences en clusters et à la généralisation des règles à l'aide de ces clusters. D'une part, nous avons construit deux méta-règles d'interprétation à partir desquelles l'ensemble de nos règles interprétatives peuvent être dérivées. La première de ces règles formalise l'interprétation de résultats expérimentaux provenant d'expériences simples (i.e. sans perturbateur), et la deuxième des résultats provenant d'expériences complexes. D'autre part, nous avons entamé une réflexion très générale sur le raisonnement entrepris lors de l'interprétation des résultats expérimentaux de la biologie des systèmes. Cette réflexion nous a permis de mettre en lumière un certain nombre d'hypothèses et d'axiomes qui sont à la base de cette interprétation. Il nous reste maintenant à dériver les deux meta-règles à partir de ces hypothèses et axiomes.

En parallèle de la méthode de construction des réseaux que nous avons introduite dans ce chapitre, nous avons proposé une méthode de complétion des graphes d'influences SBGN-AF (voir [Yam+14] et [Rou+15]). Nous n'avons par présenté cette méthode dans ce manuscrit car elle ne nous semble pas encore assez aboutie. Étant donné une carte SBGN-AF et un ensemble de résultats expérimentaux du même type que ceux présentés dans ce chapitre, cette méthode permet d'inférer un ensemble de nouvelles modulations qui, conjointement à la carte SBGN-AF, permettent d'expliquer l'ensemble des résultats expérimentaux considérés. Ces nouvelles modulations sont inférées par un raisonnement abductif sur une théorie logique formée en partie de la traduction de la carte SBGN-AF en SBGNLog-AF et d'un certain nombre d'axiomes qui formalisent des relations de transitivité entre les modulations d'un graphe d'influences. La déduction de nouvelles hypothèses par la règle de raisonnement (R14) introduite dans ce chapitre, également basée sur la transitivité des modulations, peut être vue comme

un cas particulier du raisonnement abductif employé dans cette méthode de complétion. Cette méthode de complétion pourrait sans aucun doute être adaptée aux réseaux de réactions, et être intégrée à la fois à notre méthode de construction des réseaux, et au cadre théorique formel que nous avons mentionné plus haut. L'ajout de ce volet abductif permettrait d'émettre de nouvelles hypothèses encore plus pertinentes que celles pouvant être inférées à partir des règles de raisonnement comme (R14).

Type la conclusion	Méthode	Type d'expérience	Perturbateur	Méthode de détection	Nom du type d'expérience	Prédicat
Modulation d'un processus	Mesure d'une activité	Test enzymatique			EA	$EA(X, S1, S2, E)$
			Antagoniste		ACPEA	$ACPEA(X, S1, S2, I, E)$
			Inhibiteur		ICPEA	$ICPEA(X, S1, S2, I, E)$
			siRNA		SCPEA	$SCPEA(X, S1, S2, I, E)$
		Expérience de phosphorylation		Anticorps	PA	$PA(X, Y, A, E)$
				Radio	PRA	$PRA(X, Y, A, E)$
			Antagoniste	Anticorps	ACPPA	$ACPPA(X, Y, A, I, E)$
			Inhibiteur		ICPPA	$ICPPA(X, Y, A, I, E)$
				Radio	ICPPRA	$ICPPRA(X, Y, A, I, E)$
			siRNA	Anticorps	SCPPA	$SCPPA(X, Y, A, I, E)$
	Dosage	ELISA		Anticorps	ELISA	$ELISA(X, A, E)$
			Inhibiteur		ICELISA	$ICELISA(X, A, I, E)$
			siRNA		SCELISA	$SCELISA(X, A, I, E)$
		Expérience de radio-immunologie		Radio	RIA	$RIA(X, A, E)$
			Inhibiteur		ICRIA	$ICRIA(X, A, E)$
		Western Blot		Pan-anticorps	PANWB	$PANWB(X, A, E)$
			Inhibiteur		ICPANWB	$ICPANWB(X, A, E)$
					RIPANWB	$RIPANWB(X, A, E)$
		RTPCR			RTPCR	$RTPCR(X, Y, E)$
			Inhibiteur		IRTPCR	$IRTPCR(X, Y, I, E)$
Structure	Précipitation	IP			IP	$IP(X, S1)$
	Pulldown	GSTPulldown			GSTPULLDOWN	$GSTPULLDOWN(X, S1)$
		BDPulldown			BDTPULLDOWN	$BDTPULLDOWN(X, S1)$
	FRET	FRET			FRET	$FRET(X, Y)$
	Cristallographie	Cristallographie			3D	$3D(S1)$
Localization	Fluorescence Coloration				FLUO	$FLUO(A, C, E)$
					IHC	$IHC(A, C, E)$

TABLE 4.1 – **Différents types d'expérience.** Les différents types d'expérience permettent de conclure sur la modulation d'un processus, sur une complexation ou sur la localisation d'une molécule. Pour chaque type d'expérience, lorsque c'est pertinent, le type de détecteur et de perturbateur est indiqué. Les différents clusters sont indiqués par les couleurs : à chaque cluster correspond une couleur. Les résultats d'expérience provenant de types d'expérience appartenant à un même cluster sont interprétés par les mêmes règles d'interprétation.

Troisième partie

Dynamique qualitative des réseaux moléculaires

Chapitre 5

Modélisation de la dynamique des réseaux SBGN-AF à l'aide de programmes logiques normaux du premier ordre

Sommaire

5.1 Introduction

Plusieurs sémantiques peuvent être utilisées afin de modéliser la dynamique des graphes d'influences. Les sémantiques quantitatives (comme celle des ODE) nécessitent des paramètres numériques qui doivent préalablement être mesurés ou calculés, et qui sont par conséquent difficiles à obtenir. À l'inverse, les sémantiques qualitatives ne nécessitent pas la connaissance de ces paramètres. Ces sémantiques, même si elles décrivent moins précisément la dynamique des systèmes que les sémantiques quantitatives, permettent néanmoins d'en capturer les propriétés les plus importantes, comme leurs états stables. Elles ont donc l'avantage d'être plus directement applicables aux graphes d'influences que les sémantiques quantitatives.

Une des sémantiques les plus courantes pour modéliser des graphes d'influences est la sémantique Booléenne. Avec un telle sémantique, les molécules ou activités d'un graphe d'influences peuvent être dans deux états : présent ou absent. Cette discrétisation est justifiée par la forme sigmoïdale de la courbe représentant la relation entre la concentration d'un modulateur et son effet (comme par exemple la concentration d'une molécule modulée) [TTK95]. En-deçà d'un certain seuil de concentration, un

modulateur n'a pas ou peu d'effet ; et au-delà de ce seuil de concentration, ce modulateur a un effet, qui augmente peu lorsque sa concentration augmente. Ainsi, l'effet d'un modulateur est quasi nul en-dessous du seuil, et quasi maximal au-dessus de ce seuil. Par conséquent, l'espace des concentrations peut être discrétisé par deux valeurs : en-dessous du seuil, le modulateur n'est pas présent et n'a donc pas d'effet, et au-dessus de ce seuil, le modulateur est présent et a un effet.

Le formalisme privilégié pour modéliser de tels réseaux à l'aide d'une sémantique Booléenne est le réseau Booléen (RB). Ces réseaux, depuis leur introduction par Stuart Kaufman à la fin des années 60, ont été largement étudiés et utilisés pour la modélisation des graphes d'influences.

Informellement, un RB est un ensemble de variables Booléennes associées à des fonctions Booléennes. Une fonction d'un RB modélisant un graphe d'influences décrit le comportement dynamique d'une des molécules ou activités du système représenté par le graphe d'influences. La notion de temps considérée ici est une notion de temps discret, et est modélisée par des *pas de temps*. Différents modes de mise à jour ont été considérés pour la sémantique Booléenne. Parmi ceux-ci, nous distinguons le mode de mise à jour *synchrone* et le mode de mise à jour *asynchrone*. Dans le mode de mise à jour synchrone, toutes les molécules ou activités du graphe évoluent lors d'un pas de temps, et dans le mode de mise à jour asynchrone, une seule de ces molécules ou activités évolue lors d'un pas de temps. Ces deux modes de mise à jour conduisent respectivement à distinguer deux types de dynamiques : la dynamique synchrone et la dynamique asynchrone. Historiquement, c'est le mode de mise à jour synchrone, plus simple, qui a été étudié en premier lieu, par Stuart Kauffman. Le mode de mise à jour asynchrone a lui été introduit peu après par René Thomas, et est maintenant considéré comme plus réaliste (voir [HB97] pour plus de détails), car il induit une dynamique non-déterministe. Les traces de l'une ou l'autre des dynamiques d'un RB correspondent donc à l'évolution qualitative au cours du temps des quantités des molécules ou activités du système. Une propriété importante des RBs est la présence ou non d'attracteurs : les attracteurs cycliques d'un RB correspondent à des oscillations du système, et ses points attracteurs aux états stables du système. Notons que les points attracteurs des dynamiques synchrone et asynchrone d'un RB sont les mêmes, et qu'il est ainsi plus simple de les étudier à partir de la dynamique synchrone.

La modélisation d'un graphe d'influences par un réseau Booléen ne nécessite pas de paramètres cinétiques, mais plutôt des paramètres qualitatifs qui représentent par exemple des effets coopératifs entre molécules ou activités du graphe d'influences. Ces paramètres peuvent être obtenus de différentes manières (que nous décrirons plus en détail dans la suite). Notamment, ils peuvent être déduits de principes généraux décrivant de manière informelle le comportement dynamique des molécules ou activités d'un graphe d'influences en fonction de leurs modulations. Par exemple, nous pouvons supposer que l'inhibition est toujours plus forte que l'activation, c'est-à-dire que si une molécule ou activité est inhibée par une molécule qui est présente dans le système à un temps t, alors elle sera absente du système au temps $t+1$, quel que soit l'état au temps t des molécules qui la stimulent. En considérant un ensemble de principes généraux suffisamment large, il est ainsi possible de paramétrer l'ensemble des fonctions Booléennes d'un RB modélisant un graphe d'influences.

Récemment, Katsumi Inoue a montré dans [Ino11] la correspondance entre les RB et les programmes logiques normaux (NLP) propositionnels. Il a notamment donné une traduction des RB en NLP propositionnels tels que les traces et les points attracteurs de la dynamique synchrone des RB correspondent aux orbites et modèles supportés des NLP, respectivement.

En partant de ce résultat, nous proposons un ensemble de principes généraux décrivant la dynamique Booléenne des activités d'un graphe d'influences exprimé sous forme de carte SBGN-AF, que nous formalisons en un ensemble de règles logiques. Étant donnée une carte SBGN-AF, nous montrons que cet ensemble de règles permet de calculer les traces synchrones et points attracteurs du réseau Booléen qui aurait été obtenu en modélisant cette carte avec nos principes généraux.

Le reste du chapitre est organisé comme suit. Nous donnons d'abord quelques définitions relatives

aux RB. Puis nous donnons plus de détails sur la paramétrisation des RB, et montrons qu'ils peuvent notamment être paramétrés à l'aide d'un ensemble de principes généraux. Nous montrons ensuite comment la dynamique des RB obtenus à partir de ces principes généraux peut être calculée à partir de NLPs du premier ordre formalisant ces principes. Finalement, nous discutons de la différence entre nos travaux et des travaux connexes, ainsi que du calcul des traces asynchrones avec nos programmes logiques.

Ce travail a été réalisé en collaboration avec Yoshitaka Yamamoto (université de Yamanashi) et Katsumi Inoue (NII), et a été publié dans [Rou+14].

5.2 Réseaux Booléens : définitions

Un *réseau Booléen* (RB) est défini comme un ensemble de variables Booléennes, et un ensemble de fonctions Booléennes associées à ces variables. Formellement :

Définition 5.1 (Réseau Booléen)**.** Un *réseau Booléen* est un couple (V, F), où $V = \{v_1, \ldots, v_n\}$ est un ensemble de variables Booléennes et $F = \{f_1, \ldots, f_n\}$ un ensemble de fonctions Booléennes sur les variables de V, telle que chaque f_i est associée à v_i $(1 \leq i \leq n)$.

Exemple 5.1 (Réseau Booléen)**.** $B = (V, F)$, où $V = \{a_1, a_2, a_3, a_4\}$ et $F = \{f_1, f_2, f_3, f_4\}$, avec $f_1(a_1, a_2, a_3, a_4) = \neg a_4$, $f_2(a_1, a_2, a_3, a_4) = a_2$, $f_3(a_1, a_2, a_3, a_4) = a_3$ et $f_4(a_1, a_2, a_3, a_4) = (a_1 \vee a_2) \wedge a_3$, est un réseau Booléen.

Soit un RB $B = (V, F)$, où $V = \{v_1, \ldots, v_n\}$, et $F = \{f_1, \ldots, f_n\}$. Un *état global* de ce RB est un n-uplet de valeurs Booléennes de ses variables. L'ensemble des états globaux du RB, dénoté par S, est l'ensemble $\{(v_1, \ldots, v_n) \mid v_1 \in \mathbb{B}, \ldots, v_n \in \mathbb{B}\}$. Pour un état global $s \in S$ du RB, nous dénotons par $v_i(s)$ la valeur Booléenne de v_i dans s.

Nous considérons deux mises à jour pour les RB : la mise à jour synchrone et la mise à jour asynchrone.

Avec la mise à jour synchrone, un RB passe d'un de ses états globaux $s \in S$ à un autre de ses états globaux $s' \in S$ en appliquant toutes les fonctions logiques aux valeurs de ses variables dans s simultanément, tandis qu'avec la mise à jour asynchrone, une seule fonction logique est appliquée aux valeurs de ses variables dans s à la fois, pour passer de s à s'.

Les deux définitions formelles des relations de transition synchrone et asynchrone sont les suivantes :

Définition 5.2 (Relation de transition synchrone)**.** Étant donné un RB $B = (V, F)$ où $V = \{v_1, \ldots, v_n\}$, et $F = \{f_1, \ldots, f_n\}$, la relation de tansition synchrone, notée $\rightarrow_{sy}$, est incluse dan $S \times S$ et définie de la manière suivante pour $(s, s') \in S^2$:

$$s \rightarrow_{sy} s' \triangleq \forall v_i \in V, v_i(s') = f_i(v_1(s), \ldots, v_n(s))$$

Définition 5.3 (Relation de transition asynchrone)**.** Étant donné un RB $B = (V, F)$ où $V = \{v_1, \ldots, v_n\}$, et $F = \{f_1, \ldots, f_n\}$, la relation de transition asynchrone, notée $\rightarrow_{asy}$, est incluse dans $S \times S$ et définie de la manière suivante pour $(s, s') \in S^2$:

$$s \rightarrow_{asy} s' \triangleq \exists v_i \in V : v_i(s') = f_i(v_1(s), \ldots, v_n(s)) \text{ et } \forall v_j \in V, v_j \neq v_i \Rightarrow v_j(s') = v_j(s)$$

La notion de trace et de graphe de transitions peuvent être définies à partir des relations de transitions :

Définition 5.4 (Trace d'un RB)**.** Une trace de la dynamique synchrone (resp. asynchrone) d'un RB est une série, finie ou infinie, de transitions synchrones (resp. asynchrones) de ce RB. Une telle trace est souvent dénotée par une séquence $s_1 \rightarrow_{sy} s_2 \rightarrow_{sy} \dots$ (resp. $s_1 \rightarrow_{asy} s_2 \rightarrow_{asy} \dots$) de transitions synchrones (resp. asynchrones), telle que les s_i ($i \geq 1$) sont des états globaux du RB.

Définition 5.5 (Graphe de transitions d'un RB)**.** Soit B un RB, et S l'ensemble de ses états globaux. Le graphe de transitions synchrones (resp. asynchrones) de B est un graphe dirigé dont les noeuds sont les états de S, et qui contient un arc partant d'un état global $s \in S$ et arrivant sur un état global $s' \in S$ *ssi* $s \rightarrow_{sy} s'$ (resp. $s \rightarrow_{asy} s'$).

L'ensemble des chemins (avec répétition de sommets et d'arcs) du graphe de transitions synchrone (resp. asynchrone) d'un RB est exactement l'ensemble des traces de la dynamique synchrone (resp. asynchrone) de ce RB.

Exemple 5.2 (Graphe de transitions)**.** La figure 5.1 montre le graphe de transitions de la dynamique asynchrone du RB de l'exemple 5.1, construit pour un état global initial donné (représenté en bleu sur la figure).

Définition 5.6 (Atteignabilité d'un état)**.** Soient deux états globaux s et s' d'un RB. L'état s' est *atteignable* à partir de l'état s dans la dynamique synchrone (resp. asynchrone), noté $s \rightarrow^*_{sy} s'$ (resp. $s \rightarrow^*_{asy} s'$) *ssi* $s \rightarrow_{sy} s'$ (resp. $s \rightarrow^*_{asy} s'$) ou il existe un état $s'' \in S$ tel que $s \rightarrow^*_{sy} s''$ (resp. $s \rightarrow^*_{asy} s''$) et $s'' \rightarrow_{sy} s'$ (resp. $s'' \rightarrow_{asy} s'$).

Notons que, étant donnés deux états globaux s et s' d'un RB, si $s \rightarrow^*_{sy} s'$ (resp. $s \rightarrow^*_{asy} s'$), alors soit $s = s'$, soit il existe une trace de la dynamique synchrone (resp. asynchrone) partant de s et arrivant sur s'.

Pour un état s d'un RB, nous dénotons par $X_{\rightarrow_{sy}}(s) = \{s' \mid s \rightarrow^*_{sy} s'\}$ l'ensemble des états atteignables à partir de s dans la dynamique synchrone, et par $X_{\rightarrow_{asy}}(s) = \{s' \mid s \rightarrow^*_{asy} s'\}$ l'ensemble des états atteignables à partir de s dans la dynamique asynchrone. Notons que comme la relation de transition synchrone est déterministe, pour tout état global s du RB, $|X_{\rightarrow_{sy}}(s)| = 1$.

Définition 5.7 (Attracteurs d'un RB)**.** Étant donné un RB, un ensemble A d'états globaux de ce RB est un *attracteur* pour la dynamique synchrone (resp. asynchrone) *ssi* tous les états atteignables à partir des états de A dans la dynamique synchrone (resp. asynchrone) appartiennent à A et il n'existe pas d'autre ensemble d'états strictement inclus dans A qui respecte cette propriété, i.e. :

$$\bigcup_{s \in A} X_{\rightarrow_{sy}}(s) = A \text{ (resp. } \bigcup_{s \in A} X_{\rightarrow_{asy}}(s) = A)$$

S'il n'y a qu'un seul état dans A, alors A est un *point attracteur*, et c'est un *attracteur cyclique* sinon.

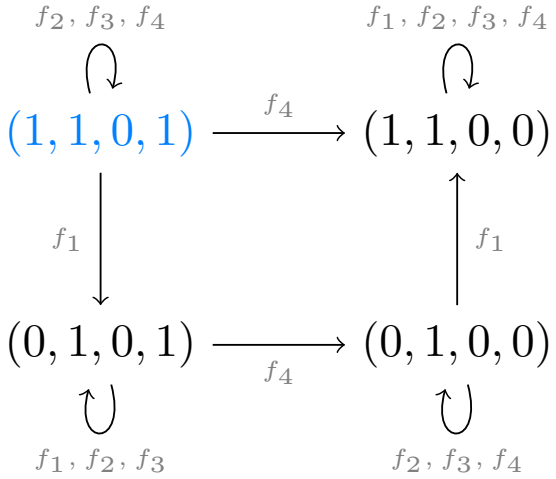


FIGURE 5.1 – **Graphe de transitions asynchrones d'un réseau Booléen.** Ce graphe montre les transitions asynchrones du réseau Booléen de l'exemple 5.1 à partir de l'état global initial représenté en bleu. Les noeuds représentent des états globaux du RB, et les arcs des transitions asynchrones entre ces états.

Nous avons cette relation bien connue entre les points attracteurs pour la dynamique synchrone d'un RB et ceux pour sa dynamique asynchrone :

Propriété 5.1. Les points attracteurs de la dynamique synchrone d'un RB sont exactement les points attracteurs de la dynamique asynchrone de ce RB.

Exemple 5.3 (Attracteurs d'un RB)**.** Le RB de l'exemple 5.1 a trois points attracteurs, qui sont les états globaux $(1,0,0,0)$,$(1,1,0,0)$ et $(0,1,1,1)$. Il a un attracteur cyclique pour sa dynamique synchrone, qui est l'ensemble d'état globaux $\{(1,0,1,0),(1,0,1,1),(0,0,1,1),(0,0,1,0)\}$, et un attracteur cyclique pour sa dynamique asynchrone, qui est exactement le même.

5.3 Des réseaux Booléens aux programmes logiques normaux propositionnels, et vice-versa

Dans [Ino11], les auteurs établissent une correspondance formelle entre les réseaux Booléens (RB) et les programmes logiques normaux (NLP) propositionnels. Ils proposent une traduction des RB vers les NLP propositionnels (et vice-versa) telle que :

- les points attracteurs du RB correspondent aux modèles supportés du NLP correspondant et
- les traces de la dynamique synchrone du RB correspondent aux orbites du NLP.

Nous donnons dans la suite la traduction des réseaux Booléens en NLP, ainsi que la traduction inverse, qui ont été introduites dans [Ino11].

Des réseaux Booléens aux programmes logiques normaux Soit $B = (V, F)$ un réseau Booléen, tel que $V = \{v_1, \ldots, v_n\}$ et $F = \{f_1, \ldots, f_n\}$. Pour chaque $v_i \in V$, supposons que la fonction Booléenne f_i qui lui est associée soit sous forme normale disjonctive, i.e. de la forme :

$$f_i(v_1, \ldots, v_n) = \bigvee_{j=1}^{l_i} B_{i,j} \text{ où } B_{i,j} = \bigwedge_{k=1}^{m_j} v_{i,j,k} \wedge \bigwedge_{k=m_j+1}^{n_j} \neg v_{i,j,k},$$

avec $v_{i,j,k} \in V$ et $n_j \geq m_j \geq 0$, et $m_j \geq 1$ ou $n_j \geq 1$, pour tout $1 \leq j \leq l_i$.
Sa traduction en NLP propositionnel est le programme $P(B)$ défini de la manière suivante :

$$P(B) \triangleq \{v_i \leftarrow B_{i,j} \mid 1 \leq j \leq l_i, v_i \in V\}$$

Pour un état global s de B, nous dénotons par $I(s)$ l'interprétation de $P(B)$ correspondante, et définie de la manière suivante :

$$I(s) \triangleq \{v_i \mid v_i(s) = 1\}$$

Nous avons les deux théorèmes suivants, introduits dans [Ino11], qui relient les traces de B aux orbites de $P(B)$, et les points attracteurs de B aux modèles supportés de $P(B)$:

Théorème 5.1. Soit B un réseau Booléen, et $P(B)$ sa traduction en NLP propositionnel. Soit s un état global de B, et $s \rightarrow_{sy} s' \rightarrow_{sy} s'' \rightarrow_{sy} \ldots$ l'unique trace partant de s dans la dynamique synchrone de B. Alors l'orbite de $I(s)$ par rapport à $P(B)$ est exactement la séquence $\langle I(s), I(s''), I(s'''), \ldots \rangle$ d'interprétations de $P(B)$.

Théorème 5.2. Soit B un réseau Booléen, et $P(B)$ sa traduction en NLP propositionnel. Soit s un état global de B. L'état s est un point attracteur de B *ssi* $I(s)$ est un modèle supporté de $P(B)$.

Des programmes logiques normaux aux réseaux Booléens Soit P un NLP propositionnel, tel que $var(P) = \{v_1, \ldots, v_n\}$.
Sa traduction en RB est le RB $B = (V, F)$, défini de la manière suivante :

$$V \triangleq \{v_1, \ldots, v_n\}$$

et

$$F \triangleq \{f_1, \ldots, f_n\}$$

où

$$f_i(v_1, \ldots, v_n) \triangleq \begin{cases} 1 \text{ s'il existe } R \in P \text{ telle que } R = v_i \leftarrow; \\ 0 \text{ si pour toute règle } R \in P, head(R) \neq v_i; \\ \bigvee_{j=1}^{k_i} B_{i,j} \text{ sinon, où } \{v_i \leftarrow B_{i,1}, \ldots, v_i \leftarrow B_{i,k_i}\} \text{ est l'ensemble} \\ \quad \text{des règles de } P \text{ concluant sur } v_i \text{ qui ne sont pas des faits.} \end{cases}$$

Pour une interprétation I de P, nous dénotons par $S(I)$ l'état global de $B(P)$ correspondant, et défini de la manière suivante :

$$S(I) = (b_1, \ldots, b_n) \text{ où } b_i = \begin{cases} 1 \text{ si } v_i \in I; \\ 0 \text{ sinon.} \end{cases}$$

Nous avons le théorème suivant, introduit dans [Ino11], qui relie les modèles supportés de P (voir le chapitre 2) aux points attracteurs de $B(P)$:

Théorème 5.3. Soit P un NLP propositionnel, et $B(P)$ sa traduction en réseau Booléen. Soit I une interprétation de P. L'interprétation I est un modèle supporté de P *ssi* $S(I)$ est un point attracteur de $B(P)$.

Finalement, nous pouvons relier les orbites de P aux traces de $B(P)$:

Théorème 5.4. Soit P un NLP propositionnel, et $B(P)$ sa traduction en réseau Booléen. Soit I_0 une interprétation de P, et $\langle I_0, I_1, I_2, \ldots \rangle$ l'orbite de I_0 par rapport à P. Alors $S(I_0) \rightarrow_{sy} S(I_1) \rightarrow_{sy} S(I_2) \rightarrow_{sy} \ldots$ est l'unique trace de la dynamique synchrone de $B(P)$ partant de $S(I_0)$.

Ce théorème n'est pas donné dans [Ino11]. Sa preuve est donnée dans l'annexe C.

5.4 Modélisation de la dynamique des graphes d'influences à l'aide de réseaux Booléens

Comme nous l'avons déjà mentionné dans l'introduction, la sémantique Booléenne est particulièrement adaptée à la modélisation de la dynamique des graphes d'influences. Le formalisme de choix pour la modélisation de tels graphes à l'aide d'une sémantique Booléenne est celui des réseaux Booléens (RB). Dans une telle modélisation, chaque activité du graphe d'influences est modélisée par une variable Booléenne, et les influences ciblant une activité donnée sont modélisées par une fonction Booléenne associée à la variable qui modélise l'activité ciblée. Ces fonctions Booléennes ne peuvent pas être construites n'importe comment, et doivent tenir compte de la nature des modulations qui entrent en jeu. Il est ainsi généralement admis le principe suivant : si une activité A a un influence positive sur une activité B, alors la variable Booléenne associée à B doit apparaître positivement dans la fonction correspondant à la variable associée à A ; et dans le cas où A a une influence négative sur B, elle doit y apparaître négativement.

Cependant, ce principe général ne permet pas de modéliser directement un graphe d'influences en un RB. En effet, s'il permet de définir quelles variables booléennes entrent en jeu dans les fonctions, il ne permet en aucun cas de définir les opérateurs logiques (de conjonction ou de disjonction) qui vont organiser ces variables en une fonction logique. La tâche consistant à définir précisément la forme des fonction logiques du RB est appelée sa *paramétrisation*.

5.4.1 Différentes méthodes de paramétrisation des réseaux Booléens

Nous recensons quatre méthodes pour paramétrer un RB. La première de ces méthodes consiste à construire le RB directement à partir de résultats expérimentaux, en même temps que le graphe d'influences. C'est la méthode principalement employée par les modélisateurs quand ils visent à modéliser un processus biologique spécifique. La difficulté de cette méthode vient principalement du fait qu'il

n'y a pas toujours d'expérience disponible pour paramétrer une certaine fonction logique. En effet, la majorité des résultats expérimentaux ne permettent bien souvent que de déduire des influences entre activités, les expériences montrant comment les influences sur une même activité interagissent étant plus lourdes à mettre en place.

La deuxième méthode utilise également des résultats expérimentaux pour paramétrer les fonctions. Plutôt que de déduire à partir de quelques expériences la paramétrisation d'une fonction Booléenne, la paramétrisation de l'ensemble du RB est calculée à partir d'un ensemble de résultats expérimentaux, en utilisant de la vérification de modèles. Par exemple, dans [Vid+12], les auteurs utilisent des mesures de changement d'activités entre un état initial et un état pseudo-stationnaire pour paramétrer des réseaux Booléens modélisant des réseaux de signalisation.

La troisième méthode consiste à choisir la paramétrisation la plus large possible, de façon à ce que la dynamique obtenue englobe la dynamique qui aurait été obtenue avec n'importe quelle autre paramétrisation. C'est cette solution que nous choisirons dans le chapitre suivant.

Enfin, la dernière méthode consiste à paramétrer les RBs à l'aide de principes généraux additionnels qui décrivent la dynamique des graphes d'influences, et qui sont choisis par le modélisateur. C'est cette solution qui est généralement choisie pour paramétrer des RB construits directement à partir de graphes d'influences, sans avoir recours à l'analyse de résultats expérimentaux.

5.4.2 Paramétrisation des réseaux Booléens à l'aide de principes généraux

Nous montrons maintenant comment un réseau Booléen (RB) modélisant une carte SBGN-AF peut être paramétré à l'aide d'un ensemble de principes généraux, que nous introduisons ci-après. Nous discutons ensuite ces principes, et donnons notamment quelques alternatives à ces derniers qui sont parfois choisies dans la littérature.

Les principes que nous avons choisis dans le cadre de cette étude sont les suivants. Soient A, B et C trois activités d'un graphe d'influences SBGN-AF.

(B1) si A a une influence positive sur B et que A est réalisée, alors le taux de B a tendance à augmenter ;

(B2) si A a une influence négative sur B et que A est réalisée, alors le taux de B a tendance à diminuer ;

(B3) si A est la source d'une stimulation nécessaire ciblant B, alors A doit être réalisée pour que B le soit ;

(B4) si A a une influence positive sur C, B a une influence négative sur C, et A et B sont toutes deux réalisées, alors le taux de C tend à diminuer ;

(B5) si B est la cible d'au moins une influence positive, alors au moins une des activités à la source d'une telle influence doit être réalisée pour que B le soit ;

(B6) si B n'est pas la cible d'une influence positive, alors aucune des sources des influences négatives la ciblant ne doit être réalisée pour que B le soit ;

(B7) si B n'est la cible d'aucune influence, alors la réalisation de B ne dépend d'aucune autre activité, et le taux de B est constant.

Les principes (B1-3) dérivent directement des définitions des différents arcs de modulation donnés dans la spécification de SBGN-AF. Le principe (B4) stipule que les inhibiteurs prennent le pas sur les activateurs, et est couramment utilisé dans la littérature (p.ex. dans [Alb04], pour la modélisation de réseaux de gènes). D'autres principes pourraient remplacer (B4) : par exemple, dans [Fay+11], les auteurs proposent de comparer le nombre d'activateurs de C au nombre d'inhibiteurs de C ; si le nombre d'activateurs est plus grand, alors le taux de C aura tendance à augmenter ; il aura tendance à diminuer sinon. Le principe (B5) est nécessaire pour que les influences positives aient un effet. Le principe (B6) permet de prendre en compte le cas suivant, parfois rencontré dans les processus

de signalisation : certaines activités n'étant pas la cible d'influences positives proviennent d'entités moléculaires qui ont une activité basale, c'est-à-dire qui sont actives sous une forme non modifiée, et qui perdent leur activité lorsqu'elles sont transformées sous l'influence d'une molécule dont l'activité est alors inhibitrice. Finalement, le principe (B7) est naturel et exprime le caractère immuable d'une activité qui ne subit aucune influence.

Ces principes généraux permettent de paramétrer un RB modélisant une carte SBGN-AF donnée. En effet, ils décrivent, pour une activité A d'une telle carte, les conditions nécessaires et suffisantes sur la réalisation des activités qui modulent A pour que A soit réalisée. Par conséquent, le RB construit à partir d'une carte SBGN-AF ne peut être paramétré que d'une seule façon à l'aide de ces principes. Nous montrons dans la suite comment un RB complètement paramétré peut être construit à partir d'une carte SBGN-AF et des principes (B1–7).

Étant donnée une carte SBGN-AF S, le RB modélisant S avec les principes généraux (B1–7) est construit en associant à chaque activité de S une variable Booléenne qui représente l'état de cette activité, et une fonction définie sur les variables associées aux modulateurs de cette activité, selon les principes (B1–7). La valeur Booléenne 0 représente l'état d'une activité où elle n'est pas réalisée, et la valeur 1 l'état où elle est réalisée.

À partir des principes généraux que nous avons choisis, nous pouvons distinguer trois cas pour modéliser la dynamique d'une activité A, suivant l'ensemble des modulateurs de A :

- Cas 1 : A n'a pas de modulateurs ;
- Cas 2 : A n'a que des inhibiteurs ;
- Cas 3 : A a des stimulateurs (et possiblement des inhibiteurs).

La fonction Booléenne décrivant la dynamique Booléenne d'une activité du cas 1 peut être dérivée du principe (B7) ; celle décrivant la dynamique d'une activité du cas 2 des principes (B2) et (B6) ; et celle décrivant la dynamique d'une activité du cas 3 par les principes (B1–5).

La source d'un arc de modulation est soit une unique activité soit un opérateur logique. Si c'est un opérateur logique, l'ensemble des opérateurs logiques et activités qui sont des ancêtres de cet opérateur et qui sont liés récursivement à celui-ci par des arcs logiques forment un arbre qui peut être vu comme une fonction logique. Ainsi, tout opérateur logique d'une carte peut être associé à une fonction logique. En considérant la sémantique des opérateurs logiques donnée dans la spécification de SBGN-AF [Mi+09], nous définissons, de manière récursive, la *satisfaction* de la fonction logique associée à un opérateur logique. Par abus de langage, nous confondons un opérateur logique avec la fonction qui lui est associée, et définissons la satisfaction des opérateurs plutôt que des fonctions logiques associées :

- un opérateur AND est satisfait si tous ses parents qui sont des activités sont réalisés, et si tous ses parents qui sont des opérateur logiques sont satisfaits ;
- un opérateur OR est satisfait si au moins un de ses parents qui sont des activités est réalisée, ou au moins un de ses parents qui sont des opérateurs logiques est satisfait ;
- un opérateur NOT est satisfait si son unique parent est une activité qui n'est pas réalisée, ou si son unique parent est un opérateur logique qui n'est pas satisfait.

Exemple 5.4 (Satisfaction des opérateur logiques)**.** La figure 5.2 montre deux modulations, chacune ayant comme source un opérateur logique. La fonction logique associée à l'opérateur qui est la source de la modulation du réseau de gauche est la fonction $(A \vee B) \wedge \neg C$. Celle du réseau de droite est la fonction $(A \wedge B) \wedge \neg C$. Supposons que A est réalisée tandis que B et C ne le sont pas. Alors

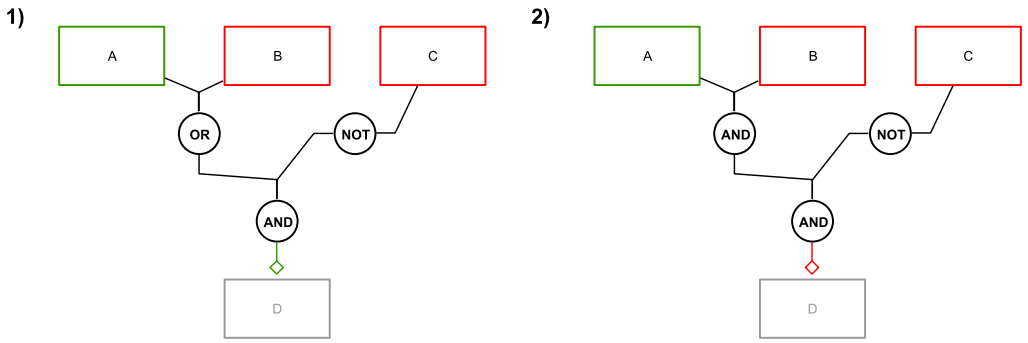


FIGURE 5.2 – **Satisfaction d'une fonction logique à la source d'une modulation.** Les activitéss représentées en vert sont réalisées, et sont associées à la valeur *vrai*, tandis que les activités représentées en rouge ne sont pas réalisées, et sont ainsi associées à la valeur $faux$. La source du réseau 1) est par conséquent satisfaite, tandis que la source du réseau 2) ne l'est pas.

l'opérateur à la source de la modulation du réseau de gauche est satisfait, tandis que celui à la source de la modulation du réseau de droite ne l'est pas.

Nous donnons maintenant la définition formelle d'un RB construit à partir d'une carte SBGN-AF S, et paramétré à l'aide des principes généraux (B1–7).

Nous utilisons les notations suivantes pour nous référer aux éléments de S :

- $\mathcal{A} = \{a_1, \ldots, a_n\}$ est l'ensemble des activités de la carte ;
- $\mathcal{O} = \{o_1, \ldots, o_q\}$ est l'ensemble des opérateurs logiques de la carte ;
- pour chaque activité $a \in \mathcal{A}$, nous dénotons par $req(a)$ (resp. $stim(a)$, $inh(a)$) les sources de l'ensemble des stimulations nécessaires (resp. influences positives, influences négatives) ciblant a ;
- pour chaque opérateur logique $o \in \mathcal{O}$, nous dénotons par $in(o)$ l'ensemble des noeuds (activités ou opérateurs logiques) sources d'un arc logique entrant sur o.

Pour une formule Booléenne f, nous dénotons par $\mathsf{DNF}(f)$ une forme normale disjonctive de f.

Définition 5.8 (Réseau Booléen construit à partir d'une carte SBGN-AF et paramétré à l'aide de principes généraux)**.** Le RB modélisant S avec les principes généraux (B1–7), dénoté par $B(S)$, est le couple (V, F) défini de la façon suivante :

$$V \triangleq \{v_i \mid a_i \in \mathcal{A}\} \text{ et } F \triangleq \{f_i \mid a_i \in \mathcal{A}\}$$

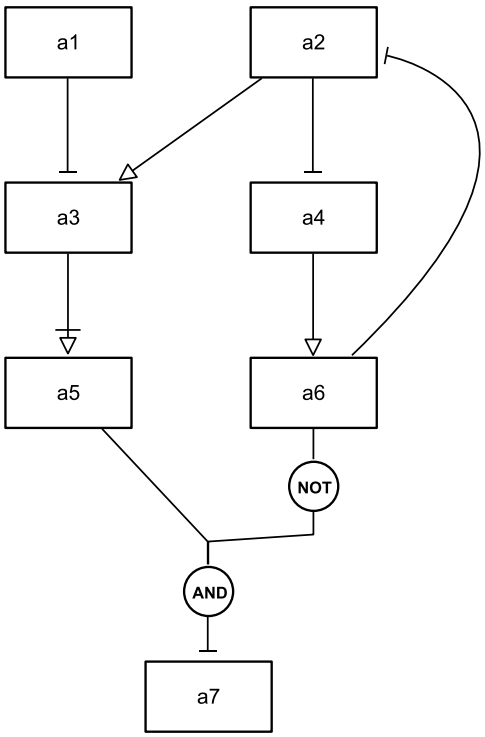
avec

$$f_i \triangleq \begin{cases} v_i \text{ si } req(a_i) = stim(a_i) = inh(a_i) = \emptyset \text{ (Cas 1) ;} \\ \mathsf{DNF}\Big(\bigwedge\limits_{i \in inh(a_i)} \neg \mathsf{logic}(i)\Big) \text{ si } req(a_i) = stim(a_i) = \emptyset \text{ et } inh(a_i) \neq \emptyset \text{ (Cas 2) ;} \\ \mathsf{DNF}\Big(\Big[\bigvee\limits_{s \in req(a_i) \cup stim(a_i)} \mathsf{logic}(s)\Big] \wedge \Big[\bigwedge\limits_{r \in req(a_i)} \mathsf{logic}(r)\Big] \wedge \Big[\bigwedge\limits_{i \in inh(a_i)} \neg \mathsf{logic}(i)\Big]\Big) \\ \quad \text{si } req(a_i) \cup stim(a_i) \neq \emptyset \text{ (Cas 3) .} \end{cases}$$

où logic est une fonction sur $\mathcal{A} \cup \mathcal{O}$ qui modélise la satisfaction des opérateur logiques, et qui est définie de la manière suivante :

$$\mathsf{logic}(q) \triangleq \begin{cases} v_i & \text{si } q = a_i, a_i \in \mathcal{A}; \\ \bigwedge\limits_{j \in in(q)} \mathsf{logic}(j) & \text{si } q \in \mathcal{O} \text{ et } q \text{ est un opérateur AND;} \\ \bigvee\limits_{j \in in(q)} \mathsf{logic}(j) & \text{si } q \in \mathcal{O} \text{ et } q \text{ est un opérateur OR;} \\ \neg \mathsf{logic}(j) & \text{si } q \in \mathcal{O}, q \text{ est un opérateur NOT et } in(q) = \{j\}. \end{cases}$$

Exemple 5.5 (Réseau Booléen paramétré à l'aide de principes généraux)**.** La figure suivante montre un exemple de carte SBGN-AF :



Le RB défini par le couple (V, F) modélisant cette carte et construit à partir de principes généraux est le suivant :

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$$

$$F = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$$

où

$$f_1(v_1, \ldots, v_7) = v_1 \ ; \ f_2(v_1, \ldots, v_7) = \neg v_6 \ ; \ f_3(v_1, \ldots, v_7) = \neg v_1 \wedge v_2 \ ;$$
$$f_4(v_1, \ldots, v_7) = \neg v_2 \ ; \ f_5(v_1, \ldots, v_7) = v_3 \ ;$$
$$f_6(v_1, \ldots, v_7) = v_4 \ ; \text{ et } f_7(v_1, \ldots, v_7) = \neg v_5 \vee v_6.$$

5.5 Modélisation de la dynamique des graphes d'influences SBGN-AF à l'aide de programmes logiques normaux du premier ordre

En partant des résultats sur la relation entre les réseaux Booleéns (RB) et les programmes logiques normaux (NLP) propositionnels introduits dans [Ino11], nous montrons que les points attracteurs et les traces finies du RB construit à partir d'une carte SBGN-AF et paramétré à l'aide des principes généraux énoncés plus haut peuvent être calculés à partir de deux NLP du permier ordre. Ces NLP sont formés d'un ensemble de règles qui formalisent nos principes généraux, du point de vue d'une sémantique Booléenne.

5.5.1 Construction des programmes logiques

Formalisation des principes généraux sous formes de règles logiques. Afin de distinguer les état successifs pris par une activité, nous introduisons une notion de temps discret, qui est modélisée par deux prédicats : $time/1$ et $next/2$. L'atome $time(T)$ signifie que T est un pas de temps, et l'atome $next(T', T)$ que le pas de temps T' est consécutif au pas de temps T. Nous modélisons l'état d'une activité par deux valeurs : la valeur 0, indiquant que l'activité n'est pas réalisée ; et la valeur 1, indiquant que l'activité est réalisée. Nous modélisons l'état d'une activité à l'aide d'un prédicat $present/2$. Pour une activité A, l'atome $present(A, T)$ signifie que A est réalisée au temps T, et $\neg present(A, T)$ que A n'est pas réalisée au temps T. Nous modélisons la satisfaction d'un opérateur logique à l'aide d'un prédicat $logic/2$. L'atome $logic(O, T)$ signifie que l'opérateur logique O est satisfait au temps T. Enfin, $\neg logic(O, T)$ signifie que O n'est pas satisfait au temps T.

À l'aide de ces prédicats, nous donnons des axiomes qui expriment la dynamique Booléenne d'une carte SBGN-AF en considérant les principes généraux édictés plus avant. L'objectif est d'exprimer les conditions à un temps T pour qu'une activité soit réalisée au temps T', T' étant le pas de temps consécutif à T. Comme nous utilisons la négation par l'échec, nous n'avons par exemple pas besoin d'écrire d'axiomes exprimant les conditions à T pour lesquelles une activité ne serait pas présente à T', i.e. d'axiomes définissant $\neg present(A, T)$, même si ce littéral $\neg present(A, T)$ apparaît dans le corps d'une règle.

Nous introduisons d'abord des règles définissant des prédicats auxiliaires qui sont nécessaires pour exprimer les règles formalisant les principes généraux, que nous présentons ensuite.

Définition des prédicats auxiliaires Soit A une activité. Nous définissons les prédicats auxiliaires suivants :

- $hasModulator(A)$ signifie que A a un modulateur ; c'est vérifié s'il existe une activité ou un opérateur logique M qui a une influence sur A :

$$hasModulator(A) \leftarrow activity(A) \wedge modulates(M, A) \tag{A1}$$

- $hasStimulator(A)$ signifie que A a un stimulateur ; c'est vérifié s'il existe une activité ou un opérateur logique S qui a une influence positive sur A :

$$hasStimulator(A) \leftarrow activity(A) \wedge stimulates(S, A) \tag{A2}$$

- $hasPresentStimulator(A, T)$ signifie que A a au moins un stimulateur qui est une activité et est réalisée à T, ou un stimulateur qui est un opérateur logique et qui est satisfait à T ; c'est vérifié

s'il existe une activité S qui a une influence positive sur A et qui est réalisée au temps T, ou s'il existe un opérateur logique S qui a une influence positive sur A et qui est satisfait au temps T :

$$hasPresentStimulator(A,T) \leftarrow time(T) \wedge activity(A) \wedge activity(S) \quad \text{(A1)}$$
$$\wedge\, stimulates(S,A) \wedge present(S,T) \quad \text{(A3)}$$

$$hasPresentStimulator(A,T) \leftarrow time(T) \wedge activity(A) \wedge logicalOperator(S) \wedge stimulates(S,A) \wedge logic(S,T); \quad \text{(A4)}$$

- $hasPresentInhibitor(A,T)$ signifie que A a au moins un inhibiteur qui est une activité et qui est réalisée à T, ou un inhibiteur qui est un opérateur logique et qui est satisfait à T ; c'est vérifié s'il existe une activité I qui a une influence négative sur A et qui est réalisée à T, ou s'il existe un opérateur logique I qui a une influence négative sur A et qui est satisfait à T :

$$hasPresentInhibitor(A,T) \leftarrow time(T) \wedge activity(A) \wedge activity(I) \wedge inhibits(I,A) \wedge present(I,T); \quad \text{(A5)}$$

$$hasPresentInhibitor(A,T) \leftarrow time(T) \wedge activity(A) \wedge logicalOperator(I) \wedge inhibits(I,A) \wedge logic(I,T); \quad \text{(A6)}$$

- $hasAbsentNecessaryStimulator(A,T)$ signifie que A a au moins un stimulateur nécessaire qui est une activité et qui n'est pas réalisée à T, ou a un stimulateur nécessaire qui est un opérateur logique et qui n'est pas satisfait à T ; c'est vérifié s'il existe une activité R qui exerce une stimulation nécessaire sur A et qui n'est pas réalisée à T, ou s'il existe un opérateur logique R qui exerce une stimulation nécessaire sur A et qui n'est pas satisfait à T :

$$hasAbsentNecessaryStimulator(A,T) \leftarrow time(T) \wedge activity(A) \wedge activity(R) \wedge necessarilyStimulates(R,A) \wedge \neg present(R,T) \quad \text{(A7)}$$

$$hasAbsentNecessaryStimulator(A,T) \leftarrow time(T) \wedge activity(A) \wedge logicalOperator(R) \wedge necessarilyStimulates(R,A) \wedge \neg logic(R,T) \quad \text{(A8)}$$

Satisfaction des opérateurs logiques Les axiomes suivants permettent de définir la satisfaction au temps T des formules Booléennes associées aux différents opérateurs logiques :

- Si O est un opératur AND qui a comme entrée une activité J qui n'est pas réalisée à T ou un opérateur logique J qui n'est pas satisfait à T, alors O non-satisfait à T :

$$notLogic(O,T) \leftarrow time(T) \wedge and(O) \wedge activity(J) \wedge input(J,O) \wedge \neg present(J,T); \quad \text{(A9)}$$

$$notLogic(O,T) \leftarrow time(T) \wedge and(O) \wedge logicalOperator(J) \wedge input(J,O) \wedge \neg logic(J,T); \quad \text{(A10)}$$

- Si O est un opérateur AND et qu'on ne peut pas déduire qu'il est non-satisfait à T, alors il est satisfait à T :

$$logic(O,T) \leftarrow time(T) \wedge and(O) \wedge \neg notLogic(O,T) \quad \text{(A11)}$$

- Si O est un opérateur OR qui a comme entrée une activité J qui est réalisée à T ou un opérateur logique J qui est satisfait à T, alors O est satisfait à T :

$$logic(O,T) \leftarrow time(T) \wedge or(O) \wedge activity(J) \wedge input(J,O) \wedge present(J,T) \quad \text{(A12)}$$

$$logic(O,T) \leftarrow time(T) \wedge or(O) \wedge logicalOperator(J) \wedge input(J,O) \wedge logic(J,T) \quad \text{(A13)}$$

- Si O est un opérateur NOT qui a comme unique entrée une activité J qui n'est pas réalisée à T ou un opérateur logique qui n'est pas satisfait à T, alors O est satisfait à T :

$$logic(O,T) \leftarrow time(T) \wedge not(O) \wedge logicalOperator(J) \wedge input(J,O) \wedge \neg logic(J,T) \quad \text{(A14)}$$

$$logic(O,T) \leftarrow time(T) \wedge not(O) \wedge activity(J) \wedge input(J,O) \wedge \neg present(J,T) \quad \text{(A15)}$$

Principes généraux Soient A une activité, T et T' deux pas de temps tels que T' soit consécutif à T. Les axiomes suivants expriment les différents principes généraux que nous avons édictés, et considérés avec une sémantique Boolénne :

- À partir de (B7) : si A n'est la cible d'aucune modulation et est réalisée à T, alors elle est réalisée à T' :

$$present(A,T') \leftarrow time(T) \wedge time(T') \wedge next(T',T) \wedge activity(A) \wedge \neg hasModulator(A) \wedge present(A,T); \quad \text{(A16)}$$

- À partir de (B2) et (B6) : si est la cible d'au moins une modulation, n'est la cible que d'influences négatives et qu'aucune des sources de ces influences qui sont des activités n'est réalisée à T, et qu'aucunes des sources qui sont des opérateurs logiques n'est satisfaite à T, alors A est réalisée à T' :

$$\begin{aligned} present(A,T') \leftarrow\ & time(T) \wedge time(T') \wedge next(T',T) \\ & \wedge activity(A) \wedge hasModulator(A) \\ & \wedge \neg hasStimulator(A) \wedge \neg hasPresentInhibitor(A,T); \end{aligned} \tag{A17}$$

- À partir de (B1–5) : si A est la cible d'au moins une modulation ; qu'aucune des sources des influences négatives ciblant A qui sont des activités n'est réalisée à T et qu'aucune des sources des influences négatives ciblant A qui sont des opérateurs logiques n'est satisfaite à T ; que toutes les sources des stimulations nécessaires ciblant A qui sont des activités sont réalisées à T et que toutes les sources des stimulations nécessaires ciblant A qui sont des opérateurs logiques sont satisfaites à T ; et qu'au moins une des sources des influences positives ciblant A qui sont des activités est réalisée à T ou qu'au moins une des sources des influences positives ciblant A qui sont des opérateurs logiques est satisfaite à T, alors A est réalisée à T' :

$$\begin{aligned} present(A,T') \leftarrow\ & time(T) \wedge time(T') \wedge next(T',T) \\ & \wedge activity(A) \wedge hasModulator(A) \\ & \wedge hasPresentStimulator(A,T) \\ & \wedge \neg hasAbsentNecessaryStimulator(A,T) \\ & \wedge \neg hasPresentInhibitor(A,T). \end{aligned} \tag{A18}$$

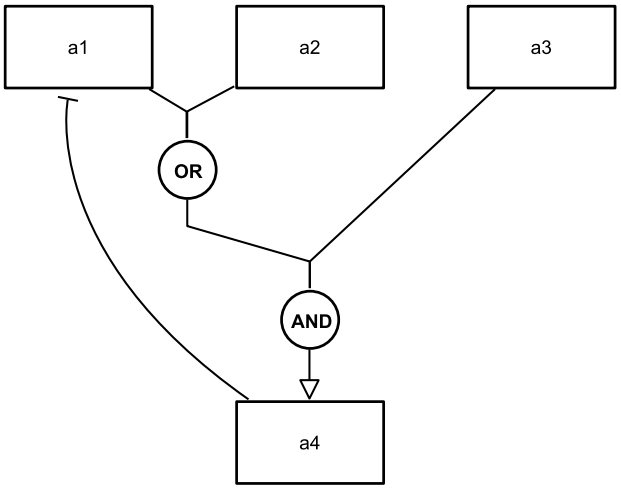
Dans la suite, nous montrons que la dynamique Booléenne d'une carte SBGN-AF peut être calculée à partir de deux NLP du premier ordre construits à partir de la traduction en SBGNLog-AF de cette carte, et des axiomes (A1–18). Le premier de ces programmes permet de calculer les traces finies de la dynamique Booléenne, tandis que le deuxième permet de calculer les points attracteurs.

Construction de deux programmes logiques Soit S une carte SBGN-AF. Nous dénotons par $B(S)$ le RB construit à partir de S en considérant les principes généraux (B1–7). Nous dénotons par Π_{ONTO} l'ensemble des axiomes ontologiques de SBGNLog-AF limités à ceux traduisant des relations is_a (par exemple, l'axiome traduisant le fait qu'une stimulation nécessaire est une influence positive). Nous dénotons par Π_A l'ensemble des axiomes (A1–18), et par $\Pi_{TRAD}(S)$ la traduction de S en SBGNLog-AF.

Finalement, nous dénotons par $\Pi(S)$ le programme qui rassemble ces trois programmes :

$$\Pi(S) \triangleq \Pi_{TRAD}(S) \cup \Pi_{ONTO} \cup \Pi_A$$

Exemple 5.6. La figure suivante montre un exemple de carte SBGN-AF :



Soit S la carte SBGN-AF représentée ci-dessus. Le RB $B(S)$, modélisant cette carte à partir des principes généraux définis plus avant, est défini par le couple (V, F), où :

$$V = \{v_1, v_2, v_3, v_4\}$$

et

$$F = \{f_1, f_2, f_3, f_4\}$$

où

$$f_1(v_1, v_2, v_3, v_4) = \neg v_4 \ ; \ f_2(v_1, v_2, v_3, v_4) = v_2 \ ;$$
$$f_3(v_1, v_2, v_3, v_4) = v_3 \ ; \text{ et } f_4(v_1, v_2, v_3, v_4) = (v_1 \wedge v_3) \vee (v_2 \wedge v_3).$$

Le programme $\Pi_{TRAD}(S)$, qui est la traduction de la carte S en SBGNLog-AF, est l'ensemble d'atomes suivants :

$activity(a_1)$	$activity(a_2)$	$activity(a_3)$	$activity(a_4)$
$or(lo_1)$	$and(lo_2)$		
$input(a_1, lo_1)$	$input(a_2, lo_1)$	$input(lo_1, lo_2)$	$input(a_3, lo_2)$
$stimulates(lo_2, a_4)$	$inhibits(a_4, a_1)$		

Le programme $\Pi(S)$ est l'union de cet ensemble d'atomes, des axiomes de SBGNLog-AF traduisant les relations is_a des ontologies de SBGN-AF, et de l'ensemble des axiomes (A1–18).

Nous transformons le programme $\Pi(S)$ en deux programmes instanciés qui pemettront respectivement de calculer les points attracteurs de $B(S)$ et les traces de longueur finie de $B(S)$.

Nous dénotons par $\Pi_{At}(S)$ le programme permettant de calculer les points attracteurs de $B(S)$. Pour un entier positif T_{max} donné, nous dénotons par $\Pi_{Tr}(S, T_{max})$ le programme permettant de calculer les traces de longueur T_{max} de $B(S)$.

- Le programme $\Pi_{At}(S)$ est obtenu à partir de $\Pi(S)$ en supprimant la notion de temps des axiomes (A1–18). En effet, comme $\Pi_{At}(S)$ est construit pour calculer les points attracteurs de $B(S)$, nous serons amenés à calculer ses points fixes, et par conséquent la notion de temps n'est pas nécessaire.

Nous obtenons donc $\Pi_{At}(S)$ en supprimant les prédicats *next* et *time* des règles de $\Pi(S)$, et en supprimant les variables de temps T et T' des prédicats *present*, *logic* et *notLogic*. Ces trois prédicats, dans le programme $\Pi_{At}(S)$, deviennent donc unaires.

- Le programme $\Pi_{Tr}(S, T_{max})$ est obtenu à partir de $\Pi(S)$ et d'un ensemble d'atomes construits à partir de T_{max}. Comme $\Pi_{Tr}(S, T_{max})$ est construit pour calculer les traces de longueur T_{max} de $B(S)$, nous devons adjoindre à ce programme l'ensemble des atomes construits à partir des symboles de prédicat *next* et *time*, en prenant en compte tous les pas de temps entre le temps 0 et le temps T_{max}. Nous dénotons cet ensemble par Π_T. Formellement :

$$\Pi_T(T_{max}) \triangleq \{time(t) \mid 0 \leq t \leq T_{max}\} \cup \{next(t+1, t) \mid 0 \leq t \leq T_{max} - 1, t \in \mathbb{N}\}$$

Le programme $\Pi_{Tr}(S, T_{max})$ est défini quant à lui défini de la manière suivante :

$$\Pi_{Tr}(S, T_{max}) \triangleq \Pi(S) \cup \Pi_T(T_{max})$$

Dans la suite, nous montrons formellement comment les points attracteurs et les traces finies de longeur T_{max} de $B(S)$ peuvent être calculées à partir de $\Pi_{At}(S)$ et $\Pi_{Tr}(S, T_{max})$, respectivement. Ces deux résultats sont établis en simplifiant les programmes $\Pi_{At}(S)$ et $\Pi_{Tr}(S, T_{max})$ à l'aide des règles de simplification (SR1–4) et de transformation (TR1–2) définies dans le chapitre 2, et en montrant la correspondance entre ces programmes simplifiés et le RB $B(S)$, à l'aide de la traduction des NLP propositionnels en RB introduite dans [Ino11], et donnée dans la section 5.3.

5.5.2 Transformation des programmes logiques

Nous notons tout d'abord que $\Pi_{At}(S)$ et $\Pi_{Tr}(S, T_{max})$ sont finiement instanciables. Nous considérons donc les versions instanciées de ces programmes. Afin de simplifier ces deux programmes, nous distinguons d'abord différents types de règles parmis ceux-ci. La figure 5.3 montre le graphe de dépendance des prédicats des programmes $\Pi_{Tr}(S, T_{max})$ et $\Pi_{At}(S)$. Ce graphe permet de visualiser la structure des deux programmes et de définir des propriétés qui nous seront utiles pour la simplification des programmes. Les deux programmes ont le même graphe de dépendance des prédicats, aux prédicats définissant la notion de temps près : $\Pi_{At}(S)$ ne comporte en effet pas d'atomes formés des symboles de prédicat *time* et *next*. Ces prédicats sont entourés en pointillés dans la figure 5.3.

À partir du graphe $G_{pred}(\Pi_{Tr}(S, T_{max}))$, nous définissons trois types de prédicats, selon qu'ils sont liés ou non au prédicat *present* dans ce graphe :

- les prédicats de *type A* sont définis récursivement de la manière suivante : un prédicat est de type A si le noeud auquel il est associé dans $G_{pred}(\Pi_{Tr}(S, T_{max}))$ n'a aucun prédécesseur ou seulement des prédécesseurs associés à des prédicats de type A ;
- l'unique prédicat de *type C* est le prédicat *present* ;
- les prédicats de *type B* sont les prédicats qui ne sont ni de type A ni de type C.

Les prédicats de type A et de type B dont des prédicats auxiliaires, définis par les axiomes (A1–15). Intuitivement, les atomes formés de symboles de prédicats de type A pourront être éliminés du corps des règles de nos deux programmes par les règles de simplification (SR1–4), et les atomes formés de symboles de prédicat de type B pourront être éliminés du corps des règles de nos deux programmes par les règles de transformation (TR1–2). Ainsi, les programmes simplifiés ne comporteront plus que des faits ou des règles formés d'instances du prédicat *present*.

Dans la figure 5.3, les prédicats de type A sont colorés en blanc, ceux de type B en gris, et l'unique prédicat de type C en noir.

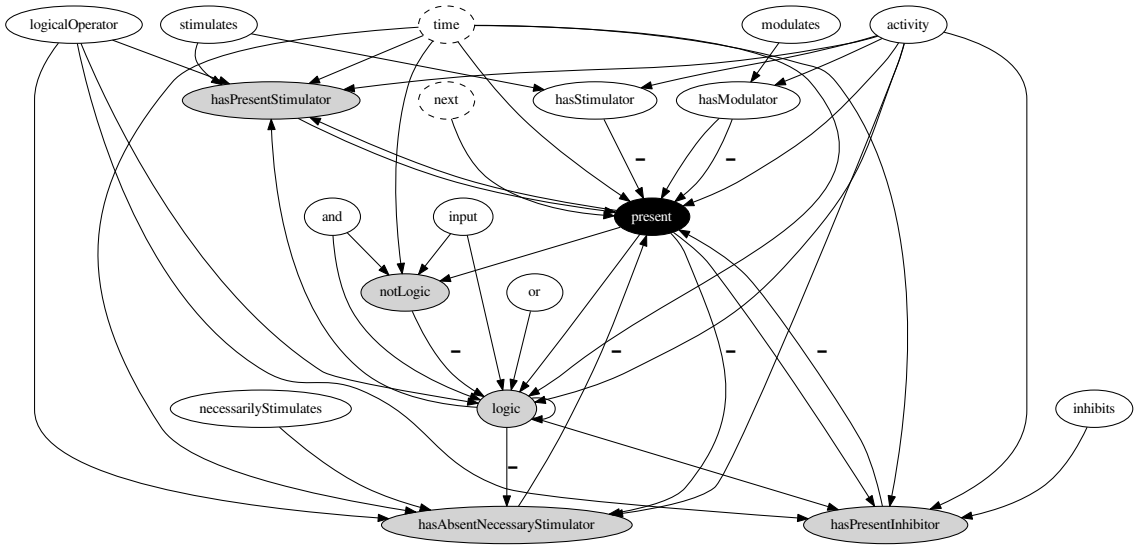


FIGURE 5.3 – **Graphe de dépendance des prédicats des programmes $\Pi_{Tr}(S, T_{max})$ et $\Pi_{At}(S)$.** Ce graphe montre la structure des programmes $\Pi_{Tr}(S, T_{max})$ et $\Pi_{At}(S)$. Les prédicats représentés en blancs sont les prédicats de type A, ceux en gris sont les prédicats de type B, et celui en noir est l'unique prédicat de type C. Les prédicats entourés de pointillés sont ceux apparaissant dans le programme $\Pi_{Tr}(S, T_{max})$ mais pas dans $\Pi_{At}(S)$.

En s'appuyant sur la définition des différents types de prédicats, nous différencions également les atomes, les littéraux, et les règles de nos deux programmes, en trois types. Un atome ou un littéral de *type A* (resp. *type B*, *type C*) est un atome construit à partir d'un prédicat de type A (resp. type B, type C). Une règle de type A (resp. type B, type C) est une règle dont la tête est de type A (resp. type B, type C). Selon ces définitions, les règles d'un des trois programmes qui sont des instances des axiomes (A1) ou (A2) sont de type A, les instances des axiomes (A3) à (A15) sont de type B, et les instances des axiomes (A16) à (A18) sont de type C.

Soit Π le programme $\Pi_{Tr}(S, T_{max})$ ou $\Pi_{At}(S)$. Nous simplifions le programme Π en appliquant les étapes suivantes :

- Étape 1 : appliquer itérativement, tant que c'est possible, les règles de simplification (SR1) à (SR4) aux règles de Π, pour obtenir le programme Π^1 ;
- Étape 2 : appliquer itérativement, tant que c'est possible, les règles de transformation (TR5) et (TR6) aux atomes de type B du corps des règles de type C de Π^1 pour obtenir Π^2 ;
- Étape 3 : supprimer toutes les règles de type A ou type B de Π^2 pour obtenir Π^f.

L'étape 1 permet d'éliminer de Π toutes les règles qui ont des atomes non supportés, et tous les atomes de type A du corps des règles, qui sont soit des instances de prédicats du domaine, soit des instances de prédicats auxiliaires.

Nous avons la propriété suivante sur les modèles supportés de Π^1 :

Propriété 5.2. Π et Π^1 ont exactement les mêmes modèles supportés.

Cette propriété découle immédiatement de la propriété 2.5.

L'étape 2 permet de remplacer les atomes de type B du corps des règles de Π_1 par leur définition. Comme tous les circuits du graphe de dépendance des atomes $G_{atom}(\Pi^1)$ de Π^1 qui contiennent un noeud associé à un atome de type B contiennent également un noeud associé à un atome de type C, tous les atomes de type B apparaissant dans le corps d'une règle sont *in fine* remplacés par des atomes de type C.

Nous avons la propriété suivante sur les modèles supportés de Π^2 :

Propriété 5.3. Les modèles supportés de Π^1 sont exactement les modèles supportés de Π^2.

Cette propriété découle immédiatement de la propriété 2.6.

Finalement, l'étape 3 permet de supprimer toutes les règles qui ne définissent pas un atome construit à partir du prédicat *present*. Nous avons la propriété suivante sur les modèles supportés de Π^f :

Propriété 5.4. Les modèles supportés de Π^2 restreints aux atomes construits à partir du prédicat *present* sont exactement les modèles supportés de Π^f.

Un sketch de preuve de cette propriété est donné dans l'annexe C.

Propriété 5.5. Les modèles supportés de Π restreints aux atomes construits à partir du prédicat *present* sont exactement les modèles supportés de Π^f.

Cette propriéré découle immédiatement des propriétés 5.2, 5.3 et 5.4.

En appliquant cette procédure de transformation aux programmes $\Pi_{Tr}(S, T_{max})$ et $\Pi_{At}(S)$, nous obtenons les deux programmes $\Pi_{Tr}(S, T_{max})^f$ et $\Pi_{At}(S)^f$, qui ne contiennent plus que le prédicat *present*.

Exemple 5.7. Soient S la carte et $\Pi(S)$ le NLP définis dans l'exemple 5.6. Le programme $\Pi_{At}(S)$ est obtenu de $\Pi(S)$ en supprimant la notion de temps de ces règles, et le programme $\Pi_{Tr}(S, T_{max})$ est obtenu à partir de $\Pi(S)$ et d'un entier T_{max} en ajoutant un ensemble d'atomes Π_T à $\Pi(S)$, qui définissent les pas de temps allant de 0 à T_{max}. Les programmes $\Pi_{At}(S)^1$ (resp. $\Pi_{Tr}(S, T_{max})^1$), $\Pi_{At}(S)^2$ (resp. $\Pi_{Tr}(S, T_{max})^2$) et $\Pi_{At}(S)^f$ (resp. $\Pi_{Tr}(S, T_{max})^f$) sont obtenus par simplifications successives de $\Pi_{At}(S)$ (resp. $\Pi_{Tr}(S, T_{max})$).

Nous donnons d'abord les programmes $\Pi_{At}(S)^1$ et $\Pi_{At}(S)^f$.

- $\Pi_{At}(S)^1$ est défini par l'ensemble des règles suivantes :

$$\begin{aligned}
\{hasModulator(a_1) \leftarrow,\\
hasModulator(a_4) \leftarrow,\\
hasStimulator(a_4) \leftarrow,\\
hasPresentStimulator(a_4) \leftarrow logic(lo_2),\\
hasPresentInhibitor(a_1) \leftarrow present(a_4),\\
logic(lo_1) \leftarrow present(a_1),\\
logic(lo_1) \leftarrow present(a_2),\\
notLogic(lo_2) \leftarrow \neg logic(lo_1),\\
notLogic(lo_2) \leftarrow \neg present(a_3),\\
logic(lo_2) \leftarrow \neg notLogic(lo_2),\\
present(a_1) \leftarrow \neg hasPresentInhibitor(a_1),\\
present(a_2) \leftarrow present(a_2),\\
present(a_3) \leftarrow present(a_3),\\
present(a_4) \leftarrow hasPresentStimulator(a_4)\}
\end{aligned}$$

- $\Pi_{At}(S)^f$ est défini par l'ensemble des règles suivantes :

$$\begin{aligned}
\{present(a_1) \leftarrow \neg present(a_4),\\
present(a_2) \leftarrow present(a_2),\\
present(a_3) \leftarrow present(a_3),\\
present(a_4) \leftarrow present(a_1) \wedge present(a_3),\\
present(a_4) \leftarrow present(a_2) \wedge present(a_3)\}
\end{aligned}$$

Nous donnons ensuite les programmes $\Pi_T(T_{max})$, $\Pi_{Tr}(S, T_{max})^1$ et $\Pi_{Tr}(S, T_{max})^f$ pour $T_{max} = 4$.

- L'ensemble d'atomes formalisant les pas de temps pour $T_{max} = 4$ est l'ensemble d'atomes suivant :

$$\Pi_T = \{time(0), time(1), time(2), time(3), time(4), next(1,0), next(2,1), next(3,2), next(4,3)\}$$

- $\Pi_{Tr}(S, 4)^1$ est défini par l'union des deux ensembles de règles suivants :

$$\begin{aligned}
\{hasModulator(a_1) \leftarrow,\\
hasModulator(a_4) \leftarrow,\\
hasStimulator(a_4) \leftarrow\}
\end{aligned}$$

et

$$\{hasPresentStimulator(a_4, t+1) \leftarrow logic(lo_2, t),$$
$$hasPresentInhibitor(a_1, t+1) \leftarrow present(a_4, t),$$
$$logic(lo_1, t+1) \leftarrow present(a_1, t),$$
$$logic(lo_1, t+1) \leftarrow present(a_2, t),$$
$$notLogic(lo_2, t+1) \leftarrow \neg logic(lo_1, t),$$
$$notLogic(lo_2, t+1) \leftarrow \neg present(a_3, t),$$
$$logic(lo_2, t+1) \leftarrow \neg notLogic(lo_2, t),$$
$$present(a_1, t+1) \leftarrow \neg hasPresentInhibitor(a_1, t),$$
$$present(a_2, t+1) \leftarrow present(a_2, t),$$
$$present(a_3, t+1) \leftarrow present(a_3, t),$$
$$present(a_4, t+1) \leftarrow hasPresentStimulator(a_4, t) \mid 0 \leq t \leq 3\}$$

- $\Pi_{Tr}(S, 4)^f$ est défini par l'ensemble des règles suivantes :

$$\{present(a_1, t+1) \leftarrow \neg present(a_4, t),$$
$$present(a_2, t+1) \leftarrow present(a_2, t),$$
$$present(a_3, t+1) \leftarrow present(a_3, t),$$
$$present(a_4, t+1) \leftarrow present(a_1, t) \wedge present(a_3, t),$$
$$present(a_4, t+1) \leftarrow present(a_2, t) \wedge present(a_3, t) \mid 0 \leq t \leq 3\}$$

5.5.3 Des programmes logiques transformés aux réseaux Booléens

Nous transformons finalement le programme $\Pi_{At}(S)$ en un RB, de la manière suivante :

- Étape 5 : Nous remplaçons dans $\Pi_{At}(S)^f$ chaque atome de la forme $present(a_i)$ par une variable propositionnelle v_i, pour obtenir un NLP propositionnel $P_{At}(S)^f$.
- Étape 6 : Nous traduisons le NLP propositionnel $P_{At}(S)^f$ en un réseau Booléen, dénoté par $B(P_{At}(S)^f)$, suivant la méthode introduite dans [Ino11] et présentée dans la section 5.3.

Nous avons le théorème suivant :

Théorème 5.5. Soit S une carte SBGN-AF, $B(S)$ le RB construit à partir de S avec les principes généraux (B1–7). Soit $\Pi_{At}(S)$ le programme défini comme précédemment, et $B(P_{At}(S)^f)$ le RB obtenu en appliquant les étapes de transformation 1–6 à $\Pi_{At}(S)$. Alors le RB $B(P_{At}(S)^f)$ est précisément le RB $B(S)$.

La preuve de ce théorème est donnée dans l'annexe C.

Exemple 5.8. Soit S la carte de l'exemple 5.6, $\Pi_{At}(S)$ le programme comme défini à l'exemple 5.7, et $B(S) = (V, F)$ le réseau Booléen obtenu à partir de S en considérant les principes généraux (B1–7),

et défini dans l'exemple 5.6. Nous rappelons que $B(S)$ est défini par

$$V = \{v_1, v_2, v_3, v_4\}$$

et

$$F = \{f_1, f_2, f_3, f_4\}$$

où

$$f_1(v_1, v_2, v_3, v_4) = \neg v_4 \ ; \ f_2(v_1, v_2, v_3, v_4) = v_2 \ ;$$
$$f_3(v_1, v_2, v_3, v_4) = v_3 \ ; \text{ et } f_4(v_1, v_2, v_3, v_4) = (v_1 \wedge v_3) \vee (v_2 \wedge v_3).$$

Soit $\Pi_{At}(S)^f$ le programme obtenu en appliquant les étapes de transformation 1–4 à $\Pi_{At}(S)$, et défini dans l'exemple 5.7. Le NLP propositionnel $P_{At}(S)$, obtenu en appliquant l'étape de transformation 5 à $\Pi_{At}(S)^f$, est défini par l'ensemble de règles suivantes :

$$\{v_1 \leftarrow \neg v_4,$$
$$v_2 \leftarrow v_2,$$
$$v_3 \leftarrow v_3,$$
$$v_4 \leftarrow v_1 \wedge v_3,$$
$$v_4 \leftarrow v_2 \wedge v_3\}$$

Le réseau Booléen $B(P_{At}(S)^f) = (V', F')$, obtenu par traduction de $P_{At}(S)^f$ selon l'étape de transformation 6, est défini par :

$$V = \{v_1, v_2, v_3, v_4\}$$

et

$$F = \{f_1, f_2, f_3, f_4\}$$

où

$$f_1(v_1, v_2, v_3, v_4) = \neg v_4 \ ; \ f_2(v_1, v_2, v_3, v_4) = v_2 \ ;$$
$$f_3(v_1, v_2, v_3, v_4) = v_3 \ ; \text{et } f_4(v_1, v_2, v_3, v_4) = (v_1 \wedge v_3) \vee (v_2 \wedge v_3).$$

$B(P_{At}(S)^f)$ est bien le RB $B(S)$.

Des propriétés 5.5 et des théorèmes 5.5, 5.3 et 5.4, nous pouvons déduire les propositions données ci-après, qui relient les points attracteurs de $B(S)$ et les traces de sa dynamique synchrone aux modèles supportés de $\Pi_{At}(S)$ et de $\Pi_{Tr}(S, T_{max})$, respectivement.

Pour une interprétation de Herbrand I de $\Pi_{At}(S)$, nous dénotons par $S(I)$ l'état global de $B(S)$ correspondant à I :

$$S(I) = (b_1, \ldots, b_n) \text{ où } \begin{cases} b_i = 1 \text{ si } present(a_i) \in I; \\ b_i = 0 \text{ sinon.} \end{cases}$$

Pour une interprétation de Herbrand I de $\Pi_{Tr}(S, T_{max})$ et un entier t, nous dénotons par $S_t(I)$ l'état global de $B(S)$ correspondant à I au temps t :

$$S_t(I) = (b_1, \ldots, b_n) \text{ où } \begin{cases} b_i = 1 \text{ si } present(a_i, t) \in I; \\ b_i = 0 \text{ sinon.} \end{cases}$$

De manière inverse, pour un état global s de $B(S)$ et un entier t, nous dénotons par $I_t(s)$ l'interprétation de Herbrand de $\Pi_{Tr}(S, T_{max})$ correspondant à s au temps t, et restreinte au prédicat *present* :

$$I_t(s) = \{present(a_i, t) \mid v_i(s) = 1\}$$

Les deux propositions qui nous intéressent sont les suivantes :

Proposition 5.1. Soit S une carte SBGN-AF, $\Pi_{At}(S)$ le programme défini à partir de S comme précédemment, et $B(S)$ le RB construit à partir de S avec les principes généraux (B1–7). Soit M une interprétation de Herbrand de $\Pi_{At}(S)$. Alors M est un modèle supporté de $\Pi_{At}(S)$ *ssi* $S(M)$ est un point attracteur de $B(S)$.

Proposition 5.2. Soit S une carte SBGN-AF, T_{max} un entier positif non nul, $\Pi_{Tr}(S, T_{max})$ le programme défini à partir de S et T_{max} comme précédemment, et $B(S)$ le RB construit à partir de S avec les principes généraux (B1–7). Soit s un état global de $B(S)$, et M l'unique modèle supporté de $\Pi_{Tr}(S, T_{max}) \cup I_0(s)$. Alors $s \to_{sy} S_1(M) \to_{sy} \cdots \to_{sy} S_{T_{max}}(M)$ est l'unique trace finie de la dynamique synchrone de $B(S)$ partant de s et arrivant sur $S_{T_{max}}(M)$.

Les preuves de ces deux propositions sont données dans l'annexe C.

En conséquence de ces deux propsoitions, nous pouvons calculer tous les points attracteurs de $B(S)$ en calculant les modèles supportés de $\Pi_{At}(S)$, et toutes les traces finies de la dynamique synchrone de $B(S)$ partant d'un état global s en calculant l'unique modèle supporté de $\Pi_{Tr}(S, T_{max}) \cup I_0(s)$, qui est également son unique modèle stable.

Le résumé des différentes transformations et de ces résultats est donné dans la figure 6.15.

Exemple 5.9 (Calcul des points attracteurs à partir des modèles supportés)**.** Soit S la carte de l'exemple 5.6, $\Pi_{At}(S)$ le programme défini à l'exemple 5.7, et $B(S)$ le RB construit à partir de S en considérant les principes généraux (B1–7).

Le programme $\Pi_{At}(S)$ a trois modèles supportés, que nous dénotons M_1, M_2 et M_3. Ces modèles supportés, restreints aux atomes construits à partir du symbole de prédicat *present*, sont les ensembles d'atomes suivants :

$$M_1 = \{present(a_1)\}$$
$$M_2 = \{present(a_1), present(a_2)\}$$
$$M_3 = \{present(a_2), present(a_3), present(a_4)\}$$

Le RB $B(S)$ a trois point attracteurs, que nous dénotons s_1, s_2 et s_3 :

$$s_1 = (1, 0, 0, 0)$$
$$s_2 = (1, 1, 0, 0)$$
$$s_3 = (0, 1, 1, 1)$$

Nous remarquons que $S(M_1) = s_1$, $S(M_2) = s_2$, et $S(M_3) = s_3$.

Exemple 5.10 (Calcul des traces finies à partir des orbites finies)**.** Soit S la carte de l'exemple 5.6. Soit $\Pi_{Tr}(S, 4)$ le programme $\Pi_{Tr}(S, T_{max})$ défini pour un entier $T_{max} = 4$, comme à l'exemple 5.7. Soit $B(S)$ le RB construit à partir de S en considérant les principes généraux (B1–7), et $s = (1, 0, 1, 0)$ un état global de $B(S)$.

L'interprétation $I_0(s)$ de $\Pi_{Tr}(S, 4)$ correspondant à s est l'ensemble d'atomes suivant :

$$I_0(s) = \{present(a_1, 0), present(a_3, 0)\}$$

L'unique modèle supporté M de $\Pi_{Tr}(S, 4) \cup I_0(s)$, restreint au prédicat *present*, est l'ensemble d'atomes suivant :

$$\{present(a_1, 0), present(a_3, 0),$$
$$present(a_1, 1), present(a_3, 1), present(a_4, 1),$$
$$present(a_3, 2), present(a_4, 2),$$
$$present(a_3, 3),$$
$$present(a_1, 4), present(a_3, 4)\}$$

La trace de longueur quatre de la dynamique synchrone de $B(S)$ partant de s est la série de transitions suivante :

$$(1, 0, 1, 0) \rightarrow_{sy} (1, 0, 1, 1) \rightarrow_{sy} (0, 0, 1, 1) \rightarrow_{sy} (0, 0, 1, 0) \rightarrow_{sy} (1, 0, 1, 0)$$

et nous remarquons que c'est exactement la trace définie par :

$$s \rightarrow_{sy} S_1(M) \rightarrow_{sy} S_2(M) \rightarrow_{sy} S_3(M) \rightarrow_{sy} S_4(M)$$

Le calcul des modèles supportés de $\Pi_{At}(S)$, ainsi que celui de l'unique modèle supporté de $\Pi_{Tr}(S, T_{max})$, peuvent se faire grâce au logiciel clingo.

Nous avons donc proposé une méthode permettant de calculer les points attracteurs et les traces finies d'un RB construit à partir d'une carte SBGN-AF et de principes généraux, à l'aide de deux NLP du premier ordre. Pour une carte SBGN-AF donnée, ces NLP sont formés d'un ensemble d'axiomes formalisant les principes généraux d'une part, et de la traduction en SBGNLog-AF de cette carte d'autre part.

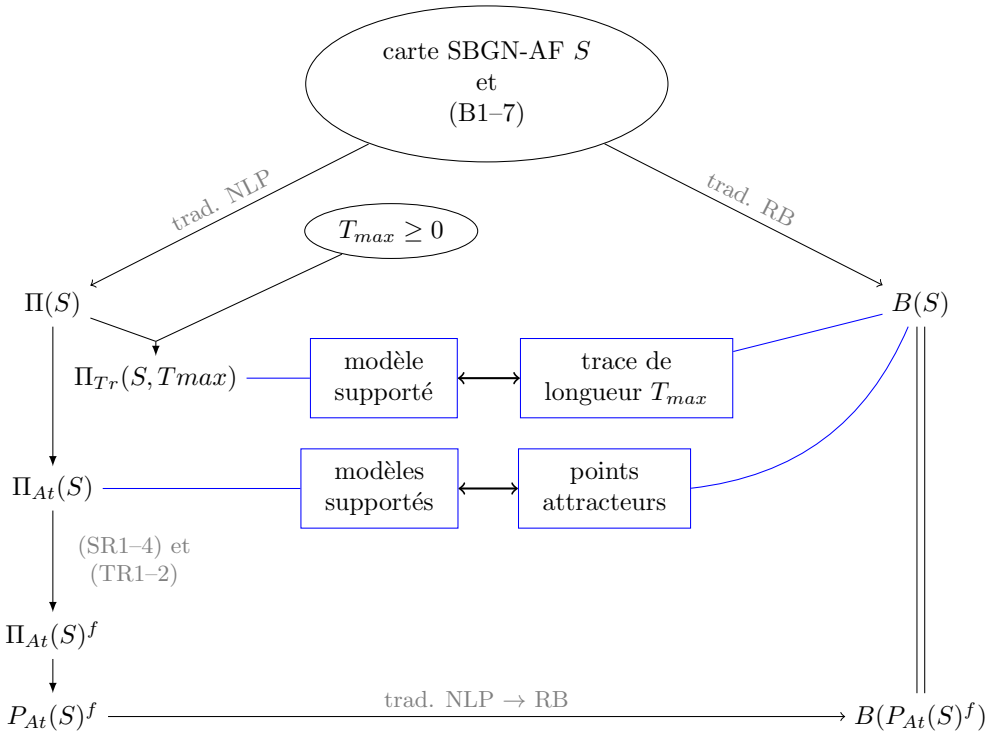


FIGURE 5.4 – **Résumé de la méthode de calcul des points attracteurs et des traces finies de la sémantique Booléenne des cartes SBGN-AF à l'aide de NLPs du premier ordre.** Les flèches (→) correspondent à des traductions, les flèches (➝) à des transformations. Finalement, la flèche de symbole (=) représente l'égalité. Le NLP du premier ordre $\Pi(S)$ et le RB $B(S)$ sont construits à partir de la carte S et des principes généraux (B1–7). Le NLP $\Pi(S)$ est transformé en deux NLP du premier ordre $\Pi_{At}(S)$ et $\Pi_{Tr}(S, T_{max})$. Les modèles supportés de $\Pi_{At}(S)$ correspondent aux points attracteurs de $B(S)$, et le modèle supporté de $\Pi_{Tr}(S, T_{max})$ correspond à une trace de longueur T_{max}. Ces résultats sont notamment démontrés par la transformation du NLP $\Pi_{At}(S)$ en un NLP propositionnel $P_{At}(S)$, qui peut lui-même être traduit en un RB $B(P_{At}(S))$. Nous avons montré que ce dernier était exactement le RB $B(S)$.

5.6 Discussion

5.6.1 Travaux connexes

De nombreux travaux utilisent la programmation logique, et en particulier ASP, pour analyser des réseaux moléculaires. Nous présentons ici quelques travaux qui analysent la dynamique des graphes d'influences à l'aide d'une sémantique Booléenne, ou directement la dynamique de réseaux Booléens, à l'aide de la programmation logique.

Le travail présenté dans ce chapitre fait suite aux travaux réalisés par Katsumi Inoue [Ino11]. Nous étendons le lien entre les réseaux Booléens (RB) et les programmes logiques normaux (NLP) propositionnels présentés dans [Ino11] aux NLP du premier ordre. Nous ne revenons pas ici sur les liens avec ces travaux, que nous avons déjà montrés dans ce chapitre.

Dans [Roc+14], les auteurs proposent une méthode de vérification des réseaux Booléens contre des formules de la logique temporelle, exprimée en ASP. Étant donné un RB, ils l'encodent en ASP en faisant correspondre à chaque fonction Booléenne du RB un ensemble de règles logiques. Ces règles logiques permettent de calculer le graphe des transitions asynchrones du RB, à partir desquelles des

formules de la logique temporelle, elles aussi encodées en ASP, peuvent être vérifiées. Contrairement à notre méthode, les auteurs de [Roc+14] prennent en entrée un réseau Booléen, qui est par nature déjà paramétré, là où nous considérons un graphe d'influences. Ainsi, même si leurs règles logiques encodant un RB sont au premier ordre, elles sont spécifiques à ce RB, et n'encodent pas des principes généraux, au contraire des axiomes que nous avons introduits. Ce travail se rapproche donc davantage des travaux de [Ino11] que des nôtres, dans le sens où le programme ASP construit est spécifique au réseau Booléen d'entrée.

D'autres travaux, utilisant la programmation logique pour analyser des RB, prennent en entrée des graphes d'influences.

Dans [Vid+12], les auteurs proposent une méthode permettant de paramétrer des réseaux Booléens construits à partir de graphes d'influences, à l'aide de mesures de changements d'activités entre un état initial et un état final stationnaire. Étant donné un graphe d'influences ne comportant pas de cycles, ils génèrent tous les réseaux Booléens modélisant ce graphe, i.e. toutes les paramétrisations possibles. Pour chaque RB ainsi créé et chaque état initial des mesures de changements d'activités, ils calculent ensuite le point attracteur atteint par le RB. Pour un état initial donné, celui-ci est unique étant donné que le graphe d'influences de départ ne contient pas de cycle. Enfin, pour chaque RB, ils confrontent les points attracteurs calculés aux états stationnaires mesurés, et calculent un score qui est d'autant plus mauvais que les points attracteurs calculés sont éloignés des états stationnaires mesurés. Ainsi, ils assignent à chaque RB issu du graphe d'influence un score, et sélectionnent le ou les RBs ayant le meilleur score. Tout ce workflow est réalisé grâce à un unique programme ASP : d'abord, le graphe d'influences d'entrée ainsi que les différents couples d'états initiaux et finaux sont encodés sous forme de faits. Puis tous les RB correspondant à une certaine paramétrisation de ce graphe d'influences sont générés grâce à une règle contenant un agrégateur dans sa tête. Ensuite, pour chaque RB généré et chaque état initial, son point attracteur est calculé grâce à des règles de propopagation. Étant donnée une variable du RB, la première règle modélise la satisfaction de la fonction Booléenne associée à cette variable, et l'autre sa falsification. Enfin, le point attracteur obtenu est comparé à l'état final mesuré, et l'écart entre les deux participe à un score qui est associé au RB. Grâce à la règle de génération des RBs, chaque modèle stable de ce programme correspond à un RB, un point attracteur, et un score. Seuls les modèles minimisant le score sont optimaux, et correspondent à des paramétrisations vraisemblables. Ces travaux sont liés aux nôtres de deux points de vue. D'abord, ils proposent également une méthode de paramétrisation des réseaux Booléens construits à partir de graphes d'influences : alors que nous utilisons des principes généraux qui amènent à une unique paramétrisation, les auteurs de [Vid+12] confrontent différentes paramétrisations possibles à des résultats expérimentaux, et choisissent celles qui paraissent les plus vraisemblables. Cette méthode de paramétrisation donne sans aucun doute des modèles plus réalistes, mais nécessitent l'analyse de résultats expérimentaux, qui sont parfois difficiles à obtenir. Ensuite, leur méthode passe par le calcul des points attracteurs d'un RB. Les règles de propagation qu'ils utilisent sont générales, i.e. elles ne dépendent pas du RB, contrairement aux règles de [Roc+14]. Cependant, elles sont à différencier de nos axiomes, étant donné qu'elles sont construites à partir des prédicats utilisés pour décrire la structure d'un réseau Booléen, et non d'un graphe d'influences.

Les travaux se rapprochant le plus des nôtres ont été publiés dans [Fay+11]. Les auteurs de cet article proposent de calculer la dynamique Booléenne synchrone des graphes d'influences à l'aide d'un programme ASP. Dans leur méthode, les règles ASP décrivant la dynamique Booléenne formalisent également un ensemble de principes généraux. Tout comme nous, ils se proposent donc de paramétrer des RB construits à partir de graphes d'influences à l'aide de principes généraux formalisés sous la forme d'axiomes, et d'en calculer la dynamique. Ils construisent pour ce faire un programme ASP similaire, dans sa forme, au programme $\Pi(S)$ que nous avons présenté dans la section précédente. Si leur programme ASP et notre programme logique sont proches, leur méthode et la nôtre comportent cependant des différences. Outre la différence entre les principes qu'ils considèrent et ceux que nous avons

donnés à la sous-section 5.4.2, nos travaux se distinguent principalement sur deux points. D'abord, les auteurs de [Fay+11] prennent en entrée des graphes d'influences ne comportant que des noeuds et des influences positives ou négatives. En considérant, de notre côté, des graphes d'influences SBGN-AF, nous prenons également en compte les influences positives et négatives, mais aussi les stimulations nécessaires, et surtout les opérateurs logiques. Ainsi, nos axiomes prennent en compte des graphes d'influences qui sont potentiellement déjà partiellement paramétrés. La seconde différence notable est que leur méthode pour le calcul des points attracteurs nécessite de calculer l'ensemble des traces finies du RB d'une longueur donnée en paramètre. Ainsi, dans leur méthode, les points attracteurs du RB ne peuvent être calculés qu'en analysant *a posteriori* les traces finies de ce RB. Or cette manière de calculer les points attracteurs n'est, dans le cas général, pas complète. Il n'y a en effet *a priori* pas moyen de connaître la longueur de trace requise pour que l'ensemble des points attracteurs d'un RB apparaissent dans les traces finies calculées. Quant à notre méthode, elle permet le calcul de l'ensemble des points attracteurs du RB sans avoir à calculer les traces de ce dernier. Cela est rendu possible par l'utilisation de la sémantique des modèles supportés.

5.6.2 Calcul des points attracteurs et des traces finies de la dynamique asynchrone

Étant donné un réseau Boléen, les points attracteurs de la dynamique asynchrone sont exactement les mêmes que ceux de la dynamique synchrone, et peuvent donc être calculés avec notre méthode.

Ce n'est pas le cas, en général, pour les traces : les traces de la dynamique synchrone sont différentes des traces de la dynamique asynchrone. Comme présenté dans [Ino11], les traces finies de la dynamique asynchrone d'un RB peuvent également être calculées grâce à sa traduction en programme logique. Cette traduction n'est cependant pas exactement la même que celle présentée dans la section 5.3, et nécessite l'ajout de règles dont la syntaxe n'appartient pas à celle des NLP, mais plutôt à celle des programmes ASP.

Dans la dynamique asynchrone, une seule fonction Booléenne est appliquée à la fois, contrairement à la dynamique synchrone où toutes les fonctions sont appliquées à la fois, ou encore à la dynamique générale, où un nombre quelconque de fonctions peuvent être appliquées à la fois. Nous nous concentrons ici sur la dynamique asynchrone, mais le calcul des traces finies de la dynamique générale suit le même principe.

Comme nous l'avons dit, dans la dynamique asynchrone, une seule fonction Booléenne est appliquée à la fois. Il faut donc choisir, à chaque pas de temps, quelle fonction appliquer ; puis l'appliquer, sans appliquer les autres. En ASP, le choix de la fonction à appliquer peut être encodé directement sous la forme d'une règle de choix. Nous donnons le détail de cet encodage dans l'annexe C.

5.7 Conclusion

Dans ce chapitre, nous avons d'abord montré comment un RB construit à partir d'une carte SBGN-AF pouvait être paramétré à l'aide d'un ensemble de principes généraux, que nous avons préalablement choisis. Nous avons ensuite montré, de manière formelle, comment la dynamique de ce RB peut être calculée à l'aide de deux NLP du premier ordre. Ces deux NLP sont formés d'axiomes formalisant les principes généraux d'une part, et de la traduction SBGNLog-AF de la carte modélisée d'autre part. Le premier de ces NLP permet de calculer les points attracteurs du RB, tandis que le deuxième permet de calculer les traces finies de la dynamique synchrone. Ces propriétés sont calculées à partir de ces deux NLP en calculant leurs modèles supportés. Cette méthode peut être mise en pratique grâce au logiciel ASP clingo, qui permet le calcul des modèles supportés d'un NLP propositionnel.

Chapitre 6

Sémantiques qualitatives pour l'analyse de la dynamique des réseaux de réactions SBGN-PD

Sommaire

6.1 Introduction

Une sémantique Booléenne pour modéliser les réseaux de réactions, dite sémantique Booléenne de Biocham, a été introduite par les auteurs de [CFS06] (voir le chapitre 1). Comme sa contrepartie pour

les graphes d'influences, les modèles construits avec cette sémantique ne nécessitent pas de paramètres cinétiques. Avec cette sémantique, chaque molécule du réseau peut être soit absente soit présente, et chaque réaction est modélisée par un ensemble de transitions Booléennes. Ces transitions modélisent notamment la production des produits d'une réaction, et la consommation éventuelle de ses réactifs.

La sémantique Booléenne de BIOCHAM est définie pour des réseaux de réactions constitués de processus moléculaires (comme les réactions chimiques, les associations/dissociations et les translocations) éventellement catalysés. De ce fait, cette sémantique ne prend pas en compte différents éléments pouvant être trouvés dans les réseaux de réactions exprimés en SBGN-PD, comme les inhibitions, qui jouent un rôle important notamment dans les processus de signalisation.

De manière générale, jusqu'à présent, aucune sémantique qualitative prenant en compte les éléments principaux des cartes SBGN-PD n'avait été proposée. C'est pourquoi nous introduisons deux nouvelles sémantiques qualitatives, nommément la *sémantiques générale* et la *sémantiques des histoires* (*stories semantics* en anglais) qui prennent en compte les principaux éléments des cartes SBGN-PD.

La sémantique générale étend la sémantique Booléenne de BIOCHAM en prenant en compte les inhibitions et les opérateurs logiques. Quant à la sémantique des histoires, elle repose sur une interprétation différente des réseaux de réactions. Elle permet de modéliser les différents états physiques (p. ex. état non modifié, état phosphorylé) des entités moléculaires plutôt que les molécules elles-mêmes. En interprétant un réseau de réactions avec cette sémantique, les états physiques d'une même entité moléculaire sont regroupés en un même ensemble abstrait appelé *histoire*, qui est modélisé par une unique variable. Cette sémantique conduit à la construction de modèles qui sont plus facilement compréhensibles d'un point de vue biologique et plus proches de la manière dont les experts perçoivent les processus biologiques, et qui conservent cependant le détail des mécanismes moléculaires entrant en jeu dans les réseaux de réactions. Les modèles construits avec ces deux sémantiques sont formalisés à l'aide de réseaux d'automates.

Le reste de ce chapitre est organisé comme suit. Nous donnons d'abord quelques définitions relatives aux réseaux d'automates. Puis nous introduisons nos deux sémantiques, ainsi que la formalisation des modèles obtenus à l'aide de réseaux d'automates. Nous donnons ensuite la relation formelle entre les dynamiques obtenues avec chacune des deux sémantiques. Ensuite, nous illustrons comment ces deux sémantiques permettent de modéliser la dynamique des réseaux de réactions en construisant et en analysant des modèles dynamiques de la régulation du cycle cellulaire par RB/E2F. Enfin, nous discutons de la relation du concept d'histoire avec d'autres notions existantes, et de l'applicabilité de nos deux sémantiques à divers types de réseaux moléculaires.

Ce travail a été réalisé en collaboration avec Laurence Calzone (Institut Curie) et Loïc Paulevé (équipe Bioinformatique du LRI), et a été publié dans [Rou+16].

6.2 Réseaux d'automates asynchrones : définitions

Un *réseau d'automates* (RA) est défini comme un ensemble d'automates, chacun formé d'un ensemble fini d'*états locaux* et d'un ensemble de *transitions locales* conditionnées par les états locaux des autres automates du réseau. Les états locaux d'un automate du réseau sont exclusifs, i.e., à chaque instant, un automate ne peut être que dans un seul de ses états locaux. L'état dans lequel est un automate à un instant t est appelé l'état *actif* de cet automate à l'instant t. L'état global d'un réseau d'automates à un instant t est donné par l'ensemble des états actifs à l'instant t des automates qui le constituent.

Plus formellement, voici la définition d'un réseau d'automates :

Définition 6.1 (Réseau d'automates)**.** Un réseau d'automates (RA) est défini par un triplet (Σ, S, T) où :

- Σ est un ensemble fini d'automates ;
- Pour chaque $\mathtt{a} \in \Sigma$, $S(\mathtt{a}) \triangleq \{\mathtt{a_i}, \cdots, \mathtt{a_j}\}$ est l'ensemble fini des états locaux de l'automate $\mathtt{a}$. Nous dénotons par $S = \prod_{\mathtt{a}\in\Sigma} S(\mathtt{a})$ l'ensemble des états globaux du RA.
- $T \subseteq \{\mathtt{a_i} \xrightarrow{\ell} \mathtt{a_j} \mid \mathtt{a} \in \Sigma, \mathtt{a_i} \in S(\mathtt{a}), \mathtt{a_j} \in S(\mathtt{a}), \ell \subset \bigcup_{\mathtt{b}\in\Sigma,\mathtt{b}\neq\mathtt{a}} S(\mathtt{b})\}$ est l'ensemble fini des transitions locales.

Exemple 6.1 (Réseau d'automates)**.** La figure 6.1 donne un exemple de RA. Ce RA est défini par le triplet (Σ, S, T) suivant :

$$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$$
$$S(\mathtt{a}) = \{\mathtt{a_0}, \mathtt{a_1}, \mathtt{a_2}\}$$
$$S(\mathtt{b}) = \{\mathtt{b_0}, \mathtt{b_1}\}$$
$$S(\mathtt{c}) = \{\mathtt{c_0}, \mathtt{c_1}\}$$
$$T = \{\mathtt{a_0} \xrightarrow{\{\mathtt{b_1}\}} \mathtt{a_1}, \mathtt{a_1} \xrightarrow{\{\mathtt{c_1}\}} \mathtt{a_2},$$
$$\mathtt{b_1} \xrightarrow{\{\mathtt{a_0},\mathtt{c_0}\}} \mathtt{b_0}, \mathtt{c_0} \xrightarrow{\{\mathtt{a_1}\}} \mathtt{c_1}\}$$

Un exemple d'état global de cet automate est l'état $(\mathtt{a_0}, \mathtt{b_1}, \mathtt{c_1})$.

Un état global d'un RA, qui est défini comme un tuple, pourra également être noté comme un ensemble d'états locaux, pour plus de commodité.

Étant donné un état global $s \in S$ d'un RA, il y a une transition de cet état vers un autre état global $s' \in S$ *ssi* il existe une transition locale $\mathtt{a_i} \xrightarrow{\ell} \mathtt{a_j} \in T$ telle que l'automate a est dans l'état a_i dans s, tous les états locaux de ℓ sont présents dans s, et s' est l'état s dans lequel a_i a été remplacé par a_j. Une telle transition globale correspond à une dynamique *asynchrone* : une seule transition locale est appliquée à la fois. Comme, depuis un état s, il est possible que plusieurs transitions locales puissent être appliquées, la dynamique est non-déterministe.

Voici la définition formelle de la relation de transition asynchrone :

Définition 6.2 (Relation de transition asynchrone)**.** Étant donné un RA (Σ, S, T), la relation de transition asynchrone, notée $\rightarrow$, est incluse dans $S \times S$ et définie de la façon suivante :

$$s \rightarrow s' \triangleq \exists \mathtt{a_i} \xrightarrow{\ell} \mathtt{a_j} \in T : \mathtt{a_i} \in s, \ell \subset s, s' = s \cup \{\mathtt{a_j}\} \setminus \{\mathtt{a_i}\}$$

Dans le cadre de cette étude, nous considérerons le mode de mise à jour dit *asynchrone*. Ce mode de mise à jour est le mode le mieux intégré dans les différents logiciels que nous utiliserons par la suite, et il présente des avantages pour la modélisation des systèmes biologiques comparé au mode de mise à jour synchrone, par exemple (voir [HB97] pour de plus amples détails).

Comme pour les réseaux Booléens au chapitre 5, la relation de transition asynchrone permet de définir les notions de trace et d'atteignabilité. Une trace est une succession de transitions asynchrones,

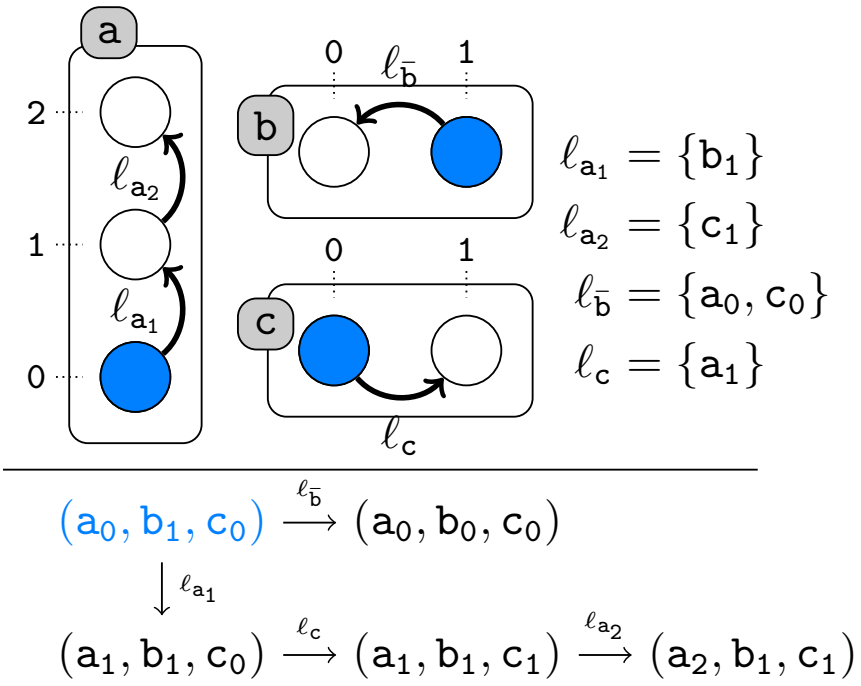


FIGURE 6.1 – **Un exemple de réseau d'automates et son graphe des transitions asynchrones construit à partir d'un état initial.** En haut : un réseau d'automates asynchrone composé de trois automates $\mathtt{a}$, $\mathtt{b}$ et $\mathtt{c}$. Les automates sont représentés par des boîtes étiquetées, et les états locaux par des cercles identifiés par des étiquettes. Par exemple, le cercle étiqueté par $\mathtt{1}$ de l'automate $\mathtt{a}$ est l'état local $\mathtt{a}_1$ de l'automate $\mathtt{a}$. Les transitions locales sont représentées par des arcs étiquetés. Une étiquette d'une transition locale représente l'ensemble des conditions qui doivent être satisfaites pour que la transition puisse se déclencher. Les états locaux représentés en bleu représentent des états locaux initiaux potentiels pour les différents automates du réseau. L'ensemble des ces états locaux initiaux constitue l'état global initial potentiel dénoté par $(\mathtt{a}_0, \mathtt{b}_1, \mathtt{c}_0)$. En bas : le graphe des transitions asynchrones du réseau d'automates représenté en haut, construit à partir d'un état global initial, repésenté en bleu. Ce graphe représente l'ensemble des transitions asynchrones qui peuvent être déclenchées successivement à partir de cet état initial global. Par exemple, à partir de l'état initial, il est possible de déclencher la transition local étiquetée par $\ell_{\mathtt{a}_1}$ ou la transition étiquétée par $\ell_{\bar{\mathtt{b}}}$. À partir de cet état, l'une de ces deux transitions sera déclenchée, de façon non-déterministe. Déclencher la transition $\ell_{\mathtt{a}_1}$ changera l'état de $\mathtt{a}$ de $\mathtt{a}_0$ à $\mathtt{a}_1$, faisant passer le réseau dans l'état global $(\mathtt{a}_1, \mathtt{b}_1, \mathtt{c}_0)$, et déclencher la transition $\ell_{\bar{\mathtt{b}}}$ changera l'état de $\mathtt{b}$ de $\mathtt{b}_1$ à $\mathtt{b}_0$, faisant passer le réseau dans l'état global $(\mathtt{a}_0, \mathtt{b}_0, \mathtt{c}_0)$.

et un état est atteignable à partir d'un autre état *ssi* il existe une trace joignant ces deux états. Plus formellement :

Définition 6.3 (Trace d'un RA)**.** Une *trace* d'un RA est une série, finie ou infinie, de transitions asynchrones de ce RA. Une telle trace est souvent dénotée par une séquence $s_1 \rightarrow s_2 \rightarrow \ldots$ de transitions globales, telle que les s_i $(i \geq 1)$ sont des états globaux du RA.

Exemple 6.2 (Traces d'un RA à partir d'un état global initial)**.** L'ensemble des traces d'un RA (possiblement à partir d'un état global initial donné) peut être représenté par un graphe dit *graphe de transitions*. La figure 6.1 montre un exemple de RA et son graphe de transition, à partir d'un état global initial représenté en bleu sur la figure. De chaque état global du RA, une seule transition locale est appliquée à la fois, le mode de mise à jour étant asynchrone. À partir de l'état initial, deux transitions peuvent être appliquées, ce qui montre le non-déterminisme de la sémantique.

Définition 6.4 (Atteignabilité d'un état)**.** Soient deux états globaux s et s' d'un RA. L'état s' est *atteignable* à partir de l'état s, noté $s \rightarrow^* s'$ *ssi* $s \rightarrow s'$ ou il existe un état $s'' \in S$ tel que $s \rightarrow^* s''$ et $s'' \rightarrow s'$. Par convention, $s \rightarrow^* s$.

6.3 La sémantique générale des réseaux SBGN-PD

Nous introduisons dans cette section la sémantique dite *générale* (trad. de l'anglais *general semantics*). Cette sémantique étend celle de BIOCHAM en prenant en compte la plupart des concepts définis par le langage SBGN-PD. Notamment, cette sémantique prend en compte les inhibitions qui sont absentes de la sémantique de BIOCHAM. Ainsi, cette sémantique permet d'interpréter n'importe quel réseau de réactions représenté sous la forme d'une carte SBGN-PD en un modèle discret. Nous donnons dans la suite de cette section la manière informelle avec laquelle sont interprétés les différents éléments d'une carte (EPN, processus, etc.) en un réseau d'automates. L'interprétation formelle d'une carte SBGN-PD en un réseau d'automates à l'aide de la sémantique générale est donnée dans la section 6.5 de ce chapitre.

6.3.1 Interprétation des EPNs, processus et modulations

Dans la sémantique générale, nous considérons qu'un EPN peut être soit absent soit présent dans le système biologique (p.ex. une cellule). Nous interprétons donc l'état d'un EPN par une valeur Booléenne. Nous aurions pu choisir d'interpréter un EPN par une valeur entière bornée, comme c'est le cas dans les réseaux de Thomas par exemple ; cependant, comme nous ne considérons en entrée que la carte SBGN-PD sans autres informations, nous n'avons *a priori* aucune connaissance sur d'éventuels effets différentiels que pourrait avoir tel ou tel EPN suivant sa quantité relative dans le système. De façon similaire aux EPNs, un processus peut avoir lieu ou non, et son état sera par conséquent également modélisé par une variable Booléenne. Quant aux modulations, elles peuvent être actives ou inactives.

Le déclenchement d'un processus, c'est-à-dire le passage, pour le processus, d'un état où il n'a pas lieu à un état où il a lieu, est conditionné par la présence des ses réactifs, et l'état, actif ou non, de ses modulations. De façon analogue aux modulations des cartes SBGN-AF, la source d'une modulation

peut être un EPN, ou un opérateur logique qui peut être associé à une formule Booléenne. Pour une modulation, nous définissons la satisfaction de cet opérateur de façon analogue à la notion définie pour les cartes SBGN-AF (voir chapitre 5) :

- un opérateur AND est satisfait si tous ses parents qui sont des EPNs sont présents, et si tous ses parents qui sont des opérateur logiques sont satisfaits ;
- un opérateur OR est satisfait si au moins un de ses parents qui sont des EPNs est présent, ou au moins un de ses parents qui sont des opérateurs logiques est satisfait ;
- un opérateur NOT est satisfait si son unique parent est un EPN qui n'est pas présent, ou si son unique parent est un opérateur logique qui n'est pas satisfait.

Si une modulation a comme source un EPN, elle est *active ssi* cet EPN est présent ; si elle a comme source un opérateur logique, elle est *active ssi* cet opérateur est satisfait.

Un processus peut être ciblé par plus d'une modulation. Dans ce cas, la logique de la modulation globale du processus est inconnue ou non spécifiée, autrement le processus serait ciblé par une unique modulation dont la source serait un opérateur logique associée à une formule Booléenne. Pour traiter ces cas-là, nous devons spécifier un principe d'interprétation générale (comme dans le chapitre 5).

Afin que la dynamique résultante de cette interprétation soit la moins contrainte possible, tout en respectant le sens biologique des différents arcs de modulations, nous avons choisi que la modulation globale d'un processus est active *ssi* :

- toutes les stimulations nécessaires du processus sont actives et
- au moins une des stimulations du processus est active ou une de ses inhibitions est inactive.

Avec l'interprétation d'une modulation globale par cette contrainte, nous sommes assurés que la dynamique obtenue contienne la dynamique qui aurait été obtenue si la fonction logique régissant la modulation globale avait été connue ou spécifiée. Notons également que nous ne prenons pas en compte les modulations qui ne sont ni des stimulations (dont les catalyses et stimulations nécessaires sont des sous-classes) ni des inhibitions : en effet, par définition, nous ne connaissons pas l'effet de telles modulations sur le processus, et ces modulations ne sont alors d'aucun effet sur la dynamique.

Notons que nous n'interprétons pas la modulation globale d'un processus comme nous avons interprété la modulation globale d'une activité au chapitre 5. Comme ici nous proposons de nouvelles sémantiques, nous veillons à ce que celles-ci soient les plus générales possibles, contrairement au chapitre précédent où nous proposions non pas une nouvelle sémantique, mais des modèles de graphes d'influences construits avec une sémantique preéxistante (la sémantique Booléenne).

6.3.2 Des cartes SBGN-PD aux réseaux d'automates

Les sémantiques qualitatives que nous proposons dans cette étude sont exprimées sous la forme de réseaux d'automates asynchrones. Nous rappelons qu'un réseau d'automates est constitué d'un ensemble d'automates, chacun formé d'un ensemble d'états locaux et d'un ensemble de transitions locales conditionnées par les états locaux des autres automates du réseau.

Dans le cadre de la *sémantique générale*, chaque EPN est associé à un unique automate avec deux états locaux, l'un étiqueté 0 (pour absent), et l'autre étiqueté 1 (pour présent). De façon analogue, à chaque processus est associé un unique automate avec deux états locaux, étiquetés 0 et 1, indiquant que le processus a lieu et que le processus n'a pas lieu, respectivement.

Les transitions locales sont construites de façon à ce que :

- un processus puisse passer d'un état où il n'a pas lieu à un état où il a lieu *ssi* tous ses réactifs (qui ne sont pas des EPNs de type source) sont présents, et sa modulation globale est active ;

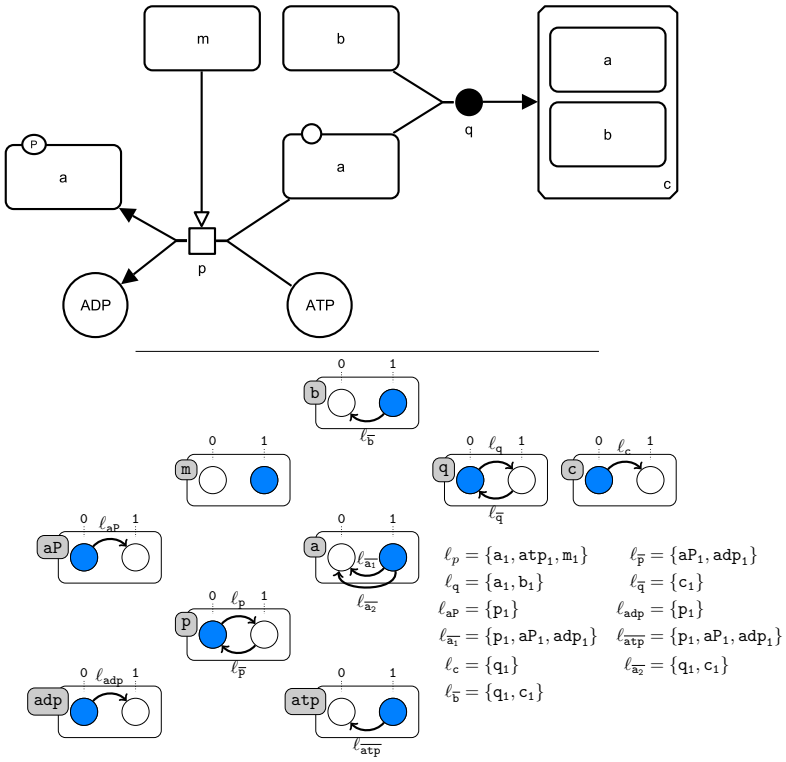


FIGURE 6.2 – **Une carte SBGN-PD modélisée par un réseau d'automates avec la sémantique générale.** En haut : un exemple de carte SBGN-PD. En bas : le réseau d'automates asynchrone qui modélise cette carte avec la sémantique générale. Les différents automates du réseau, ainsi que leurs états locaux et les transitions locales sont représentés. L'état global initial $(\mathtt{adp}_0, \mathtt{atp}_1, \mathtt{a}_1, \mathtt{aP}_0, \mathtt{b}_1, \mathtt{c}_0, \mathtt{m}_1, \mathtt{p}_0, \mathtt{q}_0)$ est représenté en bleu. Notons que dans la carte, aucun processus n'agit sur l'EPN m, et que par conséquent, aucune transition n'agit sur les états locaux de l'automate $\mathtt{m}$.

- un processus puisse passer d'un état où il a lieu à un état où il n'a pas lieu *ssi* tous ses produits (qui ne sont pas des EPNs de type puits) sont présents ;
- un EPN puisse passer d'un état absent à un état présent *ssi* il est un produit d'un processus qui a lieu ;
- un EPN puisse passer d'un état présent à un état absent *ssi* il est le réactif d'un processus qui a lieu, et tous les produits de ce processus sont présents.

Comme nous n'avons *a priori* pas d'informations sur les constantes d'équilibre des processus du réseau, nous ne forçons pas la consommation des réactifs. Cela est rendu possible par la manière dont nous construisons les transitions locales, et par le mode de mise à jour asynchrone : une transition modélisant la consommation d'un EPN peut ne pas être déclenchée avant que le processus passe dans son état où il n'a plus lieu.

Exemple 6.3 (Sémantique générale d'une carte SBGN-PD)**.** La figure 6.2 montre le modèle dynamique d'une carte SBGN-PD construit avec la sémantique générale.

6.4 La sémantique des histoires des réseaux SBGN-PD

Le langage SBGN-PD a été conçu pour modéliser des processus moléculaires, et en particulier des changements d'états physiques ou de localisation d'*entités moléculaires*. Par exemple, une protéine non phosphorylée et sa version phosphorylée peuvent être considérées comme deux états physiques de la même entité moléculaire, qui sont représentés en SBGN-PD par deux EPNs différents liés par un processus de phosphorylation. De la même façon, une entité moléculaire impliquée dans un processus d'association a un état libre et un état lié, et une entité moléculaire qui subit un processus de translocation a deux états, un pour le compartiment de départ et un pour le compartiment d'arrivée.

Une même entité moléculaire peut être impliquée dans plusieurs de ces processus, et par conséquent être associée à un certain nombre d'états différents, chacun représenté par un EPN distinct dans une carte SBGN-PD. Par exemple, dans la carte SBGN-PD donnée dans la figure 6.2, l'entité moléculaire a peut être dans trois états physiques différents : non-modifiée, phosphorylée, ou associée à b dans le complexe c.

Un état physique particulier d'une entité moléculaire peut correspondre à un état *actif* de celle-ci, c'est-à-dire un état où l'entité opère une fonction. Par exemple, dans la signalisation cellulaire, une kinase n'opère bien souvent sa fonction qu'une fois phosphorylée. Une telle activité kinase sera représentée dans un graphe d'influences par un noeud d'activité, et modélisée par une variable pouvant prendre deux valeurs dans une sémantique Booléenne : 0 (off) lorsque l'activité n'a pas lieu, et 1 (on) lorsque l'activité a lieu. Ainsi, avec une telle modélisation, une kinase sera inactive ou active, mais pas les deux en même temps. Ses deux états sont *mutuellement exclusifs*. Comme, dans notre exemple, l'état inactif correspond à la forme non phosphorylée de la kinase, et son état actif à sa forme phosphorylée, une telle modélisation impliquerait implicitement que les différentes formes de la kinase soient mutuellement exclusives.

La sémantique des histoires a pour but de modéliser les changements d'états physiques des entités moléculaires dans cette perspective. Elle contraint la sémantique générale en rendant explicitement tous les EPNs représentant différents états d'une même entité moléculaire mutuellement exclusifs.

Avant de présenter la sémantique des histoires proprement dite et la relation de cette sémantique avec la sémantique générale, nous présentons un programme logique permettant de trouver, dans une carte SBGN-PD, les ensembles d'EPNs représentant des états différents d'une même entité moléculaire. Nous appelons de tels ensembles des *histoires*.

6.4.1 Histoires et ensembles d'histoires d'une carte SBGN-PD

Une histoire d'une carte SBGN-PD est un ensemble d'EPNs de cette carte représentant chacun un état différent d'une même entité moléculaire, que nous rendrons mutuellement exclusifs dans la sémantique des histoires. Afin de respecter cette contrainte d'exclusivité mutuelle et de respecter le sens biologique des processus, les histoires elle-mêmes doivent être définies par un ensemble de contraintes.

La définition d'une histoire d'une carte SBGN-PD est la suivante :

Définition 6.5 (Histoire d'une carte SBGN-PD)**.** Étant donnée une carte SBGN-PD, une *histoire* de cette carte est une ensemble d'EPN vérifiant les contraintes suivantes :

(i) étant donnés deux EPN de cet ensemble, il existe un chemin de la carte entre ces deux EPNs tel que toutes les arêtes du chemin sont des arcs de flux et tous les EPNs du chemin appartiennent à cet ensemble ;

(ii) si un EPN de cet ensemble est un produit d'un processus, alors au moins un des réactifs de ce processus (qui peut être un EPN de type source) appartient à cet ensemble ;

(iii) étant donnés deux EPNs de cet ensemble, il n'existe pas de processus de la carte qui consomme les deux ;

(iv) étant donnés deux EPNs de cet ensemble, il n'existe pas de processus de la carte qui produise les deux.

La contrainte (i), considérée avec les contraintes (iii) et (iv), garantit que tous les EPNs d'une histoire représentent différents états d'une même entité moléculaire, qui apparaissent par transformations successives de cette entité. Quant aux contraintes (ii-iv), elles sont nécessaires pour définir une sémantique où les EPNs d'une même histoire sont mutuellement exclusifs : la contrainte (ii) garantit qu'aucun processus ne puisse produire un EPN d'une histoire sans d'abord consommer un EPN de cette même histoire ; la contrainte (iii) garantit que l'exclusion mutuelle des EPNs d'une même histoire n'empêche pas un processus d'avoir lieu dû à l'absence d'un de ses réactifs ; finalement, la contrainte (iv) garantit que l'exclusion mutuelle n'empêche pas un processus de produire tous ses produits.

Deux EPNs qui sont des états différents d'une même entité moléculaire auront souvent la même étiquette SBGN-PD, i.e. le même nom. Par conséquent, nous considérons une contrainte additionnelle, qui reste optionnelle, et qui permet d'assurer que tous les EPNs d'une même histoire partagent la même étiquette :

(v) tous les EPNs d'une même histoire partagent une même étiquette, que ce soit l'étiquette de l'EPN lui-même, ou celle d'un de ses composants si l'EPN est un complexe.

Notons qu'avec la définition que nous avons donnée d'une histoire, celle-ci peut contenir un EPN de type source ou de type puits.

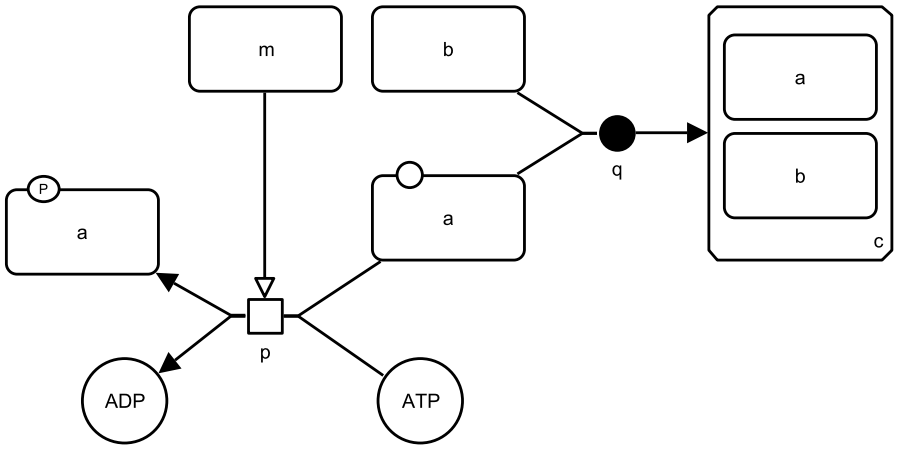
Exemple 6.4 (Histoires d'une carte SBGN-PD)**.** La figure 6.3 montre une carte SBGN-PD et l'ensemble de ses histoires comportant au moins deux EPNs, calculées avec les contraintes (i-iv). Il n'existe pas d'histoire contenant à la fois a et atp, en raison de la contrainte (iii). Une telle histoire empêcherait en effet le processus p d'avoir lieu : comme les EPNs d'une histoire ont vocation à être mutuellement exclusifs et puisque a et atp sont les réactifs de p, ses réactifs ne pourraient en conséquence jamais être présents en même temps.

De façon analogue, il n'existe pas d'histoire contenant à la fois aP et adp, cette fois-ci en raison de la contrainte (iv). Une telle histoire empêcherait le processus p de produire tous ses produits, puisque ces deux EPNs sont les produits de ce processus.

Une carte SBGN-PD peut se focaliser sur plusieurs entités moléculaires, et ainsi contenir plusieurs histoires. Nous nous intéressons donc à caractériser un ensemble d'histoires afin de modéliser les changements d'états d'un ensemble d'entités moléculaires d'une carte SBGN-PD. Comme nous voulons que les EPNs d'une histoire soient mutuellement exclusifs dans la sémantique des histoires, deux histoires modélisant deux entités moléculaires d'une même carte ne doivent pas partager un même EPN.

Définition 6.6 (Ensemble valide d'histoires d'une carte SBGN-PD)**.** Un ensemble d'histoires d'une carte SBGN-PD est dit *valide ssi* ses histoires ont une intersection vide deux à deux.

Exemple 6.5 (Ensemble valide d'histoires d'une carte SBGN-PD)**.** La figure 6.3 montre une carte SBGN-PD et les ensembles maximalement (au sens de l'inclusion) valides d'histoires de cette carte.



Ensemble d'histoires de cardinal au moins deux

$$\{\{a, aP\}, \{a, c\}, \{a, adp\}, \{aP, atp\}, \{b, c\}, \{adp, atp\}, \{a, aP, c\}, \{a, adp, c\}\}$$

Ensembles maximalement valides d'histoires

$$\{\{aP, atp\}, \{a, c\}\}; \{\{adp, atp\}, \{a, c\}\}, \{\{adp, atp\}, \{a, aP, c\}\}; \{\{a, adp, c\}, \{aP, atp\}\}; \{\{a, adp\}, \{b, c\}, \{aP, atp\}\}; \{\{a, aP\}, \{adp, atp\}, \{b, c\}\}$$

FIGURE 6.3 – **Histoires et ensembles d'histoires valides d'une carte SBGN-PD.** En haut : la carte SBGN-PD de la figure 6.2. En bas : l'ensemble des histoires de cardinal au moins deux de la carte, respectant les contraintes (i–iv), et la collection des ensemble maximalement valides d'histoires respectant les contraintes (i–iv). Les ensembles finaux d'histoires sont représentés en bleu, tandis que les ensemble d'histoires epn-maximaux sont représentés en vert.

Étant donnée une carte SBGN-PD, nous sommes intéressés à trouver un ensemble d'histoires valide qui ait un sens biologique, c'est-à-dire qui caractérise au mieux les entités moléculaires qui sont importantes à la compréhension du système biologique.

La nécessité pour un ensemble d'histoires d'être valide induit le plus souvent à choisir entre deux histoires alternatives partageant un même EPN. En particulier, les processus d'associations, en raison de la contrainte (iii), impliquent le plus souvent des histoires alternatives, une pour chaque réactif du processus, qui contiennent toutes le complexe produit par le processus.

Il s'agit donc d'abord, pour modéliser une carte SBGN-PD avec la sémantique des histoires, de choisir un ensemble valide d'histoires parmi tous les ensembles valides de la carte.

Les ensembles valides d'histoires d'une carte peuvent être calculés automatiquement. Cependant, leur nombre peut être très grand, étant donné qu'il dépend à la fois du nombre d'EPNs et du nombre d'histoires individuelles de la carte. Ainsi, choisir un ensemble valide d'histoires pertinent parmi tous les ensembles valides est en pratique irréaliste pour de grandes cartes.

Par conséquent, afin de réduire drastiquement le nombre d'ensembles valides d'histoires candidats, nous définissons deux contraintes de maximalité sur ces ensembles :

Définition 6.7 (Ensemble final d'histoires d'une carte SBGN-PD)**.** Un ensemble S d'histoires d'une carte SBGN-PD est dit *final ssi* :

- il est valide et
- il n'existe pas d'ensemble valide d'histoires $S' \neq S$ tel que pour toute histoire de S, il existe une histoire de S' qui soit un super-sensemble de cette histoire.

Définition 6.8 (Ensemble epn-maximal d'histoires d'une carte SBGN-PD)**.** Un ensemble S d'histoires d'une carte SBGN-PD est dit *epn-maximal ssi* :

- il est valide et
- il n'existe pas d'ensemble valide d'histoires $S' \neq S$ tel que le nombre total d'EPNs de S' soit supérieur au nombre total d'EPN de S.

Nous noterons que tout ensemble final d'histoires est maximalement valide, et que tout ensemble epn-maximal est final.

Exemple 6.6 (Ensemble final et epn-maximal d'histoires d'une carte SBGN-PD)**.** La figure 6.3 montre une carte SBGN-PD ainsi que ses ensembles finaux d'histoires (en bleu) et ses ensembles epn-maximaux d'histoires (en vert). L'ensemble d'histoires $\{\{a, aP\}, \{adp, atp\}\}$ est valide mais pas final, étant donné que l'ensemble d'histoires $\{\{a, aP, c\}, \{adp, atp\}\}$ est valide. Ce dernier ensemble n'est par contre pas epn-maximal : il ne contient que cinq EPNs, alors que l'ensemble $\{\{a, aP\}, \{adp, atp\}, \{b, c\}$ est valide et contient six EPNs.

Afin de réduire encore le nombre d'ensembles candidats, des contraintes supplémentaires issues de la connaissance experte peuvent être définies. Par exemple, on peut ne vouloir obtenir que des ensembles d'histoires qui contiennent telle ou telle histoire définie préalablement ; ou bien, qu'aucun des ensembles ne contienne une histoire avec un EPN donné.

6.4.2 Calcul des ensembles d'histoires d'une carte SBGN-PD

Nous avons écrit un programme Python permettant de calculer tous les ensembles valides d'histoires d'une carte SBGN-PD donnée.

Étant donné une carte SBGN-PD écrite au format SBGN-ML, le programme transforme d'abord la carte en son *graphe de composants*, à l'aide de la librairie LibSBGN [VI+12]. Le graphe de composants est construit comme suit : à chaque EPN de la carte est associé un noeud dans le graphe, étiqueté par l'identifiant de cet EPN, et est associé à un ensemble d'étiquettes formé de sa propre étiquette et des étiquettes des éventuelles sous-unités qui le composent ; à chaque couple réactif/produit de chaque processus est associé un arc dans le graphe, avec comme source le noeud associé au réactif, comme cible le noeud associé au produit, et étiqueté par l'identifiant du processus.

Le graphe de composants de la carte est ensuite formalisé à l'aide de prédicats logiques instanciés : à chaque noeud du graphe obtenu correspond un prédicat unaire de symbole *epn* avec pour argument l'étiquette identifiant le noeud, et un ensemble de prédicats binaires de symbole *label*, dont le premier argument est une étiquette de l'ensemble d'étiquettes associé au noeud, et le deuxième l'étiquette identifiant le noeud ; à chaque arc du graphe correspond un prédicat tertiaire de symbole *edge*, avec pour premier argument l'étiquette identifiant la source, pour deuxième l'étiquette identifiant la cible, et pour troisième l'étiquette de l'arc.

Le programme appelle ensuite clingo, qui exécute un programme ASP composé de l'ensemble de ces prédicats et de règles ASP qui encodent les contraintes (i–iv), et éventuellement la contrainte optionnelle (v).

Le code ASP encodant les contraintes (i–v) est donné ci-dessous.

Listing 6.1 – Programme ASP pour le calcul des histoires : contraintes (i–v)

```
1  #show gather/2.
2  %definition de la relation gather
3  gather(E1,E2):-gather(E2,E1).
4  gather(E1,E3):-gather(E1,E2);gather(E2,E3);E1!=E3.
5  %contrainte (i)
6  0 {gather(E1,E2)} 1:-edge(E1,E2,P);epn(E1);epn(E2);E1!=E2.
7  %contrainte (ii)
8  :-not gather(E1,E2);edge(E1,E2,P);gather(E2,_).
9  %contrainte (iii)
10 :-gather(E1,E2);edge(E1,E3,P);edge(E2,E4,P);E1!=E2.
11 %contrainte (iv)
12 :-gather(E1,E2);edge(E3,E1,P);edge(E4,E2,P);E1!=E2.
13 %contrainte optionnelle (v)
14 1 {cand(L,X):label(L,X)} 1:-label(_,X);epn(X).
15 same(X,Y):-cand(L,Y);cand(L,X);epn(X);epn(Y);X!=Y.
16 :-not same(X,Y);gather(X,Y);epn(X);epn(Y);X!=Y.
```

Le regroupement deux à deux des EPNs d'une même histoire est encodé par le prédicat *gather*/2. Les lignes 3–4 définissent la relation *gather*, qui est symétrique et transitive. La ligne 6 construit l'espace des solutions, tout en encodant la contrainte (i). Les lignes 8, 10 et 12 encodent les contraintes (ii), (iii) et (iv), respectivement. Finalement, les lignes 14–16 encodent la contrainte (v). Le prédicat *cand*/2 permet de choisir une étiquette pour chaque EPN. Une histoire données (construite à l'aide des constraintes (i-iv) respectera la contrainte (v) *ssi* il est possible de choisir la même étiquette pour chaque EPN de l'histoire.

Le programme ASP est exécuté par clingo, qui est appelé à travers le module gringo de Python. Les answer sets obtenus sont exactement les ensemble valides d'histoires (à une transformation syntaxique près).

La contrainte de finalité des ensembles d'histoires peut être prise en compte par post-traitement de la collection d'ensembles valides obtenue. Comme la prise en compte de cette contrainte nécessite d'abord le calcul de tous les ensembles valides d'une carte, le calcul de tous les ensembles finaux n'est possible que pour des cartes ne dépassant pas une certaine taille (aux alentours de quelques dizaines de noeuds).

Quant à la contrainte d'epn-maximalité et les contraintes issues de la connaissance experte, elles peuvent être prises en compte directement par le programme ASP, et ne nécessite pas le calcul de l'ensemble des ensembles valides d'histoires. Par conséquent, le calcul des ensembles epn-maximaux peut se faire pour des cartes de plus grande taille que celui des ensembles finaux.

Le code ASP permettant d'encoder la contrainte d'epn-maximalité est le suivant :

Listing 6.2 – Programme ASP pour le calcul des histoires : contrainte d'epn-maximalité

```
1 inStory(E1):-gather(E1,E2).
2 inStory(E2):-gather(E1,E2).
3 #maximize {1,inStory,E:inStory(E),epn(E)}.
```

Les lignes 1–2 définissent le prédicat auxiliaire $inStory/1$, qui exprime le fait qu'un EPN est dans une histoire. La ligne 3, quant à elle, est une directive de maximalité (sur la cardinalité), qui permet d'obtenir seulement les answer sets qui maximisent le nombre d'atomes de la forme $inStory(X)$.

Enfin, les contraintes issues de la connaissance experte peuvent être encodées sans ajout de règles pour certaines, et en utilisant le prédicat auxiliaire $inStory$ et de nouvelles règles pour d'autres. Par exemple, des histoires peuvent être définies prélabalement à l'aide d'instances du prédicat $gather/2$ ajoutées directement au programme, et empêcher un EPN spécifique e (désigné par une constante) d'appartenir à une histoire peut être encodé en ajoutant au programme les lignes 1–2 du Listing 6.3 et une contrainte sur e :

Listing 6.3 – Programme ASP pour le calcul des histoires : empêcher un EPN d'appartenir à une histoire

```
1 inStory(E1):-gather(E1,E2).
2 inStory(E2):-gather(E1,E2).
3 :-inStory(e).
```

6.4.3 Illustration sur la carte de l'activation d'ERK induite par $AT_{1A}R$

La figure 6.4 montre la carte de l'activation d'ERK induite par le récepteur $AT_{1A}R$, que nous avons déjà introduite dans le chapitre 3.

Afin d'illustrer le concept d'histoire et d'ensemble valide d'histoires sur un exemple réel, nous avons calculé tous les ensembles finaux d'histoires de cette carte en considérant les contraintes (i-iv). Nous n'avons obtenu que deux ensembles finaux : l'un incluant une histoire se focalisant sur chacune des deux β-arrestines ; l'autre incluant une histoire se focalisant sur le récepteur. Les histoires se focalisant sur les autres entités moléculaires de la carte étaient les mêmes dans les deux ensembles : une histoire pour la protéine G, une pour le couple PIP2/DAG, une pour la PKA, et une pour ERK.

Ces deux ensembles alternatifs sont dûs aux processus d'association faisant entrer en jeu le récepteur et chacune des deux β-arrestines. Afin de modéliser ce réseau avec la sémantique des histoires, il faudra choisir entre ces deux ensembles, c'est-à-dire choisir de se focaliser sur le récepteur ou sur les deux β-arrestines.

L'ensemble d'histoires se focalisant sur le récepteur est représenté sur la figure 6.4, où chaque histoire de l'ensemble est représentée par une couleur différente. L'histoire se focalisant sur le récepteur, en rouge dans la figure, contient tous les EPNs représentant différents états physiques du récepteur : non complexé, phosphorylé sur l'un ou l'autre de ses sites de phosphorylation, complexé à la β-arrestine 1 ou à la β-arrestine 2. Cette histoire permet donc de modéliser la succession des états physiques du récepteur. Parmi ces différents états, certains sont fonctionnels : par exemple, le récepteur, dans son état non complexé et phosphorylé sur son premier site de phosphorylation, induit la voie G en activant la protéine G, et il perd cette capacité quand il s'associe à la β-arrestine 1.

Finalement, notons que l'histoire contenant les EPN PIP2 et DAG, représentée en violet dans la figure, ne respecte pas la contrainte (v), même si cette histoire a un sens biologique. Cette contrainte peut être trop stricte pour les processus transformant des petites molécules en d'autres petites molécules, qui ont le plus souvent des étiquettes différentes.

6.4.4 Des cartes SBGN-PD aux réseaux d'automates

Nous avons défini la notion d'histoire, et vu comment des ensembles valides d'histoires d'une carte SBGN-PD pouvaient être calculés. Nous définissons maintenant la sémantique des histoires proprement dite.

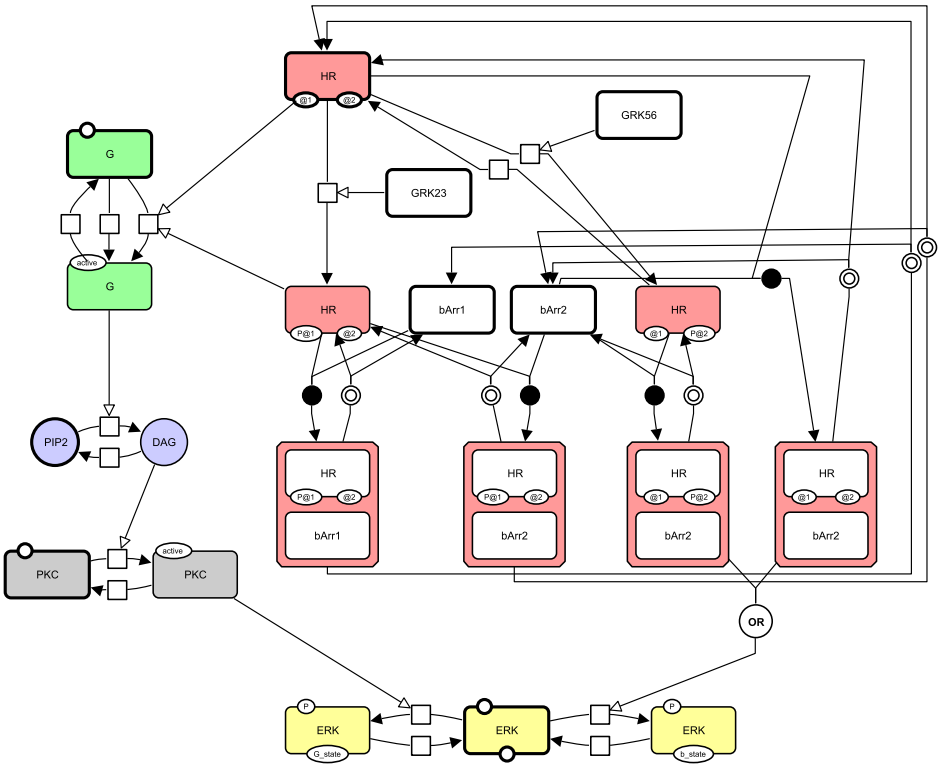


FIGURE 6.4 – **Un ensemble final d'histoires de la carte SBGN-PD de l'activation d'ERK induite par le récepteur $AT_{1A}R$.** Cette carte a déjà été représentée dans la figure 3.13. Elle représente les deux voies principales activant ERK induites par le récepteur $AT_{1A}R$ (récepteur de l'angiotensine). Ce récepteur active, d'une part, la voie de la protéine G qui active ERK, et la voie des β-arrestines d'autre part. Ces deux voies sont finement régulées par les kinases des récepteurs couplés aux protéines G (GRK) (GRK2/3 et GRK5/6), qui agissent directement sur la phosphorylation du récepteur. Afin de différencier les deux états d'ERK induits par chacune des deux voies de signalisation mentionnées plus haut, des variables d'état avec comme valeur "G" et "β" ont été ajoutées aux deux EPNs représentant ces états.
Les EPNs représentés avec une bordure épaisse représentent un état initial possible de la carte. Chaque EPN représenté en couleur appartient à une histoire, et chaque couleur est associée à une histoire différente de la carte. Les histoires représentées respectent les contraintes (i–iv). L'ensemble de ces histoires est final.
L'histoire en rose se focalise sur le récepteur HR et comprend sept états physiques différents de ce récepteur : non complexé, phosphorylé (sur un de ses deux sites), complexé à la β-arrestine 1, et complexé à la β-arrestine 2. Les autres histoires se focalisent sur ERK (en jaune), sur la protéine G (en vert), sur PIP2 (en bleu) et sur PKC (en gris).

La sémantique des histoires vise à modéliser une carte étant donné un ensemble valide d'histoires choisi parmi tous les ensembles valides de la carte. Elle ne diffère de la sémantique générale que par le traitement des EPNs qui appartiennent à une histoire, en rendant ces EPNs mutuellement exclusifs. Les EPNs n'appartenant à aucune histoire, les processus et les modulations sont modélisés exactement de la même façon que dans la sémantique générale.

Dans la sémantique générale, à chaque EPN de la carte était associé un automate ; dans la sémantique des histoires, à chaque histoire est associé un automate, et les EPNs appartenant à cette histoire sont modélisés par les états locaux de cet automate. Plus précisément, un automate modélisant une histoire est formé d'un état local pour chaque EPN non source et non puits de l'histoire, et d'un état local spécifique appelé l'*état vide* de l'histoire.

Tout état local d'un automate associé à une histoire, mis à part son état vide, correspond ainsi à un état physique de l'entité moléculaire modélisée par l'histoire. Comme ces EPNs sont modélisés par des états locaux d'un automate plutôt que par des automates, ils sont maintenant nécessairement mutuellement exlusifs. Quant à l'état vide, il modélise l'absence de cette entité moléculaire dans le système.

Les transitions locales des automates associés aux histoires sont construites de façon à ce que :

- une histoire puisse passer d'un de ses états locaux à un autre de ses états locaux, excepté l'état vide, *ssi* un processus qui a lieu consomme l'EPN auquel est associé l'état local de départ, et produit l'EPN auquel est associé l'état local d'arrivée ;
- une histoire puisse passer d'un de ses états locaux à son état vide *ssi* un processus qui a lieu consomme l'EPN associé à l'état local de départ et ne produit aucun des EPNs de l'histoire.

Étant donné que nous modélisons maintenant les histoires plutôt que les EPNs qui appartiennent à ces histoires, comme des composants distincts, et que la sémantique des processus et des modulations est définie à partir de la présence et de l'absence des EPNs qu'ils font entrer en jeu, nous devons redéfinir ces notions de présence et d'absence pour les EPNs appartenant aux histoires en fonction de l'état des histoires auxquels il appartiennent. De ce fait, un EPN appartenant à une histoire est maintenant dit *présent* si l'automate associé à l'histoire contenant cet EPN est dans l'état local associé à cet EPN ; cet EPN est dit *absent* dans le cas contraire.

Afin d'éviter d'éventuels conflits entre processus agissant sur une même histoire, c'est-à-dire entre processus qui ont des réactifs ou des produits appartenant à une même histoire, nous imposons une exclusivité mutuelle entre ces processus. Ainsi, un processus agissant sur une histoire ne pourra avoir lieu que si les autres processus agissant sur cette même histoire n'ont pas lieu.

Exemple 6.7 (Sémantique des histoires d'une carte SBGN-PD)**.** La figure 6.5 montre le modèle dynamique d'une carte SBGN-PD construit avec la sémantique des histoires. L'ensemble valide d'histoires choisi pour la modélisation contient deux histoires, une se focalisant sur l'entité a (en jaune), l'autre sur le couple atp/adp (en rose).

La contrainte d'exclusivité mutuelle des processus agissant sur une même histoire est illustrée par la présence de l'état local p_0 dans les conditions de la transition ℓ_q, et de l'état local q_0 dans celles de l_p. Sans cette contrainte, la dynamique pourrait contenir la succession des transitions suivantes, dans l'ordre : ℓ_p, ℓ_q, ℓ_2 et ℓ_3. On aurait alors produit de l'ADP sans produire la molécule phospho-a, quand bien même ces deux molécules ne sont les produits que d'un seul processus, ici p, et devraient par conséquent être produites ensemble.

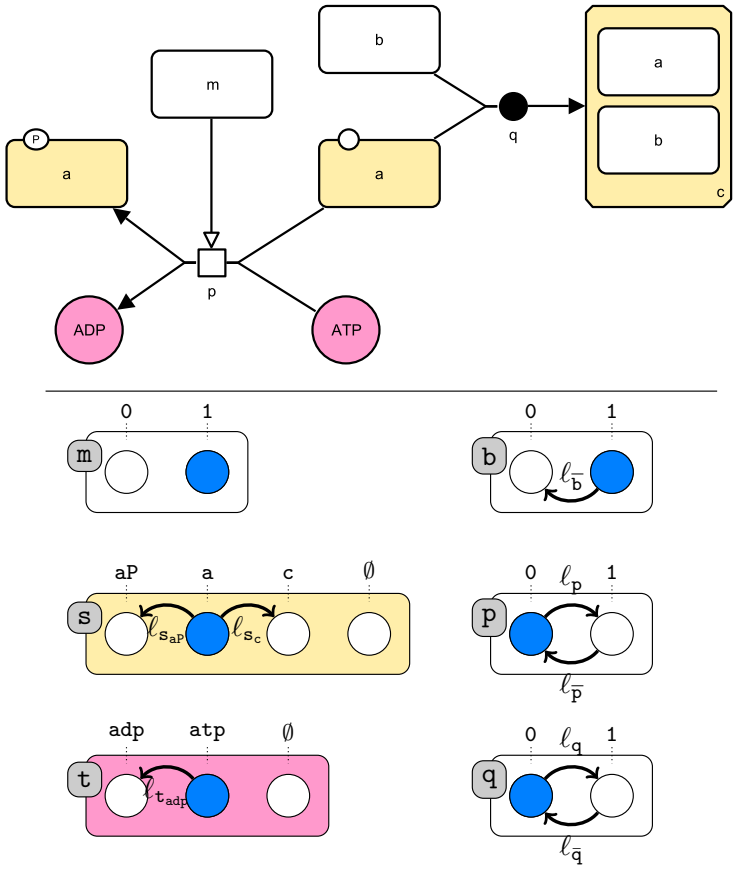


FIGURE 6.5 – **Une carte SBGN-PD modélisée par un réseau d'automates avec la sémantique des histoires.** En haut : la carte SBGN-PD de la figure 6.2. L'ensemble valide d'histoires choisi comporte deux histoires $\mathfrak{s}$ et $\mathfrak{t}$, représentées en jaune et en rose, respectivement. En bas : le réseau d'automates asynchrone qui modélise cette carte avec la sémantique des histoires, et en considérant les histoires $\mathfrak{s}$ et $\mathfrak{t}$.
L'état global initial $(b_1, m_1, p_0, q_0, s_a, t_{atp})$ est représenté en bleu. Notons que comme les processus p et q agissent tous deux sur l'histoire $\mathfrak{s}$, ils sont en conflit, et que par conséquent $q_0 \in \ell_p$ et $p_0 \in \ell_q$.

6.5 Transformation formelle des cartes SBGN-PD en RA avec les deux sémantiques

Du point de vue de la transformation des cartes SBGN-PD en réseaux d'automates, la sémantique générale peut être considérée comme un cas particulier de la sémantique des histoires où l'ensemble choisi d'histoires serait vide, étant donné que les processus, les modulations et les EPNs n'appartenant pas à une histoire sont modélisés exactement de la même façon dans les deux sémantiques. Cette propriété permet de ne définir formellement qu'une seule transformation des cartes SBGN-PD vers les réseaux d'automates, valable pour les deux sémantiques. Nous donnons cette transformation dans ce qui suit.

Soit une carte SBGN-PD. Nous utilisons les notations suivantes pour nous référer aux éléments de cette carte :

- $\mathcal{E} = \{e_1, \ldots, e_n\}$ l'ensemble des EPNs de la carte ;
- $\mathcal{P} = \{p_1, \ldots, p_m\}$ l'ensemble des processus de la carte ;
- $\mathcal{O} = \{o_1, \ldots, o_q\}$ l'ensemble des opérateurs logiques de la carte ;
- Pour chaque processus $p \in \mathcal{P}$, nous dénotons par $reac(p)$ (resp. $prod(p)$) l'ensemble des réactifs (resp. produits) de p qui ne sont pas de type source ou puits ; par $req(p)$ (resp. $stim(p)$, $inh(p)$) l'ensemble des stimulateurs nécessaires (resp. stimulateurs, inhibiteurs) qui modulent p.
- Pour chaque opérateur logique $o \in \mathcal{O}$, nous dénotons par $in(o)$ l'ensemble des noeuds (EPNs ou opérateurs logiques) sources d'un arc logique entrant sur o.
- Un EPN de type puits ou de type source sera dénoté par le symbole $\emptyset$. Et si un EPN e n'est pas de type source ou de type puits, nous noterons $e \neq \emptyset$.

Soit $\mathbb{S} = \{\mathfrak{S}_1, \ldots, \mathfrak{S}_r\}$ l'ensemble valide d'histoires choisi pour modéliser la carte SBGN-PD. Cet ensemble doit être vide si nous voulons modéliser la carte avec la sémantique générale. Nous dénotons par $\cup\mathbb{S} = \bigcup_{\mathfrak{S}\in\mathbb{S}} \mathfrak{S}$ l'ensemble des EPNs qui sont dans une histoire. Pour un processus $p \in \mathcal{P}$, nous dénotons par $\mathbb{S}(p) = \{\mathfrak{S} \in \mathbb{S}, reac(p) \cap \mathfrak{S} \neq \emptyset \vee prod(p) \cap \mathfrak{S} \neq \emptyset\}$ l'ensemble des histoires sur lesquelles p agit.

Nous encodons la carte en un réseau d'automates (Σ, S, T), en considérant l'ensemble d'histoires valides $\mathbb{S}$.

Nous rappelons que, pour un automate $\mathtt{a} \in \Sigma$, $S(\mathtt{a})$ dénote l'ensemble des états locaux de $\mathtt{a}$.

6.5.1 Encodage des automates

Pour les EPNs : À chaque EPN n'appartenant pas à une histoire sont associés un état absent et un état présent.

Ainsi, pour chaque EPN $e \in \mathcal{E}, e \notin \cup\mathbb{S}$, nous définissons un automate $\mathtt{e} \in \Sigma$ tel que $S(\mathtt{e}) = \{\mathtt{e}_0, \mathtt{e}_1\}$.

Pour les processus : À chaque processus sont associés deux états, l'un signifiant qu'il n'a pas lieu, l'autre qu'il a lieu.

Ainsi, pour chaque processus $p \in \mathcal{P}$, nous définissons un automate $\mathtt{p} \in \Sigma$ tel que $S(\mathtt{p}) = \{\mathtt{p}_0, \mathtt{p}_1\}$.

Pour les histoires : À chaque histoire sont associés un état vide et un état pour chaque EPN non source et non puits appartenant à l'histoire.

Ainsi, pour chaque histoire $\mathfrak{S} \in \mathbb{S}$, nous définissons un automate $\mathtt{S} \in \Sigma$ tel que $S(\mathtt{S}) = \{\mathtt{S}_\mathtt{e} \mid e \in \mathfrak{S}, e \neq \emptyset\} \cup \{\mathtt{S}_\emptyset\}$.

6.5.2 La logique des modulations

À chaque EPN ou opérateur logique q à la source d'une modulation, nous associons une formule Booléenne dénotée par $\mathsf{logic}(q)$. La modulation dont q est la source est active *ssi* les états des EPNs entrant en jeu dans $\mathsf{logic}(q)$ satisfont cette formule. La formule $\mathsf{logic}(q)$ est définie de la manière suivante :

$$\mathsf{logic}(q) \triangleq \begin{cases} q \text{ si } q \in \mathcal{E}; \\ \bigwedge\limits_{j \in in(q)} \mathsf{logic}(j) \text{ si } q \in \mathcal{O} \text{ et } q \text{ est un opérateur AND}; \\ \bigvee\limits_{j \in in(q)} \mathsf{logic}(j) \text{ si } q \in \mathcal{O} \text{ et } q \text{ est un opérateur OR}; \\ \neg\mathsf{logic}(j) \text{ si } q \in \mathcal{O},\ q \text{ est un opérateur NOT, et } \{j\} = in(q). \end{cases}$$

De façon analogue à la fonction logic introduite au chapitre 5, cette fonction traduit directement la sémantique des opérateurs logiques donnée dans la spécification du langage SBGN-PD [Moo+11].

Ensuite, à chaque processus p est associée une formule Booléenne $\mathsf{mod}(p)$ qui modélise la modulation globale ciblant p. Nous rappelons que la modulation globale d'un processus est active *ssi* :

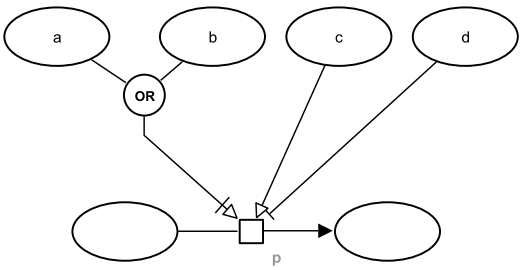
- toutes les stimulations nécessaires du processus sont actives et ;
- au moins une des stimulations du processus est active ou une de ses inhibitions est inactive.

La formule $\mathsf{mod}(p)$ est donc définie de la manière suivante :

$$\begin{aligned} \text{mod }(p) = & \bigwedge_{r \in req(p)} \mathsf{logic}(r) \\ & \wedge \Big(\bigvee_{s \in stim(p)} \mathsf{logic}(s) \vee \bigvee_{i \in inh(p)} \neg\mathsf{logic}(i) \Big) \end{aligned}$$

Finalement, pour une formule Booléenne f, nous dénotons par $\mathsf{DNF}(f)$ une forme normale disjonctive (DNF) de f.

Exemple 6.8. Nous considérons le processus, dénoté par p, de la figure suivante :



La formule $\mathsf{mod}(p)$, qui régit la modulation globale du processus p, est la formule suivante :

$$\mathsf{mod}(p) = (a \wedge b) \wedge (c \vee \neg d)$$

Une forme DNF de cette formule peut s'obtenir par distributivité :

$$\mathsf{DNF}(\mathsf{mod}(p)) = (a \wedge b \wedge c) \vee (a \wedge b \wedge \neg d)$$

6.5.3 Déclaration de conflits entre processus

Deux processus p et q agissant sur une même histoire ne peuvent pas avoir lieu en même temps. Ces processus sont dits en *conflit*. La relation de conflit est symétrique et irréflexive. Elle est définie formellement de la manière suivante :

Définition 6.9 (Relation de conflit)**.** Étant donné une carte SBGN-PD, la relation de conflit, notée $\#$, est incluse dans $\mathcal{P} \times \mathcal{P}$ et définie de la manière suivante :
Pour tout couple de processus $(p, q) \in \mathcal{P} \times \mathcal{P}$:

- $\mathbb{S}(p) \cap \mathbb{S}(q) \neq \emptyset \Rightarrow p \# q$;
- $p \# q \Rightarrow q \# p$;
- $p \# q \Rightarrow p \neq q$.

6.5.4 Encodage des transitions

Prise en compte des modulations : Pour chaque processus $p \in \mathcal{P}$, nous avons défini une formule Booléenne $\mathsf{mod}(p)$ modélisant sa modulation globale. Nous définissons maintenant la collection d'ensembles d'états locaux qui satisfont $\mathsf{mod}(p)$.

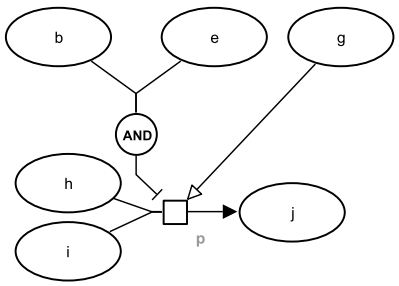
Nous définissons d'abord l'ensemble des états locaux qui satisfont un littéral donné l de $\mathsf{mod}(p)$, dénoté par $\mathsf{ls}(l)$:

$$\mathsf{ls}(l) = \begin{cases} \{\mathtt{e_0}\} \text{ si } l = \neg e, e \in \mathcal{E}, e \notin \cup\mathbb{S}; \\ \{\mathtt{e_1}\} \text{ si } l = e, e \in \mathcal{E}, e \notin \cup\mathbb{S}; \\ \{\mathtt{S_e}\} \text{ si } l = e, e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}; \\ \{\mathtt{S_f} \mid f \in \mathfrak{S}, f \neq e\} \text{ si } l = \neg e, e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}. \end{cases}$$

Puis nous définissons la collection d'ensembles d'états locaux satisfaisant $\mathsf{mod}(p)$, dénotée par $\mathsf{cond}(p)$. La formule $\mathsf{DNF}(\mathsf{mod}(p))$ est vue comme un ensemble de clauses conjonctives, et une clause comme un ensemble de littéraux. L'opérateur $\tilde{\prod}$ désigne le produit d'ensemble renvoyant un ensemble plutôt qu'un tuple.

$$\mathsf{cond}(p) = \bigcup_{c \in \mathsf{DNF}(\mathsf{mod}(p))} \tilde{\prod}_{l \in c} \mathsf{ls}(l)$$

Exemple 6.9. Nous considérons le processus, dénoté par p, de la figure suivante :



Nous considérons également les deux histoires $\mathfrak{S}_1 = \{a, b, c\}$ et $\mathfrak{S}_2 = \{d, e, f\}$, et supposons que g n'appartient à aucune histoire. Notons que les EPNs a, c, d et f ne sont pas représentés dans la carte.

La formule $\mathsf{DNF}(\mathsf{mod}(p))$, qui régit la modulation globale de p, est la formule suivante :

$$\mathsf{DNF}(\mathsf{mod}(p)) = (\neg b \wedge \neg e) \vee g$$

L'ensemble $\mathsf{cond}(p)$ des états locaux qui satisfont cette formule est alors l'ensemble suivant :

$$\begin{aligned}\mathsf{cond}(p) = \big\{\{&\mathtt{g_1}\},\\ &\{\mathtt{S_{1\emptyset}}, \mathtt{S_{2\emptyset}}\}, \{\mathtt{S_{1\emptyset}}, \mathtt{S_{2d}}\}, \{\mathtt{S_{1\emptyset}}, \mathtt{S_{2f}}\},\\ &\{\mathtt{S_{1a}}, \mathtt{S_{2\emptyset}}\}, \{\mathtt{S_{1a}}, \mathtt{S_{2d}}\}, \{\mathtt{S_{1a}}, \mathtt{S_{2f}}\},\\ &\{\mathtt{S_{1c}}, \mathtt{S_{2\emptyset}}\}, \{\mathtt{S_{1c}}, \mathtt{S_{2d}}\}, \{\mathtt{S_{1c}}, \mathtt{S_{2f}}\}\big\}\end{aligned}$$

a. Déclenchement d'un processus : Un processus peut passer d'un état où il n'a pas lieu à un état où il a lieu *ssi* tous ses réactifs sont présents (que ce soit un EPN représentant un pool d'entités moléculaires ou un EPN de type source), les processus avec lesquels il est en conflit n'ont pas lieu, et sa modulation globale est satisfaite.

Pour un processus $p \in \mathcal{P}$, nous dénotons par $\mathsf{ready}(p)$ la collection d'ensembles d'états locaux qui satisfont cette contrainte :

$$\begin{aligned}\mathsf{ready}(p) = \big\{&\{\mathtt{e_1} \mid e \in reac(p), e \notin \cup\mathbb{S}\}\\ &\cup \{\mathtt{S_e} \mid e \in reac(p), e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}\}\\ &\cup \{\mathtt{S_\emptyset} \mid reac(p) = \emptyset, prod(p) \cap \mathfrak{S} \neq \emptyset, \mathfrak{S} \in \mathbb{S}\}\\ &\cup \{\mathtt{q_0} \mid p \# q\}\\ &\cup \ell_{cond} \mid \ell_{cond} \in \mathsf{cond}(p)\big\}\end{aligned}$$

Ainsi, pour chaque processus $p \in \mathcal{P}$, et chaque ensemble d'états locaux $\ell \in \mathsf{ready}(p)$, nous définissons une transition $t \in T$ telle que $t = \mathtt{p_0} \xrightarrow{\ell} \mathtt{p_1}$.

Exemple 6.10. Nous reprenons l'exemple que nous avons utilisé pour illustrer $\mathsf{cond}(p)$. L'ensemble $\mathsf{ready}(p)$ est alors la collection d'ensembles d'états locaux suivante :

$$\begin{aligned}\mathsf{ready}(p) = \big\{\{&\mathtt{h_1}, \mathtt{i_1}, \mathtt{g_1}\},\\ &\{\mathtt{h_1}, \mathtt{i_1}, \mathtt{S_{1\emptyset}}, \mathtt{S_{2\emptyset}}\}, \{\mathtt{h_1}, \mathtt{i_1}, \mathtt{S_{1\emptyset}}, \mathtt{S_{2d}}\}, \{\mathtt{h_1}, \mathtt{i_1}, \mathtt{S_{1\emptyset}}, \mathtt{S_{2f}}\},\\ &\{\mathtt{h_1}, \mathtt{i_1}, \mathtt{S_{1a}}, \mathtt{S_{2\emptyset}}\}, \{\mathtt{h_1}, \mathtt{i_1}, \mathtt{S_{1a}}, \mathtt{S_{2d}}\}, \{\mathtt{h_1}, \mathtt{i_1}, \mathtt{S_{1a}}, \mathtt{S_{2f}}\},\\ &\{\mathtt{h_1}, \mathtt{i_1}, \mathtt{S_{1c}}, \mathtt{S_{2\emptyset}}\}, \{\mathtt{h_1}, \mathtt{i_1}, \mathtt{S_{1c}}, \mathtt{S_{2d}}\}, \{\mathtt{h_1}, \mathtt{i_1}, \mathtt{S_{1c}}, \mathtt{S_{2f}}\}\big\}\end{aligned}$$

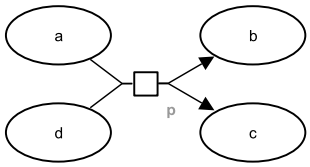
b. Arrêt d'un processus : Un processus peut passer d'un état où il a lieu à un état où il n'a pas lieu *ssi* tous ses produits sont présents (qu'ils appartiennent ou non à une histoire), et toutes les histoires qui comportent un réactif de p et aucun produit de p sont dans leur état vide.

Pour un processus $p \in \mathcal{P}$, nous dénotons par $\mathsf{done}(p)$ l'ensemble d'états locaux qui satisfait cette contrainte :

$$\begin{aligned}\mathsf{done}(p) =& \{\mathtt{e_1} \mid e \in prod(p), e \notin \cup\mathbb{S}\} \\ & \cup \{\mathtt{S_e} \mid prod(p) \cap \mathfrak{S} = \{e\}, \mathfrak{S} \in \mathbb{S}\} \\ & \cup \{\mathtt{S_\emptyset} \mid reac(p) \cap \mathfrak{S} \neq \emptyset, prod(p) \cap \mathfrak{S} = \emptyset, \mathfrak{S} \in \mathbb{S}\}\end{aligned}$$

Ainsi, pour chaque processus $p \in \mathcal{P}$, nous définissons une transition $t \in T$ telle que $t = \mathtt{p_1} \xrightarrow{\mathsf{done}(p)} \mathtt{p_0}$.

Exemple 6.11. Nous considérons le processus, dénoté par p, de la figure suivante :



Nous considérons également les deux histoires $\mathfrak{S}_1 = \{a, b\}$ et $\mathfrak{S}_2 = \{e, d\}$, et supposons que c n'appartient à aucune histoire. Notons que l'EPN e n'est pas représenté dans la carte.

Alors l'ensemble d'états locaux $\mathsf{done}(p)$ est l'ensemble suivant :

$$\mathsf{done}(p) = \{\mathtt{c_1}, \mathtt{S_{1b}}, \mathtt{S_{2\emptyset}}\}$$

c. Consommation d'un réactif n'appartenant pas à une histoire : Un EPN peut passer de son état présent à son état absent *ssi* il est réactif d'un processus qui a lieu et tous les produits de ce processus sont présents.

Ainsi, pour chaque processus $p \in \mathcal{P}$, et chaque EPN $e \in reac(p), e \notin \cup\mathbb{S}$, nous définissons une transition $t \in T$ telle que $t = \mathtt{e_1} \xrightarrow{\{\mathtt{p_1}\} \cup \mathsf{done}(p)} \mathtt{e_0}$.

d. Production d'un produit n'appartenant pas à une histoire : Un EPN peut passer de son état absent à son état présent *ssi* il est le produit d'un processus qui a lieu.

Ainsi, pour chaque processus $p \in \mathcal{P}$, et chaque EPN $f \in prod(p), e \notin \cup\mathbb{S}$, nous définissons une transition $t \in T$ telle que $t = \mathtt{f_0} \xrightarrow{\{\mathtt{p_1}\}} \mathtt{f_1}$.

e. Changement d'état d'une entité moléculaire : Pour chaque processus $p \in \mathcal{P}$ et chaque histoire $\mathfrak{S} \in \mathbb{S}(p)$ impliquée dans p :

- si $reac(p) \cap \mathfrak{S} = \{e\}$ et $prod(p) \cap \mathfrak{S} = \{f\}$, nous définissons une transition $t \in T$ telle que $t = \mathtt{S_e} \xrightarrow{\{\mathtt{p_1}\}} \mathtt{S_f}$;
- si $reac(p) \cap \mathfrak{S} = \{e\}$ et $prod(p) \cap \mathfrak{S} = \emptyset$, nous définissons une transition $t \in T$ telle que $t = \mathtt{S_e} \xrightarrow{\{\mathtt{p_1}\}} \mathtt{S_\emptyset}$;
- si $reac(p) \cap \mathfrak{S} = \emptyset$ et $prod(p) \cap \mathfrak{S} = \{f\}$, nous définissons une transition $t \in T$ telle que $t = \mathtt{S_\emptyset} \xrightarrow{\{\mathtt{p_1}\}} \mathtt{S_f}$.

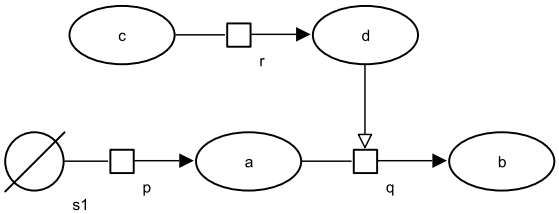
Encodage d'un état initial : Soit I un état initial *valide* du système, c'est-à-dire un ensemble d'EPNs non source et non puits de la carte présents à un instant initial donné et respectant la contrainte d'exclusivité mutuelle des EPNs d'une histoire, i.e la contrainte formelle suivante :

$$\forall (e,f) \in \mathfrak{S} \times \mathfrak{S}, \mathfrak{S} \in \mathbb{S}, e \in I \Rightarrow f \notin I$$

L'état global initial s_I du réseau d'automates est encodé à partir de I de la façon suivante :

$$\begin{aligned} s_I = &\{\mathtt{p_0} \mid p \in \mathcal{P}\} \\ &\cup \{\mathtt{e_1} \mid e \in \mathcal{E}, e \notin \cup\mathbb{S}, e \in I\} \\ &\cup \{\mathtt{e_0} \mid e \in \mathcal{E}, e \notin \cup\mathbb{S}, e \notin I\} \\ &\cup \{\mathtt{S_e} \mid e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}, e \in I\} \\ &\cup \{\mathtt{S_\emptyset} \mid \mathfrak{S} \in \mathbb{S}, \mathfrak{S} \cap I = \emptyset\} \end{aligned}$$

Exemple 6.12. Considérons la carte de la figure suivante :



Considérons également l'histoire $\mathfrak{S} = \{s_1, a, b\}$, et supposons que les EPNs c et d n'appartiennent à aucune histoire. Finalement, considérons l'état initial valide $I = \{c\}$.

Alors l'état global initial s_I est l'ensemble d'états locaux suivant :

$$s_I = \{\mathtt{p_0}, \mathtt{q_0}, \mathtt{r_0}, \mathtt{c_1}, \mathtt{d_0}, \mathtt{S_\emptyset}\}$$

6.5.5 Complexité de l'encodage

Étant donnés une carte SBGN-PD et un ensemble valide d'histoires, le nombre d'automates du RA obtenu avec notre encodage est linéaire en fonction du nombre d'EPNs et de processus de la carte. Quant à son nombre d'états globaux, il est exponentiel en fonction de ces mêmes données. Enfin, son nombre de transitions est linéaire en fonction du nombre d'EPNs et de processus, et exponentiel, étant donné un processus, en fonction du nombre d'EPNs appartenant à une histoire et apparaissant sous forme de littéraux négatifs dans les clauses de la formule associée à la modulation globale de ce processus. Le détail du calcul de la complexité de notre encodage est donné dans l'annexe D.

6.6 Relation entre la sémantique générale et la sémantique des histoires

La sémantique des histoires donne une dynamique plus contrainte que celle issue de la sémantique générale, en forçant l'exclusivité mutuelle des EPNs d'une même histoire. En particulier, la sémantique des histoires force la consommation totale des EPNs des histoires, qui sont des réactifs de processus

quelconques. Considérons un processus qui transforme une molécule a_1 en une molécule a_2, a_1 et a_2 étant deux états d'une même entité moléculaire a, et pouvant être regroupés en une histoire. Avec la sémantique générale, il est possible de produire a_2 sans consommer l'intégralité de a_1, c'est-à-dire en gardant a_1 présent dans le système ; alors qu'avec la sémantique des histoires, l'entité a passe directement de son état a_1 à son état a_2, c'est-à-dire d'un état global du système où a_1 est présent et a_2 absent à un état global où a_1 est absent et a_2 est présent. Intuitivement, la contrainte d'exclusivité mutuelle a pour conséquence que la dynamique issue de la sémantique des histoires est une sous-dynamique de celle issue de la sémantique générale. Nous donnons cette propriété plus formellement dans ce qui suit.

Soit une carte SBGN-PD. Nous utilisons les mêmes notations que précédemment pour définir les éléments de cette carte. Pour rappel, $\mathcal{E}$ désigne l'ensemble des EPNs de cette carte, $\mathcal{P}$ l'ensemble de ses processus, et $\mathcal{O}$ l'ensemble de ses opérateurs logiques.

Soit $RA_{gen} = (\Sigma_{gen}, S_{gen}, T_{gen})$ le réseau d'automates modélisant la dynamique de cette carte avec la sémantique générale. Finalement, soit $\mathbb{S}$ un ensemble valide d'histoires quelconque de cette carte, et $RA_{st} = (\Sigma_{st}, S_{st}, T_{st})$ le réseau d'automates modélisant cette même carte avec la sémantique des histoires en considéreant l'ensemble valide d'histoires $\mathbb{S}$.

Nous rappelons que par commodité, nous voyons un état global d'un RA comme un ensemble d'états locaux plutôt que comme un tuple.

À chaque état global $s \in S_{st}$ de RA_{st} correspond un état global $[\![s]\!] \in S_{gen}$ de RA_{gen}, dans lequel un EPN appartenant à une histoire est dans son état 1 (présent) *ssi* l'histoire à laquelle il appartient est dans l'état associé à cet EPN dans s, et dans son état 0 (absent) sinon. Formellement, nous avons la définition et la propriété suivantes :

Définition 6.10 (Relation $[\![]\!]$)**.** Soit $s \in S_{st}$ un état global de RA_{st}. L'ensemble d'états locaux $[\![s]\!]$ est défini de la manière suivante :

$$\begin{aligned}[\![s]\!] = &\{\mathtt{p_x} \mid \mathtt{p_x} \in s, p \in \mathcal{P}\} \\ &\cup \{\mathtt{e_x} \mid \mathtt{e_x} \in s, e \in \mathcal{E} \setminus \cup\mathbb{S}\} \\ &\cup \{\mathtt{e_1} \mid e \neq \emptyset, e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}, \mathtt{S_e} \in s\} \\ &\cup \{\mathtt{e_0} \mid e \neq \emptyset, e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}, \mathtt{S_e} \notin s\}\end{aligned}$$

Propriété 6.1. Soit $s \in S_{st}$ un état global de RA_{st}. Alors $[\![s]\!]$ est un état global de RA_{gen}, i.e. $[\![s]\!] \in S_{gen}$.

Exemple 6.13 (Relation $[\![]\!]$)**.** Considérons la carte SBGN-PD des figures 6.2 et 6.5, et l'ensemble valide d'histoires $\{\mathfrak{s}, \mathfrak{t}\}$, défini dans la figure 6.5. Considérons le RA modélisant cette carte avec la sémantique générale, dénoté par RA_{gen} et représenté dans la figure 6.2, et le RA modélisant cette même carte avec la sémantique des histoires en considérant l'ensemble valide d'histoires $\{\mathfrak{s}, \mathfrak{t}\}$, dénoté par RA_{st} et représenté dans la figure 6.5. Finalement, considérons l'état global $s = \{\mathtt{m_1}, \mathtt{b_1}, \mathtt{s_a}, \mathtt{p_0}, \mathtt{t_{atp}}, \mathtt{q_0}\}$ de RA_{st}, représenté en bleu dans le réseau d'automates de la figure 6.5.

L'ensemble d'états locaux $[\![s]\!] = \{\mathtt{p_0}, \mathtt{q_0}, \mathtt{m_1}, \mathtt{b_1}, \mathtt{a_1}, \mathtt{atp_1}, \mathtt{aP_0}, \mathtt{c_0}, \mathtt{adp_0}\}$ est bien un état global de RA_{gen}. Cet état est représenté en bleu dans le RA de la figure 6.2.

Les deux propositions suivantes établissent la relation entre la sémantique générale et la sémantique des histoires en termes d'atteignabilité :

Proposition 6.1. Soient $s, s' \in S_{st}$ deux états globaux de RA_{st} dans lesquels aucun processus n'a lieu. Si s' est atteignable à partir de s dans RA_{st}, alors $[\![s']\!]$ est atteignable à partir de $[\![s]\!]$ dans RA_{gen}.

Proposition 6.2. Soient s et s' deux états globaux de RA_{st} dans lesquels aucun processus n'a lieu. Si $[\![s']\!]$ est atteignable à partir de $[\![s]\!]$ dans RA_{gen}, alors s' n'est pas nécessairement atteignable à partir de s dans RA_{st}.

Les preuves des propositions 6.1 et 6.2 sont établies dans l'annexe D.

Nous pouvons déduire des propositions 6.1 et 6.2 des relations sur les points attracteurs des deux sémantiques.

Les points attracteurs de RA_{gen} qui ont un état global correspondant dans RA_{st} sont des points attracteurs de RA_{st} :

Propriété 6.2. Soit s un état global de RA_{st} tel que $[\![s]\!]$ soit un point attracteur de RA_{gen}. Alors s est un point attracteur de RA_{st}.

Finalement, les points attracteurs de RA_{st} ne sont pas nécessairement des points attracteurs de RA_{gen} :

Propriété 6.3. Soit s un point attracteur de RA_{st}. Alors $[\![s]\!]$ n'est pas nécessairement un point attracteur de RA_{gen}.

Ces deux propriétés sont établies dans l'annexe D.

Les figures 6.6 et 6.7 montrent les graphes de transitions des RAs donnés dans les figures 6.2 et 6.5, respectivement. Une version zoomable de la figure 6.6 est diponible à l'adresse `www.lri.fr/~rougny/sgsemgen.pdf`. Le premier graphe est le graphe de transitions du RA construit avec la sémantique générale, et le deuxième celui du RA construit avec la sémantique des histoires. Ces deux graphes de transitions ont été construits à partir des états initiaux donnés dans ces mêmes figures.

Dans les deux graphes de transitions, les états qui sont des points attracteurs sont entourés.

Le graphe de transitions du modèle construit avec la *sémantique générale* contient 88 états. Il comporte neuf attracteurs, qui sont tous des points attracteurs. Cinq de ces points attracteurs comportent les deux états locaux $\mathtt{aP}_1$ et $\mathtt{c}_1$, signifiant que dans ces états, les EPNs aP et c sont présents en même temps. Ainsi, dans le modèle construit avec la sémantique générale, il est possible de produire à la fois aP et c, l'un après l'autre. Il est possible de produire ces EPNs de deux manière différentes : soit le processus p a lieu en premier et est suivi du processus q, soit q a lieu en premier et il est suivi du processus p. Dans chacun de ces deux cas, a n'est consommé que partiellement par le premier processus qui a lieu, laissant a present dans le système (i.e. dans son état $\mathtt{a}_1$), et de ce fait la possibilité au deuxième processus d'avoir lieu. Deux des autres attracteurs contiennent les états locaux $\mathtt{aP}_1$ et $\mathtt{c}_0$, et deux autres les états $\mathtt{aP}_0$ et $\mathtt{c}_1$. Les deux premiers sont atteints lorsque le processus p a lieu en premier,

et les deux derniers lorsque le processus q a lieu en premier. Pour ces quatre derniers cas, le premier processus à avoir lieu consomme totalement a, rendant cet EPN absent du système (i.e., dans l'état $\mathtt{a}_0$), et empêchant l'autre processus d'avoir lieu.

Le modèle construit avec la *sémantique des histoires* ne comporte que 11 états, dont trois sont des points attracteurs. Ceci illustre comment la sémantique des histoires induit une sémantique de moindre taille, comparée à celle induite par la sémantique générale. Deux attracteurs contiennent l'état local $\mathtt{s}_\mathtt{c}$, signifiant que l'entité moléculaire a est dans son état complexé à b. Seulement un des attracteurs contient l'état local $\mathtt{s}_{\mathtt{aP}}$, signifiant que a est dans son état phosphorylé. Comme pour les trois points attracteurs, aucun état global ne contient en même temps les deux états locaux $\mathtt{s}_\mathtt{c}$ et $\mathtt{s}_{\mathtt{aP}}$. En effet, les EPNs c et aP appartiennent à la même histoire, et sont par conséquent mutuellement exclusifs. Parmi les deux points attracteurs contenant l'état local $\mathtt{s}_\mathtt{c}$, l'un contient $\mathtt{b}_0$, et l'autre $\mathtt{b}_1$: quand le processus q a lieu, il peut consommer l'EPN b soit partiellement soit totalement, laissant b présent dans le système ou le rendant absent du système, étant donné que b n'appartient pas à une histoire.

Les états globaux du modèle construit avec la sémantique générale qui correspondent aux états globaux du modèle construit avec la sémantique des histoires sont colorés en violet dans le graphe de transition de la figure 6.6. La partie du graphe de transition du modèle construit avec la sémantique générale qui correspond à la dynamique du modèle construit avec la sémantique des histoires est colorée en violet dans la figure 6.6. Cette partie correspond aux transitions qui modélisent la consommation totale des EPNs qui sont dans les histoires utilisées pour construire le modèle avec la sémantique des histoires. Comme la consommation totale des réactifs d'un processus et la production des produits de ce même processus requièrent plus d'étapes (i.e. de déclenchements de transitions locales) dans la sémantique générale que dans la sémantique des histoires, le nombre d'états globaux colorés en violet dans la figure 6.6 est plus grand que le nombre d'états globaux du graphe de transitions de la figure 6.7.

6.7 Application à la carte de la régulation du cycle cellulaire par RB/E2F

Dans cette section, nous illustrons comment les deux sémantiques permettent de modéliser une grande carte SBGN-PD. Nous avons choisi de modéliser la carte RB/E2F, qui contient 222 EPNs. Cette carte est représentée dans la figure 6.8. Elle a été construite et publiée par les auteurs de [Cal+08] et rendue disponible à l'adresse [E2f] au format CellDesigner en deux versions : la carte dans son intégralité, et la carte sans les processus transcriptionnels et traductionnels. C'est cette dernière carte qui est représentée dans la figure 6.8. Dans cette étude, nous considérons la carte sans les processus transcriptionnels et traductionnels, pour deux raisons. D'abord, les processus transcriptionnels et traductionnels tels que représentés au format CellDesigner ne sont pas conformes au langage SBGN-PD. Ensuite, le sous-réseau de la carte faisant entrer en jeu les différentes protéines et le sous-réseau faisant entrer en jeu les processus transcriptionnels et traductionnels sont bien distincts l'un de l'autre : seul le réseau de protéines a une influence sur la partie génétique, sans rétrocontrôle de cette dernière sur le réseau protéique. Par conséquent, la partie génétique de la carte n'a pas à proprement parler d'influence sur la dynamique du réseau.

La carte de la figure 6.8 représente les mécanismes moléculaires entrant en jeu dans la régulation du cycle cellulaire en se focalisant sur la transition G1 contrôlée par la protéine du rétinoblastome (RB) et les facteurs de transcription E2F. Le cycle cellulaire se décompose en quatre phases successives (G1, S, G2 et M) qui sont finement régulées par un ensemble de points de contrôle. RB joue un rôle crucial dans le cycle en contôlant l'entrée du cycle dans sa phase S, phase dans laquelle l'ADN est répliqué. Sa fonction moléculaire principale est l'inhibition d'E2F1, et, en inhibant E2F1, RB empêche la transcription des gènes ciblés par ce facteur de transcription. Différentes kinases cycline-

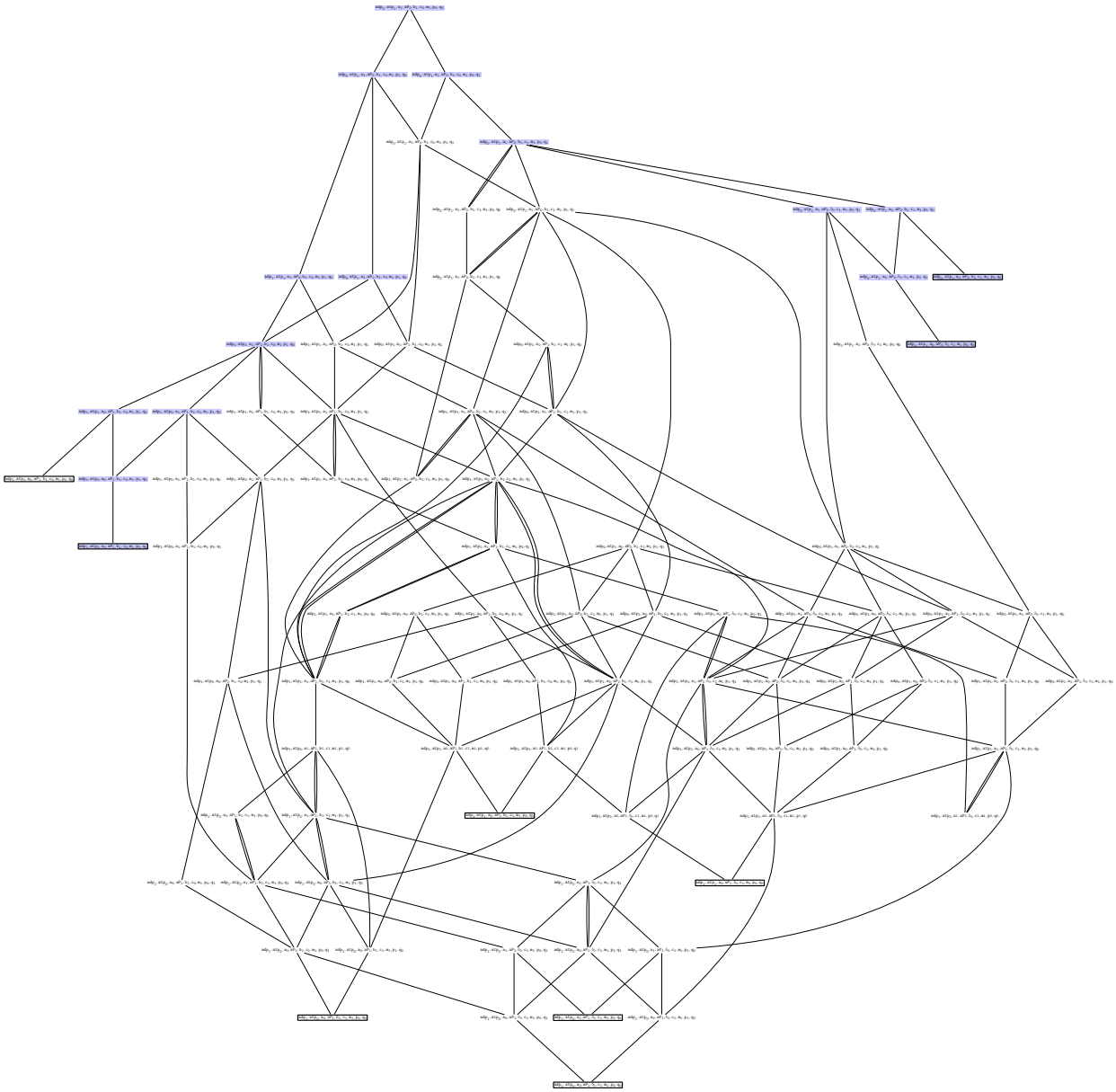


FIGURE 6.6 – **Graphe des transitions asynchrones du RA de la figure 6.2.** Ce graphe est le graphe des transitions asynchrones du RA de la figure 6.2 modélisant la carte SBGN-PD donnée dans la même figure avec la sémantique générale. Il est construit pour l'état global initial $(\mathtt{adp}_0, \mathtt{atp}_1, \mathtt{a}_1, \mathtt{aP}_0, \mathtt{b}_1, \mathtt{c}_0, \mathtt{m}_1, \mathtt{p}_0, \mathtt{q}_0)$.
Chaque noeud du graphe représente un état global du RA. Il y a un arc partant d'un noeud associé à un état global s et arrivant sur un noeud associé à un état global s' *ssi* s' est atteignable à partir de s. Les noeuds entourés d'une bordure épaisse sont associés à des points attracteurs. Les états colorés en bleu sont également présents dans le graphe des transitions asynchrones du RA modélisant la même carte mais avec la sémantique des histoires. Une version zoomable de cette figure est diponible à l'adresse www.lri.fr/~rougny/sgsemgen.pdf.

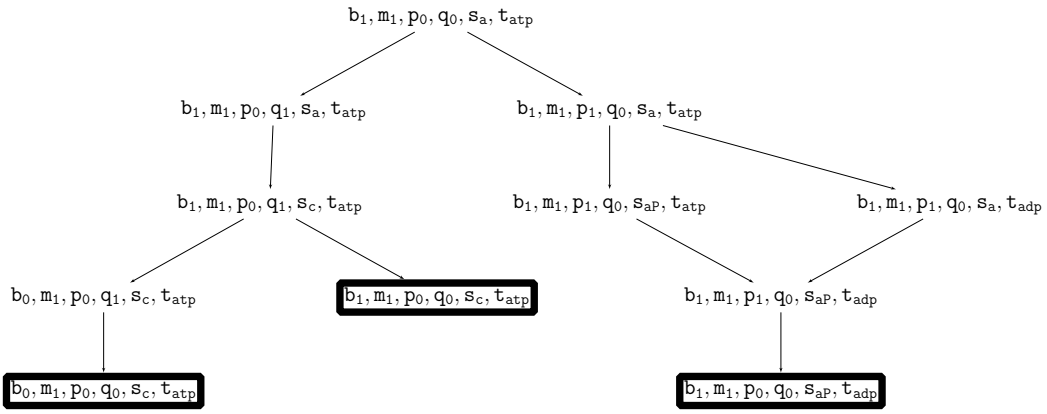


FIGURE 6.7 – **Graphe des transitions asynchrones du RA de la figure 6.5.** Ce graphe est le graphe des transitions asynchrones du RA de la figure 6.5 modélisant la carte SBGN-PD donnée dans la même figure avec la sémantique des histoires. Il est construit pour l'état global initial $(b_1, m_1, p_0, q_0, s_a, t_{atp})$.
Chaque noeud du graphe représente un état global du RA. Il y a un arc partant d'un noeud associé à un état global s et arrivant sur un noeud associé à un état global s' *ssi* s' est atteignable à partir de s. Les noeuds entourés d'une bordure épaisse sont associés à des points attracteurs.

dépendantes (CDK) interviennent dans différentes phases du cycle cellulaire et jouent un rôle majeur dans se régulation. En particulier, les CDKs phosphorylent RB, libérant ainsi petit à petit son emprise sur les facteurs E2F. Les CDKs ne sont actives qu'associées à leur cycline. Il y a six CDKs majeures dans le cycle cellulaire : CDC2 (aussi appelée CDK1), CDK2, CDK3, CDK4, CDK6 et CDK7. CDC2 est associée à la cycline B1, CDK2 aux cyclines E1 et A2, CDK3 à la cycline C, CDK4 et CDK6 à la cycline D1 et CDK7 à la cycline H. Quant aux facteurs de transcriptions E2F, certains sont des activateurs de la transcription (E2F1, E2F2 et E2F3a), et d'autres sont des inhibiteurs de la transcription (E2F3bn E2F4, E2F5, E2F6 et très probablement E2F7 et E2F8).

La stimulation des cellules par des facteurs de croissance (comme EGF) induisent un changement d'état de ces cellules, qui sortent de la phase G0 (dite phase de quiescence) et entrent dans le cycle cellulaire. Les complexes cycline D1-CDK4,6 sont activés et commencent à phosphoryler RB, responsable majeur du point de contrôle de la phase G1. Alors que la protéine RB commence à être phosphorylée, son emprise sur E2F1 s'amoindrit, et E2F1 se détache du complexe inhibiteur. E2F1 commence alors à activer la synthèse d'acteurs majeurs du cycle, via son effet sur la transcription. À la fin de la phase G1, la concentration de la cycline E1 atteint un pic, et le complexe cycline E1-CDK2 fait passer les cellules de la phase G1 à la phase S. C'est dans cette phase que l'ADN est répliqué. Après la réplication, les cellules entrent dans la phase G2, en raison principalement de l'action du complexe cycline A2-CDK2. Les cellules entrent finalement dans la phase M, phase dont le complexe cycline B1-CDC2 semble être un des régulateurs majeurs.

6.7.1 Construction de modèles avec les deux sémantiques

Afin de constuire des modèles dynamiques avec chacune des deux sémantiques, nous avons d'abord converti la carte CellDesigner dans le format SBGN-ML. Pour ce faire, nous avons utilisé la fonction d'export vers SBGN-ML du logiciel CellDesigner pour obtenir un premier fichier SBGN-ML. La fonction d'export de CellDesigner ne convertissant que les positions géographiques des noeuds et des arcs sans convertir les relations qu'entretiennent ces noeuds et ces arcs, nous avons ensuite corrigé le fichier

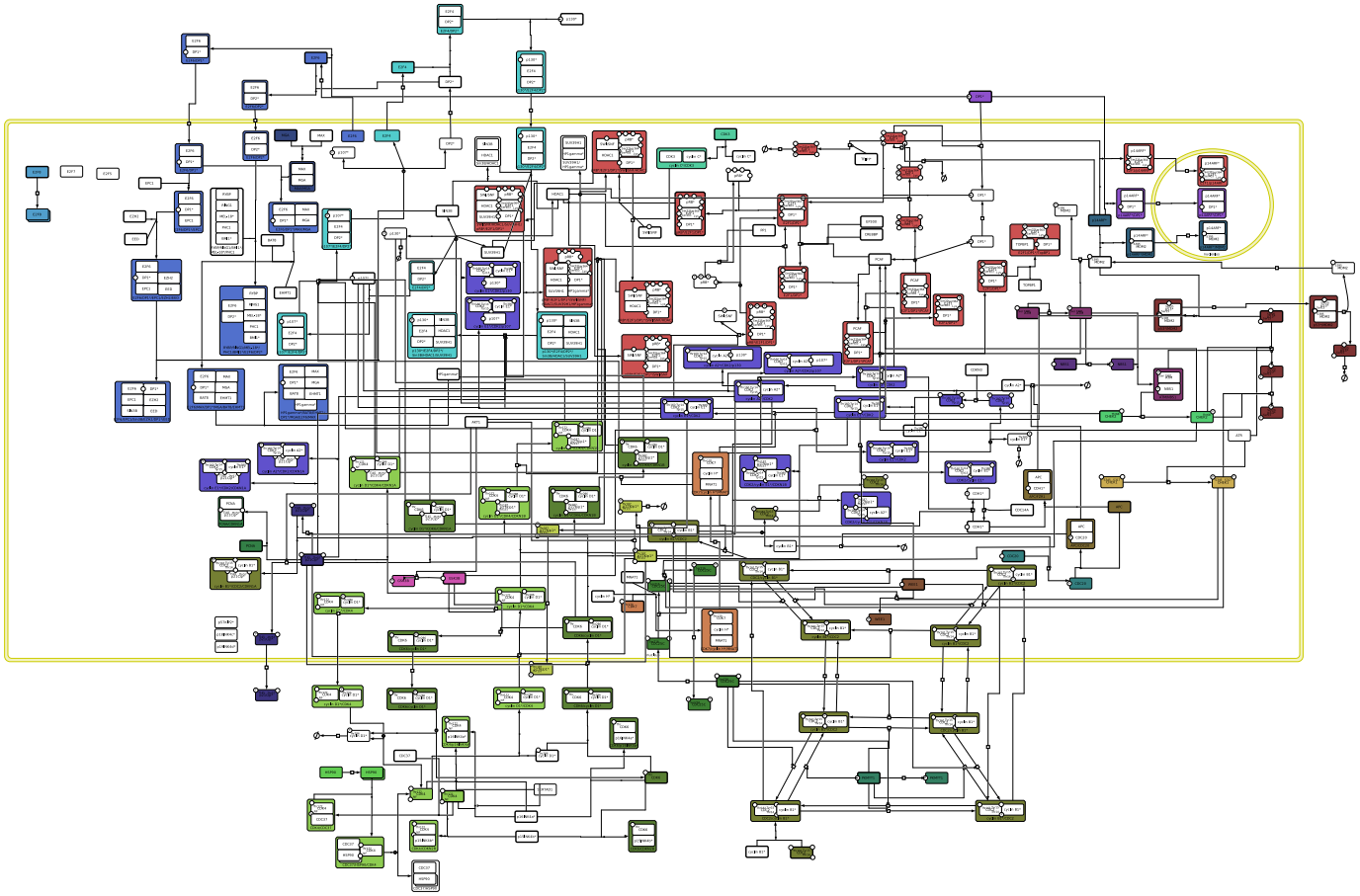


FIGURE 6.8 – **Carte SBGN-PD de la régulation du cycle cellulaire par RB/E2F.** Le cycle cellulaire est une succession de quatre phases (G1, S, G2 et M) qui sont finement régulées par les protéines de la famille des protéines pocket, dont le représentant principal est la protéine du rétinoblastome (RB). La principale fonction de cette protéine est l'inhibition des facteurs de transcription appartenant à la fammile E2F, et en particulier de la protéine E2F1. Diverses cyclines dépendantes des kinases (CDK) jouent un rôle majeur dans la régulation du cycle. Une des fonctions de ces CDK est la phosphorylation de la protéine RB, qui diminue son effet inhibiteur sur les facteurs E2F.
Cette carte est adaptée de la carte CellDesigner disponible à l'adresse [E2f]. Elle est représentée en utilisant le langage SBGN-PD.
Les EPNs avec une bordure épaisse constituent un état initial de la carte. Chaque EPN représenté en couleur appartient à une histoire, et chaque couleur est associée à une histoire différente de la carte. Les histoires représentées respectent les contraintes (i–iv). L'ensemble d'histoires représenté est epn-maximal.
Des zooms de cette carte dont donnés dans les figures 6.9–6.12.

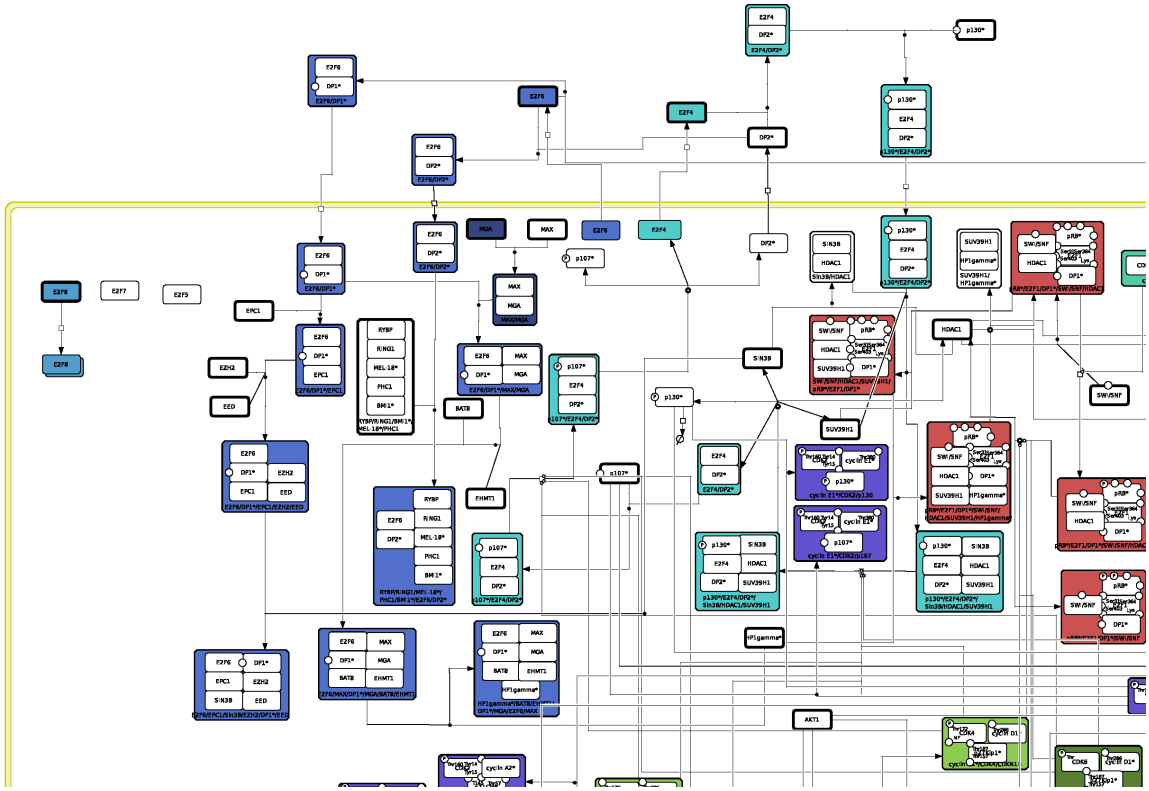


FIGURE 6.9 – **Zoom sur le carte SBGN-PD de la figure 6.6 : partie en haut à gauche.** Voir la légende de la figure 6.8.

SBGN-ML obtenu en y ajoutant les informations se trouvant dans le fichier CellDesigner d'origine et non converties.

Nous avons ensuite contruit nos deux modèles dynamiques à partir de ce fichier SBGN-ML.

Le modèle avec la sémantique générale a été construit automatiquement par transformation directe des éléments de la carte en un réseau d'automates. Le modèle obtenu contient 370 automates.

Afin de construire le modèle avec la sémantique des histoires, nous avons d'abord choisi un ensemble valide d'histoires calculé de la manière suivante. Comme les facteurs de transcriptions E2F et les CDKs jouent un rôle majeur dans la régulation du cycle, nous avons défini préalablement une histoire pour chaque CDK et une pour chaque facteur E2F, chacune contenant tous les EPNs représentant un état physique distinct de l'entité moléculaire modélisée. Nous avons également défini une histoire pour l'entité p53. Enfin, nous avons calculé tous les ensembles epn-maximaux d'histoires contenant les histoires définies préalablement. Les ensembles obtenus étaient au nombre de huit, ce qui s'explique par la présence dans la carte de trois processus d'associations avec deux réactifs chacun (nommément les couples de réactifs {MGA,MAX}, {ATM,NBS1} et {APC,CDC20}). Ces trois associations ont entraîné la formation de trois couples d'histoires alternatives, et le nombre total d'ensembles epn-maximaux résulte de la combinaison de ces trois couples. Chacun des huit ensembles obtenus contenait exactement 28 histoires pour un total de 153 EPNs, sur les 222 de la carte. Nous avons choisi l'ensemble epn-maximal contenant une histoire pour chacune des molécules MGA, ATM et APC. Cet ensemble est représenté sur la figure 6.8, où chaque couleur correspond à une histoire différente.

À partir de cet ensemble valide d'histoires et de la carte, nous avons construit automatiquement

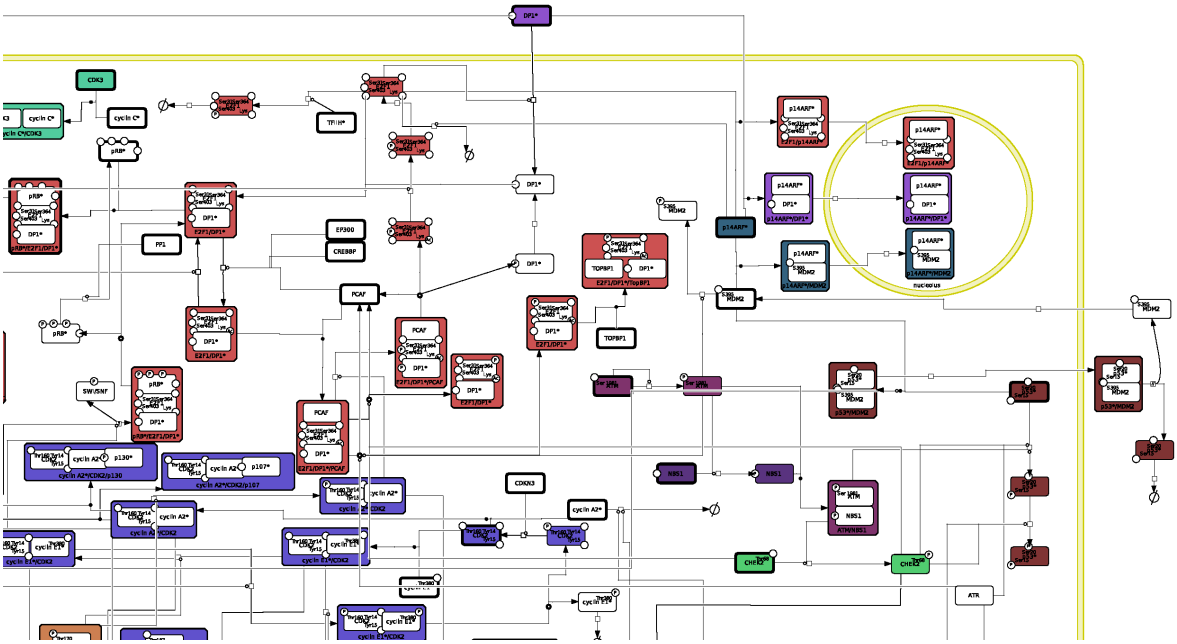


FIGURE 6.10 – **Zoom sur le carte SBGN-PD de la figure 6.6 : partie en haut à droite.** Voir la légende de la figure 6.8.

un modèle avec la sémantique des histoires. Ce modèle contient 243 automates, soit environ un tiers de moins que le modèle construit avec la sémantique générale.

6.7.2 Étude de la succession des phases du cycle cellulaire

Afin d'illustrer comment des modèles construits avec chacune de deux sémantiques peuvent être utilisés pour vérifier des propriétés intéressantes du réseau SBGN-PD sous-jacent, nous avons étudié la succession des phases du cycle cellulaire dans chacun des deux modèles. Pour ce faire, nous avons utilisé deux logiciels : Pint [Pin] et Mole [Mol]. Pint permet de réduire des modèles formalisés par des réseaux d'automates tout en conservant une propriété d'atteignabilité donnée et de convertir ces modèles en réseaux de Petri, tandis que Mole est un *déplieur* de réseaux de Petri permettant la vérification de propriétés d'atteignabilité.

Construction d'un état initial. Nous avons d'abord construit un état initial pour les deux modèles qui modélisent l'état d'une cellule dans sa phase G0, juste après qu'elle a été stimulée par un facteur de croissance, c'est-à-dire quand CDK4 et CDK6 sont présents dans le système. Un EPN est dans l'état initial *ssi* il est considéré comme présent à l'état initial. Il en est absent sinon.

Nous avons donc inclus dans l'état initial tous les EPNs qui ne peuvent pas être produits par un processus, ainsi que les EPNs qui ne peuvent être produits que par un processus qui appartient à un

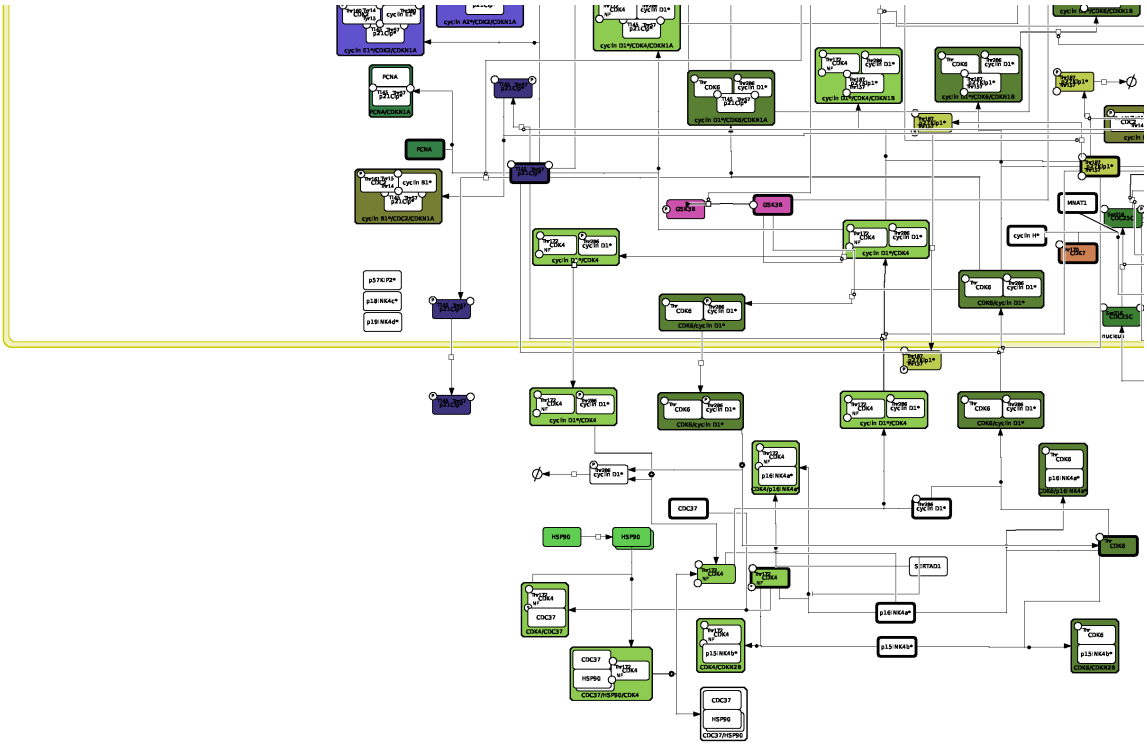


FIGURE 6.11 – **Zoom sur le carte SBGN-PD de la figure 6.6 : partie en bas à gauche.** Voir la légende de la figure 6.8.

cycle de la carte : la protéine E2F4 du cytosol et le complexe pRB-E2F1-DP1 du noyau. Les EPNs inclus dans l'état initial sont dessinés avec une bordure noire dans la carte de la figure 6.8.

Marqueurs de phases. Nous avons associé à chaque phase du cycle cellulaire un ensemble d'EPNs qui sont des *marqueurs* pour cette phase. Nous présupposons que le système (ici, une cellule ou un ensemble de cellules) est dans une certaine phase à un instant donné si au moins un des marqueurs de cette phase est présent dans le système à cet instant. Par exemple, nous avons associé à la phase G2 l'ensemble des EPNs qui représentent le complexe cycline B1-CDC2 phosphorylé ou non du cytosol.

Chacune des phases G1 et S est séparée en deux périodes appelées *précoce* et *tardive*, afin de mieux caractériser ces transitions.

Nous donnons d'abord deux définitions concernant l'atteignabilité de phases, que nous utiliserons dans l'étude de la succession des phases :

Définition 6.11 (Présence d'un marqueur de phase)**.** Soit $RA = (\Sigma, S, T)$ un réseau d'automates, construit avec l'une ou l'autre des sémantiques, et e un marqueur d'une phase. Le marqueur e est dit *présent* dans un état global $s \in S$ de RA *ssi* :

- $\mathtt{e}_1 \in s$ si $e \notin \cup\mathbb{S}$;
- $\mathtt{S}_\mathtt{e} \in s$ si $e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}$.

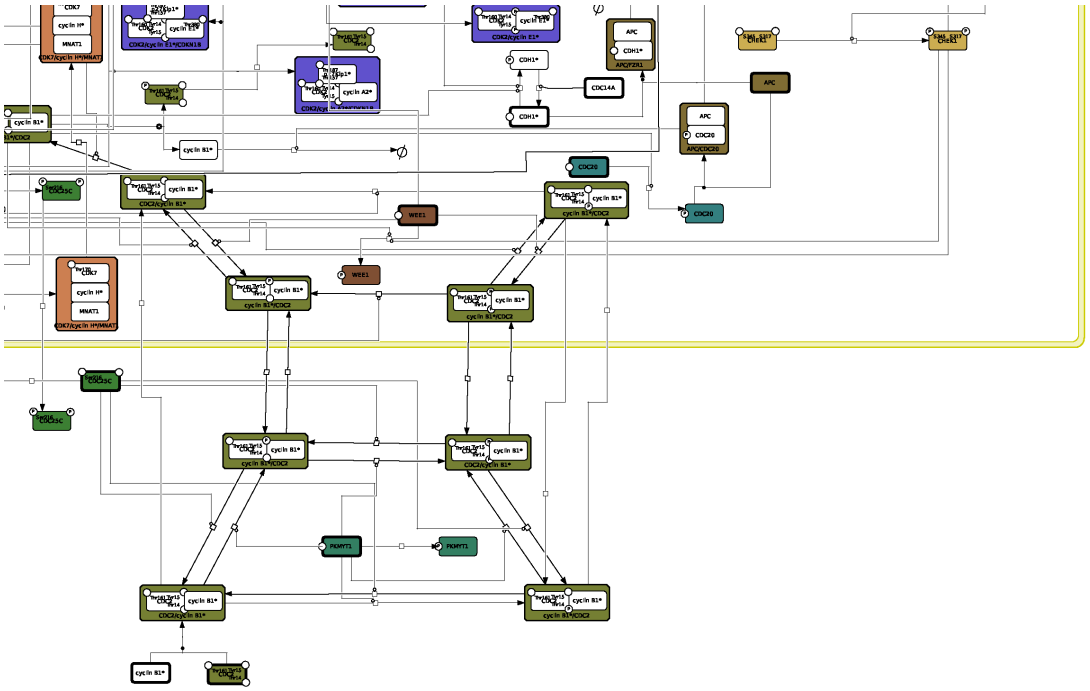


FIGURE 6.12 – **Zoom sur le carte SBGN-PD de la figure 6.6 : partie en bas à droite.** Voir la légende de la figure 6.8.

Définition 6.12 (Atteignabilité d'une phase)**.** Soient $RA = (\Sigma, S, T)$ un réseau d'automates, construit avec l'une ou l'autre des sémantiques, ph une phase du cycle, et $s \in S$ un état global de RA.

La phase ph est *atteignable* à partir de s *ssi* il existe au moins un marqueur e de ph et un état $s' \in S$ tels que $s \rightarrow^* s'$ et e est présent dans s'.

Soit s_I un état global initial de RA. La phase ph est atteignable *ssi* elle est atteignable de s_I.

Définition 6.13 (Atteignabilité simultanée de deux phases)**.** Soient $RA = (\Sigma, S, T)$ un réseau d'automates, construit avec l'une ou l'autre des sémantiques, ph et qh deux phases du cycle, et $s \in S$ un état global de RA.

Les phases ph et qh sont simultanément atteignables à partir de s *ssi* il existe un marqueur e de ph, un marqueur f de qh, et un état $s' \in S$ tels que $s \rightarrow^* s'$, et e et f sont présents dans s'.

Dans un modèle, un marqueur de phase peut être désactivé en supprimant toutes les transitions sortant et arrivant sur l'état local correspondant à l'état présent de cet EPN. Une phase peut quant à elle être désactivée en désactivant tous ses marqueurs. Par conséquent, une phase désactivée n'est plus atteignable.

Succession des phases dans les modèles initiaux. Afin de vérifier si les différentes phases du cycle cellulaire sont atteintes successivement dans chacun des deux modèles, nous avons d'abord vérifié que chacune des phases pouvait être atteinte à partir de l'état initial. Nous avons constaté que toutes les phases pouvaient être atteintes à partir de l'état initial, dans chacun des deux modèles. Nous avons ensuite vérifié que chacune des phases était encore atteignable en bloquant E2F1 dans son état initial. Comme E2F1 a un rôle central dans la régulation du cycle cellulaire, interdire tout changement d'état de cette entité aurait dû empêcher certaines phases, sinon toutes, d'être atteintes. Il s'est avéré que toutes les phases sauf la phase G1 précoce étaient encore atteignables, dans chacun des deux modèles.

Afin d'aller plus loin dans le test de la validité de nos modèles, nous avons étudié la succession des phases proprement dite, dans chacun des deux modèles. Nous nous attendions à ce que, à part pour la phase G1 précoce, toutes les phases soient atteintes successivement, et dans l'ordre. Nous avons donc testé, pour chaque phase, si elle pouvait être atteinte lorsque la phase précédente était désactivée. Toutes les phases sauf les phases G1 tardive et M étaient toujours atteignables malgré la désactivation de leurs phases précédentes.

Ces deux résultats, l'un obtenu par le blocage d'E2F1, l'autre par la désactivation, pour chaque phase, de sa phase précédente, montrent que nos modèles, construits seulement à partir de la carte SBGN-PD initiale, ne permettaient pas de reproduire une succession des phases correcte, et que, par conséquent, la succession des phases observée lors du cycle cellulaire ne semble pas pouvoir être expliquée par les seuls processus moléculaires représentés par cette carte. Il apparaît que les différentes phases du cycle peuvent être atteintes par des voies qui sont parallèles et indépendantes les unes des autres dans la carte, et que seule la séquentialité de ces voies pourrait autoriser une succession des phases correcte.

La séquentialité des voies peut être établie de plusieurs façons : par exemple, en considérant la cinétique des processus, ou en considérant des processus additionnels qui lieraient ces différentes voies entre elles.

Dans le cadre de cette étude, nous avons choisi cette dernière option. Nous avons ajouté à nos modèles des transitions modélisant différents effets transcriptionnels, et étudié la succession des phases du cycle dans ces modèles augmentés.

Succession des phases dans les modèles avec effets transcriptionnels Afin de modéliser la succession des phases observée lors du cycle cellulaire, nous avons enrichi chacun de nos deux modèles initiaux avec des effets transcriptionnels d'E2F1 connus. En effet, E2F1 agit sur la transcription de gènes dont les protéines associées ont un rôle majeur dans la régulation du cycle. Par exemple, il est connu qu'E2F1 active la transcription de CDC2 [Cal+08]. Comme la forme particulière d'E2F1 qui régule la transcription de CDC2 n'est pas connue, nous avons d'abord considéré qu'E2F1 pouvait activer la transcription de CDC2 s'il était associé seulement à DP1, ou associé à une forme phosphorylée de RB, étant donné que la forme non phosphorylée de RB inhibe E2F1. Nous avons modélisé cet effet dans chacun des deux modèles en ajoutant une transition de l'état local modélisant CDC2 absent à l'état local modélisant CDC2 présent, que nous avons conditionnée à la présence d'E2F1 dans un de ses états mentionnés ci-avant.

Nous avons ajouté ce type d'influence pour quatre des principaux régulateurs du cycle, à savoir les cyclines E1, A2, B1, et CDC2, en accord avec les effets transcriptionnels d'E2F1 observés dans la littérature [Cal+08 ; Bra+04].

Notons que ces effets transcriptionnels ne sont pas présents tels quels dans la carte CellDesigner qui comprend la partie génétique du cycle, et n'auraient pas pu être modélisés de façon concrète à partir de cette carte. En effet, ce sous-réseau ne décrit que des modulations de la transcription, sous la forme d'activations et d'inhibitions de la transcription de certains gènes en ARN opérées par des protéines génériques comme E2F1, en ne décrivant ni les processus de traduction, ni les formes particulières avec

lequelles les molécules opèrent ces modulations.

Toutes les phases du cycle étaient atteignables dans les modèles enrichis avec les effets transcriptionnels. Cependant, aucune phase, mises à part les phases G1 tardive, S précoce et G2, n'était atteignable lorsque sa phase précédente avait été préalablement désactivée. Nous sommes parvenus à empêcher l'atteignabilité de la phase S en réduisant les formes d'E2F1 ayant un effet activateur sur la transcription de la cycline A2 aux complexes où E2F1 est associé à DP1 seulement ou à la protéine RB phosphorylée trois fois. Ceci suggère que l'augmentation de la quantité de cycline A2 après la phase G1, qui induit le remplacement de la cycline E1 par la cycline A2 dans les complexes cycline E1-CDK2, pourrait être indirectement provoquée par la phosphorylation de RB sur son troisième site de phosphorylation par le complexe cycline E1-CDK2. Quant à la succession des phases S tardive et G2, elle pourrait être rétablie en ajoutant un effet positif du complexe cycline A2-CDK2 sur l'activation du complexe cycline B1-CDC2. Un tel effet a été vérifié dans la littérature (voir [GF10]), mais son mécanisme précis, et en particulier la forme dans laquelle le complexe opère son effet, reste à notre connaissance inconnu.

Finalement, nous avons vérifié, pour chacun des deux modèles enrichis, si les différentes phases du cycle pouvaient être atteintes simultanément deux à deux. Pour chaque paire de phases, nous avons vérifié s'il existait un état atteignable dans lequel un marqueur de chacune des deux phases était présent. Dans le modèle construit avec la sémantique générale, toutes les paires de phases pouvaient être atteintes simultanément, alors que dans celui construit avec la sémantique des histoires, les paires (S précoce, S tardive) et (G2, M) ne pouvaient pas l'être.

Cette différence observée entre les modèles provient de la contrainte d'exclusivité mutuelle des EPNs d'une même histoire. Si deux marqueurs associés à deux phases différentes appartiennent à la même histoire, alors il se peut que ces deux phases ne soient pas atteignables simultanément. Cette dernière analyse illustre comment la sémantique des histoires peut aider à raisonner sur des processus biologiques pour lesquels des états fonctionnels successifs d'entités moléculaires clés peuvent être liés à des événements biologiques observés à une échelle plus large.

6.8 Workflow

Touts les scripts et et toutes les commandes utilisés pour les analyses présentées ci-dessus sont disponibles à l'adresse `https://github.com/pauleve/sbgnpd2an-suppl`. Le workflow de notre méthode est représenté dans la figure 6.13.

Pour n'importe quelle carte SBGN-PD écrite au format SBGN-ML, les ensembles valides d'histoires peuvent être calculés. Puis deux modèles dynamiques peuvent être construits, l'un avec la sémantique générale, l'autre avec la semantique des histoires, en prenant en compte un ensemble valide d'histoires choisi parmi les ensembles calculés.

Les modèles dynamiques obtenus peuvent ensuite être analysés par une variété de méthodes et de logiciels, et en particulier vérifiés, comme c'est le cas dans notre illustration.

6.9 Discussion

6.9.1 Travaux connexes

Des notions similaires aux histoires peuvent être trouvées dans la littérature.

Composants moléculaires. Dans [PMMS15], les auteurs présentent un algorithme semi-automatique pour trouver les *composants moléculaires* d'un réseau de réactions. Ils définissent un composant moléculaire comme une entité biologique pouvant se retrouver sous la forme de différentes espèces moléculaires

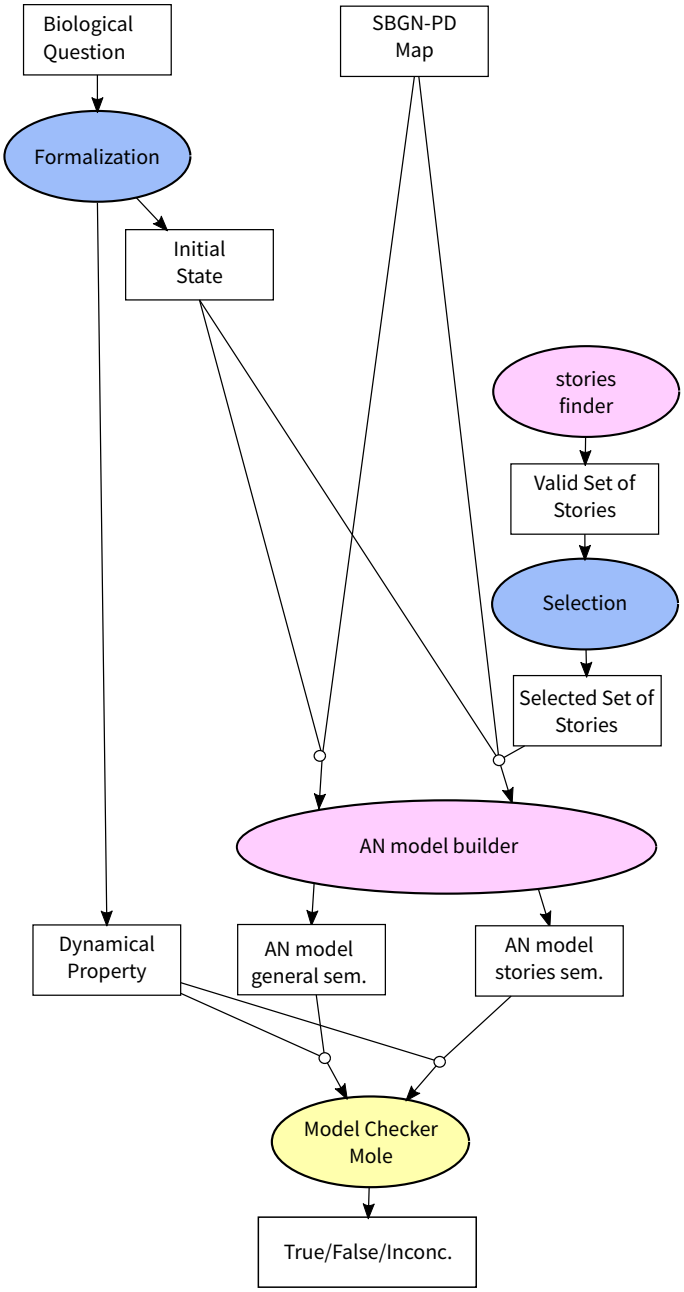


FIGURE 6.13 – **Workflow de notre méthode.** Les rectangles représentent des objets et les ellipses des tâches. Les tâches colorées en rose sont celles développées dans le cadre des travaux présentés dans ce chapitre ; celles colorées en bleu correspondent à des interventions de l'utilisateur ; finalement, celles colorées en jaune sont exécutées par l'intermédiaire d'outils diponibles publiquement.

dans le réseau. Ainsi, un composant moléculaire est un nom d'espèce associé à un ensemble d'espèces moléculaires qui partagent ce nom. Leur algorithme pour trouver de tels composants repose sur la loi de conservation de la masse, et procède itérativement comme suit :

- choisir arbitrairement une réaction du réseau qui n'a pas encore été examinée ;
- associer chaque réactif de la réaction à un produit de la réaction distinct, ou à un produit divisé en deux parties (en ajoutant de nouveau symboles), et mémoriser ces associations et divisions ;
- mettre à jour les associations déjà réalisées dans les autres réactions en accord avec les nouvelles associations et divisions.

Dans le cas où des ambiguïtés surviendraient lors de la phase d'association des réactifs aux produits, l'algorithme demande à l'utilisateur quelle est la bonne association.

Les histoires respectant les contraintes (i-iv) que nous avons définies et les composants moléculaires définis par les auteurs de [PMMS15] visent tous deux à modéliser les changements d'états d'entités moléculaires. La différence principale entre ces deux notions est que les éléments d'un même composant ne sont pas contraints à l'exclusivité mutuelle, comme c'est le cas pour les EPNs d'une même histoire. Ainsi, les composants moléculaires ne sont pas construits à partir de contraintes définies pour rendre compatbile une contrainte d'exclusivité mutuelle et une sémantique pour la dynamique comme c'est le cas pour les histoires, et dans le cas général, ne peuvent pas directement être utilisés pour construire un modèle dynamique qualitatif.

Nous illustrons cette différence sur un petit exemple. Nous considérons un réseau evec deux processus : le premier processus transforme un EPN A en un EPN B, et le second processus est une association entre A et B pour donner le complexe C.

Considérant ces deux processus, il n'y a qu'une seule histoire respectant les contraintes (i-iv) : $\{A, C\}$. Avec ce même réseau, l'algorithme de [PMMS15] calcule automatiquement un seul composant, associé à l'ensemble d'espèces $\{A, B, C_A, C_B\}$, où C_A et C_B sont les parties provenant de la division de C. Ce composant n'est pas pertinent pour construire un modèle dynamique avec la sémantique des histoires : associé à un seul automate dont les états locaux seraient les éléments du composant, A et B ne seraient jamais présents en même temps dans le système, et par conséquent le processus d'association ne pourrait jamais avoir lieu. Ainsi, la notion de composant moléculaire n'est pas aussi pertinente que celle des histoires si l'on vise à construire des modèles dynamiques, comme dans notre cas.

BiNoM. Dans [Cal+08], les auteurs décomposent le réseau RB/E2F en 16 modules à l'aide du plugin de Cytoscape [Sha+03] nommé BiNoM [Zin+08], de la façon suivante. D'abord, des modules sont construits en décomposant le réseau en sous-réseaux, chacun se focalisant sur une entité moléculaire. Les sous-réseaux obtenus qui se recoupent par plus de 30% sont ensuite fusionnés. Finalement, les modules obtenus à la phase précédente sont modifiés à la main afin de donner à chacun un sens biologique. Dans la plupart des cas, les modules obtenus intègrent une protéine d'intérêt et toutes ses formes modifiées (phosphorylées, acétylées, etc.), ainsi que ses modulateurs (p.ex. des kinases). Des influences entre modules sont dérivées des influences des molécules qui les composent, et l'ensemble des modules et influences forment une carte modulaire, semblable à un réseau d'influences.

En résumé, l'approche utilisée par BiNoM se focalise sur la structure du réseau détaillé fourni par la carte SBGN-PD. Elle abstrait et simplifie ce réseau pour construire un graphe d'influences afin d'identifier des motifs clés du réseau de départ, comme des boucles d'interactions négatives, qui peuvent être responsables de certains comportements dynamiques, mais sans fournir de modèle dynamique. Au contraire, la sémantique des histoires conserve le niveau de détail de la carte SBGN-PD initiale, tout en donnant des contraintes sur sa dynamique.

6.9.2 Deux sémantiques pour différents types de réseaux

La sémantique générale étend celle de BIOCHAM en prenant en compte les inhibitions. Cette sémantique peut être appliquée à n'importe quelle carte SBGN-PD, c'est-à-dire à n'importe quel réseau modélisant des processus moléculaires précis. C'est le cas pour la majorité des réseaux métaboliques, et certains réseaux de signalisation ou de régulation, comme ceux présentés dans ce chapitre.

Quant à la sémantique des histoires, elle a un champ d'application plus restreint. Elle ne peut en effet être appliquée qu'aux réseaux modélisant des changements d'états physiques d'entités moléculaires. Les réseaux présentant de telles transformations sont principalement des réseaux de signalisation ou de régulation. Utiliser la sémantique des histoires pour modéliser des réseaux métaboliques aurait peu de sens de manière générale, vu qu'il est difficile dans ces réseaux de définir des entités moléculaires ayant plusieurs états physiques distincts. Néanmoins, l'application de cette sémantique à certains réseaux métaboliques pourrait tout de même être envisagée. Prenons l'exemple du processus de la photosynthèse chez les plantes. Il s'agit d'un processus qui fait entrer un jeu une succession de réactions d'oxydo-réduction, avec comme entrée une molécule d'eau, de l'énergie et du CO_2, en sortie de l'O_2 et une molécule carbonée réduite, et comme intermédiaire un ensemble de protéines (comme les photosystèmes I et II) et de coenzymes (comme le $NADP^+$). Ce processus peut être vu comme un flux d'électrons entre les réactifs initiaux et les produits finaux, où chacun des intermédiaires recevra des électrons de l'intermédiaire précédent pour les donner à l'intermédiaire suivant. Ainsi, les électrons, qui passent de molécule en molécule, pourraient être vus comme une entité moléculaire, et suivis par une histoire, dont les éléments seraient les molécules qui recoivent les électrons.

En conclusion, la sémantique générale a un champ d'application plus large que la sémantique des histoires. Elle peut naturellement être appliquée aux réseaux métaboliques, ce qui n'est pas le cas pour l'autre sémantique. Cependant, appliquée au réseaux qui modélisent des transformations physiques successives d'entités moléculaires, la sémantique des histoires permet de construire des modèles de dimension réduite comparé à la sémantique générale. De plus, la sémantique des histoires pourrait permettre de mieux modéliser les processus biologiques comportant des événements discrets successifs, comme c'est la cas pour la régulation du cycle cellulaire.

6.9.3 Sémantique des histoires et sémantique Booléenne pour des réseaux d'influences

Comme nous l'avons montré dans le chapitre 3, les réseaux de signalisation représentés en SBGN-PD peuvent être transformés en graphes d'influences représentés en SBGN-AF. Nous discutons ici de la relation entre la sémantique des histoires appliquée à une carte SBGN-PD et la sémantique Booléenne appliquée à la carte SBGN-AF obtenue par transformation de cette carte SBGN-PD. Nous prenons comme exemple la carte SBGN-PD de l'activation d'ERK de la figure 6.4, que nous avons transformée en une carte SBGN-AF représentée dans la figure 3.14, au chapitre 3.

La sémantique des histoires, de par sa définition, semble bien adaptée à la modélisation des réseaux de signalisation. Dans les cascades de signalisation représentées en SBGN-PD, comme la voie G de la carte de l'activation d'ERK, chaque entité moléculaire apparait le plus souvent sous la forme de deux EPNs différents, chacun représentant un état physique différent de l'entité moléculaire. Une telle entité pourra alors être modélisée par une histoire contenant les deux EPNs. Par conséquent, le réseau d'automates construit avec la sémantique des histoires et modélisant une telle voie contiendrait un automate formé de deux états locaux pour chaque entité moléculaire. Un des états locaux de l'automate modéliserait l'état actif de l'entité moléculaire, et l'autre son état inactif.

Ces voies sont transformées en SBGN-AF en des voies linéaires, constituées d'une activité pour chaque entité moléculaire de la voie. Une modélisation d'une telle voie représentée en SBGN-AF par un réseau Booléen amènerait à modéliser chaque activité par une variable Booléenne. La notion d'histoire

pourrait alors être confondue avec celle d'activité moléculaire, et un automate modélisant une histoire serait analogue à une variable Booléenne modélisant une activité.

Cependant, cette correspondance ne tient pas pour des voies de signalisation plus complexes, comme la voie β-arrestine de la carte figure 6.4. En effet, dans ces voies, les entités moléculaires ont plus de deux états physiques différents, et plusieurs états physiques d'une même entité peuvent être des états actifs de cette entité. Ainsi, pour ces voies, à une histoire correspondrons plusieurs activités, et par conséquent, les automates modélisant les histoires ne seront plus analogues aux variables Booléennes modélisant les activités. Par exemple, pour la voie β-arrestine, nous avons défini une histoire qui modélise l'entité moléculaire HR. Dans le RA construit avec la sémantique des histoires, cette entité est modélisée par un unique automate. Or cette entité correspond à six activités dans la carte SBGN-AF obtenue par transformation de la carte SBGN-PD d'origine. Par conséquent, le réseau Booléen construit à partir de la carte SBGN-AF contiendra six variables Booléennes, toutes modélisant cette entité.

6.9.4 Taille des histoires et nombre d'EPNs total

Le calcul des ensembles valides d'histoires pour la carte RB/E2F suggère qu'il existe un compromis entre le nombre d'histoires d'un ensemble valide et leur taille. Afin de modéliser cette carte, nous avons calculé un ensemble epn-maximal étant donné un certain nombre d'histoires construites au préalable. Cet ensemble contient 28 histoires pour un total de 153 EPNs. Or les ensembles epn-maximaux calculés sans définition au préalable d'histoires contiennent tous 42 histoires pour un total de 162 EPNs, c'est-à-dire plus d'histoires, mais des histoires contenant en moyenne moins d'EPNs. Ainsi, pour cet exemple, augmenter la taille des histoires semble causer une diminution du nombre d'histoires des ensembles epn-maximaux. Ce compromis peut être illustré par un exemple simple. Considérons une carte SBGN-PD contenant deux processus, l'un consommant A pour produire B, l'autre consommant B et C pour produire D. Étant données les contraintes (i-iv), cette carte a deux ensembles maximalement valides d'histoires : $\big\{\{A,B\},\{C,D\}\big\}$ et $\big\{\{A,B,D\}\big\}$. Seul le premier ensemble est epn-maximal, et nous observons que son ratio nombre d'EPNs total/nombre d'histoires est inférieur à celui du deuxième ensemble (4/2 contre 3/1).

6.9.5 Encodage des deux sémantiques à l'aide de réseaux de Petri

Le formalisme des réseaux d'automates est proche de celui des réseaux de Petri. En effet, tout RA peut être traduit en un réseau de Petri (RP) équivalent. Étant donné un RA, son encodage en un RP se fait de la manière suivante : tout état local de tout automate du RA est encodé en une place dans le RP ; toute transition locale du RA est encodée en une transition dans le RP avec un arc entrant et un arc sortant de cette place ; les conditions des transitions locales du RA sont encodées par des arcs de lecture dans le RP.

Exemple 6.14 (Encodage d'un réseau d'automates asynchrone en un réseau de Petri équivalent)**.** La figure 6.14 montre l'encodage du RA de l'exemple 6.1 en un RP équivalent. Même si nous les avons représentées côte à côte, les places $\mathtt{a}_1$, $\mathtt{a}_2$ et $\mathtt{a}_3$ sont *a priori* indépendantes les unes des autres. Ce n'est qu'en montrant que ces places sont mutuellement exclusives qu'on peut déduire qu'elles modélisent les états locaux du composant a.

Ainsi, toute sémantique exprimée à l'aide de réseaux d'automates peut être exprimée, avec une expressivité équivalente, à l'aide de réseaux de Petri. Cependant, les réseaux de Petri ne permettent pas de regrouper explicitement des états locaux sous forme de composants (i.e. de parties appartenant

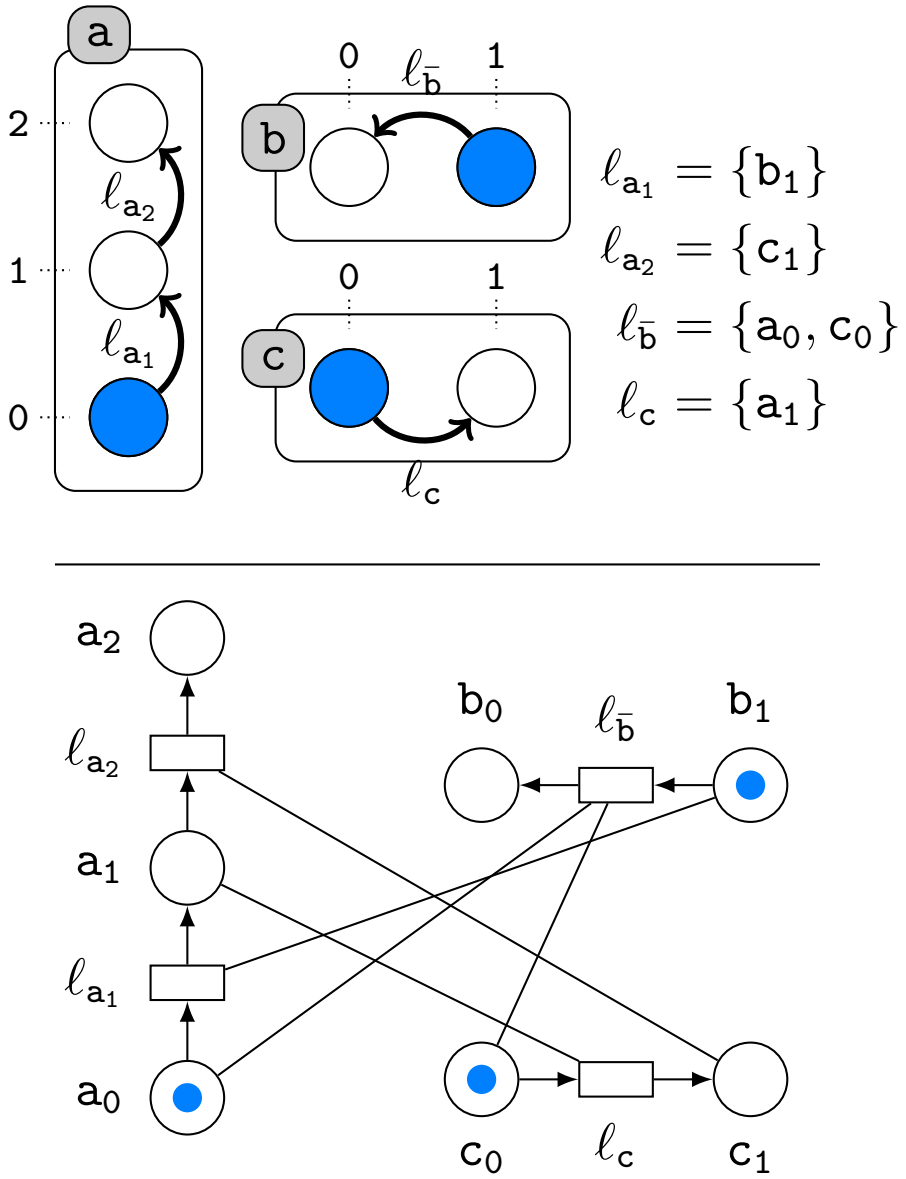


FIGURE 6.14 – **Encodage d'un réseau d'automates en réseau de Petri.** En haut : le réseau d'automates de la figure 6.1. En bas : le réseau de Petri (RP) correspondant. Les cercles représentent des *places*, les rectangles des *transitions*. Les flèches représentent des *arcs* entrant sur les transitions et sortant des transitions, et les arêtes représentent des *arcs de lecture*. Finalement, les points bleus représentent des *jetons*.
Un *marquage* du RP est une distribution de jetons dans les places. Une transition peut être déclenchée *ssi* toutes les places liées à la transition par un arc entrant ou un arc de lecture ont au moins un jeton. Quand une transition est déclenchée, tous les jetons sont retirés des places liées à cette transition par un arc entrant.
Dans le RP, toutes les places sont *a priori* indépendantes les unes des autres, et par conséquent, les composants `a`, `b` et `c` restent implicites. Ils ne peuvent être découverts qu'en calculant les ensembles de places mutuellement exclusives du RP, à partir de sa structure et de sa dynamique.

à un système), au contraire des réseaux d'automates. Les composants d'un réseaux de Petri peuvent être vus comme les ensembles de places mutuellement exclusives de ce réseau (i.e. à un instant donné, il n'y a qu'une seule place d'un tel ensemble qui soit marquée). Or les places d'un réseaux de Petri sont *a priori* indépendantes les unes des autres, et ce n'est que par un calcul non trivial que ces ensembles peuvent être explicités, en prenant en compte la structure du réseau, et un éventuel marquage initial.

6.10 Conclusion et perspectives

Nous avons introduit deux nouvelles sémantiques qualitatives pour modéliser la dynamique des réseaux de réactions. Ces sémantiques prennent en compte les concepts les plus importants du langage SBGN-PD. La sémantique générale est une sémantique Booléenne qui étend la sémantique de BIOCHAM en considérant des réseaux de réactions pouvant comporter des inhibitions et des opérateur logiques. Quant à la sémantique des histoires, elle offre un nouveau point de vue sur les réseaux de réactions, en se basant sur le concept d'histoire. Une histoire d'un réseau de réactions est un ensemble d'états physiques d'une même entité moléculaire qui apparaissent par transformation de cette entité. En rendant ces états physiques mutuellement exclusifs, la sémantique des histoires se place d'une certaine façon à mi-chemin entre la sémantique générale et la sémantique Booléenne des graphes d'influences. Cette sémantique prend tout son sens pour la modélisation de la dynamique des réseaux de signalisation ou de régulation qui comportent de nombreuses protéines pouvant être dans différents états post-traductionnels. Les modèles obtenu à l'aide de cette sémantique sont de moindre taille comparé à ceux obtenus à l'aide de la sémantique générale, et prennent cependant en compte le détail des mécanismes moléculaires propres aux réseaux de réaction.

La définition formelle des histoires est basée sur un certain nombre de contraintes structurelles des réseaux de réaction, qui ont pour objectif d'empêcher certains comportements (ou l'absence de comportements) dans la dynamique obtenue à l'aide de la sémantique des histoires. Des contraintes additionnelles pourraient être considérées. Par exemple, dans la définition actuelle des histoires, nous n'interdisons pas le cas où une histoire contient le réactif d'un processus et l'unique stimulateur de ce même processus. Or une telle histoire empêcherait le processus d'avoir lieu. D'autres contraintes de ce type pourraient sans doute être ajoutées. Il faudrait, pour les trouver, définir formellement nos exigences vis-à-vis des processus, qui traduiraient par exemple que la contrainte d'exclusivité mutuelle ne doit pas empêcher un processus de se déclencher localement. Des contraintes additionnelles pourraient ensuite être dérivées à partir de ces exigences.

Nous projetons également de développer un logiciel, qui, à terme, prendra en compte l'intégralité du workflow présenté dans ce chapitre. Ce logiciel permettra notamment de visualiser tous les ensembles valides d'histoires calculés, et facilitera le choix nécessaire de l'ensemble d'histoires adéquat pour la modélisation avec la sémantique des histoires.

Conclusion et perspectives

Sommaire

Conclusion

Un des concepts au cœur de la biologie des systèmes est celui de *réseau moléculaire*. Un réseau moléculaire représente les relations qu'entretiennent des entités moléculaires participant à un processus biologique donné. Deux tâches fondamentales de la biologie des systèmes sont la construction de réseaux moléculaires à partir de données expérimentales d'une part, et l'analyse de la dynamique de ces réseaux d'autre part. Un des objectifs principaux de la biologie des systèmes est donc de proposer des méthodes automatiques permettant de résoudre ces deux tâches, capables de prendre en compte l'ensemble des expériences et des concepts de la biologie des systèmes, et qui reposent sur des standards.

Nos contributions à ces deux tâches sont données ci-après, et sont résumées dans la figure 6.15.

Dans le premier chapitre, nous avons introduit deux nouveaux langages de la logique du premier ordre qui permettent d'exprimer les réseaux de réactions et les graphes d'influences. Ces deux langages, que nous avons nommés SBGNLog-PD et SBGNLog-AF, ont été construits directement à partir des langages SBGN-PD et SBGN-AF, respectivement. Nous avons ensuite présenté un cas d'utilisation de ces langages, en proposant une méthode de transformation automatique des réseaux de réactions en graphes d'influences, basée sur ASP.

Dans le deuxième chapitre, nous avons proposé une méthode de construction des réseaux de signalisation à l'échelle des mécanismes moléculaires, basée sur l'interprétation automatique de résultats expérimentaux de différents types, qui procède d'une démarche déductive. Nous avons également montré comment le processus d'interprétation pouvait être inversé pour proposer des plans expérimentaux permettant de tester une hypothèse biologique donnée (raisonnement abductif). Nous avons illustré notre méthode en reconstruisant le réseau de signalisation induit par le récepteur de la FSH, et proposé un ensemble de plans expérimentaux pour valider une hypothèse sur la phosphorylation de MEK par p38MAPK, que nous avions préalablement émise à l'aide de notre méthode.

Dans le troisième chapitre, nous avons montré comment un réseau Booléen construit à partir d'une carte SBGN-AF et paramétré à l'aide de principes généraux pouvait être exprimé sous la forme de deux programmes logiques normaux du premier ordre, formés notamment d'axiomes traduisant ces principes généraux. Nous avons ensuite proposé une méthode de calcul des points attracteurs et des traces finies de ce réseau Booléen à l'aide de ces deux programmes logiques.

Dans le quatrième chapitre, nous avons proposé deux nouvelles sémantiques qualitatives pour la dynamique des réseaux de réactions. Ces sémantiques prennent en compte les principaux concepts du langage SBGN-PD. La première de ces sémantiques, dite sémantique générale, étend la sémantique

Booléenne de BIOCHAM en prenant notamment en compte les inhibitions et les opérateurs logiques. Quant à la deuxième sémantique, dite sémantique des histoires, elle permet de modéliser l'ensemble des états physiques d'une même entité moléculaire par une unique variable. Nous avons illustré comment ces deux sémantiques pouvaient être utilisées en construisant et en étudiant des modèles dynamiques de la carte de la régulation du cycle cellulaire par RB/E2F, qui contient plus de 200 noeuds. Nous avons en particulier illustré comment les modèles construits à l'aide de ces sémantiques pouvaient être vérifiés.

L'ensemble de nos travaux de thèse portent donc sur deux tâches fondamentales de la biologie des systèmes, qui sont la construction des réseaux moléculaires d'une part, et leur analyse d'autre part, et que nous avons présentées plus haut.

Les méthodes permettant la réalisation de l'une ou de l'autre tâche sont le plus souvent quantitatives, et reposent sur des valeurs numériques qui sont parfois difficiles à obtenir. *A contrario*, nous avons proposé un ensemble de méthodes qualitatives, qui ne reposent pas sur ces valeurs numériques. Si les formalismes qualitatifs ont largement prouvé leur utilité pour l'analyse de la dynamique des réseaux, ce n'est pas le cas pour la construction de ces derniers. Le formalisme de la logique du premier ordre, que nous avons utilisé dans notre contribution à cette tâche, s'est révélé suffisamment expressif pour raisonner de façon précise sur les mécanismes moléculaires sous-jacents aux processus biologiques. De plus, nous avons pu tirer profit des différents modes de raisonnement (ici, déductif et abductif), qui vont de paire avec ce formalisme, pour bâtir une méthode permettant non seulement la construction de réseaux moléculaires détaillés, mais aussi la proposition de plans expérimentaux pour tester une hypothèse biologique donnée. Nous avons également utilisé, dans chacun des travaux présentés dans ce manuscrit, la programmation ASP, qui nous a permis d'exprimer et de réaliser, de façon simple, un ensemble de tâches de raisonnement plus ou moins complexes.

Un autre point commun à l'ensemble des méthodes que nous avons proposées est la prise en compte de réseaux exprimés dans un standard de la biologie des systèmes, qui est la notation SBGN. D'un point de vue théorique, l'utilisation de ce standard nous a permis de donner une base conceptuelle commune à l'ensemble de nos travaux. D'un point de vue plus pratique, l'utilisation systématique des langages SBGN nous a permis d'intégrer nos différentes méthodes dans un même framework, qui est schématisé dans la figure 6.15 : la dynamique des réseaux de réactions SBGN-PD construits à l'aide de notre méthode du chapitre 4 peut être analysée à l'aide des sémantiques qualitatives que nous avons proposées dans le chapitre 6. Ces réseaux peuvent également être transformés automatiquement en graphes d'influences SBGN-AF par notre méthode du chapitre 3 ; et leur dynamique Booléenne peut ensuite être calculée par les programmes logiques normaux que nous avons introduits au chapitre 5. Aussi, l'utilisation de la notation SBGN permet d'inscrire nos différents travaux dans l'effort de la communauté de la biologie des systèmes pour faciliter l'échange et la réutilisation des réseaux et modèles produits par la discipline.

Perspectives

Dans la suite, nous donnons quelques perspectives générales à nos différents travaux.

Prendre en compte les différents sens apportés aux modulations. Comme nous l'avons déjà mentionné dans les perspectives du chapitre 4, nous avons entrepris l'élaboration d'un cadre théorique pour notre méthode de construction des réseaux de réactions, cadre dans lequel nous souhaiterions également pouvoir fonder notre méthode de complétion des graphes d'influences. L'élaboration d'un tel cadre théorique nécessite de définir précisément les relations de modulation, qui sont un des concepts centraux de la biologie des systèmes. Or, au cours de nos différents travaux, nous nous sommes aperçu

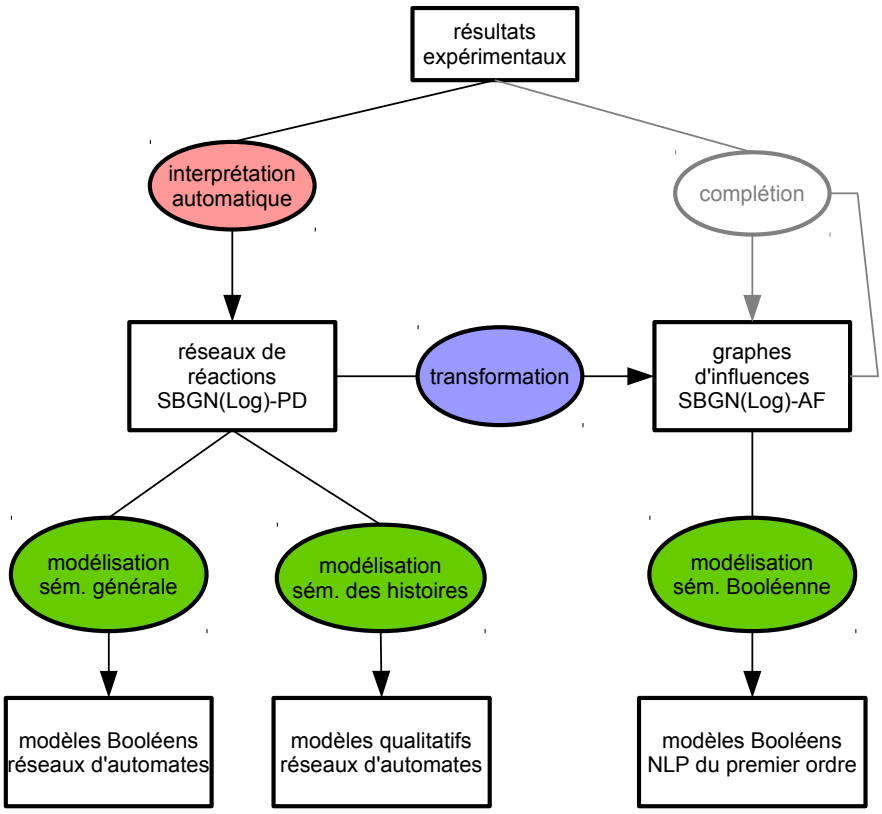


FIGURE 6.15 – **Schéma récapitulatif de nos différents travaux.** Les rectangles représentent des objets d'étude de la biologie des systèmes, et les ellipses des processus. Le travail que nous avons réalisé sur la complétion (représenté en gris) n'a pas été présenté dans ce manuscrit.

que le sens que nous apportions aux modulations dépendait de l'étude menée. L'exemple qui nous semble le mieux illustrer cette différence de sens apportée aux modulations est le suivant. Considérons deux activités A et B, et deux résultats expérimentaux relatifs à ces activités. Supposons que le premier de ces résultats nous amène à conclure que A stimule B, et le deuxième que A inhibe B. Les deux énoncés qui ont été déduits sont, du point de vue de la biologie tout à fait compatibles, et ont été déduits indépendamment l'un de l'autre. Cependant, lorsqu'on essaye d'interpréter ces deux modulations en termes de causalité, ils deviennent incompatibles. Par exemple, ces deux modulations pourraient être interprétées par la conjonction d'énoncés suivante : "la présence de A cause la présence de B" et "la présence de A cause l'absence de B". Or il n'est bien entendu pas possible que la présence de A cause à la fois la présence de B et son absence. La solution employée pour produire un énoncé causal qui ait du sens est alors d'interpréter ces deux modulations en même temps, c'est-à-dire d'apporter du sens au couple de modulations plutôt qu'à chacune de ces modulations considérées indépendamment l'une de l'autre. C'est la manière qui est par exemple employée pour construire des réseaux Booléens, où c'est la modulation globale d'une activité qui est interprétée, c'est-à-dire l'ensemble des modulations ciblant une activité. Par conséquent, lorsque des modulations sont le produit de l'interprétation de résultats expérimentaux, nous leur donnons un sens individuel ; et lorsque ces dernières sont interprétées par des énoncés de causalité, nous donnons un sens à plusieurs modulations considérées ensemble plutôt qu'à chacune d'entre elles.

De même, vu la façon dont nous déduisons les modulations à partir de résultats expérimentaux, la validité d'un énoncé comme "A stimule B" dépend *a priori* du système biologique dans lequel est

opérée l'activité, et de l'état de ce système.

L'ensemble de ces considérations nous a permis d'ébaucher une distinction des différents sens apportés aux modulations, qu'il nous reste encore à approfondir. Cette distinction nous semble être une condition requise à l'établissement d'un cadre commun à l'ensemble de nos travaux, ainsi qu'à la poursuite de travaux sur la fusion des réseaux moléculaires SBGN, que nous présentons ci-après.

Fusion des réseaux SBGN. Nous avons introduit dans le chapitre 3 une méthode permettant la transformation des réseaux de réactions SBGNLog-PD en graphes d'influences SBGNLog-AF, à l'aide d'un programme ASP. Nous souhaiterions, sur le même principe, proposer une méthode de fusion de deux cartes SBGN-PD ou deux cartes SBGN-AF. Comme les connaissances sur un système sont le plus souvent représentées sous la forme de réseaux moléculaires, une telle méthode permettrait un transfert des connaissances d'un système à l'autre. Elle permettrait ainsi de produire de réseaux moléculaires plus complets.

Une telle méthode ne consisterait pas en une "simple" fusion de graphes, pour deux raisons, que nous donnons brièvement ci-après. Premièrement, un même concept biologique peut être représenté de façon plus ou moins précise. Prenons l'exemple du processus de phosphorylation d'ERK sur un site particulier. Il est possible de représenter ce processus en spécifiant le site de phosphorylation ; mais il est aussi possible de ne pas spécifier le site dans sa représentation. Si deux cartes SBGN-PD que l'on veut fusionner représentent chacune ce processus, la première en mentionnant le site de phosphorylation, et la deuxième sans le mentionner, il faudra être capable de reconnaître que c'est le même processus qui est représenté, afin de n'avoir qu'une seule représentation du processus (de préférence la plus précise) dans la carte issue de la fusion. Il faudra donc, dans un premier temps, établir une relation d'ordre "plus précis que" entre différents motifs représentant les mêmes concepts biologiques. Certaines instances de cette relation sont évidentes, comme celles impliquant des motifs représentant des concepts organisés par une ontologie. D'autres sont plus complexes, notamment lorsqu'elles impliquent des motifs représentant des processus moléculaires.

Deuxièmement, comme nous l'avons suggéré plus haut, la fusion de cartes nécessite selon nous d'avoir préalablement établi des distinctions claires entre les différents sens apportés aux modulations, et notamment de déterminer dans quelle mesure un énoncé comme "A stimule B" peut avoir un sens indépendamment de la prise en compte d'un système biologique et d'un de ses états particuliers. Considérons par exemple trois activités A, B et C, ainsi que deux cartes SBGN-AF, que l'on veut fusionner. La première de ces carte représente la stimulation de B par A, et la deuxième la stimulation de C par B. En fusionnant ces cartes sans précaution, on serait amené à contruire une carte contenant la stimulation de C par B, et celle de B par A. Mais est-on certain que, dans les conditions dans lesquelles il a été établi que A stimule B, B peut bien stimuler C ? Cette question est, à nos yeux, essentielle, et nécessite de distinguer de prime abord le sens apporté aux modulations d'une carte qui ont été toutes établies dans des conditions expérimentales plus ou moins identiques (c'est-à-dire, établies pour des des systèmes qui sont semblables), du sens des modulations d'une carte obtenue par fusion. Cette distinction nous paraît en premier lieu importante en vue de la construction de modèles causaux (Booléens par exemple). En effet, il est *a priori* possible que, dans le système où on a montré que A stimule B, la présence de A cause l'absence de C (nous ne pouvons pas le savoir avant d'avoir réalisé l'expérience adéquate). Or une interprétation causale de la carte fusionnée sans distinction de sens aucune pourrait nous amener à un modèle qui montre exactement le contraire.

Modélisation des cartes SBGN-ER. L'ensemble des travaux que nous avons présentés dans ce manuscrit portent sur les graphes d'influences SBGN-AF et les réseaux de réactions SBGN-PD. Nous avons tout juste mentionné SBGN-ER, qui est le dernier des langages SBGN. SBGN-ER permet de représenter des relations entre entités moléculaires (comme des interactions ou des modulations) sans

représenter des processus moléculaires précis. Outre le fait que chacun des trois langages permet de représenter des concepts différents, le langage SBGN-ER a une particularité supplémentaire. En effet, la sémantique de ce langage est donnée dans sa spécification, c'est-à-dire que la définition de la syntaxe du langage est accompagnée des règles indiquant comment il faut interpréter les différents concepts représentés par ce langage. Par exemple, la spécification de ce langage donne deux sémantiques alternatives pour la stimulation. Une stimulation, dont la source est une entité E et la cible une relation R, signifie que : "si E existe alors R est renforcée" ("if E exists then R is reinforced"), ou bien encore que "si E existe alors la probabilité de R est augmentée" (if E exists then the probability of R is increased"). Le langage SBGN-ER est donc accompagné d'une sémantique plus ou moins ambigue. La deuxième des interprétations de la stimulation fournie dans la spécification donne même un cadre d'interprétation mathématique : une entité peut exister, et sera donc *a priori* modélisée par une valeur Booléenne, et une relation peut avoir lieu avec une certaine probabilité. De manière générale, la sémantique d'une carte SBGN-ER est définie de la manière suivante : les différents éléments (entités et relations) de la carte peuvent exister ou non (i.e. être "vrais" comme indiqué parfois dans la spécification), et les relations entre les éléments de la carte doivent être interprétées par des règles logiques (non fournies par la spécification) qui lient l'existence d'un certain élément d'une carte à l'existence ou à la probabilité d'exister d'un autre élément de la carte. Une première étude exploratoire de cette sémantique nous a conduit à relever de possibles incohérences (notamment pour les relations entre une entité et une autre relation). Cependant, il est sans doute envisageable de corriger ces incohérences si elles sont avérées et de construire une sémantique correcte à partir de celle de la spécification.

Une fois qu'une sémantique correcte sera définie, il sera possible de modéliser les cartes SBGN-ER à l'aide d'un formalisme adéquat. À notre connaissance, aucune étude n'a jusqu'à présent proposé de modéliser des cartes SBGN-ER suivant la sémantique introduite dans la spécification. Cependant, dans [DCO14], les auteurs proposent de modéliser des cartes MIM [Koh99], qui sont très proches des cartes SBGN-ER (ces dernières étant très fortement inspirées des cartes MIM), par des règles logiques du premier ordre. La logique du premier ordre semble être le formalisme de choix pour formaliser les règles interprétant les relations d'une carte lorsqu'on ne considère pas les notions de probabilité faisant partie des définitions des relations. Pour prendre en compte à la fois la notion de règle et la notion de probabilité de la sémantique, il faudrait vraisemblablement se tourner vers un formalisme logique probabiliste. Une première recherche de formalismes adéquats nous a amené à considérer deux langages : le langage CP-logic [VDB09], qui permet de formaliser des énoncés causaux sous forme de règles logiques et de les interpréter dans un contexte probabiliste, et celui des programmes logiques Bayésiens [KDR07], qui permet d'adosser à un programme logique un réseau Bayésien. Les cartes SBGN-ER seraient ainsi modélisées par des programmes logiques faisant entrer en jeu des relations de probabilité. Deux types d'analyses pourraient être menées à partir de tels modèles. D'abord, si les relations de probabilité sont connues, il serait possible de donner la probabilité d'existence d'une entité donnée sachant l'existence d'un ensemble d'entités ou de relations. Ensuite, sans qu'il y ait nécessité cette fois-ci de connaître précisément les probabilités, il serait possible de donner un ordre partiel des modèles de tels programmes selon leur probabilité, et ainsi d'ordonner les états globaux de la carte selon leur probabilité.

Nous donnons maintenant quelques perspectives à plus long terme.

Visualisation des réseaux. Afin de mieux les appréhender et les comprendre, les réseaux moléculaires ont besoin d'être visualisés, sous forme de représentations graphiques. Même si de nombreux logiciels permettent déjà la visualisation des réseaux moléculaires (p. ex. Vanted/SBGN-ED [CKS10], CellDesigner [Fun+08], Cytoscape [Sha+03], SBGNViz [Sar+15]), cette tâche reste un des problèmes actuels de la biologie des systèmes (et plus largement, la visualisation des graphes est un domaine

de recherche bien vivant de l'informatique). Au cours de nos travaux, nous avons été confrontés au problème de la mise en forme automatique des réseaux moléculaires, en particulier lors de la construction ou la transformation de réseaux (chapitres 3 et 4). En effet, les deux méthodes que nous avons présentées construisent des réseaux moléculaires formalisés en logique du permier ordre, et le passage de ce formalisme à une représentation graphique, qui permettrait sans doute d'avoir une meilleure compréhension de ces réseaux, nécessite la mise en forme automatique de cette représentation. Or peu de logiciels permettent une mise en forme satisfaisante des réseaux, et en particulier des réseaux de réactions. La plupart des méthodes qui ont été proposées et implémentées reposent sur des algorithmes de mise en forme de graphes qui ne prennent pas en compte les spécificités structurelles des réseaux et les règles parfois implicites de mise en forme utilisées lors de la construction manuelle de représentations graphiques de ces réseaux. Les spécificités structurelles dépendent de la nature des réseaux : un réseau métabolique est composé de réactions qui s'enchaînent linéairement, alors qu'un réseau de signalisation est composé de réactions en cascade. Ces deux types de réseaux ne seront ainsi *a priori* pas représentés graphiquement de la même façon. Quant aux règles implicites de mise en forme, elles définissent la manière habituelle de représenter les concepts de la biologie des systèmes, au-delà du choix des glyphes. Par exemple, les processus sont habituellement représentés horizontalement plutôt que verticalement. Le respect de ces règles est nécessaire à l'obtention d'un représentation graphique des réseaux facilement compréhensible. Une difficulté de la mise en forme est donc d'abord l'identification de ces spécificités structurelles et de ces règles parfois implicites, qui doivent ensuite être prises en compte dans le développement des méthodes.

De plus, certaines méthodes actuelles (par exemple celles utilisées dans CellDesigner) donnent des résultats satisfaisants pour de petits réseaux, mais peinent à donner une mise en forme lisible pour des réseaux composés de centaines de molécules (problème de passage à l'échelle).

Extraction automatique de résultats expérimentaux de la littérature. Les résultats expérimentaux relatifs aux réseaux induits par les récepteurs à la FSH et à l'EGF que nous avons interpétés automatiquement dans le chapitre 4 ont été extraits de la littérature à la main. Cette tâche est longue et fastidieuse, et mériterait d'être automatisée. L'extraction automatique et massive d'expériences à partir d'articles de la littérature permettrait en outre la création de banques d'expériences qui pourraient ensuite être utilisées par diverses méthodes de la biologie des systèmes. C'est d'ailleurs l'objectif du projet *Biosystémique* initié par Anne Poupon (équipe BIOS - INRA Centre Val de Loire), qui porte sur l'extraction d'expériences par fouille du corps du texte et des légendes d'articles biologiques relatifs à la voie des β-arrestines induite par les GPCR.

La fouille de données dans les articles biologiques est un domaine florissant de la biologie des systèmes. Cependant, si de nombreuses méthodes permettent l'extraction de connaissances biologiques de la littérature (voir [RSG09]), les méthodes permettant l'extraction de résultats expérimentaux (non interprétés) sont à notre connaissance beaucoup plus rares. Cette tâche peut sans doute se révéler complexe vu que l'extraction de résultats nécessite d'obtenir des informations de différentes natures qui proviennent de différentes sources. En effet, en biologie de la signalisation par exemple, un résultat publié dans un article consiste le plus souvent en une image accompagnée d'une légende décrivant l'expérience réalisée, d'un protocole décrivant comment elle a été réalisée, et d'une description du résultat, parfois déjà issue d'une interprétation de celui-ci, dans le corps du texte. Afin d'extraire un résultat, il faudrait alors employer à la fois des techniques de traitement du langage naturel et du traitement d'images. Une première étape pour l'extraction de résultats expérimentaux serait l'extraction d'informations sur les expériences qui ont été réalisées, sans le souci du résultat lui-même ou de son interprétation, extraction qui nécessite principalement le traitement du texte contenu dans des légendes des figures. Une telle méthode permettrait alors d'indiquer au biologiste que telle ou telle expérience a

été réalisée dans tel article, et que le résultat obtenu à partir de cette expérience se trouve dans telle figure de cet article.

Bibliographie

[AE+12] Zahira Aslaoui-Errafi, Sarah Cohen-Boulakia, Christine Froidevaux, Pauline Gloaguen, Anne Poupon, Adrien Rougny et Meriem Yahiaoui. "Towards a logic-based method to infer provenance-aware molecular networks". In : *Proceedings of the 1st ECML/PKDD International workshop on Learning and Discovery in Symbolic Systems Biology (LDSSB)*. 2012, p. 103–110.

[Ala+09] Hena Alam, Jennifer Weck, Evelyn Maizels, Youngkyu Park, Eun Jig Lee, Margaret Ashcroft et Mary Hunzicker-Dunn. "Role of the phosphatidylinositol-3-kinase and extracellular regulated kinase pathways in the induction of hypoxia-inducible factor (HIF)-1 activity and the HIF-1 target vascular endothelial growth factor in ovarian granulosa cells in response to follicle-stimulating hormone". In : *Endocrinology* 150.2 (2009), p. 915–928.

[Alb04] Réka Albert. "Boolean modeling of genetic regulatory networks". In : *Complex networks*. Lecture Notes In Physics. Springer, 2004, p. 459–481.

[Alt+13] Tomer Altman, Michael Travers, Anamika Kothari, Ron Caspi et Peter D Karp. "A systematic comparison of the MetaCyc and KEGG pathway databases". In : *BMC Bioinformatics* 14.112 (2013).

[AO03] Réka Albert et Hans G Othmer. "The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster". In : *Journal of theoretical biology* 223.1 (2003), p. 1–18.

[Ash+00] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig et al. "Gene Ontology : tool for the unification of biology". In : *Nature genetics* 25.1 (2000), p. 25–29.

[Ban+07] Mukesh Bansal, Vincenzo Belcastro, Alberto Ambesi-Impiombato et Diego Di Bernardo. "How to infer gene networks from expression profiles". In : *Molecular Systems Biology* 3.1 (2007), p. 78.

[Bid91] Nicole Bidoit. "Negation in rule-based database languages : a survey". In : *Theoretical Computer Science* 78.1 (1991), p. 3–83.

[BKL08] Christel Baier, Joost-Pieter Katoen et Kim Guldstrand Larsen. *Principles of model checking*. MIT press, 2008.

[Bra+04] Adrian P Bracken, Marco Ciro, Andrea Cocito et Kristian Helin. "E2F target genes : unraveling the biology". In : *Trends in biochemical sciences* 29.8 (2004), p. 409–417.

[Büc+13] Finja Büchel, Nicolas Rodriguez, Neil Swainston, Clemens Wrzodek, Tobias Czauderna, Roland Keller, Florian Mittag, Michael Schubert, Mihai Glont, Martin Golebiewski et al. "Path2Models : large-scale generation of computational models from biochemical pathway maps". In : *BMC Systems Biology* 7.1 (2013), p. 1.

[Cal+08] Laurence Calzone, Amélie Gelay, Andrei Zinovyev, François Radvanyi et Emmanuel Barillot. "A comprehensive modular map of molecular interactions in RB/E2F pathway". In : *Molecular Systems Biology* 4.1 (2008).

[Car+10] Etienne Caron, Samik Ghosh, Yukiko Matsuoka, Dariel Ashton-Beaucage, Marc Therrien, Sébastien Lemieux, Claude Perreault, Philippe P Roux et Hiroaki Kitano. "A comprehensive map of the mTOR signaling network". In : *Molecular Systems Biology* 6.1 (2010).

[Cas+08] Ron Caspi, Hartmut Foerster, Carol A Fulcher, Pallavi Kaipa, Markus Krummenacker, Mario Latendresse, Suzanne Paley, Seung Y Rhee, Alexander G Shearer, Christophe Tissier et al. "The MetaCyc Database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases". In : *Nucleic Acids Research* 36.suppl 1 (2008), p. D623–D631.

[CE81] Edmund M Clarke et E Allen Emerson. "Design and synthesis of synchronization skeletons using branching time temporal logic". In : *Logic of Programs*. Springer. 1981, p. 52–71.

[CFS06] Laurence Calzone, François Fages et Sylvain Soliman. "BIOCHAM : an environment for modeling biological systems and formalizing experimental knowledge". In : *Bioinformatics* 22.14 (2006), p. 1805–1807.

[Cha+02] Christophe Chassagnole, Naruemol Noisommit-Rizzi, Joachim W Schmid, Klaus Mauch et Matthias Reuss. "Dynamic modeling of the central carbon metabolism of Escherichia coli". In : *Biotechnology and Bioengineering* 79.1 (2002), p. 53–73.

[Cha+11] Claudine Chaouiya, Aurélien Naldi, Elisabeth Remy et Denis Thieffry. "Petri net representation of multi-valued logical regulatory graphs". In : *Natural Computing* 10.2 (2011), p. 727–750.

[Cha+13] Claudine Chaouiya, Duncan Bérenguier, Sarah M Keating, Aurélien Naldi, Martijn P Van Iersel, Nicolas Rodriguez, Andreas Dräger, Finja Büchel, Thomas Cokelaer, Bryan Kowal et al. "SBML qualitative models : a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools". In : *BMC Systems Biology* 7.1 (2013), p. 1.

[Cha+14] Thomas Chatain, Stefan Haar, Loïg Jezequel, Loïc Paulevé et Stefan Schwoon. "Characterization of reachable attractors using Petri net unfoldings". In : *Computational Methods in Systems Biology*. Springer, 2014.

[CKS10] Tobias Czauderna, Christian Klukas et Falk Schreiber. "Editing, validating and translating of SBGN maps". In : *Bioinformatics* 26.18 (2010), p. 2340–2341.

[CL14] Chin-Liang Chang et Richard Char-Tung Lee. *Symbolic logic and mechanical theorem proving*. Academic press, 2014.

[CM03] William Clocksin et Christopher S Mellish. *Programming in PROLOG*. Springer, 2003.

[Cré+01] Pascale Crépieux, Sebastien Marion, Nadine Martinat, Véronique Fafeur, Yves Le Vern, Dominique Kerboeuf, Florian Guillou et Eric Reiter. "The ERK-dependent signalling is stage-specifically modulated by FSH, during primary Sertoli cell maturation". In : *Oncogene* 20.34 (2001), p. 4696–4709.

[CS04] Hao Chen et Burt M Sharp. "Content-rich biological network constructed by mining PubMed abstracts". In : *BMC bioinformatics* 5.1 (2004), p. 1–13.

[CTM05] Jean-Michel Couvreur et Yann Thierry-Mieg. "Hierarchical decision diagrams to exploit model structure". In : *Formal Techniques for Networked and Distributed Systems - FORTE 2005*. Lecture Notes In Computer Science. Springer, 2005, p. 443–457.

[Cza+13] Tobias Czauderna, Michael Wybrow, Kim Marriott et Falk Schreiber. "Conversion of KEGG metabolic pathways to SBGN maps including automatic layout". In : *BMC bioinformatics* 14.1 (2013), p. 1.

[DCO14] Robert Demolombe, Luis Fariñas del Cerro et Naji Obeid. "Logical modeling of biological systems". In : Wiley, 2014. Chap. A Logical Model for Molecular Interaction Maps, p. 125–166.

[Dem+10] Emek Demir, Michael P Cary, Suzanne Paley, Ken Fukuda, Christian Lemer, Imre Vastrik, Guanming Wu, Peter D'Eustachio, Carl Schaefer, Joanne Luciano et al. "The BioPAX community standard for pathway data sharing". In : *Nature biotechnology* 28.9 (2010), p. 935–942.

[DP60] Martin Davis et Hilary Putnam. "A computing procedure for quantification theory". In : *Journal of the ACM (JACM)* 7.3 (1960), p. 201–215.

[E2f] *RB/E2F pathway*. URL : http://bioinfo-out.curie.fr/projects/rbpathway/ (visité le 08/02/2016).

[Eis+98] Michael B Eisen, Paul T Spellman, Patrick O Brown et David Botstein. "Cluster analysis and display of genome-wide expression patterns". In : *Proceedings of the National Academy of Sciences* 95.25 (1998), p. 14863–14868.

[Eke+02] Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, José Meseguer et Kemal Sonmez. "Pathway logic : Symbolic analysis of biological signaling". In : *Pacific Symposium on Biocomputing*. T. 7. 2002, p. 400–412.

[Fay+11] Timur Fayruzov, Jeroen Janssen, Dirk Vermeir, Chris Cornelis et Martine De Cock. "Modelling gene and protein regulatory networks with answer set programming". In : *First International journal of data mining and bioinformatics* 5.2 (2011), p. 209–229.

[Flo+15] Åsmund Flobak, Anaïs Baudot, Elisabeth Remy, Liv Thommesen, Denis Thieffry, Martin Kuiper et Astrid Lægreid. "Discovery of drug synergies in gastric cancer cells predicted by logical modeling". In : *PLoS Computational Biology* 11.8 (2015), e1004426.

[FS08] François Fages et Sylvain Soliman. "Abstract interpretation and types for systems biology". In : *Theoretical Computer Science* 403.1 (2008), p. 52–70.

[Fun+08] Akira Funahashi, Yukiko Matsuoka, Akiya Jouraku, Mineo Morohashi, Norihiro Kikuchi et Hiroaki Kitano. "CellDesigner 3.5 : a versatile modeling tool for biochemical networks". In : *Proceedings of the IEEE* 96.8 (2008), p. 1254–1265.

[Geb+08] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub et Sven Thiele. *A user's guide to gringo, clasp, clingo, and iclingo*. 2008.

[GF10] Delquin Gong et James E Ferrell. "The roles of cyclin A2, B1, and B2 in early and late mitotic events". In : *Molecular biology of the cell* 21.18 (2010), p. 3149–3161.

[GG10] Clare E Giacomantonio et Geoffrey J Goodhill. "A Boolean model of the gene regulatory network underlying Mammalian cortical area development". In : *PLoS Computational Biology* 6.9 (2010), e1000936.

[Gil77] Daniel T Gillespie. "Exact stochastic simulation of coupled chemical reactions". In : *The journal of physical chemistry* 81.25 (1977), p. 2340–2361.

[Glo+11] Pauline Gloaguen, Pascale Crépieux, Domitille Heitzler, Anne Poupon et Eric Reiter. "Mapping the follicle-stimulating hormone-induced signaling networks". In : *Frontiers in endocrinology* 2 (2011), p. 45.

[Glo12] Pauline Gloaguen. *Inférence automatique de modèles de voies de signalisation à partir de données expérimentales*. These, 2012.

[HB97] Inman Harvey et Terry Bossomaier. "Time out of joint : Attractors in asynchronous random boolean networks". In : *Proceedings of the Fourth European Conference on Artificial Life*. 1997, p. 67–75.

[Hei+12] Domitille Heitzler, Guillaume Durand, Nathalie Gallay, Aurélien Rizk, Seungkirl Ahn, Jihee Kim, Jonathan D Violin, Laurence Dupuy, Christophe Gauthier, Vincent Piketty et al. "Competing G protein-coupled receptor kinases balance G protein and β-arrestin signaling". In : *Molecular Systems Biology* 8.1 (2012), p. 590.

[Hil+12] Steven M Hill, Yiling Lu, Jennifer Molina, Laura M Heiser, Paul T Spellman, Terence P Speed, Joe W Gray, Gordon B Mills et Sach Mukherjee. "Bayesian inference of signaling network topology in a cancer cell line". In : *Bioinformatics* 28.21 (2012), p. 2804–2810.

[HS14] Anja Hartmann et Falk Schreiber. "Integrative analysis of metabolic models–from structure to dynamics". In : *Frontiers in Bioengineering and Biotechnology* 2 (2014).

[Huc+03] Michael Hucka, Andrew Finney, Herbert M Sauro, Hamid Bolouri, John C Doyle, Hiroaki Kitano, Adam P Arkin, Benjamin J Bornstein, Dennis Bray, Athel Cornish-Bowden et al. "The systems biology markup language (SBML) : a medium for representation and exchange of biochemical network models". In : *Bioinformatics* 19.4 (2003), p. 524–531.

[IDN13] Katsumi Inoue, Andrei Doncescu et Hidetomo Nabeshima. "Completing causal networks by meta-level abduction". In : *Machine learning* 91.2 (2013), p. 239–277.

[Ino11] Katsumi Inoue. "Logic programming for Boolean networks". In : *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. T. 22. 1. 2011, p. 924.

[Ino92] Katsumi Inoue. "Linear resolution for consequence finding". In : *Artificial Intelligence* 56.2 (1992), p. 301–353.

[Jac87] François Jacob. *La logique du vivant : une histoire de l'hérédité*. Gallimard, 1987.

[JN13] Nick Juty et Nicolas le Novère. "Systems biology ontology". In : *Encyclopedia of Systems Biology* (2013), p. 2063–2063.

[Kau69] Stuart A Kauffman. "Metabolic stability and epigenesis in randomly constructed genetic nets". In : *Journal of theoretical biology* 22.3 (1969), p. 437–467.

[KDR07] Kristian Kersting et Luc De Raedt. "Bayesian Logic Programming : Theory and Tool". In : *Statistical Relational Learning*. MIT Press, 2007.

[KG00] Minoru Kanehisa et Susumu Goto. "KEGG : kyoto encyclopedia of genes and genomes". In : *Nucleic Acids Research* 28.1 (2000), p. 27–30.

[Kit02] Hiroaki KITANO. "Systems biology : a brief overview". In : *Science* 295.5560 (2002), p. 1662–1664.

[Kit03] Hiroaki KITANO. "A graphical notation for biochemical networks". In : *Biosilico* 1.5 (2003), p. 169–176.

[Koh99] Kurt W KOHN. "Molecular interaction map of the mammalian cell cycle control and DNA repair systems". In : *Molecular biology of the cell* 10.8 (1999), p. 2703–2734.

[Kwi+06] Marta KWIATKOWSKA, Gethin NORMAN, David PARKER, Oksana TYMCHYSHYN, John HEATH et Eamonn GAFFNEY. "Simulation and verification for computational modelling of signalling pathways". In : *Proceedings of the 2006 Winter Simulation Conference*. IEEE. 2006, p. 1666–1674.

[Li+08] Shenghua LI, Paul BRAZHNIK, Bruno SOBRAL et John J TYSON. "A quantitative study of the division cycle of Caulobacter crescentus stalked cells". In : *PLoS Computational Biology* 4.1 (2008), e9.

[LMH09] Laurence LOEWE, Stuart MOODIE et Jane HILLSTON. "Quantifying the implicit process flow abstraction in SBGN-PD diagrams with Bio-PEPA". In : *arXiv :0910.1410* (2009).

[LN+06] Nicolas LE NOVERE, Benjamin BORNSTEIN, Alexander BROICHER, Melanie COURTOT, Marco DONIZELLI, Harish DHARURI, Lu LI, Herbert SAURO, Maria SCHILSTRA, Bruce SHAPIRO et al. "BioModels Database : a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems". In : *Nucleic acids research* 34.suppl 1 (2006), p. D689–D691.

[LN+09] Nicolas LE NOVERE, Michael HUCKA, Huaiyu MI, Stuart MOODIE, Falk SCHREIBER, Anatoly SOROKIN, Emek DEMIR, Katja WEGNER, Mirit I ALADJEM, Sarala M WIMALARATNE et al. "The systems biology graphical notation". In : *Nature Biotechnology* 27.8 (2009), p. 735–741.

[LN+11] Nicolas LE NOVÈRE, Emek DEMIR, Huaiyu MI, Stuart MOODIE et Alice VILLEGER. "Systems biology graphical notation : entity relationship language Level 1 (Version 1.2)." In : *Nature Precedings* (2011).

[Mi+09] Huaiyu MI, Falk SCHREIBER, Nicolas LE NOVÈRE, Stuart MOODIE et Anatoly SOROKIN. "Systems biology graphical notation : activity flow language level 1". In : *Nature Precedings* (2009).

[Mol] *Mole - Petri net unfolder*. URL : http://www.lsv.ens-cachan.fr/\textasciitildeschwoon/tools/mole/ (visité le 08/02/2016).

[Moo+11] Stuart MOODIE, Nicolas LE NOVÈRE, Emek DEMIR, Huaiyu MI et Alice VILLEGER. "Systems biology graphical notation : process description language Level 1". In : *Nature Precedings* (2011).

[MRH12] Wolfgang MARWAN, Christian ROHR et Monika HEINER. "Petri nets in Snoopy : A unifying framework for the graphical display, computational modelling, and simulation of bacterial regulatory networks". In : *Bacterial Molecular Networks : Methods and Protocols* (2012), p. 409–437.

[MS07] Florian MARKOWETZ et Rainer SPANG. "Inferring cellular networks–a review". In : *BMC Bioinformatics* 8.6 (2007).

[Nab+10] Hidetomo NABESHIMA, Koji IWANUMA, Katsumi INOUE et Oliver RAY. "SOLAR : An automated deduction system for consequence finding". In : *AI communications* 23.2-3 (2010), p. 183–203.

[Nal+11] Aurélien Naldi, Elisabeth Remy, Denis Thieffry et Claudine Chaouiya. "Dynamically consistent reduction of logical regulatory graphs". In : *Theoretical Computer Science* 412.21 (2011), p. 2207–2218.

[NED03] Svetlana Novichkova, Sergei Egorov et Nikolai Daraselia. "MedScan, a natural language processing engine for MEDLINE abstracts". In : *Bioinformatics* 19.13 (2003), p. 1699–1706.

[Nig+15] Vivek Nigam, Robin Donaldson, Merrill Knapp, Tim McCarthy et Carolyn Talcott. "Inferring Executable Models from Formalized Experimental Evidence". In : *Computational Methods in Systems Biology*. Lecture Notes in Computer Science. Springer, 2015, p. 90–103.

[Nis01] Darryl Nishimura. "BioCarta". In : *Biotech Software & Internet Report : The Computer Software Journal for Scient* 2.3 (2001), p. 117–120.

[Oda+05] Kanae Oda, Yukiko Matsuoka, Akira Funahashi et Hiroaki Kitano. "A comprehensive pathway map of epidermal growth factor receptor signaling". In : *Molecular Systems Biology* 1.1 (2005).

[OTP10] Jeffrey D Orth, Ines Thiele et Bernhard Ø Palsson. "What is flux balance analysis ?" In : *Nature Biotechnology* 28.3 (2010), p. 245–248.

[PAK13] Loïc Paulevé, Geoffroy Andrieux et Heinz Koeppl. "Under-Approximating Cut Sets for Reachability in Large Scale Automata Networks". In : *Computer Aided Verification*. T. 8044. Lecture Notes in Computer Science. Springer, 2013, p. 69–84.

[Pe'+01] Dana Pe'er, Aviv Regev, Gal Elidan et Nir Friedman. "Inferring subnetworks from perturbed expression profiles". In : *Bioinformatics* 17.suppl 1 (2001), S215–S224.

[Pic+08] Alexander R Pico, Thomas Kelder, Martijn P Van Iersel, Kristina Hanspers, Bruce R Conklin et Chris Evelo. "WikiPathways : pathway editing for the people". In : *PLoS Biology* 6.7 (2008), e184.

[Pin] *Pint - Static analyzer for dynamics of Automata Networks*. URL : http://loicpauleve.name/pint (visité le 08/02/2016).

[PMMS15] Giovanni Pardini, Paolo Milazzo et Andrea Maggiolo-Schettini. "Component identification in biochemical pathways". In : *Theoretical Computer Science* 587 (2015), p. 104–124.

[PMR11] Loïc Paulevé, Morgan Magnin et Olivier Roux. "Refining dynamics of gene regulatory networks in a stochastic π-calculus framework". In : *Transactions on computational systems biology xiii*. Lecture Notes in Computer Science. Springer, 2011, p. 171–191.

[PMR12] Loïc Paulevé, Morgan Magnin et Olivier Roux. "Static analysis of biological regulatory networks dynamics using abstract interpretation". In : *Mathematical Structures in Computer Science* 22.04 (2012), p. 651–685.

[Prz04] Nataša Przulj. "Graph theory approaches to protein interaction data analysis". In : *Knowledge Discovery in proteomics*. CRC Press, 2004.

[RCB06] Adrien Richard, Jean-Paul Comet et Gilles Bernot. "Formal methods for modeling biological regulatory networks". In : *Modern Formal Methods and Applications*. Springer, 2006, p. 83–122.

[Roc+14] Alexandre Rocca, Nicolas Mobilia, Éric Fanchon, Tony Ribeiro, Laurent Trilling et Katsumi Inoue. In : *Logical modeling of biological systems*. Wiley, 2014. Chap. Asp for construction and validation of regulatory biological networks, p. 167–206.

[Rou+14] Adrien Rougny, Christine Froidevaux, Yoshitaka Yamamoto et Katsumi Inoue. In : *Logical modeling of biological systems*. Wiley, 2014. Chap. Analyzing SBGN-AF Networks Using Normal Logic Programs, p. 325–361.

[Rou+15] Adrien Rougny, Yoshitaka Yamamoto, Hidetomo Nabeshima, Gauvain Bourgne, Anne Poupon, Katsumi Inoue et Christine Froidevaux. "Completing signaling networks by abductive reasoning with perturbation experiments". In : *Proceedings of the 25st International Conference on Inductive Logic Programming*. 2015.

[Rou+16] Adrien Rougny, Christine Froidevaux, Laurence Calzone et Loïc Paulevé. "Qualitative dynamics semantics for SBGN process description". In : *BMC Systems Biology* 10.42 (2016).

[RSG09] Andrey Rzhetsky, Michael Seringhaus et Mark B Gerstein. "Getting started in text mining : part two". In : *PLoS Comput Biol* 5.7 (2009), e1000411.

[RSI12] Oliver Ray, Takehide Soh et Katsumi Inoue. "Analyzing pathways using ASP-based approaches". In : *Algebraic and Numeric Biology*. Springer, 2012, p. 167–183.

[Sac+05] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger et Garry P Nolan. "Causal protein-signaling networks derived from multiparameter single-cell data". In : *Science* 308.5721 (2005), p. 523–529.

[Sar+15] Mecit Sari, Istemi Bahceci, Ugur Dogrusoz, Selcuk Onur Sumer, Bülent Arman Aksoy, Özgün Babur et Emek Demir. "SBGNViz : a tool for visualization and complexity management of SBGN process description maps". In : *PLoS One* 10.6 (2015), e0128985.

[Sch+02a] Birgit Schoeberl, Claudia Eichler-Jonsson, Ernst Dieter Gilles et Gertraud Müller. "Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors". In : *Nature Biotechnology* 20.4 (2002), p. 370–375.

[Sch+02b] Stefan Schuster, Thomas Pfeiffer, Ferdinand Moldenhauer, Ina Koch et Thomas Dandekar. "Exploring the pathway structure of metabolism : decomposition into subnetworks and application to Mycoplasma pneumoniae". In : *Bioinformatics* 18.2 (2002), p. 351–361.

[Sco+06] Jacob Scott, Trey Ideker, Richard M Karp et Roded Sharan. "Efficient algorithms for detecting signaling pathways in protein interaction networks". In : *Journal of Computational Biology* 13.2 (2006), p. 133–144.

[SCS02] Ida Schomburg, Antje Chang et Dietmar Schomburg. "BRENDA, enzyme data and metabolic information". In : *Nucleic Acids Research* 30.1 (2002), p. 47–49.

[SDF99] Stefan Schuster, Thomas Dandekar et David A Fell. "Detection of elementary flux modes in biochemical networks : a promising tool for pathway analysis and metabolic engineering". In : *Trends in Biotechnology* 17.2 (1999), p. 53–60.

[Sha+03] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski et Trey Ideker. "Cytoscape : a software environment for integrated models of biomolecular interaction networks". In : *Genome research* 13.11 (2003), p. 2498–2504.

[Sno89] El Houssine Snoussi. "Qualitative dynamics of piecewise-linear differential equations : a discrete mapping approach". In : *Dynamics and stability of Systems* 4.3-4 (1989), p. 565–583.

[SR+09] Julio Saez-Rodriguez, Leonidas G Alexopoulos, Jonathan Epperlein, Regina Samaga, Douglas A Lauffenburger, Steffen Klamt et Peter K Sorger. "Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction". In : *Molecular Systems Biology* 5.331 (2009).

[Thi+13] Ines Thiele, Neil Swainston, Ronan MT Fleming, Andreas Hoppe, Swagatika Sahoo, Maike K Aurich, Hulda Haraldsdottir, Monica L Mo, Ottar Rolfsson, Miranda D Stobbe et al. "A community-driven global reconstruction of human metabolism". In : *Nature Biotechnology* 31.5 (2013), p. 419–425.

[Tho73] René Thomas. "Boolean formalization of genetic control circuits". In : *Journal of theoretical biology* 42.3 (1973), p. 563–585.

[TT95] Denis Thieffry et René Thomas. "Dynamical behaviour of biological regulatory networks—II. Immunity control in bacteriophage lambda". In : *Bulletin of Mathematical Biology* 57.2 (1995), p. 277–297.

[TTK95] René Thomas, Denis Thieffry et Marcelle Kaufman. "Dynamical behaviour of biological regulatory networks—I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state". In : *Bulletin of mathematical biology* 57.2 (1995), p. 247–276.

[VCS13] Torsten Vogt, Tobias Czauderna et Falk Schreiber. "Translation of SBGN maps : process description to activity flow". In : *BMC Systems Biology* 7.115 (2013).

[VDB09] Joost Vennekens, Marc Denecker et Maurice Bruynooghe. "CP-logic : A Language of Causal Probabilistic Events and Its Relation to Logic Programming". In : *CoRR* abs/0904.1672 (2009).

[VI+12] Martijn P Van Iersel, Alice C Villéger, Tobias Czauderna, Sarah E Boyd, Frank T Bergmann, Augustin Luna, Emek Demir, Anatoly Sorokin, Ugur Dogrusoz, Yukiko Matsuoka et al. "Software support for SBGN maps : SBGN-ML and LibSBGN". In : *Bioinformatics* 28.15 (2012), p. 2016–2021.

[Vid+12] Santiago Videla, Carito Guziolowski, Federica Eduati, Sven Thiele, Niels Grabe, Julio Saez-Rodriguez et Anne Siegel. "Revisiting the training of logic models of protein signaling networks with ASP". In : *Computational Methods in Systems Biology*. Springer. 2012, p. 342–361.

[Wal+11a] Dagmar Waltemath, Richard Adams, Daniel A Beard, Frank T Bergmann, Upinder S Bhalla, Randall Britten, Vijayalakshmi Chelliah, Michael T Cooling, Jonathan Cooper, Edmund J Crampin et al. "Minimum information about a simulation experiment (MIASE)". In : *PLoS Computational Biology* 7.4 (2011), e1001122.

[Wal+11b] Dagmar Waltemath, Richard Adams, Frank T Bergmann, Michael Hucka, Fedor Kolpakov, Andrew K Miller, Ion I Moraru, David Nickerson, Sven Sahle, Jacky L Snoep et al. "Reproducible computational biology experiments with SED-ML-the simulation experiment description markup language". In : *BMC Systems Biology* 5.198 (2011).

[Wit+12] Ulrike Wittig, Renate Kania, Martin Golebiewski, Maja Rey, Lei Shi, Lenneke Jong, Enkhjargal Algaa, Andreas Weidemann, Heidrun Sauer-Danzwith, Saqib Mir et al. "SABIO-RK—database for biochemical reaction kinetics". In : *Nucleic acids research* 40.D1 (2012), p. D790–D796.

[Yam+14] Yoshitaka YAMAMOTO, Adrien ROUGNY, Hidetomo NABESHIMA, Katsumi INOUE, Hisao MORIYA, Christine FROIDEVAUX et Koji IWANUMA. "Completing SBGN-AF Networks by Logic-Based Hypothesis Finding". In : *Proceedings of the first Proceedings of the first International Conference on Formal Methods in Macro-Biology*. Springer. 2014, p. 165–179.

[Yur+06] Anton YURYEV, Zufar MULYUKOV, Ekaterina KOTELNIKOVA, Sergei MASLOV, Sergei EGOROV, Alexander NIKITIN, Nikolai DARASELIA et Ilya MAZO. "Automatic pathway building in biological association networks". In : *BMC bioinformatics* 7.171 (2006).

[Zha+08] Ranran ZHANG, Mithun Vinod SHAH, Jun YANG, Susan B NYLAND, Xin LIU, Jong K YUN, Réka ALBERT et Thomas P LOUGHRAN. "Network model of survival signaling in large granular lymphocyte leukemia". In : *Proceedings of the National Academy of Sciences* 105.42 (2008), p. 16308–16313.

[Zin+08] Andrei ZINOVYEV, Eric VIARA, Laurence CALZONE et Emmanuel BARILLOT. "BiNoM : a Cytoscape plugin for manipulating and analyzing biological networks". In : *Bioinformatics* 24.6 (2008), p. 876–877.

[Zup+03] Blaz ZUPAN, Janez DEMSAR, Ivan BRATKO, Peter JUVAN, John A HALTER, Adam KUSPA et Gad SHAULSKY. "GenePath : a system for automated construction of genetic networks from mutant data". In : *Bioinformatics* 19.3 (2003), p. 383–389.

Annexe A

Annexe du chapitre 2

Sommaire

A.1 Preuves

A.1.1 Sketch de preuve de la propriété 2.5

Propriété. Soit Π un NLP et $R \in \Pi$ une règle de Π. Soit Π' le NLP obtenu en appliquant une des règles de simplification (SR1–4) à R. Π et Π' ont exactement les mêmes modèles supportés.

Sketch de preuve. L'application itérée des règles de simplification (SR1–4) aux règles de Π est une extension de la procédure David-Putnam compatible avec la sémantique des modèles supportés. Nous rappelons qu'une interprétation M de Π (resp. Π') est un modèle supporté de Π (resp. Π') *ssi* $T_\Pi(M) = M$ (resp. $T_{\Pi'}(M) = M$). Nous procédons en montrant que si $T_\Pi(M) = M$, alors $T_{\Pi'}(M) = M$, et inversement, par double inclusion.

A.1.2 Sketch de preuve de la propriété 2.6

Propriété. Soit Π un NLP et $R \in \Pi$ une règle de Π. Soit Π' le NLP obtenu en appliquant une des règles de transformation (TR1–2) à un atome du corps de R. Π et Π' ont exactement les mêmes modèles supportés.

Sketch de preuve. Nous procédons, pour chacune des règles (TR1) et (TR2), en montrant que si $T_\Pi(M) = M$, alors $T_{\Pi'}(M) = M$, et inversement, par double inclusion.

Annexe B

Annexes du chapitre 3

Sommaire

B.1 Création des unités d'informations lors de la transformation des cartes SBGN-PD en cartes SBGN-AF

L'étiquette d'une unité d'information d'une activité peut soit être vide, soit comporter le nom de l'EPN dont elle provient. Vue la façon dont nous créons les étiquettes des activités, deux activités de même nature et provenant de deux EPNs différents mais représentant différents états d'une même entité moléculaire auront la même étiquette, étant donné que ces dits EPNs partagent eux-mêmes la même étiquette. Afin de distinguer ces activités l'une de l'autre, nous concaténons à l'étiquette de l'unité d'information de chaque activité, initialement formée de l'étiquette de l'EPN dont elle provient, l'ensemble des variables d'états de cet EPN, représentées sous forme de chaînes de caractères. Pour ce faire, nous ordonnons d'abord l'ensemble des variables d'états d'un EPN suivant l'ordre lexicographique sur la variable de la variable d'état (qui, nous le rappelons, comporte également une valeur). Cet ordre est donné par deux prédicats, min et inf, et calculé grâce à la fonction python `inf(const1,const2)` directement intégrée au code ASP. Cette fonction, étant donnée deux constantes `const1` et `const2` qui peuvent être des chaînes de caractères, renvoie `1` si `str(const1)< str(const2)`, et `0` sinon. Le prédicat $inf(E, Var1, Var2)$ signifie que la variable $Var1$ de l'EPN E est inférieure, suivant l'ordre lexicographique, à la variable $Var2$, et $min(E, Var)$ qu'il n'existe pas de variable de E inférieure à Var. Ces deux prédicats sont définis de la manière suivante :

$$\begin{aligned} inf(E, Var1, Var2) &:\!- \; epn_{PD}(E);\; stateVariable_{PD}(E, Val1, Var1);\; stateVariable_{PD}(E, Val2, Var2);\\ &\qquad @inf(Var1, Var2) = 1.\\ min(E, Var) &:\!- \; stateVariable_{PD}(E, Val, Var);\; \text{not } inf(E, , Var). \end{aligned}$$

Pour un EPN ayant au moins une variable d'état, l'ordre lexicographique de ses variables d'état, calculé sur la variable, est toujours total, étant donné que toutes les variables des variables d'état d'un même EPN sont différentes les unes des autres. Ainsi, pour un tel EPN, le prédicat min sera toujours défini.

Les variables d'états sont représentées sous la forme de chaînes de caractères de la manière suivante. Étant donnée une variable d'état d'un EPN E formalisée par le prédicat $stateVariable(E, Val, Var)$, elle est représentée par :

- une chaîne de caractères vide si $Val = unset$;
- la chaîne "$Val@N$" si $Val \neq unset$ et Var est de la forme $undefined(E, N)$;
- la chaîne "$Val@Var$" sinon.

La transformation des variables d'état d'un EPN en chaînes de caractères, et leur concaténation dans l'ordre lexicographique à l'étiquette de cet EPN sont réalisées en même temps par les règles données ci-après. Le prédicat *uoiLabel* donne, pour un EPN, l'étiquette, partiellement créée, de l'unité d'information de l'activité que cet EPN opère éventuellement. Son premier argument est l'identifiant de l'EPN, le deuxième l'étiquette en cours de création, et le troisième un nombre indiquant l'étape de la création. À l'étape 0, l'étiquette de l'unité d'information est l'étiquette de l'EPN ; à l'étape 1, l'étiquette est la concaténation de l'étiquette obtenue à l'étape 1 de la chaîne de caractères représentant la première variable d'état ; et ainsi de suite, jusqu'à ce que toutes les variables d'état aient été prises en compte.

La règle suivante correspond à l'étape 0 :

$$uoiLabel(E, L, 0) \text{ :- } label_{PD}(E, L);\ epn_{PD}(E).$$

Les règles suivantes correspondent à l'étape 1 :

$$uoiLabel(E, L, 1) \text{ :- } Val = unset;\ stateVariable_{PD}(E, Val, Var);\ min(E, Var);\ uoiLabel(E, L, 0);\ epn_{PD}(E).$$

$$uoiLabel(E, @concat(Val, "@", Var, "\_", L), 1) \text{ :- not } undef(E, Var);\ Val! = unset;\ stateVariable_{PD}(E, Val, Var);\ min(E, Var);\ uoiLabel(E, L, 0);\ epn_{PD}(E).$$

$$uoiLabel(E, @concat(Val, "@", N, "\_", L), 1) \text{ :- } Val! = unset;\ stateVariable_{PD}(E, Val, undefined(E, N));\ min(E, Var);\ uoiLabel(E, L, 0);\ epn_{PD}(E).$$

Enfin, les trois règles suivantes correspondent à toutes les étapes d'après la première concaténation :

$$uoiLabel(E, L, N + 1) \text{ :- not } undef(E, Var1);\ Val1 = unset;\ N = \{inf(E, Var2, Var1) : stateVariable_{PD}(E, Val2, Var2)\};\ uoiLabel(E, L, N);\ \text{not } min(E, Var1);\ stateVariable_{PD}(E, Val1, Var1);\ epn_{PD}(E).$$

$$uoiLabel(E, @concat(Val1, "@", Var1, "\_", L), N+1) \text{ :- not } undef(E, Var1);\ Val1! = unset;\\ N = \{inf(E, Var2, Var1) : stateVariable_{PD}(E, Val2, Var2)\};\\ uoiLabel(E, L, N);\ \text{not } min(E, Var1);\ stateVariable_{PD}(E, Val1, Var1);\\ epn_{PD}(E).$$

$$uoiLabel(E, @concat(Val1, "@", N, "\_", L), N+1) \text{ :- } Val1! = unset;\\ N = \{inf(E, Var2, Var1) : stateVariable_{PD}(E, Val2, Var2)\};\\ uoiLabel(E, L, N);\ \text{not } min(E, Var1);\\ stateVariable_{PD}(E, Val1, undefined(E, N));\ epn_{PD}(E).$$

Une fois les étiquettes des unités d'information créées, nous créons l'unité d'information de chaque activité à partir du type de l'EPN qui opère cette activité, et de l'étiquette obtenue. Nous donnons comme exemple la règle permettant de créer l'unité d'information d'une activité opérée par une macromolécule. L'étiquette de l'unité d'information obtenue à l'étape N est l'étiquette finale s'il n'existe pas d'étiquette créée à l'étape $N+1$:

$$uoi_a f(a(E, R), macromolecule, L) \text{ :- not } uoiLabel(E, \_, N+1);\ uoiLabel(E, L, N);\\ hasActivity(E, R);\ macromolecule_{PD}(E).$$

Annexe C

Annexe du chapitre 5

Sommaire

C.1 Preuves

C.1.1 Preuve du théorème 5.4

Théorème. Soit P un NLP propositionnel, et $B(P)$ sa traduction en réseau Booléen. Soit I_0 une interprétation de P, et $\langle I_0, I_1, I_2, \ldots\rangle$ l'orbite de I_0 par rapport à P. Alors $S(I_0) \rightarrow_{sy} S(I_1) \rightarrow_{sy} S(I_2) \rightarrow_{sy} \ldots$ est l'unique trace de la dynamique synchrone de $B(P)$ partant de $S(I_0)$.

Preuve. Soit P un NLP propositionnel, tel que $var(P) = \{v_1, \ldots, v_n\}$. Soit $B(P)$ le RB obtenu à partir de P par la traduction introduite dans [Ino11] et donnée dans la section 5.3. Soient I une interprétation de P et s' l'état global de $B(P)$ tel que $S(I) \rightarrow_{sy} s'$. Soit $v_i \in var(P)$ une variable propositionnelle de P. Par construction de $B(P)$, $v_i \in V$. Nous montrons d'abord que $s' = S(T_P(I))$.

- Supposons que $v_i \in T_P(I)$. Alors il existe une règle R de P telle que $head(R) = v_i$ et :
  - $body(R) = \emptyset$ ou
  - $body(R) = B_{i,j}$, avec $B^+_{i,j} \subseteq I$ et $B^-_{i,j} \cap I = \emptyset$.

  Dans le premier cas, $f_i(v_1, \ldots, v_n) = 1$ et $f_i(v_1(S(I)), \ldots, v_n(S(I)) = 1$. Par conséquent, $v_i(s') = 1$.

  Dans le deuxième cas, $f_i(v_1, \ldots, v_n) = B_{i,j} \bigvee_{1 \leq l \leq k_i, l \neq j} B_{i,l}$. Comme $B^+_{i,j} \subseteq I$ et $B^-_{i,j} \cap I = \emptyset$, $f_i(v_1(S(I)), \ldots, v_n(S(I)) = 1$. Par conséquent $v_i(s') = 1$.

  Donc, dans les deux cas, $v_i(s') = 1$.
- Supposons maintenant que $v_i \notin T_P(I)$. Encore, deux cas se présentent :
  - soit il n'existe pas de règle $R \in P$ telle que $head(R) = v_i$. Alors, par construction de $B(P)$, $f_i(v_1, \ldots, v_n) = 0$ et $f_i(v_1(S(I)), \ldots, v_n(S(I)) = 0$. Par conséquent, $v_i(s') = 0$.

— soit il existe $R \in P$ telle que $head(R) = v_i$. Alors toutes les règles de P telles que $head(R) = v_i$ sont de la forme $v_i \leftarrow B_{i,j}$, et pour chacune de ces règles, $B^+_{i,j} \not\subseteq I$ ou $B^-_{i,j} \cap I \neq \emptyset$. Par construction de $B(P)$, $f_i(v_1, \ldots, v_n) = \bigvee_{j=1}^{k_i} B_{i,j}$. Comme, pour tout $1 \leq j \leq k_i$, $B^+_{i,j} \not\subseteq I$ ou $B^-_{i,j} \cap I \neq \emptyset$, $f_i(v_1(S(I)), \ldots, v_n(S(I)) = 0$. Par conséquent, $v_i(s') = 0$.

Nous concluons donc que $v_i(s') = 0$.

Finalement, nous concluons que $s' = S(T_P(I))$, et donc que $S(I) \rightarrow_{sy} S(T_P(I))$.

Soit I_0 une interprétation de P et $\langle I_0, I_1, I_2, \ldots \rangle$ l'orbite de I_0 par rapport à P. Par définition des orbites, $I_i = T_P(I_{i-1})$ pour $i > 0$. Par conséquent, $S(I_0) \rightarrow_{sy} S(I_1) \rightarrow_{sy} S(I_2) \rightarrow_{sy} \ldots$ est l'unique trace de la dynamique synchrone de $B(P)$ partant de $S(I_0)$.

C.1.2 Sketch de preuve de la propriété 5.4

Propriété. Les modèles supportés de Π^2 restreints aux atomes construits à partir du prédicat *present* sont exactement les modèles supportés de Π^f.

Sketch de preuve. Pour un NLP Π ne contenant que des règles de type A, type B et type C, nous dénotons par Π_A (resp. Π_B, Π_C) l'ensemble des règles de type A (resp. type B, type C) de Π. D'abord nous choisissons un modèle supporté M de Π^2, et nous dénotons par M_C l'ensemble des atomes de M restreints au prédicat *present* (i.e au seul prédicat de type C). Nous montrons que $T_{\Pi^f}(M_C) = M_C$ en remarquant que toutes les règles de type C de Π^2 sont dans Π^f, et que toutes les règles de Π^f n'ont que des atomes de type C dans leur corps. Nous concluons sur le fait que M_C est un modèle supporté de Π^f. Nous choisissons ensuite un modèle supporté M_C de Π^f. Nous construisons une interprétation de Herbrand $M = M_C \cup T_{\Pi^2_B}(M_C) \cup T_{\Pi^2_A}(M_C)$ de Π^2, et nous montrons que $T_{\Pi^2}(M) = M$, en remarquant que les règles de Π^2_B n'ont que des atomes de type C dans leur corps, et que les règles de Π^2_C sont des faits. Nous concluons que M est un modèle supporté de Π^2.

C.1.3 Preuve du théorème 5.5

Théorème. Soit S une carte SBGN-AF, $B(S)$ le RB construit à partir de S avec les principes généraux (B1–7). Soit $\Pi_{At}(S)$ le programme défini comme précédemment, et $B(P_{At}(S)^f)$ le RB obtenu en appliquant les étapes de transformation 1–6 à $\Pi_{At}(S)$. Alors le RB $B(P_{At}(S)^f)$ est précisément le RB $B(S)$.

Preuve. Nous définissons une nouvelle classe de NLP que nous appelons *NLP étendus aux DNF*, qui sont formés de règles dont le corps est une DNF. Formellement, les règles d'un tel programme sont de la forme :

$$H \leftarrow C_1 \vee \cdots \vee C_n$$

où les C_i sont des conjonctions de littéraux.

Pour une règle R de cette forme, nous dénotons par $head(R) = H$ la tête de R, et par $body(R) = C_1 \vee \cdots \vee C_n$ le corps de R.

À tout NLP correspond un NLP étendu aux DNF. Soit Π un NLP. Nous dénotons par $DNF(\Pi)$ le NLP étendu aux DNF correpondant à Π, et défini comme suit :

$$DNF(\Pi) = \{h \leftarrow \bigvee_{S \in \Pi, head(S) = h} body(S) \mid h \in atom(\Pi), \exists R \in \Pi : head(R) = h\}$$

Nous définissons deux nouvelles règles de transformation, dénotées (TR1$_{DNF}$) et (TR2$_{DNF}$), que nous allons utiliser à la place des règles de transformation (TR1) et (TR2) dans la suite. Ces deux règles sont définies pour des NLP étendus aux DNF de la manière suivante. Soit Π_{DNF} un NLP étendu aux DNF, et R une règle de ce programme.

(TR1$_{DNF}$) si R est de la forme $h \leftarrow (b \wedge C) \vee D$, où C est une conjonction de littéraux, D est une DNF, h et b sont des atomes tels que $h \neq b$ et il existe une règle S telle que $head(S) = b$, alors remplacer R par :

$$h \leftarrow DNF\Big(\big([\bigvee_{S \in \Pi_{DNF}, head(S)=b} body(S)] \wedge C\big) \vee D\Big)$$

(TR2$_{DNF}$) si R est de la forme $h \leftarrow (\neg b \wedge C) \vee D$, où C est une conjonction de littéraux, D est une DNF, h et b sont des atomes tels que $h \neq b$ et il existe une règle S telle que $head(S) = b$, alors remplacer R par :

$$h \leftarrow DNF\Big(\big(\neg[\bigvee_{S \in \Pi_{DNF}, head(S)=b} body(S)] \wedge C\big) \vee D\Big)$$

Lemme C.1. Soient Π et Π' deux NLP. Il existe une transformation (TR1) (resp. (TR2)) de Π à Π' *ssi* il existe une transformation (TR1$_{DNF}$)(resp. (TR2$_{DNF}$)) de $DNF(\Pi)$ à $DNF(\Pi')$.

Preuve. Soit Π un NLP, et R une règle de Π de la forme $h_1 \leftarrow C_{1,1}$ où $C_{1,1} = l \wedge C$, l étant un littéral formé de l'atome h_2 et C une conjonction de littéraux. Soit $\Pi_1 = \{h_1 \leftarrow C_{1,1}, \ldots, h_1 \leftarrow C_{1,n_1}\}$ l'ensemble des règles de Π dont la tête vaut h_1, et $\Pi_2 = \{h_2 \leftarrow C_{2,1}, \ldots, h_2 \leftarrow C_{2,n_2}\}$ l'ensemble des règles de Π dont la tête vaut h_2. Nous dénotons par $\Pi_3 = \Pi \setminus (\Pi_1 \cup \Pi_2)$ l'ensemble des règles de Π qui ne concluent ni sur h_1 ni sur h_2.

La seule règle R_1 du programme $DNF(\Pi)$ dont la tête vaut h_1 est la règle

$$h_1 \leftarrow \bigvee_{k=1}^{n_1} C_{1,k}$$

et la seule règle R_2 de $DNF(\Pi)$ dont la tête vaut h_2 est la règle

$$h_2 \leftarrow \bigvee_{l=1}^{n_2} C_{2,l}$$

Nous avons $DNF(\Pi) = \{R1\} \cup \{R2\} \cup DNF(\Pi_3)$.

- (TR1) Supposons que $l = h_2$. En appliquant la transformation (TR1) à l'atome h_2 du corps de $R \in \Pi$ nous remplaçons, dans Π, la règle R par l'ensemble de règles suivantes :

$$\{h_1 \leftarrow C_{2,l} \wedge C \mid 1 \leq l \leq n_2\}$$

Par cette transformation, nous obtenons alors le programme Π' suivant :

$$\begin{aligned}\Pi' =& \{h_1 \leftarrow C_{2,l} \wedge C \mid 1 \leq l \leq n_2\} \\ & \cup \{h_1 \leftarrow C_{1,k} \mid 2 \leq k \leq n_1\} \\ & \cup \{h_2 \leftarrow C_{2,l} \mid 1 \leq l \leq n_2\} \\ & \cup \Pi_3\end{aligned}$$

Le programme $DNF(\Pi')$ est alors constitué de $DNF(\Pi_3)$ et des deux règles suivantes :

$$\begin{aligned}h_1 &\leftarrow (\bigvee_{l=1}^{n_2}[C_{2,l} \wedge C]) \vee \bigvee_{k=2}^{n_1} C_{1,k} \\ h_2 &\leftarrow \bigvee_{l=1}^{n_2} C_{2,l}\end{aligned}$$

En appliquant la transformation (TR1$_{DNF}$)à l'atome h_2 du corps de $R_1 \in DNF(\Pi)$ nous remplaçons, dans $DNF(\Pi)$, la règle R_1 par la règle suivante :

$$h_1 \leftarrow DNF\Big((\bigvee_{l=1}^{n_2}[C_{2,l}] \wedge C) \vee \bigvee_{k=2}^{n_1} C_{1,k}\Big)$$

c'est à dire

$$h_1 \leftarrow (\bigvee_{l=1}^{n_2}[C_{2,l} \wedge C]) \vee \bigvee_{k=2}^{n_1} C_{1,k}$$

Par cette transformation, nous obtenons alors le programme $DNF(\Pi)'$, qui est exactement le programme $DNF(\Pi')$.

- (TR2) Supposons que $l = \neg h_2$. En appliquant la transformation (TR2) à l'atome h_2 du corps de $R \in \Pi$ nous remplaçons, dans Π, la règle R par l'ensemble de règles suivantes :

$$\bigcup_{b_1 \in C_{2,1}, \ldots, b_{n_2} \in C_{2,n_2}} \{h_1 \leftarrow \bigwedge_{l=1}^{n_2} \neg b_l \wedge C\}$$

Par cette transformation, nous obtenons alors le porgramme Π' suivant :

$$\begin{aligned}\Pi' = & \bigcup_{b_1 \in C_{2,1}, \ldots, b_{n_2} \in C_{2,n_2}} \{h_1 \leftarrow \bigwedge_{l=1}^{n_2} \neg b_l \wedge C\} \\ & \cup \{h_1 \leftarrow C_{1,k} \mid 2 \leq k \leq n_1\} \\ & \cup \{h_2 \leftarrow C_{2,l} \mid 1 \leq l \leq n_2\} \\ & \cup \Pi_3\end{aligned}$$

Le programme $DNF(\Pi')$ est alors constitué de $DNF(\Pi_3)$ et des deux règles suivantes :

$$h_1 \leftarrow \Big(\bigvee_{b_1 \in C_{2,1},\ldots,b_{n_2} \in C_{2,n_2}} [\bigwedge_{l=1}^{n_2} \neg b_l \wedge C] \Big) \vee \bigvee_{k=2}^{n_1} C_{1,k}$$

$$h_2 \leftarrow \bigvee_{l=1}^{n_2} C_{2,l}$$

En appliquant la transformation (TR2_{DNF}) à l'atome h_2 du corps de $R_1 \in DNF(\Pi)$ nous remplaçons, dans $DNF(\Pi)$, la règle R_1 par la règle suivante :

$$\begin{aligned}
& h_1 \leftarrow DNF\Big(\big(\neg(\bigvee_{l=1}^{n_2} [C_{2,l}]\big) \wedge C\big) \vee \bigvee_{k=2}^{n_1} C_{1,k} \Big) \\
=& h_1 \leftarrow DNF\Big(\big(\bigwedge_{l=1}^{n_2} [\neg C_{2,l}] \wedge C\big) \vee \bigvee_{k=2}^{n_1} C_{1,k} \Big) \\
=& h_1 \leftarrow DNF\Big(\big(\big(\bigvee_{b_1 \in C_{2,1},\ldots,b_{n_2} \in C_{2,n_2}} [\bigwedge_{l=1}^{n_2} \neg b_l] \big) \wedge C\big) \vee \bigvee_{k=2}^{n_1} C_{1,k} \Big) \\
=& h_1 \leftarrow \Big(\bigvee_{b_1 \in C_{2,1},\ldots,b_{n_2} \in C_{2,n_2}} \bigwedge_{l=1}^{n_2} [\neg b_l \wedge C] \Big) \vee \bigvee_{k=2}^{n_1} C_{1,k}
\end{aligned}$$

Par cette transformation, nous obtenons alors le programme $DNF(\Pi)'$, qui est exactement le programme $DNF(\Pi')$.

Par conséquent, il existe une transformation (TR1) (resp. (TR2)) de Π vers Π' *ssi* il existe une tranformation (TR1_{DNF}) (resp. (TR2_{DNF})) de $DNF(\Pi)$ vers $DNF(\Pi')$. □

Nous montrons maintenant la proposition, à l'aide des nouvelles règles de transformation que nous avons introduites.

Soit S une carte SBGN-AF. Nous dénotons par $\mathcal{A} = \{a_1, \ldots, a_n\}$ l'ensemble des activités de S, par $\mathcal{O} = \{o_1, \ldots, o_m\}$ l'ensemble des opérateur logiques de S, par $\mathcal{O}_{AND}$ (resp. $\mathcal{O}_{OR}$, $\mathcal{O}_{NOT}$) l'ensemble des opérateurs logiques AND de S (resp. OR, NOT). Pour une activité $a_i \in \mathcal{A}$ de S, nous dénotons par $req(a_i)$ (resp. $stim(a_i)$, $inh(a_i)$) les sources de l'ensemble des stimulations nécessaires (resp. stimulations, inhibitions) ciblant a_i. Pour un opérateur logique $o_i \in \mathcal{O}$ de S, nous dénotons par $in(o_i)$ l'ensemble des noeuds (activités ou opérateurs logiques) à la source d'un arc logique ciblant o_i.

zsh :1 : command not found : :w Soient $\Pi_{TRAD}(S)$ la traduction de S en SBGNLog-AF, Π_{ONTO} l'ensemble des axiomes ontologiques de SBGNLog-AF limités à ceux traduisant les relations is_a, et Π_A l'ensemble des axiomes (A1–18). Soit $\Pi(S)$ le programme défini par :

$$\Pi(S) \triangleq \Pi_{TRAD}(S) \cup \Pi_{ONTO} \cup \Pi_A$$

Finalement, soit $\Pi_{At}(S)$ le programme obtenu de $\Pi(S)$ en supprimant la notion de temps de ce programme.

Nous appliquons les trois étapes de transformation à $\Pi_{At}(S)$ pour obtenir le programme $\Pi_{At}(S)^f$.

• Étape 1 : nous appliquons iérativement les règles de simplification (SR1–4) aux règles de $\Pi_{At}(S)$. Nous obtenons le programme $\Pi_{At}(S)^1$ constitué de l'ensemble des règles suivantes :

$$\{hasModulator(a_i) \leftarrow \mid a_i \in \mathcal{A}, req(a_i) \cup stim(a_i) \cup inh(a_i) \neq \emptyset\} \quad \text{(E1)}$$
$$\cup\{hasStimulator(a_i) \leftarrow \mid a_i \in \mathcal{A}, req(a_i) \cup stim(a_i) \neq \emptyset\} \quad \text{(E2)}$$
$$\cup\{hasPresentStimulator(a_i) \leftarrow present(s) \mid a_i \in \mathcal{A}, s \in req(a_i) \cup stim(a_i), s \in \mathcal{A}\} \quad \text{(E3)}$$
$$\cup\{hasPresentStimulator(a_i) \leftarrow logic(s) \mid a_i \in \mathcal{A}, s \in req(a_i) \cup stim(a_i), s \in \mathcal{O}\} \quad \text{(E4)}$$
$$\cup\{hasPresentInhibitor(a_j) \leftarrow present(i) \mid a_j \in \mathcal{A}, i \in inh(a_j), i \in \mathcal{A}\} \quad \text{(E5)}$$
$$\cup\{hasPresentInhibitor(a_j) \leftarrow logic(i) \mid a_j \in \mathcal{A}, i \in inh(a_j), i \in \mathcal{O}\} \quad \text{(E6)}$$
$$\cup\{hasAbsentNecessaryStimulator(a_i) \leftarrow \neg present(r) \mid a_i \in \mathcal{A}, r \in req(a_i), r \in \mathcal{A}\} \quad \text{(E7)}$$
$$\cup\{hasAbsentNecessaryStimulator(a_i) \leftarrow \neg logic(r) \mid a_i \in \mathcal{A}, r \in req(a_i), r \in \mathcal{O}\} \quad \text{(E8)}$$
$$\cup\{notLogic(o_i) \leftarrow \neg present(j) \mid o_i \in \mathcal{O}_{AND}, j \in in(o_i), j \in \mathcal{A}\} \quad \text{(E9)}$$
$$\cup\{notLogic(o_i) \leftarrow \neg logic(j) \mid o_i \in \mathcal{O}_{AND}, j \in in(o_i), j \in \mathcal{O}\} \quad \text{(E10)}$$
$$\cup\{logic(o_i) \leftarrow \neg notLogic(o_i) \mid o_i \in \mathcal{O}_{AND}\} \quad \text{(E11)}$$
$$\cup\{logic(o_i) \leftarrow present(j) \mid o_i \in \mathcal{O}_{OR}, j \in in(o_i), j \in \mathcal{A}\} \quad \text{(E12)}$$
$$\cup\{logic(o_i) \leftarrow logic(j) \mid o_i \in \mathcal{O}_{OR}, j \in in(o_i), j \in \mathcal{O}\} \quad \text{(E13)}$$
$$\cup\{logic(o_i) \leftarrow \neg present(j) \mid o_i \in \mathcal{O}_{NOT}, \{j\} = in(o_i), j \in \mathcal{A}\} \quad \text{(E14)}$$
$$\cup\{logic(o_i) \leftarrow \neg logic(j) \mid o_i \in \mathcal{O}_{NOT}, \{j\} = in(o_i), j \in \mathcal{O}\} \quad \text{(E15)}$$
$$\cup\{present(a_i) \leftarrow present(a_i) \mid a_i \in \mathcal{A}, req(a_i) \cup stim(a_i) \cup inh(a_i) = \emptyset\} \quad \text{(E16)}$$
$$\cup\{present(a_i) \leftarrow \neg hasPresentInhibitor(a_i) \mid a_i \in \mathcal{A}, req(a_i) \cup stim(a_i) = \emptyset, inh(a_i) \neq \emptyset\} \quad \text{(E17)}$$
$$\cup\{present(a_i) \leftarrow hasPresentStimulator(a_i) \wedge \neg hasAbsentNecessaryStimulator(a_i)$$
$$\wedge \neg hasPresentInhibitor(a_i) \mid a_i \in \mathcal{A}, req(a_i) \cup stim(a_i) \neq \emptyset\} \quad \text{(E18)}$$

• Étape 2 : par simplicité pour la démonstration, nous n'appliquons par les règles de transformation (TR1) et (TR2), mais les règles (TR1_{DNF})et (TR2_{DNF}). Dans les règles de (E16), il n'y a aucun atome à remplacer. Dans les règles de (E17), nous appliquons (TR2_{DNF}) à l'atome $hasPresentInhibitor(a_i)$. Nous remplaçons donc (E17) par l'ensemble de règles suivantes :

$$\Big\{present(a_j) \leftarrow DNF\Big(\neg\big((\bigvee_{i \in inh(a_j), i \in \mathcal{A}} present(i)) \vee (\bigvee_{i \in inh(a_j), i \in \mathcal{O}} logic(i))\big)\Big)$$
$$\mid a_j \in \mathcal{A}, req(a_j) \cup stim(a_j) = \emptyset, inh(a_j) \neq \emptyset\Big\}$$
$$= \Big\{present(a_j) \leftarrow DNF\Big((\bigwedge_{i \in inh(a_j), i \in \mathcal{A}} \neg present(i)) \wedge (\bigwedge_{i \in inh(a_j), i \in \mathcal{O}} \neg logic(i))\Big)$$
$$\mid a_j \in \mathcal{A}, req(a_j) \cup stim(a_j) = \emptyset, inh(a_j) \neq \emptyset\Big\} \quad \text{(E17}')$$

Dans les règles de (E18), nous appliquons (TR1_{DNF}) à l'atome $hasPresentStimulator(a_i)$, et ($\text{TR2}_{DNF}$) aux atomes $hasPresentInhibitor(a_i)$ et $hasAbsentNecessaryStimulator(a_i)$. Nous remplaçons donc

(E18) par l'ensemble de règles suivantes :

$$\Big\{present(a_j) \leftarrow DNF\Big(\big[\big(\bigvee_{s\in req(a_j)\cup stim(a_j),s\in\mathcal{A}} present(s)\big) \vee \big(\bigvee_{s\in req(a_j)\cup stim(a_j),s\in\mathcal{O}} logic(s)\big)\big]$$
$$\wedge \big[\neg\big(\big(\bigvee_{r\in req(a_j),r\in\mathcal{A}} \neg present(r)\big) \vee \big(\bigvee_{r\in req(a_j),r\in\mathcal{O}} \neg logic(r)\big)\big)\big]$$
$$\wedge \big[\neg\big(\big(\bigvee_{i\in inh(a_j),i\in\mathcal{A}} present(i)\big) \vee \big(\bigvee_{i\in inh(a_j),i\in\mathcal{O}} logic(i)\big)\big)\Big)$$
$$\mid a_j \in \mathcal{A}, req(a_j)\cup stim(a_j) \neq \emptyset\Big\}$$
$$=\Big\{present(a_j) \leftarrow DNF\Big(\big[\big(\bigvee_{s\in req(a_j)\cup stim(a_j),s\in\mathcal{A}} present(s)\big) \vee \big(\bigvee_{s\in req(a_j)\cup stim(a_j),s\in\mathcal{O}} logic(s)\big)\big]$$
$$\wedge \big[\big(\bigwedge_{r\in req(a_j),r\in\mathcal{A}} present(r)\big) \wedge \big(\bigwedge_{r\in req(a_j),r\in\mathcal{O}} logic(r)\big)\big]$$
$$\wedge \big[\big(\bigwedge_{i\in inh(a_j),i\in\mathcal{A}} \neg present(i)\big) \wedge \big(\bigwedge_{i\in inh(a_j),i\in\mathcal{O}} \neg logic(i)\big)\big]\Big)$$
$$\mid a_j \in \mathcal{A}, req(a_j)\cup stim(a_j) \neq \emptyset\Big\} \qquad (E18')$$

Il nous reste à remplacer itérativement les atomes de la forme $logic(o_i)$ du corps des règles de (E17′) et (E18′) par leur définition, à l'aide des règles de transformation ($TR1_{DNF}$) et ($TR2_{DNF}$), pour obtenir les ensembles de règles (E17″) et (E18″).

- Si $o_i \in \mathcal{O}_{AND}$, $logic(o_i)$ est remplacé par :

$$\neg notLogic(o_i)$$

Le littéral $\neg notLogic(o_i)$ est à son tour remplacé à l'aide de la règle ($TR2_{DNF}$) par :

$$\neg\big(\big(\bigvee_{j\in in(o_i),j\in\mathcal{A}} \neg present(j)\big) \vee \big(\bigvee_{j\in in(o_i),j\in\mathcal{O}} \neg logic(j)\big)\big)$$
$$=\big(\bigwedge_{j\in in(o_i),j\in\mathcal{A}} present(j)\big) \wedge \big(\bigwedge_{j\in in(o_i),j\in\mathcal{O}} logic(j)\big) \qquad (AND)$$

- Si $o_i \in \mathcal{O}_{OR}$, $logic(o_i)$ est remplacé par :

$$\big(\bigvee_{j\in in(o_i),j\in\mathcal{A}} present(j)\big) \vee \big(\bigvee_{j\in in(o_i),j\in\mathcal{O}} logic(j)\big) \qquad (OR)$$

- Si $o_i \in \mathcal{O}_{NOT}$, $logic(o_i)$ est remplacé par :

$$\neg present(j) \text{ si } o_i \in \mathcal{O}_{NOT}, \{j\} = in(o_i) \text{ et } j \in \mathcal{A};$$
$$\neg logic(j) \text{ si } o_i \in \mathcal{O}_{NOT}, \{j\} = in(o_i) \text{ et } j \in \mathcal{O}. \qquad (NOT)$$

Nous définissons une fonction récursive logic^{FO} sur $\mathcal{A} \cup \mathcal{O}$ telle que :

$$\mathsf{logic}^{FO}(q) = \begin{cases} present(q) \text{ si } q \in \mathcal{A} \ ; \\ \bigwedge\limits_{j \in in(q)} \mathsf{logic}^{FO}(j) \text{ si } q \in \mathcal{O}_{AND} \ ; \\ \bigvee\limits_{j \in in(q)} \mathsf{logic}^{FO}(j) \text{ si } q \in \mathcal{O}_{OR} \ ; \\ \neg\mathsf{logic}^{FO}(j) \text{ si } q \in \mathcal{O}_{NOT} \text{ et } \{j\} = in(q) \ . \end{cases}$$

Nous remarquons que remplacer itérativement les atomes de la forme $logic(o_i)$ dans le corps des règles des ensembles (E17′) et (E18′) par (AND), (OR) et (NOT) revient à remplacer ces atomes par $\mathsf{logic}^{FO}(o_i)$. Nous pouvons donc écrire les ensembles (E17″) et (E18″) à l'aide de la fonction logic^{FO} :

$$\left\{ present(a_j) \leftarrow DNF\Big(\bigwedge_{i \in inh(a_j)} \neg\mathsf{logic}^{FO}(i) \Big) \mid a_j \in \mathcal{A}, req(a_j) \cup stim(a_j) = \emptyset, inh(a_j) \neq \emptyset \right\} \quad \text{(E17″)}$$

$$\left\{ present(a_j) \leftarrow DNF\Big(\Big[\bigvee_{s \in req(a_j) \cup stim(a_j)} \mathsf{logic}^{FO}(s) \Big] \wedge \Big[\bigwedge_{r \in req(a_j)} \mathsf{logic}^{FO}(r) \Big] \wedge \Big[\bigwedge_{i \in inh(a_j)} \neg\mathsf{logic}^{FO}(i) \Big] \Big) \right.$$
$$\left. \mid a_j \in \mathcal{A}, req(a_j) \cup stim(a_j) \neq \emptyset \right\} \quad \text{(E18″)}$$

Nous obtenons un programme $\Pi_{At}(S)^2_{DNF}$ formé de l'union des ensembles de règles (E1–16), (E17″) et (E18″).

Par le lemme C.1, il existe un programme $\Pi_{At}(S)^2$ obtenu par application itérative des règles de transformation (TR1) et (TR2) tel que :

$$DNF(\Pi_{At}(S)^2) = \Pi_{At}(S)^2_{DNF}$$

Comme plus aucune règle de transformation (TR1_{DNF}) ou (TR2_{DNF}) n'est applicable aux atomes du corps des règles de type C de $\Pi_{At}(S)^2_{DNF}$, plus aucune règle de transformation (TR1) ou (TR2) n'est applicable aux règles de type C de $\Pi_{At}(S)^2$. Le programme $\Pi_{At}(S)^2$ est donc exactement le programme qui aurait été obtenu en appliquant l'étape 2 avec les règles de transformation (TR1) et (TR2).

• Étape 3 : nous supprimons toutes les règles de $\Pi_{At}(S)^2_{DNF}$ qui ne sont pas de type C pour obtenir un programme $\Pi_{At}(S)^f_{DNF}$. Les seules règles de type C de $\Pi_{At}(S)^2_{DNF}$ étant celles des ensembles (E16), (E17″) et (E18″), $\Pi_{At}(S)^f_{DNF}$ est formé de ces trois ensembles de règles.

De plus, le programme $\Pi_{At}(S)^f_{DNF}$ vérifie l'égalité suivante :

$$\Pi_{At}(S)^f_{DNF} = DNF(\Pi_{At}(S)^f)$$

• Nous construisons maintenant $B(\Pi_{At}(S)^f)$ à partir de $\Pi_{At}(S)^f_{DNF}$.

Nous remarquons d'abord la propriété suivante. Soit P un NLP propositionnel tel que $var(P) = \{v_1, \ldots, v_u\}$, et $B(P) = (V, F)$ le RB obtenu à partir de P par la traduction introduite dans [Ino11] et donnée dans la section 5.3. Si $v_i \in var(P)$ est une variable de P telle qu'il existe au moins une règle dans P qui conclut sur v_i, et aucune des règles de P concluant sur v_i n'est un fait, alors nous avons l'égalité suivante concernant la fonction $f_i \in F$ et le programme $DNF(P)$:

$$f_i(v_1, \ldots, v_u) = body(R)$$

où R est la règle de $DNF(P)$ concluant sur v_i.

Par conséquent, pour un programme P dont les variables de P respectent la propriété précédente, nous pouvons définir $B(P)$ à partir de $DNF(P)$:

$$V \triangleq \{v_i \mid v_i \in var(DNF(P))\}$$

et

$$F \triangleq \{f_i = body(R) \mid v_i \in var(DNF(P)), head(R) = v_i\}$$

Nous transformons d'abord $\Pi_{At}(S)^f_{DNF}$ en un NLP propositionnel étendu aux DNF $P_{At}(S)^f_{DNF}$ en remplaçant dans $\Pi_{At}(S)^f_{DNF}$ tous les atomes de la forme $present(a_i)$ par la variable propositionnelle v_i.

Nous définissons la fonction logic^P, définie sur $\mathcal{A} \cup \mathcal{O}$, qui est définie comme la fonction logic^{FO} mais en prenant en compte ce remplacement :

$$\mathsf{logic}^P(q) = \begin{cases} v_i \text{ si } q = a_i, a_i \in \mathcal{A} \ ; \\ \bigwedge\limits_{j \in in(q)} \mathsf{logic}^P(j) \text{ si } q \in \mathcal{O}_{AND} \ ; \\ \bigvee\limits_{j \in in(q)} \mathsf{logic}^P(j) \text{ si } q \in \mathcal{O}_{OR} \ ; \\ \neg\mathsf{logic}^P(j) \text{ si } q \in \mathcal{O}_{NOT} \text{ et } \{j\} \in in(q) \ . \end{cases}$$

Le programme $P_{At}(S)^f_{DNF}$ est formé des ensembles de règles suivants :

$$\left\{ v_i \leftarrow v_i \mid a_i \in \mathcal{A}, req(a_i) \cup stim(a_i) \cup inh(a_i) = \emptyset \right\} \tag{E16^P}$$

$$\cup \left\{ v_i \leftarrow DNF\Big(\bigwedge_{i \in inh(a_i)} \neg\mathsf{logic}^P(i) \Big) \mid a_i \in \mathcal{A}, req(a_i) \cup stim(a_i) = \emptyset, inh(a_i) \neq \emptyset \right\} \tag{E17^P}$$

$$\cup \left\{ v_i \leftarrow DNF\Big(\Big[\bigvee_{s \in req(a_i) \cup stim(a_i)} \mathsf{logic}^P(s) \Big] \wedge \Big[\bigwedge_{r \in req(a_i)} \mathsf{logic}^P(r) \Big] \wedge \Big[\bigwedge_{i \in inh(a_i)} \neg\mathsf{logic}^P(i) \Big] \Big) \right.$$
$$\left. \mid a_i \in \mathcal{A}, req(a_i) \cup stim(a_i) \neq \emptyset \right\} \tag{E18^P}$$

La même transformation peut être appliquée à $\Pi_{At}(S)^f$ pour obtenir un programme propositionnel $P_{At}(S)^f$ tel que :

$$DNF(P_{At}(S)^f) = P_{At}(S)^f_{DNF}$$

Comme pour toute variable $v_i \in var(P_{At}(S)^f)$, il y a au moins une règle de $P_{At}(S)^f$ qui conclut sur v_i, et aucune règle de P concluant sur v_i n'est un fait, nous construisons $B(\Pi_{At}(S)^f)$ à partir de $P_{At}(S)^f_{DNF}$. Nous obtenons le RB $B(P_{At}(S)^f)$ tel que :

$$V = \{v_i \mid a_i \in \mathcal{A}\}$$

et

$$F = \{f_i \mid a_i \in \mathcal{A}\}$$

où

$$f_i = \begin{cases} v_i \text{ si } req(a_j) = stim(a_j) = inh(a_j) = \emptyset; \\ DNF\Big(\bigwedge\limits_{i \in inh(a_j)} \neg \mathsf{logic}^P(i)\Big) \text{ si } req(a_j) = stim(a_j) = \emptyset \text{ et } inh(a_j) = \emptyset; \\ DNF\Big(\big[\bigvee\limits_{s \in req(a_j) \cup stim(a_j)} \mathsf{logic}^P(s)\big] \wedge \big[\bigwedge\limits_{r \in req(a_j)} \mathsf{logic}^P(r)\big] \wedge \big[\bigwedge\limits_{i \in inh(a_j)} \neg \mathsf{logic}^P(i)\big]\Big) \text{ si } req(a_j) \cup stim(a_j) \neq \emptyset. \end{cases}$$

En remarquant que la fonction logic^P est définie exactement de la même façon que la fonction logic, nous concluons $B(\Pi_{At}(S)^f)$ est exactement $B(S)$.

C.1.4 Preuve de la proposition 5.1

Proposition. Soit S une carte SBGN-AF, $\Pi_{At}(S)$ le programme défini à partir de S comme précédemment, et $B(S)$ le RB construit à partir de S avec les principes généraux (B1–7). Soit M une interprétation de Herbrand de $\Pi_{At}(S)$. Alors M est un modèle supporté de $\Pi_{At}(S)$ *ssi* $S(M)$ est un point attracteur de $B(S)$.

Preuve. Cette proposition découle immédiatement de la propriété 5.5 sur la conservation des modèles supportés par les règles de simplification et de transformation des NLP que nous avons introduites, du théorème 5.5 et du théorème 5.3.

C.1.5 Preuve de la proposition 5.2

Proposition. Soit S une carte SBGN-AF, T_{max} un entier positif non nul, $\Pi_{Tr}(S, T_{max})$ le programme défini à partir de S et T_{max} comme précédemment, et $B(S)$ le RB construit à partir de S avec les principes généraux (B1–7). Soit s un état global de $B(S)$, et M l'unique modèle supporté de $\Pi_{Tr}(S, T_{max}) \cup I_0(s)$. Alors $s \rightarrow_{sy} S_1(M) \rightarrow_{sy} \cdots \rightarrow_{sy} S_{T_{max}}(M)$ est l'unique trace finie de la dynamique synchrone de $B(S)$ partant de s et arrivant sur $S_{T_{max}}(M)$.

Preuve. Soit S une carte SBGN-AF et T_{max} un entier positif non nul. Soient $\Pi_{At}(S)$ le programme défini à partir de S et $\Pi_{Tr}(S, T_{max})$ le programme défini à partir de S et T_{max}, comme dans la sous-section 5.5.1. Soient $\Pi_{At}(S)^f$ et $\Pi_{Tr}(S, T_{max})^f$ les programmes obtenus par transformation de $\Pi_{At}(S)$ et $\Pi_{Tr}(S, T_{max})$, comme dans la sous-section 5.5.2. Soit $P_{At}(S)^f$ le NLP propositionnel obtenu à partir de $\Pi_{At}(S)^f$, comme dans la sous-section 5.5.3. Soit $B(S)$ le RB construit à partir de S avec les principes généraux (B1–7). Finalement, soit s un état global de $B(S)$. Nous dénotons par $I(s) = \{v_i \mid v_i(s) = 1\}$ l'interprétation de Herbrand de $P_{At}(S)^f$ correspondant à s.

Nous montrons d'abord que $\Pi_{Tr}(S, T_{max})^f$ permet de calculer les orbites de longueur T_{max} de $P_{At}(S)^f$.

Avec des transformations similaires à celles effectuées dans la preuve du théorème 5.5, nous pouvons montrer le lemme suivant, qui donne une relation de structure entre $\Pi_{Tr}(S, T_{max})^f$ et $P_{At}(S)^f$:

Lemme C.2. Pour toute règle $R = v_i \leftarrow v_{j,1} \wedge \cdots \wedge v_{j,k} \wedge \neg v_{j,k+1} \wedge \cdots \wedge \neg v_{j,q}$ (avec $q \geq k \geq 0$, et $k \geq 1$ ou $q \geq 1$) de $P_{At}(S)^f$ et pour tout $0 \leq t < T_{max}$, $\Pi_{Tr}(S, T_{max})^f$ contient exactement une règle de la forme :

$$present(a_i, t+1) \leftarrow present(a_{j,1}, t) \wedge \cdots \wedge present(a_{j,k}, t) \wedge \neg present(a_{j,k+1}, t) \wedge \cdots \wedge \neg present(a_{j,q}, t)$$

De plus, $\Pi_{Tr}(S, T_{max})^f$ ne contient pas d'autres règles.

Soit I_0 une interprétation de Herbrand de $P_{At}(S)^f$. Nous dénotons par $\Pi_{Tr}(S, T_{max}, I_0)^f$ le programme suivant :

$$\Pi_{Tr}(S, T_{max}, I_0)^f \triangleq \Pi_{Tr}(S, T_{max})^f \cup \{present(a_i, 0) \mid v_i \in I_0\}$$

Comme $\Pi_{Tr}(S, T_{max}, I_0)^f$ est fortement stratifié, il n'a qu'un seul modèle supporté. Nous dénotons ce modèle par M, et pour $0 \leq t \leq T_{max}$, nous dénotons par M_t l'ensemble de variables propositionnelles de $P_{At}(S)^f$ défini de la manière suivante :

$$M_t = \{v_i \mid present(a_i, t) \in M\}$$

Nous montrons par récurrence sur t ($0 \leq t \leq T_{max}$) que pour toute interprétation I_t de l'orbite $\langle I_0, \ldots, I_{T_{max}} \rangle$ de I_0 par rapport à $P_{At}(S)^f$, $I_t = M_t$.

- Cas $t = 0$. Nous montrons par double inclusion que $M_0 = I_0$.
  - Soit $v_i \in M_0$ une variable propositionnelle. Alors, par définition, $present(a_i, 0) \in M$. Étant donnée la construction de $\Pi_{Tr}(S, T_{max}, I_0)^f$ et le lemme C.2, aucune autre règle de $\Pi_{Tr}(S, T_{max}, I_0)^f$ que le fait $present(a_i, 0) \leftarrow$ ne conclut sur $present(a_i, 0)$, et ce fait n'appartient pas à $\Pi_{Tr}(S, T_{max})^f$. Donc $v_i \in I_0$.
  - Soit $v_i \in I_0$ une variable propositionnelle. Alors, par construction de $\Pi_{Tr}(S, T_{max}, I_0)^f$, la règle $present(a_i, 0) \leftarrow$ appartient à $\Pi_{Tr}(S, T_{max}, I_0)^f$. Par conséquent $present(a_i, 0) \in M$, et $v_i \in M_0$.

  Finalement, $I_0 = M_0$.
- Cas $t = n$, $0 < n \leq T_{max}$. Supposons que $M_{n-1} = I_{n-1}$. Nous montrons par double inclusion que $M_n = I_n$.
  - Soit $v_i \in M_n$ une variable propositionnelle. Alors, par définition, $present(a_i, n) \in M$. Comme M est un modèle supporté de $\Pi_{Tr}(S, T_{max}, I_0)^f$, $T_{\Pi_{Tr}(S,T_{max},I_0)^f}(M) = M$. Il existe donc une règle $R \in \Pi_{Tr}(S, T_{max}, I_0)^f$ telle que $head(R) = present(a_i, n)$, $body^+(R) \subseteq M$ et $body^-(R) \cap M = \emptyset$. Or $M_{n-1} = I_{n-1}$, donc pour tout atome de la forme $present(a_j, n-1)$ appartenant à $body^+(R)$ et tel que $v_j \in var(P_{At}(S)^f)$, $v_j \in I_{n-1}$, et pour tout atome de la forme $present(a_k, n-1)$ appartenant à $body^-(R)$ et tel que $v_k \in var(P_{At}(S)^f)$, $v_k \notin I_{n-1}$. Par le lemme C.2, il existe donc une règle R' de $P_{At}(S)^f$ telle que $head(R') = v_i$, pour tout $v_j \in body^+(R)$, $v_j \in I_{n-1}$, et pour tout $v_k \in body^-(R')$, $v_k \notin I_{n-1}$. D'après la définition de l'opérateur de conséquence immédiate, on a donc $v_i \in T_{P_{At}(S)^f}(I_{n-1})$. Or $T_{P_{At}(S)^f}(I_{n-1}) = I_n$, donc $v_i \in I_n$.
  - Soit $v_i \in I_n$ une variable propositionnelle. Comme $I_n = T_{P_{At}(S)^f}(I_{n-1})$, il existe une règle $R \in P_{At}(S)^f$ telle que $head(R) = v_i$, $body^+(R) \subseteq I_{n-1}$ et $body^-(R) \cap I_{n-1} = \emptyset$. Par le lemme C.2, il existe donc une règle $R' \in \Pi_{Tr}(S, T_{max}, I_0)^f$ telle que :

— $head(R') = present(a_i, n)$;
— $body^+(R') = \{present(a_j, n-1) \mid v_j \in body^+(R)\}$;
— $body^-(R') = \{present(a_k, n-1) \mid v_k \in body^-(R)\}$.

Comme $body^+(R) \subseteq I_{n-1}$, $M_{n-1} = I_{n-1}$, par définition de M_n, $body^+(R') \subseteq M$. Comme $body^-(R) \cap I_{n-1} = \emptyset$, $M_{n-1} = I_{n-1}$et que pour tout variable propositionnelle v_l, $v_l \in M_{n-1}$ *ssi* $present(a_l, n-1) \in M$, par définition de M_n, nous avons $body^-(R') \cap M = \emptyset$. Nous avons donc, par définition de l'opérateur de conséquence immédiate, $present(a_i, n) \in T_{\Pi_{Tr}(S,T_{max},I_0)^f}(M)$. Or M est un modèle supporté de $\Pi_{Tr}(S, T_{max}, I_0)^f$, donc $T_{\Pi_{Tr}(S,T_{max},I_0)^f}(M) = M$. Par conséquent, $present(a_i, n) \in M$, et $v_i \in M_n$.

Nous avons donc $M_n = I_n$.

Par conséquent, pour tout $0 \leq t \leq T_{max}$, $M_t = I_t$.

Nous rappelons les notations suivantes. Pour un interprétation de Herbrand de I de $P_{At}(S)^f$, nous dénotons par $S(I)$ l'état global de $B(S)$ correspondant, et défini de la manière suivante :

$$S(I) = (b_1, \ldots, b_n) \text{ où } \begin{cases} b_i = 1 \text{ si } present(a_i) \in I; \\ b_i = 0 \text{ sinon.} \end{cases}$$

Pour une interprétation de Herbrand I de $\Pi_{Tr}(S, T_{max})$ et un entier t, nous dénotons par $S_t(I)$ l'état global de $B(S)$ correspondant à I au temps t :

$$S_t(I) = (b_1, \ldots, b_n) \text{ où } \begin{cases} b_i = 1 \text{ si } present(a_i, t) \in I; \\ b_i = 0 \text{ sinon.} \end{cases}$$

Nous supposons maintenant que $S(I_0) = s$. D'après les théorèmes 5.5 et 5.4, $S(I_0) \to_{sy} \cdots \to_{sy} S(I_{T_{max}})$ est l'unique trace finie de la dynamique synchrone de $B(S)$ partant de $S(I_0)$ et arrivant sur $S(I_{T_{max}})$. Or, pour tout $0 \leq t \leq T_{max}$, $M_t = I_t$. Par conséquent, cette trace finie peut se réécrire sous la forme $S(M_0) \to_{sy} \cdots \to_{sy} S(M_{T_{max}})$. Comme, pout tout $0 \leq t \leq T_{max}$, $S(M_t) = S_t(M)$, nous concluons finalement que $S_0(M) \to_{sy} \cdots \to_{sy} S_{T_{max}}(M)$ est l'unique trace finie de la dynamique synchrone de $B(S)$ partant de s et arrivant sur $S_{T_{max}}(M)$.

C.2 Encodage ASP pour le calcul des points attracteurs et des traces finies de la dynamique asynchrone

Étant donné un graphe d'influences S, le RB $B(S)$ modélisant S d'après les principes généraux (B1–7) comporte une fonction Booléenne par activité de S. Choisir une fonction Booléenne à appliquer à chaque pas de temps revient donc à choisir exactement une activité de S. Ce choix peut être encodé par la règle ASP suivante :

$$1\{apply(A, T) : activity(A)\}1 \text{ :- } time(T), t < tmax. \tag{A19}$$

Ensuite, le déclenchement des axiomes (A16–18) ne peut se faire que pour l'activité qui a été choisie au temps T. Ceci peut être encodé en ajoutant à chacun de ces axiomes la condition $apply(A, T)$, pour former les nouveaux axiomes suivants :

$$\begin{aligned} present(A, T') \text{ :- } & time(T); time(T'); next(T', T); activity(A); \\ & apply(A, T); \text{ not } hasModulator(A); present(A, T). \end{aligned} \tag{A20}$$

$$present(A,T') \text{ :- } time(T);\ time(T');\ next(T',T);\ activity(A);\ apply(A,T);\ hasModulator(A);\ \text{not } hasStimulator(A);\ \text{not } hasPresentInhibitor(A,T). \quad \text{(A21)}$$

$$present(A,T') \text{ :- } time(T);\ time(T');\ next(T',T);\ activity(A);\ apply(A,T);\ hasModulator(A);\ hasPresentStimulator(A,T);\ \text{not } hasAbsentNecessaryStimulator(A,T);\ \text{not } hasPresentInhibitor(A,T). \quad \text{(A22)}$$

Enfin, une activité qui n'a pas été choisie au temps T conserve sa valeur au temps T', qui est le successeur de T :

$$present(A,T') \text{ :- } time(T);\ time(T');\ next(T',T);\ activity(A);\ \text{not } apply(A,T);\ present(A,T). \quad \text{(A23)}$$

Maintenant, considérons le programme $\Pi(S)' = \Pi_{TRAD}(S)' \cup \Pi'_{ONTO} \cup \Pi'_A$ où :

- $\Pi_{TRAD}(S)'$ est la traduction de S en SBGNLog-AF écrite en ASP ;
- Π'_{ONTO} est l'ensemble des axiomes ontologiques de SBGNLog-AF écrits en ASP ;
- Π'_A est l'ensemble des axiomes (A1–A15) écrits en ASP et des axiomes (A19–A23) nouvellement définis.

Considérons également un entier positif T_{max}, le programme ASP $\Pi_T(T_{max})'$, qui est la version ASP du programme logique $\Pi_T(T_{max})$, et le programme ASP $\Pi_{Tr}(S,T_{max})' = \Pi(S)' \cup \Pi_T(T_{max})'$.

Alors, de la même façon que les traces finies de la dynamique synchrone peuvent être calculées à partir de $\Pi_{Tr}(S,T_{max})$, les traces de sa dynamique synchrone peuvent être calculées à partir de $\Pi_{Tr}(S,T_{max})'$.

Nous avons la propriété suivante :

> **Propriété C.1.** Soit s un état global de $B(S)$, T_{max} un entier positif non nul, et $\{M_1,\ldots,M_n\}$ l'ensemble des modèles stables de $\Pi_{Tr}(S,T_{max})'$. Alors l'ensemble $\{s \rightarrow_{sy} S_1(M_i) \rightarrow_{sy} \cdots \rightarrow_{sy} S_{T_{max}}(M_i) \mid 1 \leq i \leq n\}$ est l'ensemble des traces finies de taille T_{max} partant de s.

Notons qu'à deux modèles stables de $\Pi_{Tr}(S,T_{max})'$ peut correspondre la même trajectoire. En effet, chaque modèle stable M_i contient un ensemble d'atomes $\{apply(a_{i,t},t) \mid a_{i,t} \in V, 0 \leq t < T_{max}\}$ tel que, pour deux modèles stables M_i et M_j donnés, il existe un entier t pour lequel $a_{i,t} \neq a_{j,t}$. Or choisir, à un instant t donné, c'est-à-dire à partir d'un état global donné, d'appliquer telle fonction d'un RB plutôt que telle autre n'influe pas sur la trace partant de cet état, si appliquer l'une ou l'autre fonction ne change pas les valeurs des variables qui leur sont associées, i.e. ne change pas l'état global du RA.

Annexe D

Annexe du chapitre 6

Sommaire

D.1 Complexité de l'encodage

Dans la suite de cette section, nous donnons une évaluation de la complexité de notre encodage. Nous calculons, étant donnés une carte SBGN-PD et un ensemble valide d'histoires, le nombre d'automates, d'états globaux et de transitions nécessaires pour encoder cette carte en considérant cet ensemble d'histoires.

Nous rappelons les notations suivantes pour nous référer aux éléments de la carte :

- $\mathcal{E} = \{e_1, \ldots, e_n\}$ l'ensemble des EPNs de la carte ;
- $\mathcal{P} = \{p_1, \ldots, p_m\}$ l'ensemble des processus de la carte ;
- $\mathcal{O} = \{o_1, \ldots, o_q\}$ l'ensemble des opérateurs logiques de la carte ;
- Pour chaque processus $p \in \mathcal{P}$, nous dénotons par $reac(p)$ (resp. $prod(p)$) l'ensemble des réactifs (resp. produits) de p qui ne sont pas de type source ou puits ; par $req(p)$ (resp. $stim(p)$, $inh(p)$) l'ensemble des stimulateurs nécessaires (resp. stimulateurs, inhibiteurs) qui modulent p.
- Pour chaque opérateur logique $o \in \mathcal{O}$, nous dénotons par $in(o)$ l'ensemble des noeuds (EPNs ou opérateurs logiques) sources d'un arc logique entrant sur o.
- Un EPN de type puits ou de type source sera dénoté par le symbole $\emptyset$.

Nous dénotons par $\mathbb{S} = \{\mathfrak{S}_1, \ldots, \mathfrak{S}_r\}$ l'ensemble valide d'histoires considéré, et par $\cup\mathbb{S} = \bigcup_{\mathfrak{S}\in\mathbb{S}} \mathfrak{S}$ l'ensemble des EPNs qui sont dans une histoire. Pour un processus $p \in \mathcal{P}$, nous dénotons par $\mathbb{S}(p) = \{\mathfrak{S} \in \mathbb{S}, reac(p) \cap \mathfrak{S} \neq \emptyset \vee prod(p) \cap \mathfrak{S} \neq \emptyset\}$ l'ensemble des histoires impliquées dans p.

Nous encodons la carte considérée en un RA défini par le triplet (Σ, S, T), où Σ est l'ensemble des automates de ce RA, S l'ensemble de ses états globaux, et T l'ensemble de ses transitions locales, suivant l'encodage défini dans la section 6.5.

Nombre d'automates : D'après notre encodage :

- pour chaque EPN $e \in \mathcal{E}, e \notin \cup\mathbb{S}$, nous définissons un automate $\mathtt{e} \in \Sigma$ tel que $S(\mathtt{e}) = \{\mathtt{e_0}, \mathtt{e_1}\}$;
- pour chaque processus $p \in \mathcal{P}$, nous définissons un automate $\mathtt{p} \in \Sigma$ tel que $S(\mathtt{p}) = \{\mathtt{p_0}, \mathtt{p_1}\}$;
- pour chaque histoire $\mathfrak{S} \in \mathbb{S}$, nous définissons un automate $\mathtt{S} \in \Sigma$ tel que $S(\mathtt{S}) = \{\mathtt{S_e} \mid e \in \mathfrak{S}, e \neq \emptyset\} \cup \{\mathtt{S}_\emptyset\}$.

Par conséquent, le nombre d'automates est le suivant :

$$\begin{aligned} |\Sigma| &= (|\mathcal{E}| - |\cup\mathbb{S}|) + |\mathcal{P}| + |\mathbb{S}| \\ &\leq |\mathcal{E}| + |\mathcal{P}| \end{aligned}$$

Le nombre d'automates est donc borné par le nombre d'EPNs et le nombre de processus.

Nombres d'états globaux : Nous rappelons que pour un RA dont l'ensemble d'automates est Σ, nous dénotons, pour un automate $\mathtt{a} \in \Sigma$, son ensemble d'états locaux par $S(\mathtt{a}) \triangleq \{\mathtt{a_i}, \cdots, \mathtt{a_j}\}$, et que l'ensemble d'états globaux de ce RA est défini par :

$$S = \prod_{\mathtt{a} \in \Sigma} S(\mathtt{a})$$

D'après notre encodage, nous avons donc :

$$\begin{aligned} |S| &= \Big| \prod_{\mathtt{a} \in \Sigma} S(\mathtt{a}) \Big| \\ &= \Big| \prod_{e \in \mathcal{E}, e \notin \cup\mathbb{S}} S(\mathtt{e}) \times \prod_{p \in \mathcal{P}} S(\mathtt{p}) \times \prod_{\mathfrak{S} \in \mathbb{S}} S(\mathtt{S}) \Big| \end{aligned}$$

Or par définition, tous les états locaux des automates sont disjoints. Nous avons donc :

$$|S| = \prod_{e \in \mathcal{E}, e \notin \cup\mathbb{S}} |S(\mathtt{e})| \times \prod_{p \in \mathcal{P}} |S(\mathtt{p})| \times \prod_{\mathfrak{S} \in \mathbb{S}} |S(\mathtt{S})|$$

Comme, pour toute histoire $\mathfrak{S} \in \mathbb{S}$, $|S(\mathtt{S})| \leq |\mathfrak{S}| + 1$, nous avons :

$$\begin{aligned} |S| &\leq \prod_{e \in \mathcal{E}, e \notin \cup\mathbb{S}} |S(\mathtt{e})| \times \prod_{p \in \mathcal{P}} |S(\mathtt{p})| \times \prod_{\mathfrak{S} \in \mathbb{S}} (|\mathfrak{S}| + 1) \\ &\leq \prod_{e \in \mathcal{E}, e \notin \cup\mathbb{S}} 2 \times \prod_{p \in \mathcal{P}} 2 \times \prod_{\mathfrak{S} \in \mathbb{S}} (|\mathfrak{S}| + 1) \\ &\leq 2^{|\mathcal{E} \setminus \cup\mathbb{S}|} \times 2^{|\mathcal{P}|} \times \prod_{\mathfrak{S} \in \mathbb{S}} (|\mathfrak{S}| + 1) \end{aligned}$$

Or $\cup\mathbb{S} \subseteq \mathcal{E}$ et pour tout couple d'histoires $(\mathfrak{S}_i, \mathfrak{S}_j) \in \mathbb{S}^2$ tel que $i \neq j$, $\mathfrak{S}_i \cap \mathfrak{S}_j = \emptyset$, donc :

$$|S| \leq 2^{|\mathcal{E}|} \times 2^{-\sum\limits_{\mathfrak{S}\in\mathbb{S}}|\mathfrak{S}|} \times 2^{|\mathcal{P}|} \times \prod_{\mathfrak{S}\in\mathbb{S}}(|\mathfrak{S}|+1)$$
$$\leq \frac{2^{|\mathcal{E}|} \times 2^{|\mathcal{P}|} \times \prod\limits_{\mathfrak{S}\in\mathbb{S}}(|\mathfrak{S}|+1)}{\prod\limits_{\mathfrak{S}\in\mathbb{S}} 2^{|\mathfrak{S}|}}$$

Comme, pour toute histoire $\mathfrak{S} \in \mathbb{S}$, $|\mathfrak{S}| + 1 \leq 2^{|\mathfrak{S}|}$, nous avons finalement :

$$|S| \leq 2^{|\mathcal{E}|} \times 2^{|\mathcal{P}|}$$

Le nombre d'états globaux est donc exponentiel en fonction du nombre d'EPNs et de processus.

Nombre de transitions : D'après notre encodage (donné à la section 6.5), nous construisons, pour chaque processus p :

- $|\mathsf{ready}(p)|$ transitions pour le déclenchement de p (cf. a.) ;
- une transition pour l'arrêt de p (cf. b.) ;
- une transition pour chaque réactif de p n'appartenant pas à une histoire (cf. c.) ;
- une transition pour chaque produit de p n'appartenant pas à une histoire (cf. d.) ;
- une transition pour chaque histoire impliquée dans p (cf. e.).

Par conséquent, le nombre total de transitions est le suivant :

$$|T| = \sum_{p\in\mathcal{P}} \Big(|\mathsf{ready}(p)| + 1 + |reac(p) \setminus \cup\mathbb{S}| + |prod(p) \setminus \cup\mathbb{S}| + |\mathbb{S}(p)| \Big)$$

Or nous avons :

$$\begin{aligned} |\mathsf{ready}(p)| = \Big| & \{\{\mathsf{e}_1 \mid e \in reac(p), e \notin \cup\mathbb{S}\} \\ & \cup \{\mathsf{S}_\mathsf{e} \mid e \in reac(p), e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}\} \\ & \cup \{\mathsf{S}_\emptyset \mid reac(p) = \emptyset, prod(p) \cap \mathfrak{S} \neq \emptyset, \mathfrak{S} \in \mathbb{S}\} \\ & \cup \{\mathsf{q}_0 \mid p \# q\} \\ & \cup \ell_{cond} \mid \ell_{cond} \in \mathsf{cond}(p)\} \Big| \\ = & |\mathsf{cond}(p)| \end{aligned}$$

Par conséquent, le nombre total de transitions est :

$$|T| = \sum_{p\in\mathcal{P}} |\mathsf{cond}(p)| + 1 + |reac(p) \setminus \cup\mathbb{S}| + |prod(p) \setminus \cup\mathbb{S}| + |\mathbb{S}(p)|$$

Or nous avons :

$$|\mathsf{cond}(p)| = \Big| \bigcup_{c \in \mathsf{DNF}(\mathsf{mod}(p))} \tilde{\prod_{l \in c}} \mathsf{ls}(l) \Big|$$

Dans le pire des cas, pour deux clauses c et c' de $\mathsf{DNF}(\mathsf{mod}(p))$, nous avons $\tilde{\prod_{l \in c}} \mathsf{ls}(l) \cap \tilde{\prod_{l \in c'}} \mathsf{ls}(l) = \emptyset$, et pour deux littéraux l et l' d'une même clause c de $\mathsf{DNF}(\mathsf{mod}(p))$, $\mathsf{ls}(l) \cap \mathsf{ls}(l') = \emptyset$ donc :

$$|\mathsf{cond}(p)| \leq \sum_{c \in \mathsf{DNF}(\mathsf{mod}(p))} \tilde{\prod_{l \in c}} |\mathsf{ls}(l)|$$

Or, pour un littéral l d'une clause $c \in \mathsf{DNF}(\mathsf{mod}(p))$, nous avons :

$$|\mathsf{ls}(l)| = \begin{cases} |\{e \mid e \ \in \mathfrak{S}, e \neq \emptyset\}| \text{ si } l = \neg e, e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}; \\ 1 \text{ sinon;} \end{cases}$$

Nous pouvons alors donner un majorant à $\tilde{\prod_{l \in c}} |\mathsf{ls}(l)|$, et donc à $|\mathsf{cond}(p)|$.

Étant donné une clause $c \in \mathsf{DNF}(\mathsf{mod}(p))$, nous dénotons par $negE(c) = \{e \mid l = \neg e, l \in c, e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}\}$ l'ensemble des EPNs appartenant à une histoire et apparaissant dans un littéral négatif de c, et par $negS(c) = \{\mathfrak{S} \mid \mathfrak{S} \in \mathbb{S}, e \in \mathfrak{S}, e \in negE(c)\}$ l'ensemble des histoires auxquelles ces EPNs appartiennent. Notons que pour une clause c, $negE(c)$ et $negS(c)$ peuvent être vides.

Nous avons alors la relation suivante, pour une clause $c \in \mathsf{DNF}(\mathsf{mod}(p))$:

$$\tilde{\prod_{l \in c}} |\mathsf{ls}(l)| \leq max\big(\{1\} \cup \{|\mathfrak{S}| \mid \mathfrak{S} \in negS(c)\}\big)^{|negE(c)|}$$

Par conséquent :

$$|\mathsf{cond}(p)| \leq \sum_{c \in \mathsf{DNF}(\mathsf{mod}(p))} max\big(\{1\} \cup \{|\mathfrak{S}| \mid \mathfrak{S} \in negS(c)\}\big)^{|negE(c)|}$$

et

$$|T| \leq \sum_{p \in \mathcal{P}} \Big(\big(\sum_{c \in \mathsf{DNF}(\mathsf{mod}(p))} max\big(\{1\} \cup \{|\mathfrak{S}| \mid \mathfrak{S} \in negS(c)\}\big)^{|negE(c)|} \big) + 1 + |reac(p) \setminus \cup\mathbb{S}| + |prod(p) \setminus \cup\mathbb{S}| + |\mathbb{S}(p)| \Big)$$

En conclusion, le nombre de transitions est linéaire en fonction du nombre de processus, et pour chaque processus, linéaire en fonction du nombre de clauses apparaissant dans la formule associée à la modulation globale de ce processus, et exponentiel en fonction du nombre de littéraux négatifs apparaissant dans les clauses de cette même formule et représentant des EPNs appartenant à une histoire. Le terme exponentiel est dû à la manière dont nous avons défini l'absence d'un EPN d'une histoire : un EPN d'une histoire est absent *ssi* un autre EPN de cette histoire est présent ou cette histoire est dans son état vide.

D.2 Preuves

Soit une carte SBGN-PD. Nous utilisons les mêmes notations qu'à la section précédente pour désigner les éléments de cette carte. $\mathcal{E}$ désigne l'ensemble des EPNs de cette carte, $\mathcal{P}$ l'ensemble de ses processus, et $\mathcal{O}$ l'ensemble de ses opérateurs logiques.

Soit $RA_{gen} = (\Sigma_{gen}, S_{gen}, T_{gen})$ le réseau d'automates modélisant la dynamique de cette carte avec la sémantique générale. Finalement, soit $\mathbb{S}$ un ensemble valide d'histoires quelconque de cette carte, et $RA_{st} = (\Sigma_{st}, S_{st}, T_{st})$ le réseau d'automates modélisant cette même carte avec la sémantique des histoires en considéreant l'ensemble valide d'histoires $\mathbb{S}$.

Nous rappelons que par commodité, nous voyons un état global d'un RA comme un ensemble d'états locaux plutôt que comme un tuple.

Dans ce qui suit, comme pour les deux réseaux d'automates, nous distinguerons les fonctions utilisées pour encoder RA_{st} de celles utilisées pour encoder RA_{gen}, en ajoutant le suffixe *st* aux premières, et le suffixe *gen* aux secondes. Nous étendons la définition de $[\![]\!]$ aux états globaux partiels et aux ensembles d'états globaux (partiels).

D.2.1 Sketch de preuve de la proposition 6.1

Proposition. Soient $s, s' \in S_{st}$ deux états globaux de RA_{st} dans lesquels aucun processus n'a lieu. Si s' est atteignable à partir de s dans RA_{st}, alors $[\![s']\!]$ est atteignable à partir de $[\![s]\!]$ dans RA_{gen}.

Afin de montrer cette propriété, nous avons d'abord besoin du lemme suivant : si un processus $p \in \mathcal{P}$ peut se déclencher dans un état global $s \in S_{st}$ de RA_{st}, alors il peut se déclencher dans l'état $[\![s]\!] \in S_{gen}$ de RA_{gen}. Formellement :

Lemme D.1. Soient $s \in S_{st}$ un état global de RA_{st} et $p \in \mathcal{P}$ un processus. S'il existe un ensemble d'états locaux l_{st} tel que $\mathtt{p_0} \xrightarrow{l_{st}} \mathtt{p_1} \in T_{st}$ et $l_{st} \subseteq s$, alors il existe un ensemble d'états locaux l_{gen} tel que $\mathtt{p_0} \xrightarrow{l_{gen}} \mathtt{p_1} \in T_{gen}$ et $l_{gen} \subseteq [\![s]\!]$.

Preuve. S'il existe l_{st} tel que $\mathtt{p_0} \xrightarrow{l_{st}} \mathtt{p_1}$, alors, vu notre encodage, $l_{st} \in \mathsf{ready_{st}}(p)$. Par conséquent, étant donnée la définition de $\mathsf{ready_{st}}$, il existe deux ensembles d'états locaux k_{st} et m_{st} tels que $l_{st} = k_{st} \cup m_{st}$, $m_{st} \in \mathsf{cond_{st}}(p)$ et :

$$\begin{aligned} k_{st} = &\{\mathtt{e_1} \mid e \in reac(p), e \notin \cup\mathbb{S}\} \\ &\cup \{\mathtt{S_e} \mid e \in reac(p), e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}\} \\ &\cup \{\mathtt{S_\emptyset} \mid reac(p) = \emptyset, prod(p) \cap \mathfrak{S} \neq \emptyset, \mathfrak{S} \in \mathbb{S}\} \\ &\cup \{\mathtt{q_0} \mid p \# q\} \end{aligned}$$

Soit k_{gen} un ensemble d'états locaux, défini par :

$$k_{gen} = \{\mathtt{e_1} \mid e \in reac(p), e \in \mathcal{E}\}$$

Étant donnée la définition de $[\![]\!]$, nous avons :

$$k_{gen} = [\![k_{st}]\!] \setminus \{\mathsf{q_0} \mid p\#q\}$$

Comme $k_{gen} \subseteq [\![k_{st}]\!]$ et $k_{st} \subseteq s$, nous déduisons que $k_{gen} \subseteq [\![s]\!]$.

Par définition de $\mathsf{cond_{st}}(p)$, il existe une clause c^* de $\mathsf{mod_{st}}(p)$ telle que :

$$m_{st} \in \tilde{\prod_{l \in c^*}} \mathsf{ls_{st}}(l)$$

Soit m_{gen} un ensemble d'états locaux défini par :

$$m_{gen} = \{\mathsf{e_0} \mid l = \neg e, e \in \mathcal{E}, l \in c^*\} \cup \{\mathsf{e_1} \mid l = e, e \in \mathcal{E}, l \in c^*\}$$

Comme $\mathsf{cond_{gen}}(p)$ est défini par

$$\begin{aligned}\mathsf{cond_{gen}}(p) &= \bigcup_{c \in \mathsf{DNF}(\mathsf{mod}(p))} \tilde{\prod}_{l \in c} \mathsf{ls_{gen}}(l) \\ &= \Big\{\{\mathsf{e_0} \mid l = \neg e, e \in \mathcal{E}, l \in c\} \cup \{\mathsf{e_1} \mid l = e, e \in \mathcal{E}, l \in c\} \mid c \in \mathsf{DNF}(\mathsf{mod}(p))\Big\},\end{aligned}$$

nous avons $m_{gen} \in \mathsf{cond_{gen}}(p)$.
Nous montrons maintenant que $m_{gen} \subseteq [\![s]\!]$. Soit $\mathsf{e_x} \in m_{gen}$ un élément de m_{gen}.

- Si $\mathtt{x} = \mathtt{1}$, alors il existe un littéral l tel que $l = e$ et $l \in c^*$.
  - Si $e \notin \cup\mathbb{S}$, alors $\mathsf{e_1} \in m_{st}$, et par conséquent $\mathsf{e_1} \in [\![s]\!]$.
  - Si $e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}$, alors $\mathsf{S_e} \in m_{st}$. Comme $m_{st} \in s$, d'après la définition de $[\![]\!]$, $\mathsf{e_x} \in [\![s]\!]$.
- Si $\mathtt{x} = \mathtt{0}$, alors il existe un littéral l tel que $l = \neg e$ et $l \in c^*$.
  - Si $e \notin \cup\mathbb{S}$, alors $\mathsf{e_0} \in m_{st}$, et par conséquent $\mathsf{e_0} \in [\![s]\!]$.
  - Si $e \in \mathfrak{S}, \mathfrak{S} \in \mathbb{S}$ alors d'après la définition de $\mathsf{ls_{st}}$, il existe forcément un EPN $f \in \mathcal{E}$ tel que $f \neq e$, $f \in \mathfrak{S}$ et $\mathsf{S_f} \in m_{st}$. Or $m_{st} \subseteq s$, donc $\mathsf{S_f} \in s$. Comme, par définition, un état global contient exactement un état local de chaque automate, $\mathsf{S_e} \notin s$. Par définition de $[\![]\!]$, nous avons donc $\mathsf{e_0} \in [\![s]\!]$.

Par conséquent, $\mathsf{e_x} \in [\![s]\!]$, et $m_{gen} \subseteq [\![s]\!]$.

Soit l_{gen} un ensemble d'état locaux tel que $l_{gen} = k_{gen} \cup m_{gen}$. Comme $k_{gen} \subseteq [\![s]\!]$ et $m_{gen} \subseteq [\![s]\!]$, nous avons $l_{gen} \subseteq [\![s]\!]$. De plus, par définition de $\mathsf{ready_{gen}}$, $l_{gen} \in \mathsf{ready_{gen}}(p)$, et suivant notre encodage, $\mathsf{p_0} \xrightarrow{l_{gen}} \mathsf{p_1} \in T_{gen}$. □

Par le lemme D.1, tout processus qui peut être déclenché à partir de s peut être déclenché à partir de $[\![s]\!]$. Nous remarquons d'abord que pour chaque trace de la dynamique de RA_{st} partant d'un état s et arrivant sur un état s', telle qu'aucun processus n'a lieu ni dans s ni dans s', si un processus p a été déclenché dans cette trace, alors toutes les transitions locales modélisant la production de produits de p (appartenant à une histoire ou non) correspondent forcément à une transition globale de cette trace. En effet, un processus ne peut s'arrêter (i.e. cesser d'avoir lieu) que si toutes les transitions locales modélisant la production de ses produits ont bien été déclenchées. De plus, ces transitions locales, ainsi que les transitions locales modélisant la consommation des réactifs du processus p, ne

sont conditionnées que par l'état du processus p pour les premières, et l'état du processus p et des produits pour les deuxièmes. Par conséquent, une fois qu'un processus s'est déclenché, son déroulement est indépendant de l'état des autres éléments de la carte, et il est inexorable. Ainsi, pour toute trace t_{st} comme mentionnée plus haut et tout processus p qui a été déclenché dans cette trace, il existe une trace t'_{st} partant de s et arrivant sur s' telle que toutes les transitions locales modélisant la production des produits de p, et les transitions de t_{st} modélisant la consommation des réactifs de p sont directement consécutives au déclenchement de p dans t'_{st}. Ce résultat peut être obtenu en réordonnant la trace t_{st} afin d'obtenir la trace t'_{st}.

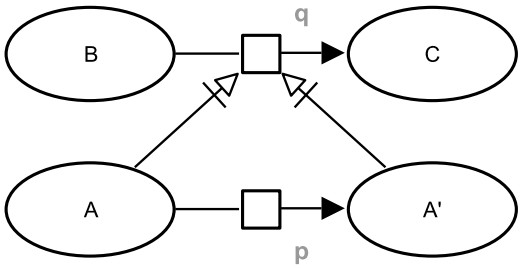
À partir de la trace t'_{st}, nous pouvons construire une séquence de transitions locales de RA_{gen} telle que la trace correspondante parte de $[\![s]\!]$ et arrive sur $[\![s']\!]$, en déplaçant notamment les transitions locales modélisant la consommation d'un EPN qui appartient à une histoire juste avant la transition locale modélisant l'arrêt du processus concerné. Cette séquence est construite à partir de t'_{st} de la manière suivante :

- en remplaçant dans t_{st} chaque transition locale qui ne concerne pas une histoire par la transition locale de RA_{gen} correspondante ;
- en remplaçant dans t_{st} chaque transition locale de la forme $\mathtt{S_e} \xrightarrow{\{\mathtt{p_1}\}} \mathtt{S_f}$ par la transition $\mathtt{f_0} \xrightarrow{\{\mathtt{p_1}\}} \mathtt{f_1}$;
- en remplaçant dans t_{st} chaque transition locale de la forme $\mathtt{S_\emptyset} \xrightarrow{\{\mathtt{p_1}\}} \mathtt{S_f}$ par la transition $\mathtt{f_0} \xrightarrow{\{\mathtt{p_1}\}} \mathtt{f_1}$;
- en ajoutant dans t_{st}, juste avant une transition locale de la forme $\mathtt{p_1} \xrightarrow{\mathsf{done_{st}}(p)} \mathtt{p_0}$, une transition locale de la forme $\mathtt{e_1} \xrightarrow{\{\mathtt{p_1}\}\cup\mathsf{done_{gen}}(p)} \mathtt{e_0}$ pour chaque e tel que $e \in reac(p)$ et $e \in \mathbb{S}$.

D.2.2 Preuve de la proposition 6.2

Proposition. Soient s et s' deux états globaux de RA_{st} dans lesquels aucun processus n'a lieu. Si $[\![s']\!]$ est atteignable à partir de $[\![s]\!]$ dans RA_{gen}, alors s' n'est pas nécessairement atteignable à partir de s dans RA_{st}.

Nous montrons cette proposition à l'aide d'un contre-exemple.



Considérons une unique histoire $\mathfrak{S} = \{A, A'\}$.

Soient $s = \{\mathtt{S_A}, \mathtt{B_1}, \mathtt{C_0}, \mathtt{p_0}, \mathtt{q_0}\}$ et $s' = \{\mathtt{S_{A'}}, \mathtt{B_1}, \mathtt{C_1}, \mathtt{p_1}, \mathtt{q_1}\}$ deux états globaux de RA_{st}. Alors s' n'est pas atteignable à partir de s dans la dynamique de RA_{st}.En effet, la production de c nécessite le déclenchement du processus q. Or q, pour se déclencher, nécessite à la fois la présence de A et la présence de A', ce qui n'est pas possible vu que ces deux EPNs appartiennent à la même histoire. Par contre, l'état $[\![s']\!] = \{\mathtt{A_0}, \mathtt{A'_1}, \mathtt{B_1}, \mathtt{C_1}, \mathtt{p_1}, \mathtt{q_1}\}$ est atteignable à partir de l'état $[\![s]\!] = \{\mathtt{A_1}, \mathtt{A'_0}, \mathtt{B_1}, \mathtt{C_0}, \mathtt{p_0}, \mathtt{q_0}\}$ dans la dynamique de RA_{gen}, étant donné que A et A' peuvent être présents ensemble dans cette sémantique. La trace suivante permet d'atteindre $[\![s']\!]$ à partir de $[\![s]\!]$:

$\{\mathtt{A}_1, \mathtt{A'}_0, \mathtt{B}_1, \mathtt{C}_0, \mathtt{p}_0, \mathtt{q}_0\} \to \{\mathtt{A}_1, \mathtt{A'}_0, \mathtt{B}_1, \mathtt{C}_0, \mathtt{p}_1, \mathtt{q}_0\} \to \{\mathtt{A}_1, \mathtt{A'}_1, \mathtt{B}_1, \mathtt{C}_0, \mathtt{p}_1, \mathtt{q}_0\} \to \{\mathtt{A}_1, \mathtt{A'}_1, \mathtt{B}_1, \mathtt{C}_0, \mathtt{p}_1, \mathtt{q}_1\}$
$\to \{\mathtt{A}_0, \mathtt{A'}_1, \mathtt{B}_1, \mathtt{C}_0, \mathtt{p}_1, \mathtt{q}_1\} \to \{\mathtt{A}_0, \mathtt{A'}_1, \mathtt{B}_1, \mathtt{C}_1, \mathtt{p}_1, \mathtt{q}_1\}$

D.2.3 Sketch de peuve de la propriété 6.2

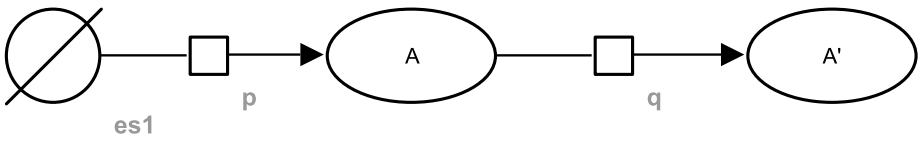
Propriété. Soit s un état global de RA_{st} tel que $[\![s]\!]$ soit un point attracteur de RA_{gen}. Alors s est un point attracteur de RA_{st}.

Sketch de preuve. Nous remarquons d'abord que si $[\![s]\!]$ est un point attracteur de RA_{gen}, alors aucun processus n'a lieu dans $[\![s]\!]$. En effet, la transition locale modélisant la production d'un produit d'un processus p est conditionnée à l'état local $\mathtt{p}_1$ de p, et l'arrêt de p est conditionné à la présence de tous ses produits. Par conséquent, un processus p finira toujours par s'arrêter. Finalement, par le lemme D.1, si un processus peut être activé dans s, alors il peut être activé dans $[\![s]\!]$. Comme $[\![s]\!]$ est un point attracteur, aucun processus ne peut se déclencher dans $[\![s]\!]$, et donc dans s.

D.2.4 Preuve de la propriété 6.3

Propriété. Soit s un point attracteur de RA_{st}. Alors $[\![s]\!]$ n'est pas forcément un point attracteur de RA_{gen}.

Nous montrons cette propriété à l'aide d'un contre-exemple. Considérons la carte SBGN-PD suivante :



Considérons une unique histoire $\mathfrak{S} = \{es1, A, A'\}$.

L'état $s = \{\mathtt{S}_{\mathtt{A'}}, \mathtt{p}_0, \mathtt{q}_0\}$ est un point attracteur de RA_{st}. Par contre, l'état $[\![s]\!] = \{\mathtt{A}_0, \mathtt{A'}_1, \mathtt{p}_0, \mathtt{q}_0\}$ n'est pas un point attracteur de RA_{gen}, étant donné que la transition locale $\mathtt{p}_0 \xrightarrow{\emptyset} \mathtt{p}_1$ peut être déclenchée.