Analyse symbolique et inférence de modèles métaboliques

Razanne Issa

université de BORDEAUX

THÈSE PRÉSENTÉE
POUR OBTENIR LE GRADE DE

DOCTEUR DE
L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET
D'INFORMATIQUE
SPÉCIALITÉ INFORMATIQUE

par Razanne ISSA

ANALYSE SYMBOLIQUE ET INFÉRENCE DE MODÈLES MÉTABOLIQUES

Sous la direction de : David SHERMAN
(Co-encadrant : Pascal Durrens)

Soutenue le 10/07/2015

Membres du jury :

Alain DENISE	Professeur (U. Paris-Sud)	Rapporteur
Christian RETORÉ	Professeur (U. Montpellier)	Rapporteur
Emmanuel TALLA	Maître de conf. (U. Aix-Marseille)	Examinateur
Guillaume BLIN	Professeur (U. Bordeaux)	Examinateur
Pascal DURRENS	Chargé de recherche (CNRS)	Co-encadrant
David SHERMAN	Directeur de recherche (Inria)	Directeur de thèse

Titre Analyse symbolique et inférence de modèles métaboliques

Résumé L'objectif de cette thèse est de proposer une nouvelle méthode de construction de modèles métaboliques dans le contexte de la génomique comparée. Nous avons développé un outil, ab-pantograph, permettant l'inférence de modèles métabolique se basant sur la logique abductive. Pour ce faire, nous avons introduit une représentation logique de modèles métaboliques minimaux enzymatiques, puis à partir d'un modèle métabolique dit de référence, nous avons dérivé un modèle minimal enzymatique explicite accompagné d'association de gènes. Enfin, en couplant ce modèle métabolique au génome d'un organisme cible, nous inférons par abduction un modèle enzymatique pour cet organisme cible accompagné d'un ensemble d'associations de gènes, modèle que l'on veut congruent à celui que l'on aurait pu obtenir en ayant toutes les informations pour l'organisme cible. L'outil proposé, ab-pantograph, a été développé en utilisant la programmation logique par contraintes et Hyprolog.

Mots-clefs Réseaux métaboliques, modèles métaboliques, construction de modèles logique abductive, inférence, associations de gènes, Hyprolog

Title symbolic analysis and inference of metabolic models

Abstract The objective of this thesis is to propose a new method of constructing metabolic models in the context of comparative genomics. We have developed a tool, ab-pantograph, allowing the inference of metabolic models based on the Abductive logic. To do this, we have introduced a logical representation of minimal enzymatic metabolic models and from a metabolic model called reference, we derived an explicit enzymatic minimal model accompanied by gene association. Finally, by coupling this metabolic modele with the genome of a target organism, we infer abductively a model enzyme for this target organism accompanied by a set of gene associations, pattern one wants congruent to that which is could have obtained by having all the information to the target organism. The proposed tool, ab-pantograph, has been developed using constraint logic programming and Hyprolog.

Keywords Metabolic networks, metabolic models, models construction, gene-protein-reaction(GPR) associations, Hyprolog

Laboratoire Bordelais de Recherche en Informatique - UMR 5800

Remerciements

Je souhaite adresser mes remerciements à mon directeur de thèse Monsieur David Sherman, qui m'a soutenu durant ces années et qui a su me guider et m'orienter dans mes recherches. Je remercie également Monsieur Pascal Durrens, co-encadrant de thèse pour tout l'intérêt qu'il a porté au suivi de mes travaux.

J'exprime aussi toute ma gratitude à Messieurs Alain Denise et Christian Retoré, pour avoir accepté d'être les rapporteurs de ma thèse, et pour toutes les remarques riches en conseils constructifs qui m'ont permis de réaliser les amendements au manuscrit original. Un grand merci aussi à Monsieur Emmanuel Talla d'avoir pris le temps d'examiner et de s'intéresser à mon travail et à Monsieur Guillaume Blin, tant pour avoir été présent au sein du jury en tant qu'examinateur, que pour tout le temps qu'il a pu passer pour me soutenir dans la rédaction du manuscrit.

Je tiens particulièrement à remercier toutes les personnes avec qui j'ai pu tisser des liens d'amitié au cours de ces années de recherches : des amis comme Allyx Fontaine, Jérôme Charton, Ludovic Paulhac, Laetitia Bourgeade, Mahmoud Gargouri et Eve Garnaud pour leur soutien moral et pour tous les moments de joie, de déceptions ou de doutes que nous avons partagé ensemble. Merci aussi à Florent Berthaut, Marie Beurton-Aimar, Simon Naulin, Cédric Teyton pour leur amitié fidèle.

Je n'oublie pas non plus Pascal Desbarats et Aymeric Vincent qui ont toujours su se rendre disponibles pour m'épauler dans la réalisation du manuscrit et pour m'apporter leurs indispensables encouragements.
Mes pensées amicales et nostalgiques vont aussi vers toutes les personnes avec qui j'ai pu travailler en équipe, en particulier David Auber et Anne-laure Gautier, mais aussi à tous les personnels permanents, non-permanents ou administratifs que j'ai rencontré au sein du LaBRI et d'Inria, sans qui ces dernières années auraient sans nul doute eu moins de saveur.

Un grand merci à tout mes amis du cercle extra-professionnel qui me sont chers et que j'ai pu connaître au cours de ces dernières années passées en France, je pense à Merian Nassra, Buchra Hamelin et Marc Delavaud.

Enfin, mes pensées les plus affectives vont à ma famille résidant en Syrie, dont le soutien indéfectible m'aura permis de surmonter les moments les plus difficiles.

Je conclus ces remerciements par le plus important, même si parfois le vocabulaire me manque pour exprimer toute l'intensité des émotions dissimulées derrière ces mots, je souhaite exprimer toute ma gratitude envers mes parents pour tous les sacrifices qu'ils ont concédé à faire pour la réussite de leurs enfants, et ce malgré un contexte souvent difficile. La distance nous sépare, mais mon coeur est avec vous.

Cette thèse vous est dédiée.

Table des matières

Introduction

Cette thèse concerne la construction de modèles métaboliques dans le contexte de la génomique comparée.

Le métabolisme se réfère à l'ensemble des processus biochimiques qui se produisent au sein des organismes, communément appelées *réactions*. Ce sont ces dernières qui permettent aux organismes (i) de se développer et de se reproduire, (ii) de maintenir leurs structures, et (ii) de s'adapter à leurs milieux de vie. Une réactions transforme des molécules, spontanément ou sous l'influence de molécules catalyseurs appelées enzymes.

Un *réseau métabolique* décrit un ensemble de réactions et les connections entre elles induites par les molécules qu'elles aient en commun, en montrant les enchainements entre réactions, le réseau métabolique aide les biologistes à comprendre les flux de molécules produits et consommés par les réactions. Cette compréhension sert d'appui pour l'identification de voies métaboliques dont chacun part d'un ensemble de molécules "entrées" et produit à la fin un ensemble de molécules "sorties". Les réseaux métaboliques sont utilisés pour optimiser les flux pour les applications biotechnologiques, pour identifier les cibles médicamenteuses, *etc*.

Un *modèle métabolique* est une représentation mathématique du réseau métabolique qui permet la simulation numérique, l'étude des flux ou de la topologie, typiquement sous forme d'équations différentielles ou de graphes bipartis. Ce sont les modèles métaboliques à l'échelle du génome qui permettent de faire le lien entre les informations biochimiques issues du génome et les phénotypes métaboliques. Ce lien peut être dérivé en s'appuyant sur un cadre d'interprétations solides de données expérimentales relatives à des états métaboliques et permettant des expériences *in silico* avec le métabolisme complet de la cellule. Cela autorise la simulation de comportements phénotypiques d'un micro-organisme dans différentes conditions environnementales et génétiques, ce qui constitue un outil important dans l'ingénierie métabolique.

L'étude des systèmes biologiques à l'échelle du génome peut nous aider à comprendre les processus biologiques fondamentaux qui régissent l'activité des organismes vivants et régulent leurs interactions avec l'environnement.

Compte tenu de l'importance des modèles métaboliques dans les applications biomédicales de recherche et de biotechnologie, un grand nombre de travaux et d'éventuelles plates-formes de calcul associées ont été proposés pour attaquer le problème de la reconstruction de modèles métaboliques. Jusqu'à il y a peu de temps encore, le processus de reconstruction des modèles métaboliques était difficile et onéreux. En effet les experts

devaient travailler sur de longues périodes, couvrant souvent plusieurs années, avec l'objectif final de mettre en relation, en les disposant dans des réseaux, toutes les pièces d'un puzzle complexe constitué de centaines de réactions biochimiques. Ce travail colossale n'aboutissant généralement qu'à la résolution d'une petite partie de ce qu'une cellule est capable de faire dans son ensemble.

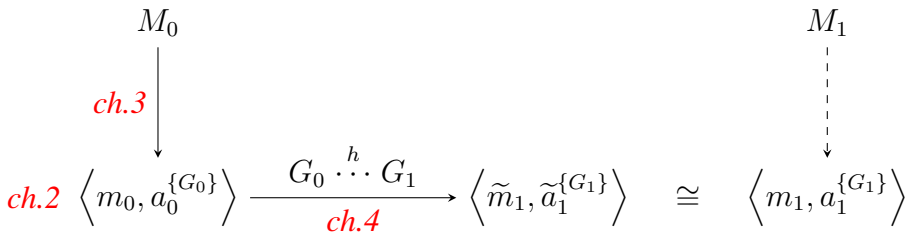
L'émergence des techniques de séquençage haut débit et des masses de données associées n'a pas simplifié le problème. En effet, d'un côté, ces données massives peuvent permettre d'imaginer reconstituer les réseaux métaboliques à l'échelle du génome entier pour un nombre croissant d'organismes. De l'autre, de par la complexité, la variété et le volume des données à traiter, l'automatisation du processus de la reconstruction de modèles métaboliques est devenue indispensable pour l'étude de la biologie des systèmes du métabolisme.

Cette automatisation de la reconstruction de modèles métaboliques est très souvent basée sur l'utilisation de pipelines structurés faisant usage de plusieurs outils de bioinformatique. Ces outils concernent typiquement (*i*) l'annotation et la ré-annotation de génome, (*ii*) la recherche d'homologie, (*iii*) l'intégration et l'interrogation de données du domaine disponibles dans une multitude de base de données accessibles via internet, (iv) la localisation spatiale des protéines, et bien autres. Malgré une quantité d'outils disponibles toujours plus importante, certaines des étapes de reconstruction de modèles métaboliques sont toujours faites par des processus semi-automatisés nécessitant des corrections manuelles effectuées par des experts.

Analyse et inférence de modèles

Les travaux effectués dans le cadre de cette thèse portent sur le développement d'un outil ab-Pantograph permettant l'inférence de modèles métaboliques. Plus précisément, comme nous le verrons par la suite, il s'agit ici d'inférer des modèles métaboliques à l'aide de la logique abductive. Le principe général de l'outil est le suivant :
(i) Pour soutenir l'inférence que nous développons dans ab-Pantograph, nous introduisons une représentation logique de *modèles métaboliques* minimaux enzymatiques (*ch*.2).
(ii) A partir d'un modèle métabolique complet d'un organisme dit de référence, il s'agit de dériver un modèle minimal enzymatique explicite accompagné *d'associations de gènes*. Ces associations de gènes, comme nous le détaillerons par la suite, représentent des connaissances implicites sur le modèle métabolique complet et permettront de prendre de décisions pendant l'inférence des réactions du modèle métabolique de l'organisme cible. On peut décrire le modèle comme un ensemble de réactions, un ensemble d'annotations de gènes ainsi qu'un ensemble d'associations de gènes par réaction. Ces données représente le modèle métabolique de notre organisme de référence (*ch*.3).
(iii) Couplant ce modèle métabolique au génome de l'organisme cible et à une fonction d'homologie (au niveau des gènes), nous *inférons par abduction* un modèle enzymatique de l'organisme cible accompagné d'un ensemble d'associations de gènes. L'objectif est que ce dernier modèle soit congruent à celui que l'on aurait pu obtenir, à l'instar de ce que l'on a effectué pour l'organisme de référence, à partir du modèle complet de l'organisme

cible (que l'on cherche à produire et dont on ne dispose donc pas) (*ch.4*).
Cela peut se résumer par le diagramme suivant :



où M est un modèle complet, et un uplet $\langle m, a^{\{G\}} \rangle$ est sa projection sur un modèle métabolique m annoté par des formules booléennes a définis sur un ensemble de variables G. Le diagramme montre notre objectif, que le modèle $\langle \widetilde{m}_1, \widetilde{a}_1^{\{G_1\}} \rangle$ que nous inférons est congruent au modèle théorique $\langle m_1, a_1^{\{G_1\}} \rangle$ que nous aurions obtenu si nous avions disposé d'un modèle complet M_1.

Structure de la thèse

Le manuscrit est organisé de la manière suivante comme indiqué sur le diagramme précédent. Le premier chapitre nous donne un aperçu des réseaux métaboliques. Nous y présenterons un état de l'art de la problématique de modélisation des réseaux métaboliques et de leurs reconstructions. Dans le deuxième chapitre, nous présenterons une représentation logique des modèles métaboliques offrant une description formelle et une sémantique de ces derniers. Cette représentation a l'avantage de proposer une définition et une manipulation faciles des opérations d'inférences, et sera utilisée comme une base pour notre problématique d'inférence de nouveaux modèles. Le troisième chapitre présente un critère formel permettant de mieux formaliser les formules d'association de gènes, de sorte que la connaissance correspondante peut être plus facilement extraite par d'autres processus d'inférence. Enfin, dans le quatrième chapitre, nous nous attacherons à présenter une méthode permettant l'inférence de modèles métaboliques en se basant sur la logique et le raisonnement abductifs. Nous utiliserons, pour ce faire, la programmation logique par contraintes et Hyprolog. Nous terminons par une synthèse des résultats obtenus et une présentation des orientations pour les futurs travaux.

Chapitre 1

Préliminaires

Dans ce chapitre, nous allons commencer par décrire des principes des biologie qui nous permettront ensuite de présenter un aperçu des réseaux métaboliques. Puis nous introduirons la programmation logique et la logique abductive. Enfin, nous présenterons les méthodes classiques de modélisation et de construction de réseaux métaboliques.

1.1 Réseaux métaboliques

Avec la quantité de connaissances existantes aujourd'hui, l'augmentation de la dimension et de la complexité des systèmes biologiques, la modélisation s'avère être une approche efficace pour comprendre la complexité de ces systèmes. Modéliser un système biologique revient à parler de "**réseau biologique**" . Cette notion abstraite est utilisée pour modéliser des connaissances et représenter des processus biologiques.

L'étude des réseaux biologiques, leur modélisation, analyse et leur visualisation sont des tâches importantes en sciences de la vie aujourd'hui. La compréhension de ces réseaux est essentielle pour donner un sens biologique à beaucoup de données complexes qui sont actuellement produites.

Cette importance croissante des réseaux biologiques est manifeste par le nombre croissant de travaux de recherche et de publications traitant cette problématique. La plupart des réseaux biologiques est loin d'être complète et généralement difficile à interpréter en raison de la complexité des relations et des spécifités des données.

La visualisation de réseau est une méthode fondamentale qui aide les scientifiques à comprendre les réseaux biologiques et à découvrir les propriétés importantes des processus biochimiques sous-jacents. Il existe une grande variété de méthodes de visualisation, l'objectif principal de la visualisation est de faciliter la compréhension des éléments modélisés par le réseau et par conséquent de choisir la visualisation la plus adaptée.

Plusieurs réseaux métaboliques très importants sont liés à des molécules telles que l'ADN, l'ARN, les protéines et les métabolites et leurs interactions : e.g., les réseaux de régulation, les réseaux d'interaction protéine-protéine et des réseaux métaboliques. Les réseaux de régulation des gènes et de transduction du signal décrivent comment les gènes peuvent être activés ou réprimés et donc quelles protéines sont produites dans la cellule

à un moment donné. Une telle régulation peut être causée par des protéines dites régulatrices ou des signaux externes. Les réseaux d'interaction protéine-protéine représentent les interactions entre les protéines telles que la construction de complexes protéiques et l'activation d'une protéine par une autre protéine. Les réseaux métaboliques montrent comment les métabolites sont transformés, par exemple pour produire de l'énergie ou pour la synthèse de substances spécifiques. Dans le cadre de cette thèse nous nous interesserons uniquement aux réseaux métaboliques.

Les réactions métaboliques sont fondamentales pour les processus de la vie, ces réactions répondent au besoin de molécules et d'énergie pour que les processus cellulaires puissent avoir lieu. Un grand nombre de réactions se produisent à tout moment dans les cellules vivantes, et le produit d'une réaction est habituellement utilisé par une autre réaction, donc les réactions métaboliques sont fortement interconnectées et regroupées en voies et réseaux métaboliques.

On appelle *réaction métabolique* la transformation, généralement catalysée par des *enzymes*, de substances chimiques (métabolites) appelés *substrats* ou *réactants* en d'autres métabolites appelées *produits* (c.f Figure 1.1).

$$\text{substrats} \xrightarrow{\text{enzymes}} \text{produits}$$

FIGURE 1.1 – Illustration d'une réaction métabolique.

Les enzymes jouent un rôle de catalyseur biologique, c'est-à-dire un composé qui facilite une réaction sans en modifier les produits et sans être consommé. Elles sont donc capables d'accélérer jusqu'à des millions de fois les réactions chimiques du métabolisme, sans pour autant modifier l'équilibre formé. Les enzymes sont généralement constituées de molécules protéiques issues de la traduction du génome à l'aide de facteurs de transcription (c.f. Figure 1.2(d)).

En pratique, une réaction peut être catalysée par plusieurs enzymes (isozymes). Les isozymes diffèrent généralement par leurs propriétés cinétiques et la façon dont elles sont régulées. Notons que certaines enzymes sont très spécifiques et ne catalysent qu'une seule réaction, alors que d'autres sont peu spécifiques (également appelées *"à large spectre"*) et peuvent catalyser toute une classe de réactions. On différencie les réactions nécessitant une enzyme, appelées réactions catalysées, de celles n'en néccessitant pas, appelées réactions spontanées.

Le concept de réversibilité, qui est une propriété de la plupart des réactions, a été introduit par *Claud Louis Berthollet* en 1803 : la réaction peut se produire dans les deux sens (c.f. Figure 1.2(b)). Etant donnée une réaction réversible R entre A et B (tous deux à la fois produit et substrat), le sens d'application de R sera de A vers B si la concentration en A est plus grande que celle en B ; de B vers A sinon. Dans l'état stable, la réaction réversible transforme, à la même vitesse, A en B et B en A ; conduisant ainsi à un équilibre.

Malgré tout, certaines réactions sont irréversibles (c.f. Figure 1.2(a)). Dans ce cas, les substrats sont convertis en produits mais ces derniers ne peuvent pas être reconvertis

en substrats. Un exemple d'une réaction irréversible est la combustion. Dans ces réactions, les réactifs sont complètement consommés, les produits sont formés et la réaction s'achève. Notons que ces réactions sont très rares en biologie.

Nous appellons *compartiment cellulaire* une partie fermée dans le cellule qui est entourée par une membrane. La subdivision des cellules dans des compartiments ou des parties isolés permet à la cellule de créer des environnements spécialisés pour des fonctions spécifiques. Ces compartiments peuvent être des organites (ou *"organelles"*), c'est-à-dire des structures spécialisées qui se chargent d'un ensemble de tâches à l'intérieur de la cellule, ou des régions locales de la cellule définie par la concentration des molécules ou des caractéristiques physiques distinctes et de proportions. La présence de compartiments spécialisés dans la cellule est une caractéristique très importante qui sépare les eucaryotes des procaryotes. Les procaryotes peuvent être souvent décrits comme un seul compartiment : le cytosol. Les eucaryotes, quant à eux, contiennent une variété de compartiments. Des espèces moléculaires ont besoin d'être transportées entre les compartiments, soit par des protéines spécialisées, soit par des réactions spontanées au travers des pores des membranes. Ces réactions qui transportent des espèces moléculaires entre les compartiments sont appelées *réactions de transport* (c.f. Figure 1.2(c)).

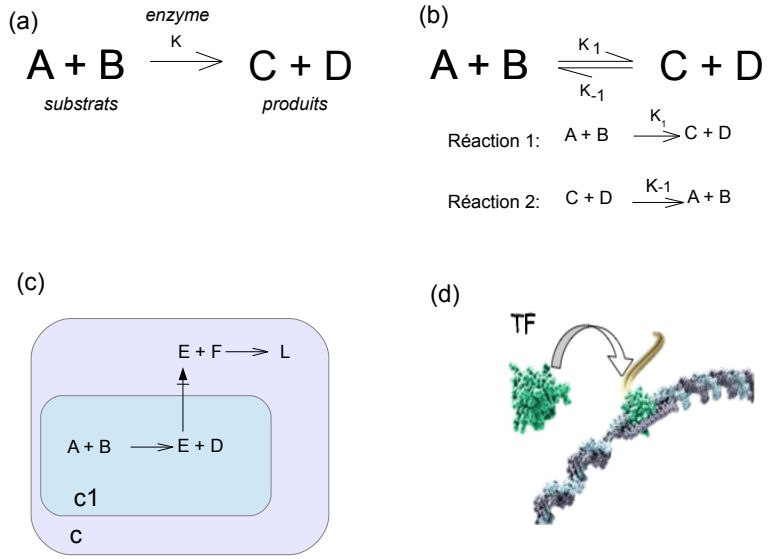


FIGURE 1.2 – Différents types de réaction métabolique : (a) Une réaction irreversible transforme les métabolites A et B en C et D, (b) une réaction reversible transforme A et B en C et D dans un sens et C et D en A et B dans l'autre, (c) une réaction de transport qui transporte la molécule E du compartiment C1 dans le compartiment C, (d) un facteur de transcription se fixe sur le promoteur d'un gène, et cela va traduire le gène en proteine.

On appelle *voie métabolique* un sous-réseau connecté extrait du réseau métabolique et représentant une fonction biologique particulière ou définie par des limites fonctionnelles.

Les voies métaboliques sont définies comme des séries de réactions chimiques successives qui, à partir d'un ensemble de substrats, forment des produits spécifiques. Le plus souvent, ces voies sont associées à des fonctions biologiques précises (comme la biosynthèse d'un acide aminé, la dégradation des sucres).

Formellement, $P = (R_1, ..., R_n)$ est une voie métabolique si $\forall i \in [1, n]$, la réaction R_i a au moins un produit qui est un substrat de la réaction R_{i+1}.

Chacune des enzymes de ces réactions peut éventuellement participer à plusieurs voies métaboliques. La longueur d'une voie métabolique est le nombre de réactions entre le précurseur et les métabolites finaux de la voie.

Bien que cette définition formelle permette de définir de manière unique une voie métabolique, les frontières des voies métaboliques sont généralement difficiles à définir car une réaction peut appartenir à plusieurs voies métaboliques. On en déduit que les voies métaboliques ne sont pas des ensembles disjoints. La définition des voies métaboliques reste floue et dépend des experts et des organismes considérés.

Dans cette thèse, nous cherchons à analyser les réseaux métaboliques. Pour ce faire, il nous faut les représenter dans un modèle adéquat.

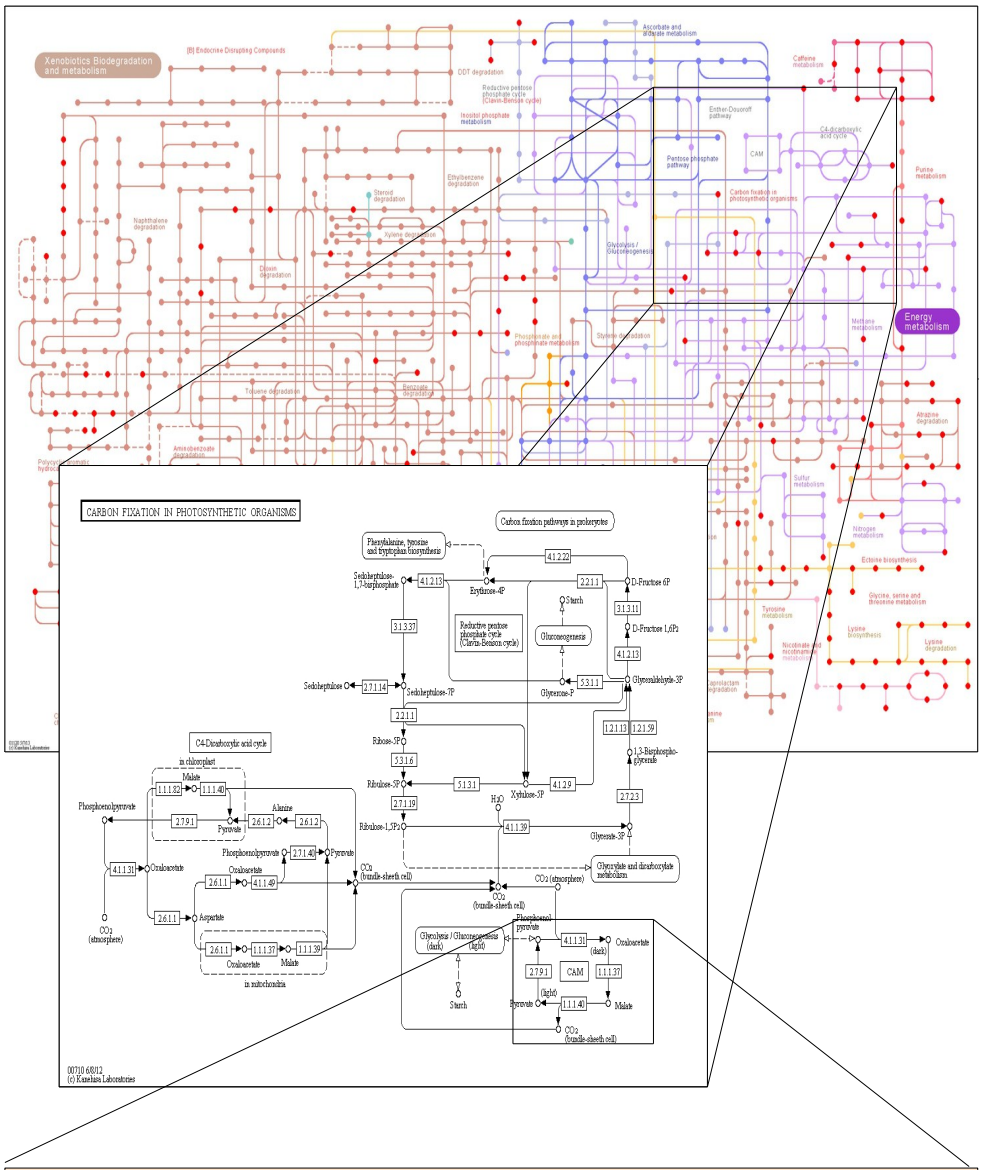
1.2 Modélisation des réseaux métaboliques

La modélisation d'un domaine scientifique est un processus continu d'observation des phénomènes, la compréhension de ceux-ci selon un modèle actuellement choisi et l'utilisation de cette compréhension pour améliorer le modèle général actuel du domaine. Dans ce processus de développement d'un modèle scientifique, on commence avec un modèle relativement simple qui peut être encore amélioré et étendu avec la répétition du processus. Tout modèle des phénomènes à tout stade de son développement peut être incomplète dans sa description. La tâche est alors d'utiliser les informations données par des observations expérimentales pour améliorer et éventuellement compléter cette description.

La modélisation consiste à créer une représentation simplifiée des phénomènes communément appelée *le modèle*. En biologie des systèmes, un modèle est une construction artificielle qui reproduit les comportements et les propriétés spécifiques d'un système que l'on souhaite étudier. Les modèles peuvent avoir différentes formes : e.g., physiques, logiques ou mathématiques.

Avec l'explosion des données expérimentales en biologie, il y a eu beaucoup de tentatives pour développer des modèles mathématiques pour la description des fonctions cellulaires (que ce soit les fonctions globales ou les fonctions de processus cellulaires individuels).

On définira un *métabolisme* comme l'ensemble complet de réactions chimiques qui se produisent dans un organisme vivant afin de maintenir sa vie. Les enzymes ont un rôle principal dans ce processus car elles sont responsables de la catalysation des réactions chimiques. Les relations enzyme-réaction peuvent être utilisées pour la reconstruction d'un réseau de réactions, ce qui conduit à un modèle métabolique du métabolisme. La construction du réseau d'un métabolisme implique des étapes telles que l'annotation



4.1.1.31 : H2O + phosphoenolpyruvate + CO2 → phosphate + oxaloacetate

1.1.1.37 : oxaloacetate + N ADH + H + → (S) − malate + N AD+

1.1.1.40 : (S) − malate + N ADP + → pyruvate + CO2 + N ADP H + H +

2.7.9.1 : AT P + pyruvate + phosphate → AM P + phosphoenolpyruvate + diphosphate

FIGURE 1.3 – Réseau métabolique complet pour *Y. lipolytica* extrait de la base de données KEGG.

fonctionnelle du génome, l'identification des réactions associées et la détermination de leur stoechiométrie, l'affectation de la localisation, la détermination de la composition de la biomasse, l'estimation des besoins en énergie, et la définition des contraintes du modèle. Elle est donc essentiellement dépendante de l'information présente dans le génome de l'organisme correspondant et dans la littérature disponible sur le sujet. On parle alors de modèles à l'échelle du génome qui ont suscité un intérêt considérable du fait de leur capacité à éclaircir les caractéristiques cellulaires et conduire à une meilleure compréhension des systèmes biologiques.

Les modèles métaboliques en particulier ont été largement utilisés pour étudier les voies métaboliques complexes afin de mieux comprendre les systèmes microbiens et de concevoir des stratégies pour l'ingénierie de diverses applications biotechnologiques. Similaires aux réseaux métaboliques, les modèles des réseaux de transcription et de signalisation ont également été reconstruits pour élucider les interactions réglementaires et pour comprendre davantage la réponse des systèmes à divers stimuli environnementaux.

Cependant, un vrai modèle genome-scale qui intègre toutes ces caractéristiques dans un modèle complet n'a pas encore été construit. Pour le moment, les modèles existants de réseaux ont contribué individuellement à la connaissance de leurs domaines respectifs et à notre compréhension des systèmes biologiques.

Etant donné que les réseaux métaboliques et leur régulation sont complexes, une analyse intuitive des systèmes biologiques est une tâche difficile. Différentes méthodes de modélisation mathématique permettent de faire face à cette complexité. La question est donc de déterminer les outils mathématiques les plus adaptés pour un objectif donné. Les systèmes biologiques étant complexes, la tentation d'inclure tous les détails dans le modèle mathématique est grande. Toutefois, il est impossible d'être complet mais des simplifications et approximations seront nécessaires. En outre, le but de la modélisation n'est pas seulement de trouver une description valide du système donné, mais également d'en effectuer l'analyse et la simulation. Par conséquent, les décisions relatives à des simplifications et approximations devraient être fondées sur plusieurs facteurs, tels que la validité de la description du système, la commodité mathématique, et les objectifs de la modélisation.

Deux principaux groupes de techniques de modélisation sont utilisés :

1. **Les méthodes quantitatives** qui visent à capter les changements des quantités de métabolites et d'enzymes en représentant les processus cellulaires : e.g., la cinétique chimique d'une réaction.

2. **Les méthodes qualitatives** qui visent simplement à modéliser la présence d'interactions et analyser la structure. Ces modèles peuvent nécessiter peu de données quantitatives. Par exemple, les réactions sont modélisées comme des transformations chimiques, sans représentation des cinétiques chimiques des réactions.

Ci-après, nous présenterons deux exemples de méthodes quantitatives et un exemple de méthode qualitative, resp. les matrices stoechiométriques, les équations différentielles et les graphes.

1.2.1 Les matrices stoechiométriques

Etant donnée une réaction, on appelle *coefficient stoechiométrique* (ou le nombre stoechiométrique dans IUPAC [MM97]) d'un métabolite ou d'une espèce chimique, le nombre de molécules qui participent à cette réaction. C'est le nombre qui est affecté à cette espèce dans l'équation chimique considérée.

Par exemple dans la réaction qui forme l'ammoniac (NH_3), exactement une molécule d'azote (N_2) réagit avec trois molécules d'hydrogène (H_2) afin de produire deux molécules de (NH_3) :

$$N_2 + 3H_2 \rightarrow 2NH_3$$

Dans cette réaction, le nombre stoechiométrique de l'azote est 1, celui de l'hydrogène est 3, et celui de l'ammoniac est égal de 2. A noter que pour faciliter de notation, lorsque le nombre stoechiométrique est égal à 1, il n'est pas écrit.

On appelle *matrice stoechiométrique* la matrice S à n colonnes et m lignes contenant les coefficients stoechiométriques d'un modèle, où n est le nombre de réactions et m le nombre de métabolites internes (participant à ces réactions). Chaque colonne de la matrice correspond à une réaction chimique ou une réaction de transport, où les valeurs non nulles identifient les métabolites participants à la réaction et définissent les coefficients stoechiométriques correspondant.

L'élément s_{ij} dans la matrice représente le coefficient stoechiométrique du métabolite i dans la réaction j. La matrice contient également une directivité, avec la convention de signe suivante : négatif pour les substrats et positif pour les produits (c.f. Figure 1.4). La matrice stoechiométrique d'une réaction permet de représenter dans quelles proportions les différents métabolites réagissent ou se forment.

En tenant compte des lignes de la matrice à la place des colonnes, la matrice stoechiométrique peut également être considérée comme la liste des réactions auxquelles un metabolite donné participe. Cette interprétation est utile pour définir les équilibres de masses pour chaque métabolite dans le réseau. L'équilibre de masses peut être défini en termes du flux à travers chaque réaction et de la stoechiométrie de cette réaction sous la forme :

$$\frac{d\mathbf{x}}{dt} = \mathbf{S}.\mathbf{v} \tag{1.1}$$

où $\mathbf{v}$ représente le vecteur de flux avec des éléments correspondant à des flux dans des réactions données (colonnes) de $\mathbf{S}$.

À l'état stable, la variation d'un composé $\mathbf{x}$ dans le temps $\mathbf{t}$ dans toutes les réactions du système devient nulle. Cette hypothèse est pertinente pour la plupart des réactions intracellulaires car elles sont généralement beaucoup plus rapides que la vitesse de variation dans les phénotypes cellulaires qui en résultent, tels que la croissance cellulaire et du processus dynamique. L'hypothèse de l'état stable élimine donc la dérivée temporelle de l'équation (1.1), ce qui donne :

$$\mathbf{S}.\mathbf{v} = 0 \tag{1.2}$$

Cette équation nécessite que chaque métabolite soit consommé dans la même quantité qu'il est produit.

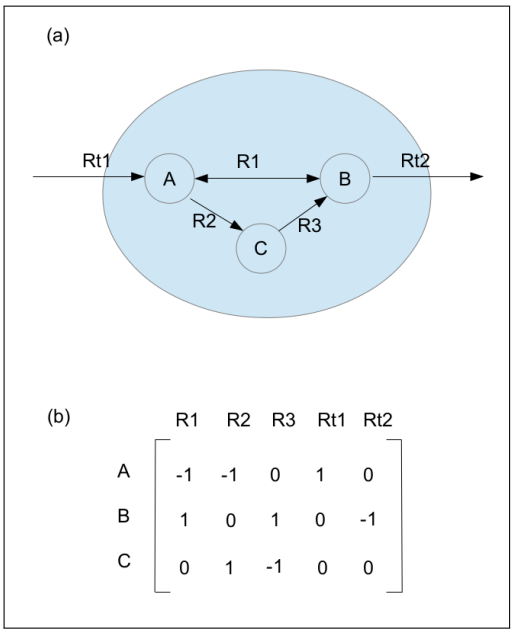


FIGURE 1.4 – (a) Un petit réseau de réactions constitué de trois métabolites (A, B et C), trois réactions métaboliques (R1, R2, R3) et deux réactions de transport (Rt1, Rt2). (b) La matrice stoechiométrique pour ce modèle.

1.2.2 Équations différentielles

Les équations différentielles représentent le formalisme le plus couramment utilisé pour décrire quantitativement la cinétique d'un système dynamique. La cinétique de la régulation d'un réseau est alors modélisée par un ensemble d'équations exprimant le taux de production de chaque composant du système (ARN, protéines, autres molécules) comme une fonction des concentrations des autres composants. Mais la non-linéarité des fonctions utilisées rend ce type de formalisme réfractaire à l'analyse mathématique.

Les modèles *d'équations différentielles (ODEs)* sont des modèles dynamiques qui fournissent une bonne précision et offrent des bonnes représentations des lois de cinétique biochimique. Dans ce type de modèle, on considère que la structure du réseau est connue et on s'intéresse uniquement à sa dynamique.

Le changement des concentrations des métabolites dans une réaction métabolique avec le temps, peut être décrit à l'aide d'ODEs. Par exemple, pour la réaction formant l'ammoniac (NH_3) présentée précédement, on obtient :

$$\frac{dN_2}{dt} = -\nu, \frac{dH_2}{dt} = -3\nu, \frac{dNH_3}{dt} = 2\nu.$$

Cela signifie que la dégradation de N_2 avec la vitesse ν est accompagnée par la dégradation de H_2 avec le triple de la vitesse et avec la production de NH_3 avec une vitesse doublée.

Le système dynamique pour un réseau métabolique se compose de m métabolites et r réactions. Il est décrit par le système d'équations suivant :

$$\frac{dS_i}{dt} = \sum_{j=1}^{r} n_{ij}\nu_j \; for \; i = 1, ..., m$$

La quantité n_{ij} est le coefficient stoechiométrique pour le métabolite i dans une réaction j.

La difficulté d'obtenir des constantes suffisamment précises pour les coefficients stoechiométriques, la complexité de la résolution des nombreuses équations résultantes, et le manque d'outils pour réviser automatiquement ces modèles, ont motivé l'utilisation de techniques plus qualitatives de raisonnement, comme les graphes.

1.2.3 Les graphes

Un graphe $G(V, E)$ est un objet mathémathique où V est l'ensemble des noeuds, et E est l'ensemble des arêtes connectant une paire de noeuds. Une arête est une paire non ordonnée de noeud (graphe non orienté), ou elle est une paire ordonnée de noeud appelé arc (graphe orienté ou direct).

Modéliser un réseau métabolique par un graphe revient à choisir quels sont les éléments du réseau qu'on associe aux noeuds et aux arêtes (ou aux arcs), où les éléments d'un réseau métabolique sont les métabolites (on les appelle ici les entités), les enzymes, et les réactions. Les réseaux métaboliques peuvent être associés à différents types de graphes, nous présentons ici quelques exemples.

Graphe des enzymes

Dans le graphe des enzymes, les sommets correspondent à des enzymes, où il existe une arête entre deux enzymes si elles catalysent deux réactions qui ont des composés en commun.

Ce graphe est moins utilisé que les graphes d'entités et de réactions (présentés ci-après), car il ne donne pas une représentation claire du point de vue biologiste. Une des difficultés d'utilisation de ce graphe est qu'une enzyme peut catalyser plusieurs réactions qui sont parfois éloignées dans un réseau ou dans une voie métabolique en termes du nombre des entités pour passer de l'une à l'autre, alors que ces réactions sont présentées proches dans le graphe des enzymes. Une autre difficulté est d'avoir des réactions qui sont catalysées par plusieurs enzymes, dans ce cas ces réactions seront dupliquées. En fait, l'utilisation du graphe des enzymes perd l'information structurelle qui peut être ignorée quand on s'intéresse uniquement aux enzymes et aux relations entre elles.

Graphe des entités

Un graphe des entités (dites aussi composés) est un modèle de réseau métabolique où les noeuds correspondent aux entités, et où il existe une arête entre deux entités s_1, s_2 s'il existe une réaction qui consomme s_1, et produit s_2.

Cette présentation fournit des informations sur les relations entre les entités comme par exemple la connection ou les distances entre les entités Figure 1.5 (b).

Graphe des réactions

Un graphe des réactions est un modèle de réseau métabolique où les noeuds correspondent aux réactions, et où il existe une arête entre deux réactions r_1, r_2 s'il existe une entité produite par r_1 et consommée par r_2 Figure 1.5 (c). Ce graphe est utile pour analyser les propriétés des relations entre les réactions.

Le graphe des entités et le graphe des réactions sont parfois ambigus car plusieurs réseaux différents peuvent avoir le même graphe. Par exemple, la Figure 1.6 illustre deux réseaux représentés par le même graphe des entités.

Pour éviter cette confusion on peut ajouter des étiquettes sur les arêtes des graphes Figure 1.7 (a). Ces étiquettes identifient des réactions sur les arêtes correspondantes du graphe des entités, et identifient des entités sur les arêtes correspondantes du graphe des réactions. Dans un graphe des entités, une réaction avec n substrats et m produits est représenté par $m \times n$ arcs ayant comme étiquette cette réaction. Une autre façon de résoudre cette ambiguité, c'est d'utiliser un modèle de graphe plus expressif. Par exemple, les graphes bipartis et les hypergraphes, qui sont équivalents en termes de quantité d'informations modélisées, et permettent le même type d'analyse structurale.

Graphe biparti des réactions et des entités

Un graphe biparti est un modèle de réseau métabolique dont l'ensemble des noeuds a une partition de deux ensembles disjoints U et R, où U correspond aux entités et R aux réactions, et tel qu'il existe une arête entre une entité u et une réaction r si cette entité est consommée ou produite par cette réaction Figure 1.5 (d). Le graphe biparti fournit une représentation qui permet plus d'un seul chemin entre les mêmes entités à travers différentes réactions, qui est une importante caractéristique pour différents mécanismes complexes.

Hypergraphe des réactions et des entités

Un Hypergraphe est une généralisation d'un graphe dans lequel une arête peut connecter n'importe quel nombre de noeuds. La Figure 1.8 montre un hypergraphe non orienté qui représente le réseau donné. Dans un hypergraphe orienté, une hyperarête orientée correspond à plusieurs départs et plusieurs arrivées pour une réaction. Pour modéliser un réseau métabolique par un hypergraphe, on associe les entités aux noeuds et les réactions aux hyperarêtes.

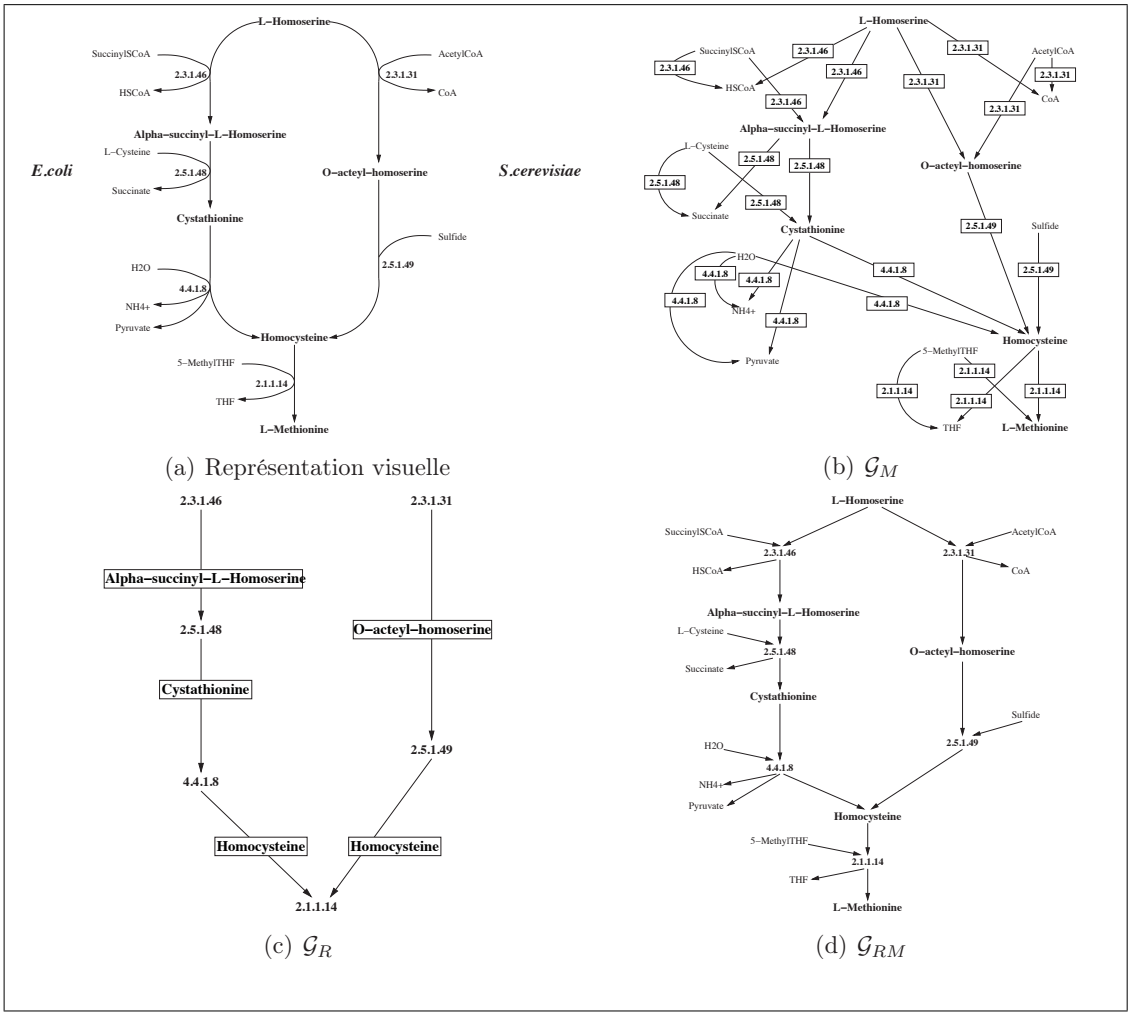


FIGURE 1.5 – Un ensemble de réactions présenté par différents graphes (emprunté à [Boy04] et adapté de [vHWGW02]) - (a) représentation classique, (b) le graphe des entités présente les métabolites en noeuds et les réactions en arêtes, il existe un noeud par entité et tous les couples d'entitiés (substrat, produit) de la même réaction sont reliés par une arête, (c) dans le graphe des réactions, il existe un noeud par réaction et deux réactions sont reliées si elles ont une entité en commun, (d) graphe bipartite où chaque composé et chaque réaction sont associés à un noeud.

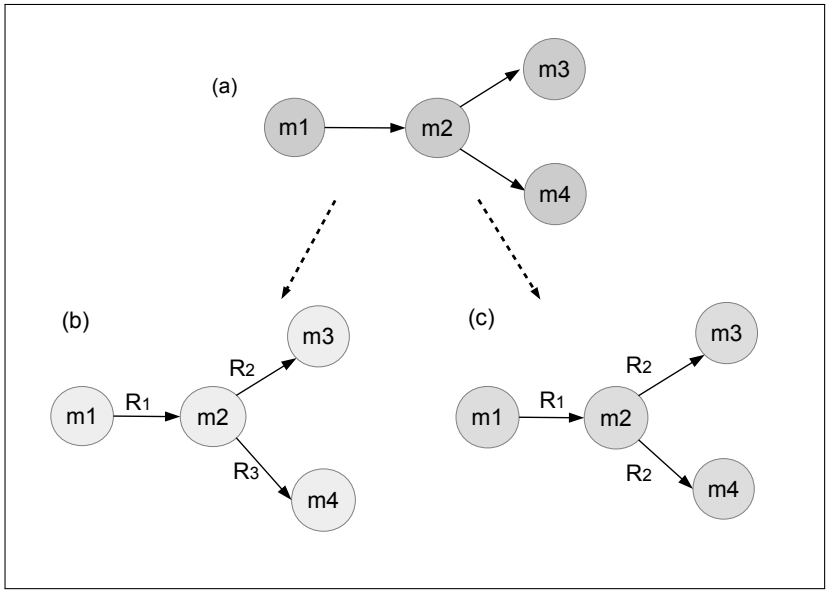


FIGURE 1.6 – Un ensemble d'entités présenté par un graphe dans (a). Ce graphe peut être traité en deux façons différents représentant deux réseaux différents des réactions. (b) réseau 1 : R1 : $m_1 \rightarrow m_2$, R2 : $m_2 \rightarrow m_3$, R3 : $m_2 \rightarrow m_4$, et (c) réseau 2 : R1 : $m_1 \rightarrow m_2$, R2 : $m_2 :\rightarrow m_3 + m_4$.

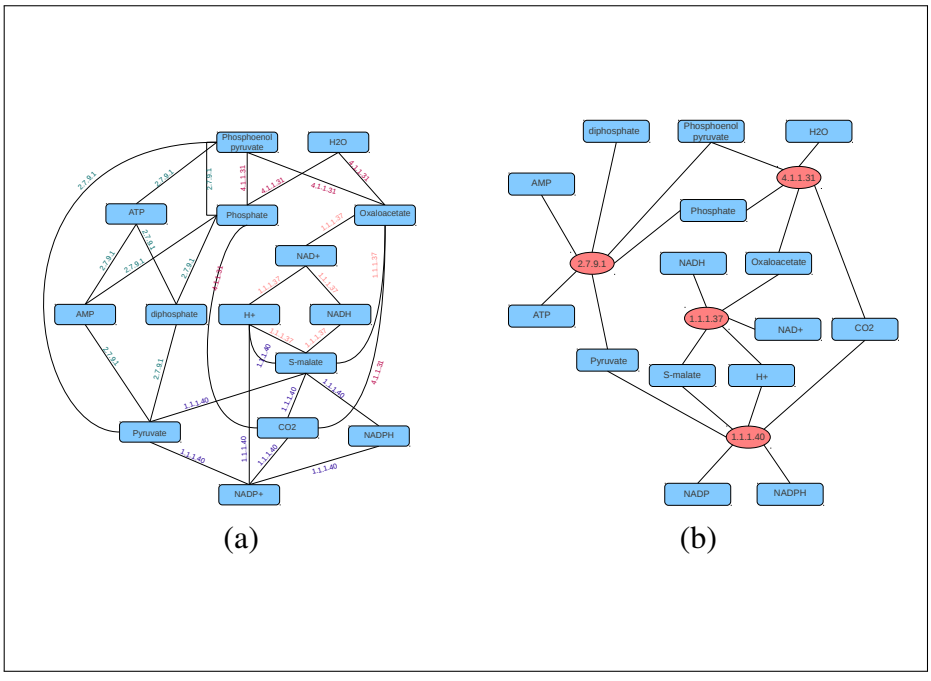


FIGURE 1.7 – Représentations de réseau métaboliques par (a) graphes simples entre nœuds de métabolites, où les réactions engendrent chacune plusieurs arcs ; (b) graphes bipartis entre métabolites et réactions, où les réactions sont aussi des nœuds.

Même pour les graphes bipartis et l'hypergraphe, l'utilisation d'arêtes non orientées dans le cas d'une réaction reversible présente toujours une ambiguïté (plusieurs réseaux peuvent avoir le même graphe).

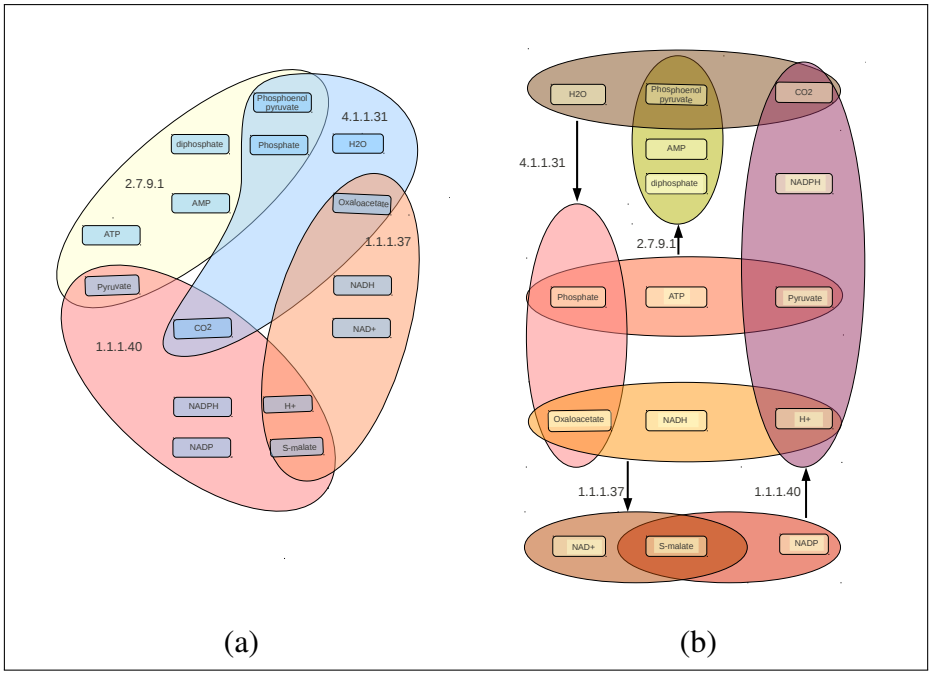


FIGURE 1.8 – Représentations par hypergraphes : (a) une réaction est représentée par un hyperarête entre métabolites simples ; (b) les métabolites consommés ou produits par une réaction sont regroupés dans des hypernœuds, liés par réactions simples.

L'orientation Les modèles peuvent être orientés ou non orientés, selon que les réactions sont réversibles ou pas. Dans le cas d'un réseau où toutes les réactions sont irréversibles (resp. réversibles), on peut modéliser le réseau par un graphe orienté (resp. non orienté). Dans le cas où certaines réactions sont réversibles et d'autres ne le sont pas, on doit utiliser un graphe mixte.

Pour la modélisation, on choisit un modèle de graphe correspondant le mieux à la situation et aux informations que l'on cherche. Si par exemple on s'intéresse aux enzymes et aux relations entre elles, on choisit le graphe des enzymes. En revanche, on choisit le graphe des réactions si l'on s'intéresse à la connecxion des réactions mais pas la façon dont elles sont connectées. Finalement, le choix du graphe biparti devient important si on s'intéresse aux quels métabolites sont connectés aux quelles réactions.

1.3 Construction de réseaux métaboliques

La croissance considèrable du nombre de génomes séquencés et les progrès récents dans les domaines de la bioinformatique et la biologie des systèmes ont fourni plusieurs modèles métaboliques à l'échelle du génome qui ont été utilisés pour fournir des méthodes de simulation de phénotype. Compte tenu de leur importance dans les recherches biomédicales et les applications à la biotechnologie, beaucoup d'attention a été donnée sur le développement du calcul pour la reconstruction rapide des réseaux métaboliques. Ces reconstructions représentent des bases de connaissances structurées représentant l'information pertinente sur les transformations biochimiques qui ont lieu dans les organismes cibles spécifiques.

Les modèles métaboliques à l'échelle du gènome sont devenus un outil indispensable pour l'étude de la biologie des systèmes du métabolisme. Ces modèles offrent un moyen de traduire rapidement les connaissances détaillées de milliers de processus enzymatiques dans des prédictions quantitatives du comportement des cellules entières, ce qui permet la simulation de leurs phénotypes dans des conditions environnementales et génétiques distinctes.

Ces modèles recueillent des informations concernant les différentes entités cellulaires. Tous les modèles disposent d'informations de base sur les réactions métaboliques et les métabolites concernés (y compris stoechiométrie et de l'information de réversibilité), et dans de nombreux cas, le compartiment où les réactions se produisent. La plupart des modèles métaboliques à l'échelle du génome incluent également des informations sur le niveau de transcription / traduction, y compris les enzymes qui catalysent les réactions, informations sur les peptides composant les complexes de protéines et, enfin, les gènes codant pour ces peptides [FHT+09].

Les reconstructions de réseaux métaboliques sont devenues un outil indispensable pour l'étude de la biologie des systèmes du métabolisme [CKR+04, TPVP05, AKV+04, PPL+06, BHRP05, FP08] Pour plus d'informations sur les modèles métaboliques à l'échelle du gènome récents, leur génération automatisée, des améliorations et des applications, le lecteur peut se référer à [KSK+12, CdVK+12, ZSRM12].

Toutes les méthodes de reconstruction du réseau métabolique d'un organisme prennent pour point de départ la séquence complète de son génome. Un point essentiel de la reconstruction est la détection puis l'annotation des gènes codant pour des enzymes le long du génome.

Le processus de reconstruction de modèles et réseaux métaboliques suit trois étapes principales comme le montre la Figure 1.9.

En utilisant des données provenant de différentes bases de données biologiques, la première version de modèles métaboliques peut être reconstruite de façon automatisée comme cela est effectué dans SEED [DOD+13], BioNetBuilder [ACDL+07], et ReMatch [PAR+08]. Ces méthodes peuvent intégrer des données de plusieurs bases de données. La première version de modèles métaboliques peut également être produite de façon semi-automatisée comme dans RAVEN [ALS+13], et MicrobeFlux [FXCT12]. Ces méthodes

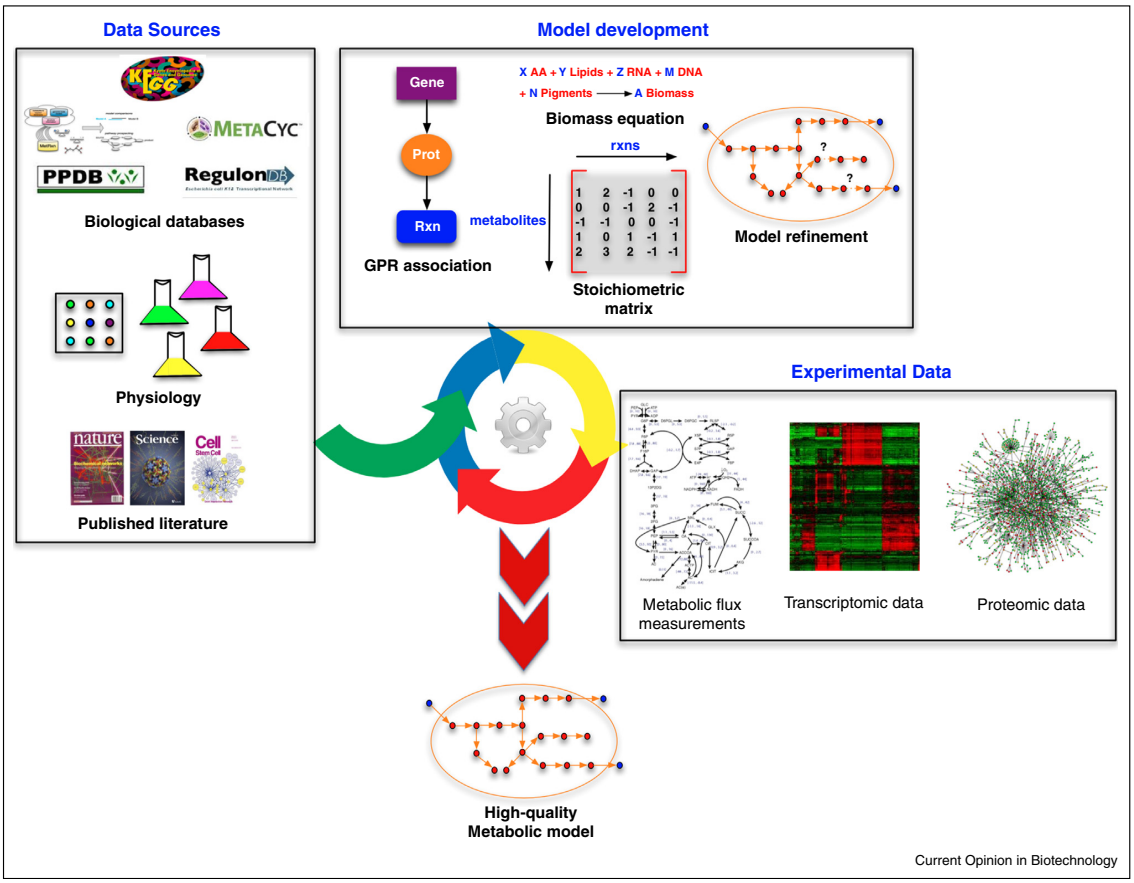


FIGURE 1.9 – Schéma de développement d'un modèle métabolique de haute qualité illustré de [SCM14] : la première étape consiste à extraire des données de différentes bases de données biologiques, la physiologie et la biochimie de l'organisme ainsi que la littérature publiée. Dans l'étape suivante, les associations gènes-proteins-réactions sont établies, l'équation de la biomasse est décrite sur la base de mesures expérimentales et le modèle est représenté sous la forme d'une matrice stoéchiométrique. En outre, les lacunes dans le modèle sont identifiées et réconciliées selon des techniques de comblement des lacunes établies. Enfin, dans la troisième étape des mesures expérimentales à haut débit, telles que les données transcriptomiques et protéomiques, sont utilisées pour améliorer la précision du modèle.

font usage non seulement des bases de données disponibles, mais aussi des modèles publiés des espèces étroitement liées.

Dans l'ensemble, les méthodes automatisées sont très utiles pour l'élaboration des modèles initiaux, alors que les méthodes semi-automatiques peuvent affiner ces premières versions de modèles pour les amener à un niveau d'achèvement requis.

Le type de reconstruction automatique le plus utilisé est la reconstruction par homologies. Son principe est d'attribuer à chaque gène de l'organisme cible sa(ses) fonctions enzymatique(s) (et le nombre EC associé [Bai00]) pour associer une réaction enzymatique à un ensemble des réactions présentes dans l'organime de référence. A partir de cet ensemble de réactions enzymatiques existantes, un réseau est produit. Le principe de ce fonctionnement est illustré sur la Figure 1.10. Les méthodes qui utilisent ce type de reconstruction nécessitent un modèle existant connu comme référence. Ces méthodes sont appelées *soustractive* : les réactions non-conservées seront perdues, par rapport au modèle de référence, mais de nouvelles réactions ne seront pas détectées. Quelques programmes qui mettent en oeuvre cette idée sont pathologic [KPR02], metaSHARK [PSMW05], IdentiCS [SZ04], et AUTOGRAPH [BBV07].

En se basant sur l'annotation du gènome, des voies métaboliques génériques peuvent être utilisées (e.g les voies métaboliques dans la base de données KEGG [KG00]). Une des méthodes adaptant cette idée est SEED [DFB$^+$07] qui utilise une conservation des sous-systèmes et des voies de KEGG comme une base pour le reconstruction de modèles métaboliques et des familles de protéines généralisées pour décider la conservation des gènes.

Une autre approche est la reconstruction supervisée basée sur la prédiction de graphes (un réseau est présenté sous la forme d'un graphe), [BBV07]. Pathologic produit un draft de modèle pour un organisme en analysant la conservation des voies par rapport à un autre organisme [PK02]. La différence pour cette approche c'est qu'elle met l'accent sur la conservation de la voie contre la conservation des réactions individuelles dans les autres méthodes.

AUTOGRAPH, quant à elle, exploite les modèles métaboliques existants en tant que points de départ pour la reconstruction semi-automatisée. Elle utilise Inparanoid [ORS05] comme source de preuves de conservation de fonctionnalités entre les organismes.

1.4 Programmation logique

La programmation logique est une forme de programmation déclarative car elle s'attache davantage au *quoi* qu'au *comment*. La principale différence entre la programmation logique et les langages de programmation classiques repose sur l'exécution des calculs sous forme de preuves logiques. Elle est particulièrement adaptée aux besoins de l'intelligence artificielle, dont elle est un des principaux outils.

Un programme écrit dans un langage de programmation logique est exprimé en termes de relations, représentées comme des faits et des règles sur certains domaines du problème. Il se compose d'un ensemble d'axiomes et d'une déclaration d'objectif. Les règles d'inférence sont appliquées pour déterminer si les axiomes sont suffisants pour assurer la vérité de l'énoncé de l'objectif. L'exécution de ce programme correspond à la construction d'une preuve de l'objectif déclaré par les axiomes. Dans ce langage de programmation, les règles sont écrites sous forme de clauses :

$$H :- B1, \ldots, Bn.$$

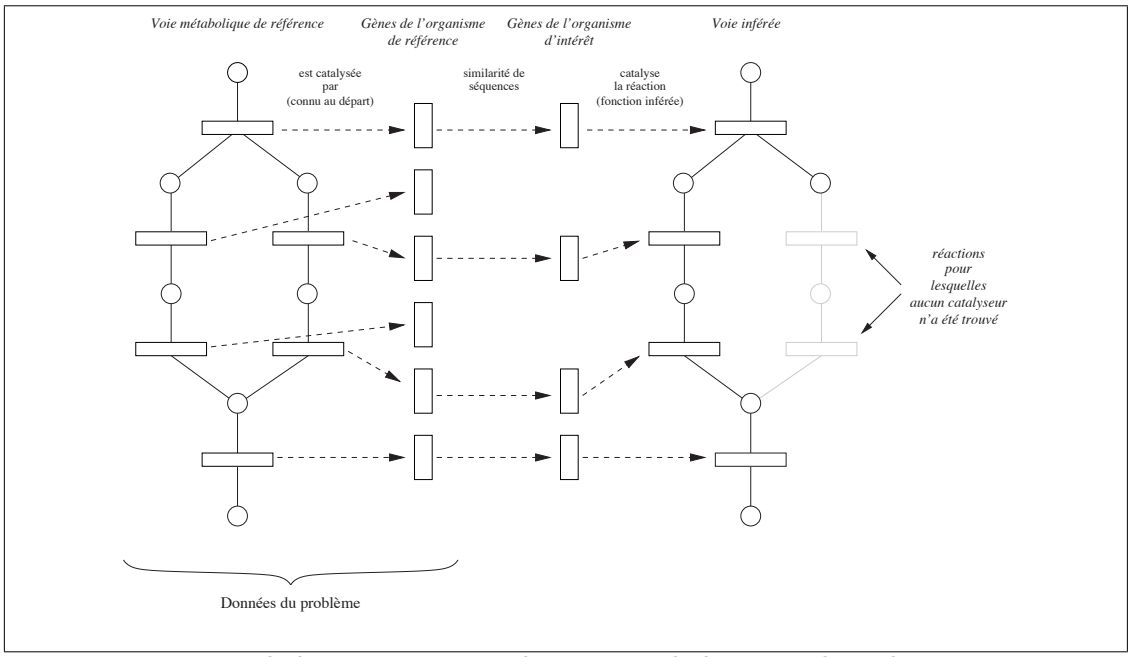


FIGURE 1.10 – Principe de la reconstruction des réactions métaboliques par homologie adapté de [Boy04].Les réactions à reconstruire est supposée entiérement connue dans l'organisme de référence, la similarité de séquences est utilisée afin de trouver dans l'organisme cible les gènes responsables de la catalyse de ces réactions. Si le nombre de catalyseurs trouvés est suffisant, la réaction est alors considérée comme présente dans l'organisme cible.

et elles sont interprétées comme : "Pour prouver H, il faut prouver chacun des $B1, \ldots, Bn$." et peuvent s'exprimer par

$$H \texttt{ if } B1 \texttt{ and } \ldots \texttt{ and } Bn.$$

Par la suite, une *clause* est une séquence finie de littéraux $L_1, \ldots, L_n$ que l'on écrit par convention comme une disjonction $L_1 \vee \ldots \vee L_n$. On appelle *clause de Horn* toute clause comportant au plus un littéral positif. Une clause de Horn est d'autre part appelée *négative* si elle n'inclut pas de littéraux positifs, ou *définie* si elle contient exactement un seul littéral positif.

Une clause de Horn définie $R = L \vee \neg L_1 \vee \ldots \vee \neg L_n$ est souvent appelée *règle* et est notée $L \leftarrow L_1, \ldots, L_n$. L'atome L et les atomes $L_1, \ldots, L_n$ sont respectivement appelés la tête et le corps de la règle. Lorsque le corps de la règle est vide (i.e. $n = 0$), la clause est appelé *fait* et est simplement noté L.

Un programme logique se compose d'un ensemble de clauses de Horn définies, c'est à dire de formules logiques de la forme :

$$A :- B_1, \ldots, B_n$$

où les B_i et A sont des prédicats de la forme $p(t_1, \ldots, t_m)$ où p représente le nom du prédicat et les t_i ses arguments (i.e., des variables, des constantes ou des termes).

La programmation logique avec contraintes [FHK+92, VH91, MS98] combine les avantages de la programmation logique et ceux de la résolution de contraintes en ajoutant des contraintes à la programmation logique standard. Ces contraintes peuvent être définie par tout type de formules logiques dans le corps des clauses.

1.4.1 Langages de programmation logique

Prolog créé par *Alain Colmerauer* et *Philippe Roussel* [CR14] est l'un des principaux langages de programmation logique. Le nom **Prolog** est un acronyme de **PRO**grammation en **LOG**ique. Le langage de programmation Prolog est né d'un projet, dont le but premier n'était pas de faire un langage de programmation mais de traiter les langages naturels, en l'occurrence le Français. Ce projet a donné naissance à un Prolog préliminaire en 1971 et un Prolog plus définitif à la fin de l'année 1972.

En Prolog, le programme logique est exprimé en termes de relations, et un calcul est lancé en exécutant une requête sur ces relations. Le mécanisme de base de Prolog est le suivant : étant donné un programme logique P et un but B, Prolog construit une preuve (par résolution) de B étant donné P.

Il est basé sur le calcul des prédicats du premier ordre ; cependant il est restreint dans sa version initiale à n'accepter que les clauses de Horn (les versions modernes de Prolog acceptent des prédicats plus complexes).

L'utilisateur en Prolog définit une base de connaissances de faits et de règles, pour être utilisée par l'interpréteur pour répondre à des questions. Les relations et les requêtes sont construites en utilisant le seul type de données de Prolog, qui est le *terme*. Les termes sont des atomes, des nombres, des variables ou des termes composés. Les relations dans un programme Prolog sont définies par deux types de clauses : les **prédicats** et les **règles**.

Un prédicat consiste en une tête et un nombre d'arguments. Dans `geneactif(ccp1)`, par exemple, le "`geneactif`" est la tête, et "`ccp1`" est l'argument. En se basant sur ce fait nous pouvons demander à un interpréteur Prolog des requêtes simples comme :

```
?- geneactif(ccp1).
     oui.
?- geneactif(X).
     X = ccp1;
     fail.
```

Dans ce second exemple, à la question "`geneactif(X)`" l'interpréteur propose la réponse "`X = ccp1`" unifiant la variable "`X`" à l'atome "`ccp1`", et c'est un succès pour Prolog. Après cette première réponse, on peut demander s'il y a d'autres réponses en utilisant le « ; » (symbole de la disjonction), ici l'interpréteur répond qu'il n'en trouve pas. Quand Prolog ne peut pas prouver un fait, la recrecherche échoue comme dans l'exemple :

```
?- geneactif(ccp2).
     fail.
```

Les prédicats sont généralement définis pour exprimer les faits que le programme connaît à propos d'un domaine, et l'usage de prédicats requiert une certaine convention. Par exemple, la sémantique des deux prédicats suivants n'est pas immédiate :

```
genecode(ccp1,ccp1p).
genecode(ccp1p,ccp1).
```

Dans les deux prédicats "`genecode`" est la tête, et "`ccp1`" et "`ccp1p`" sont les arguments. Nous pouvons lire le premier prédicat comme " le gène ccp1 code la protèine ccp1p" et le second comme " la protèine est codée par le gène ccp1". Les deux interprétations sont vrais en Prolog, mais l'ordre des arguments compte. Il est important de respecter le même ordre au sein d'un même programme.

Le second type de clause est la règle qui est de la forme :

```
Tête :- Corps.
```

Où ":−" correspond à "si" ; ce qui induit la signification suivante pour la règle : "La Tête est vraie si le Corps est vrai".

Exemple d'une règle avec des variables :

```
proteine(P) :- geneactif(G),genecode(G,P).
```

Cette règle signifie que "une protéine est présente si le gène codant pour cette protéine est active".

Un programme en Prolog est une collection de faits et de règles qui forment une base de connaissances. On utilise un programme de Prolog en lui posant des requêtes ; autrement dit, en posant des questions au sujet des informations stockées dans la base de connaissances.

L'information négative ne peut être exprimée dans la clause logique d'Horn. Cependant, Prolog fournit l'opérateur de négation par échec (en anglais NAF pour negation as failure, ou NBF pour negation by failure) pour trouver une preuve. Rappelons que la négation par échec est une règle d'inférence non monotone en programmation logique, utilisée pour la dérivation de `not(p)` à partir de l'échec de la dérivation de `p`. En d'autres termes, pour démontrer `not(p)` Prolog va tenter de démontrer `p`. Si la démonstration de p échoue, Prolog considère que `not(p)` est démontrée.

Pour Prolog, `not(p)` signifie que la formule `p` n'est pas démontrable, et non que c'est une formule fausse. C'est ce que l'on appelle l'hypothèse du *monde clos* qui est utilisée pour vérifier qu'une formule n'est pas vraie, et non pas pour détérminer la valeur d'une variable. La négation par l'échec ne doit être utilisée que sur des prédicats dont les arguments sont déterminés et à des fins de vérification.

Plusieurs environnements Prolog sont disponibles, dans cette thèse nous utilisons, SWI-Prolog. SWI-Prolog a été créé en 1987 par *Jan Wielemaker* et offre un environnement Prolog complet. Le développement SWI-Prolog a été guidé par les besoins des applications du monde réel et est largement utilisé dans la recherche et l'éducation ainsi qu'au sein d'applications commerciales (`http://www.swi-prolog.org/`).

Constraint Handling Rules (CHR) [Frü98, FA03, FR09] est un langage déclaratif à base de règles, constitué de règles avec des gardes, appelées contraintes, qui transforment des multi-ensembles de relations jusqu'à ce qu'il n'y ait plus de changement. CHR est déclaratif puisque ses règles peuvent être considérées comme des formules logiques.

Initialement prévu pour la programmation par contraintes, CHR trouve des applications dans différents domaines dont *le raisonnement abductif*. CHR étend Prolog avec un ensemble de contraintes. Les règles d'un programme CHR servent de règles de réécriture sur les ensembles de contraintes.

CHR ne s'impose pas nécessairement comme un nouveau langage de programmation, mais plutôt comme une extension de langage (au sein de la syntaxe du langage hôte), qui est souvent Prolog. Dans la langage hôte, les contraintes de CHR peuvent être affichées dans les règles de CHR, et les déclarations du langage hôte peuvent être incluses.

Un programme de CHR est un ensemble fini de règles comprenant des contraintes, où les règles peuvent être de trois types : règles de Simplification, de Propagation ou de Simpagation.

Les règles de simplification sont de la forme suivante :

$$H_1, \ldots, H_i \Longleftrightarrow G_1, \ldots, G_j | B_1, \ldots, B_k$$

où $i > 0, j \geq 0, k \geq 0$ et où la tête est une séquence (conjonction) non vide de contraintes de CHR, les gardes $G_1, \ldots, G_j$ forment une conjonction de contraintes intrinsèques, et le corps $B_1, \ldots, B_k$ qui est l'objectif est une conjunction de contraintes de CHR et des contraintes intrinsèques. Une règle de simplification signifie que la tête est vraie si et seulement si le corps est vrai. Opérationnellement, lorsque la garde est vraie alors la tête et le corps sont équivalents et on remplace la tête par le corps. Les contraintes intrinsèques sont des contraintes prédéfinies par le langage de base, tandis que les contraintes de CHR sont définies par les règles de CHR.

Les règles de propagation sont de la forme suivante :

$$H_1, \ldots, H_i \Longrightarrow G_1, \ldots, G_j | B_1, \ldots, B_k$$

Une règle de propagation est une implication, à condition que la garde soit vraie, et elle signifie que le corps est vrai si la tête est vraie. Pratiquement, lorsque les contraintes dans la tête sont dans l'ensemble des contraintes, et la garde est vraie, les contraintes dans le corps sont ajoutées à l'ensemble des contraintes.

Les règles de simpagation sont de la forme suivante :

$$H_1, \ldots, H_l \backslash H_{l+1}, \ldots, H_i \Longleftrightarrow G_1, \ldots, G_j | B_1, \ldots, B_k$$

où $l > 0$ et les autres symboles ont la même signification que ceux dans les règles de simplification. La règle de simpagation est équivalente logiquement à la règle de simplification :

$$H_1, \ldots, H_l, H_{l+1}, \ldots, H_i \Longleftrightarrow G_1, \ldots, G_j | B_1, \ldots, B_k, H_1, \ldots, H_l$$

Pratiquement, lorsque les contraintes dans la tête sont dans l'ensemble des contraintes, et la garde est vraie, $H_1, \ldots, H_l$ reste dans l'ensembl des contraintes, et $H_{l+1}, \ldots, H_i$ sont remplacés par $B_1, \ldots, B_k$.

Le programme CHR suivante, dans la syntaxe Prolog, contient quatre règles qui mettent en oeuvre un solveur pour une contrainte moins ou égal

réflexivité X $\leq$ X <=> true. (X $\leq$ Y)

une règle de simplification : elle exprime que, si un fait se trouve dans le stock, il peut être éliminé. comme (X $\leq$ X).

antisymétrie X $\leq$ Y, Y $\leq$ X <=> X = Y. (si X $\leq$ Y et Y $\leq$ X, alors X = Y)

une règle de simplification : elle exprime que, si deux faits se trouvent dans le stock (X $\leq$ Y et Y $\leq$ X), ils peuvent être remplacer par le seul fait (X = Y). Et cette contrainte d'égalité est considéré comme intégré.

transitivité X $\leq$ Y, Y $\leq$ Z ==> X $\leq$ Z. (si X $\leq$ Y et Y $\leq$ Z, alors X $\leq$ Z)

une règle de propagation : contrairement à la simplification, il ne supprime rien du stock de contraintes, au lieu, elle ajoute un troisième fait (X $\leq$ Z), lorsque les deux faits (X $\leq$ Yet Y $\leq$ Z) y sont.

idempotence X $\leq$ Y $\setminus$ X $\leq$ Y <=> true.

une règle de simpagation : quand elle trouve des faits dupliqués, elle les supprime du stock.

Il y a une analogie évidente entre les abductibles plus les contraintes d'intégrité et les CHR plus les règles [AC01].

1.4.2 Le raisonnement abductif

Dans le cadre de cette thèse, nous proposons une nouvelle approche de construction de réseaux métaboliques basée sur la logique abductive (c.f. Chapitre 4). Nous commençons ici par une introduction du raisonnement abductif.

L'abduction [DK02] est un type de raisonnement (inférence ou argument). Si la notion de l'abduction tire en effet ses racines chez Aristote ou de certains passages de Laplace, c'est bien Peirce qui l'a théorisée [Fan70], [Pei55], et a identifié trois formes distinguées de raisonnement : la déduction, l'induction et l'abduction.

La *déduction* est un processus analytique basé sur l'application de règles générales sur des cas particuliers, qui procède par nécessité (la conclusion est nécessairement présente dans les prémisses).

L'*induction* est un raisonnement produisant des généralisations. En effet, il consiste à généraliser un raisonnement ou une observation à partir de cas singuliers. Ici, les conclusions sont plus générales que les prémisses.

L'*abduction*, quant à elle, est une autre forme d'inférence synthétique. Elle correspond à l'inférence de la meilleure explication.

Partant d'un fait inattendu, l'abduction permet de déduire une nouvelle hypothèse sur ce qui pourrait expliquer ce fait.
En aucune manière, l'abduction, à elle seule, ne permet de dire si une hypothèse est vraie ou fausse. C'est à partir de la déduction, puis de l'induction comme étape finale que la question de la vérité pourra être abordée. En d'autres termes, l'abduction ne porte que sur le possible (ou l'impossible).

Les définitions originales proposées par Pierce *et al.* [Pei74] sont *"Deduction proves that something must be ; Induction shows that something actually is operative ; Abduction merely suggests that something may be."*

Nous considèrons l'exemple de Kakas *et al.* [KKT92] défini sur les ensembles d'observations $\mathcal{O}$, de règles $\mathcal{R}$ et de contraintes $\mathcal{C}$ suivant :

$\mathcal{O}$ = {shoes are wet}
$\mathcal{R}$ contient les règles suivantes :

grass is wet **if** *it rained*
grass is wet **if** *sprinkler was on*
shoes are wet **if** *grass is wet*

ainsi que la contrainte $\mathcal{C}$ = {*false* **if** it rained **and** the sun was shining}.

Nous souhaitons expliquer l'observation "my shoes are wet". Une explication correspond à un ensemble d'hypothèses qui avec les connaissances explicites impliquent l'observation donnée.

{*rained last night*} est une explication possible.{*sprinkler was on*} est une explication alternative. C'est une forme de raisonnement non-monotone, car les explications qui sont compatibles avec un état de la base de connaissances peuvent devenir incompatibles suite à l'ajout d'une nouvelle information. Dans l'exemple précédent si on ajoute la nouvelle information :

The sun was shining

et la contrainte :

false ***if*** *it rained* ***and*** *the sun was shining*.

qui interdit proposer l'hypothèse {it rained} quand on a le fait {the sun was shining} l'explication {*rained last night*} peut devenir fausse, et l'explication alternative {*sprinkler was on*} peut être la vraie raison pour l'observation donnée.

L'existence de multiples explications est une caractéristique générale de raisonnement abductif, et la sélection des explications "préférés" est un problème important.

L'abduction peut ainsi mettre en évidence des mécanismes, ce qui crée un lien possible avec la recherche qualitative, dans le Diagnostic, la Planification, le Langage Naturel et l'Apprentissage Automatique.

Chapitre 2

Représentation logique

Ce chapitre a pour but de donner une définition formelle et sémantique des descriptions de modèles métaboliques utilisés comme base pour l'inférence de nouveaux modèles. L'objectif est de définir une représentation dans laquelle les opérations d'inférence sont faciles à définir et à manipuler.

Après avoir défini différents méthodes de représentation (Section 2.1), nous définissons une représentation de modèles métaboliques, ainsi que les interprétations sémantiques en termes de SBGN et en termes de relations logiques dans la Section 2.2. Dans la Section 2.2.3, nous définissons les opérations souhaitées d'édition du modèle pour montrer comment cette représentation peut être utilisée afin d'inférer ou d'améliorer des modèles.

2.1 Méthodes de représentation

2.1.1 Notation graphique de système biologie (System Biology Graphical Notation - SBGN)

SBGN [LNHM+09, LNMS+08], est un langage visuel développé par une communauté de biochimistes, des modélisateurs et informaticiens (`http://sbgn.org`). L'objectif de SBGN est de standardiser la représentation graphique/visuelle des processus biochimiques et cellulaires essentiels étudiés en systèmes biologiques. Il définit un ensemble de symboles avec une sémantique précise, ainsi que des règles syntaxiques détaillées définissant leur utilisation. Il décrit également la manière dont ces informations graphiques doivent être interprétées.

La standardisation de notations graphiques pour décrire les interactions biologiques est une étape importante vers la transmission précise et efficace des connaissances biologiques entre différentes communautés. Jusqu'au développement de SBGN, aucune convention n'existait pour définir précisément comment dessiner les schémas représentant les interactions entre les gènes et les molécules d'une manière cohérente, correcte et sans ambiguïté.

Inclure tous les aspects pertinents d'un système biologique dans un seul diagramme conduit à un schéma désespérément compliqué et incompréhensible aux lecteurs humains. Pour cette raison, il existe trois langages en SBGN correspondant à trois *points de vue* différents du même modèle : "Process Description", "Entity Relationship" et "Activity Flow". Ces *points de vue* sont représentés par des classes différentes d'informations, où chaque language définit un ensemble complet de symboles avec une sémantique précise, ainsi que des règles syntaxiques détaillées concernant la construction et l'interprétation des plans.

Dans le langage **SBGN Process Description**, chaque noeud représente un état donné d'une espèce. Par conséquent, une espèce peut apparaître plusieurs fois dans le même schéma si elle représente la même entité dans des états différents. Ce langage, facile à comprendre, est approprié pour représenter les aspects temporels d'interaction. Pour autant, il est difficile de repérer les interactions impliquant une entité donnée puisqu'elle peut apparaitre plusieurs fois dans le diagramme.

Au contraire, dans le langage **SBGN Entity Relationship**, une espèce donnée apparaît une seule fois. Ce langage est utile pour comprendre les relations impliquant chaque entité, mais l'évolution temporelle des événements est difficile, voir impossible, à suivre du fait de l'absence de description des séquences d'événements.

Finalement, le langage **Activity Flow** est un langage hybride entre les deux précédents. Il est particulièrement commode pour représenter l'effet de perturbations, soit génétiques ou environnementale.

Un exemple présente la description au format SBGN des voies métaboliques de la Glycolyse issu du [LNMS+08] est fourni en Figure 2.11.

Le format SBGN n'est pas adapté à la problématique d'inférence de modèle du fait de sa représentation à grain trop fin pour une manipulation de données à grande échelle.

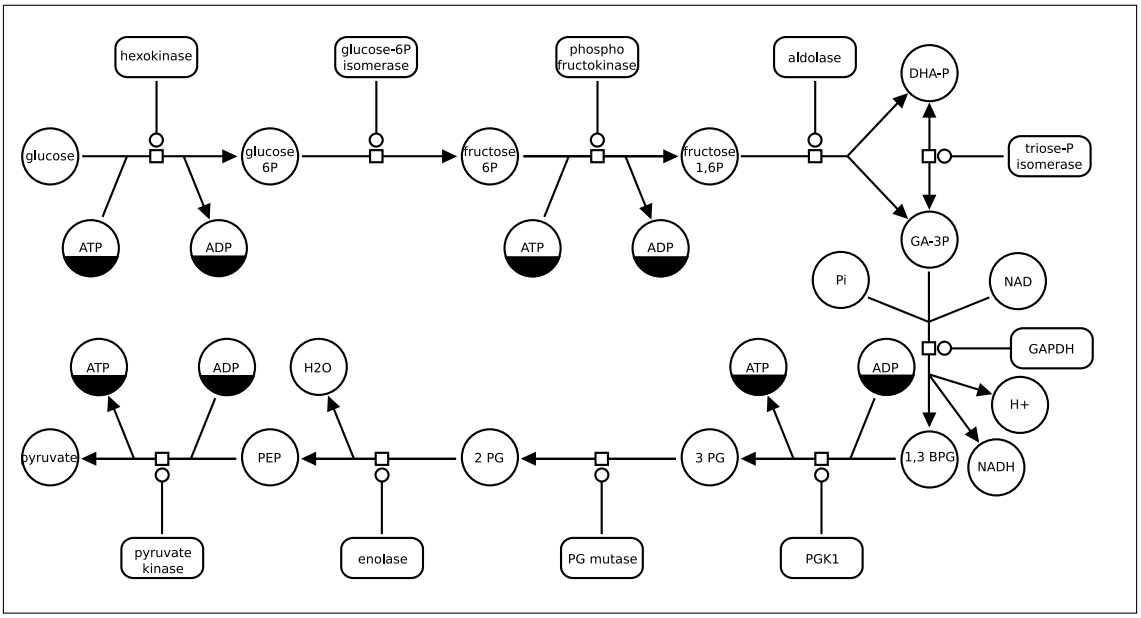


FIGURE 2.11 – Exemple sur la Glycolyse issu de [LNMS+08] illustrant l'utilisation de SBGN pour décrire les voies métaboliques. Le schéma ci-dessus représente la Glycolyse qui est un processus libérant l'énergie de catabolisme du glucose. Dans cet exemple, les macromolécules ne sont que des catalyseurs, et les autres molécules sont consommées ou produites. Notons que les noeuds comportant une bande noire sont des marqueurs indiquant que ce dernier peut apparaître plusieurs fois dans le schéma.

2.1.2 Langage de marquage des systèmes biologiques (Systems Biology Markup Language - SBML)

SBML est un langage informatique de balisage utilisé pour représenter des modèles biologiques [1]. Il est orienté vers des systèmes décrivant où les entités biologiques sont impliquées et modifiées par des processus qui se produisent au fil du temps (e.g., un réseau de réactions biochimiques). Un modèle SBML peut représenter différentes classes de phénomènes biologiques, y compris les chemins de signalisation cellulaire, les réseaux métaboliques, des réactions biochimiques, la régulation génétiques, *etc*.

Comme indiqué dans [HFS+03], l'utilisation de SBML offre de nombreux avantages. SBML permet (1) l'utilisation de plusieurs outils sans avoir à réécrire des modèles adaptés à chaque outil, (2) aux modèles d'être partagés et publiés sous une forme indépendante de l'environnement logiciel utilisé par l'utilisateur et (3) d'assurer la survie de modèles au-delà de la durée de vie du logiciel utilisé pour les créer.

Le squelette commun à tout modèle SBML est le suivant.

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<sbml xmlns="..." level="..." metaid="..." version="...">
<model id="..." name="..." metaid="...">
    <listOfUnitDefinitions>
      ...
    </listOfUnitDefinitions>
    <listOfCompartments>
    ...
    </listOfCompartments>
    <listOfSpecies>
    ...
    </listOfSpecies>
    <listOfParameters>
    ...
    </listOfParameters>
    <listOfRules>
    ...
    </listOfRules>
    <listOfReactions>
    ...
    </listOfReactions>
</model>
</sbml>
```

Dans l'exemple suivant, nous présentons une réaction du modèle `BIOMD0000000010` extraite de BioModels Database (`http://www.ebi.ac.uk/biomodels-main/`)

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
```

1. `http://sbml.org/Basic_Introduction_to_SBML`

```
<sbml xmlns="..." level="..." metaid="_492719" version="4">
  <model id="BIOMD0000000010" name="Kholodenko2000 - MAPK feedback"
  metaid="_000001">
    ...
   </reaction>
   <reaction id="J9" name="dephosphorylation of MAPK-P" metaid="_584815"
   reversible="false">
     <annotation>
       <rdf:RDF xmlns:rdf="..." xmlns:bqmodel="http://
          biomodels.net/model-qualifiers/" xmlns:bqbiol=
          "http://biomodels.net/biology-qualifiers/">
          <rdf:Description rdf:about="#_584815">
            <bqbiol:isVersionOf>
              <rdf:Bag>
               <rdf:li rdf:resource="http://identifiers.org/.../3.1.3.16"/>
               <rdf:li rdf:resource="http://identifiers.org/.../GO:0006470"/>
              </rdf:Bag>
            </bqbiol:isVersionOf>
          </rdf:Description>
       </rdf:RDF>
     </annotation>
     <listOfReactants>
       <speciesReference species="MAPK_P" metaid="_653547"/>
     </listOfReactants>
     <listOfProducts>
       <speciesReference species="MAPK" metaid="_653559"/>
     </listOfProducts>
     <kineticLaw metaid="_653571">
       <math xmlns="http://www.w3.org/1998/Math/MathML">
         <apply>
           <divide/>
           <apply>
             <times/>
             <ci> uVol </ci>
             <ci> V10 </ci>
             <ci> MAPK_P </ci>
           </apply>
           <apply>
             <plus/>
             <ci> KK10 </ci>
             <ci> MAPK_P </ci>
           </apply>
         </apply>
       </math>
       <listOfParameters>
         <parameter id="V10" metaid="_001365" value="0.5"/>
```

```
          <parameter id="KK10" metaid="_001367" value="15"/>
        </listOfParameters>
      </kineticLaw>
      </reaction>
      ...
    </listOfReactions>
  </model>
</sbml>
```

2.2 Représentation logique de modèles

Dans cette section, nous présentons notre proposition de représentation logique de modèles métaboliques qui considère la réaction comme un ensemble de faits. Notons que cette représentation est adaptée à l'inférence de modèle, comme nous le verrons dans le Chapitre 4.

2.2.1 Domaines sémantiques

Notons *EPNs* l'ensemble de toutes les espèces et $\mathfrak{C}$ l'ensemble des compartiments. Alors, une réaction est définie comme un processus transformant une ou plusieurs éspèces de *EPNs* (appelées *substrats*), en une ou plusieurs espèces de *EPNs* (appelées *produits*) contrôlé par d'autres entités de *EPNs* qui catalysent, inhibent, ou modifient cette réaction.

Pour décrire formellement notre modèle, nous définissions, en premier lieu, un ensemble de faits. Soit $\mathit{substrate}(\mathrm{g}, c, n)$ (resp. $\mathit{produce}(\mathrm{g}, c, n)$) la notation du fait correspondant à la présence, dans le compartiment c, de l'espèce g comme substrat (resp. produit) d'une réaction avec le coefficient stoéchiométrique n. Similairement,
soit $\mathit{catalyzed}(\mathrm{g}, c)$ (resp. $\mathit{inhibited}(\mathrm{g}, c)$, $\mathit{modulated}(\mathrm{g}, c)$, $\mathit{stimulated}(\mathrm{g}, c)$) la notation du fait correspondant à la catalysation (resp. inhibition, modulation, stimulation), par l'espèce g, d'une réaction dans le compartiment c.

A partir de ces notations, nous pouvons définir formellement les domaines (au sens mathématique du terme) suivants :

$$Substrat = \{\mathit{substrate}(\mathrm{g}, c, n) | \, (g, c, n) \in \mathit{EPNs} \times \mathfrak{C} \times \mathbb{N}^*\}$$

$$Produit = \{\mathit{produce}(\mathrm{g}, c, n) | \, (g, c, n) \in \mathit{EPNs} \times \mathfrak{C} \times \mathbb{N}^*\}$$

$$Catalyseur = \{\mathit{catalyzed}(\mathrm{g}, c) | \, (g, c) \in \mathit{EPNs} \times \mathfrak{C}\}$$

$$Inhibiteur = \{\mathit{inhibited}(\mathrm{g}, c) | \, (g, c) \in \mathit{EPNs} \times \mathfrak{C}\}$$

$$Modulateur = \{\mathit{modulated}(\mathrm{g}, c) | \, (g, c) \in \mathit{EPNs} \times \mathfrak{C}\}$$

$$Stimuleur = \{\mathit{stimulated}(\mathrm{g}, c) | \, (g, c) \in \mathit{EPNs} \times \mathfrak{C}\}$$

On appelera rôle l'union de ces domaines définie formellement par

ROLE $= Substrat \cup Produit \cup Catalyseur \cup Inhibiteur \cup Modulateur \cup Stimuleur$.

L'ensemble des réactions possibles, noté $\mathfrak{R}$, peut alors être défini par $\mathfrak{R} = \{A | A \subseteq \text{ROLE}\}$.

Notons que étant donnée une réaction $r \in \mathfrak{R}$, il existe un arc entrant étiqueté dans un graphe de réactions, et plus précisément, un ensemble de demi-arcs du domaine ROLE.

Les domaines précédement définis peuvent être réduits à une réaction donnée $r \in \mathfrak{R}$ comme suit :

$$\begin{aligned} Sub(r) &= \{s \in Substrat \mid s \in r\} \\ Prod(r) &= \{p \in Produit \mid p \in r\} \\ Catz(r) &= \{c \in Catalyseur \mid c \in r\} \\ Inh(r) &= \{i \in Inhibiteur \mid i \in r\} \\ Modul(r) &= \{m \in Modulateur \mid m \in r\} \\ Stimul(r) &= \{s \in Stimuleur \mid s \in r\} \end{aligned}$$

Formellement, la réaction illustrée en Figure 2.12 peut s'écrire comme suit.

$$\begin{aligned} r = \{&\textit{substrate}(\text{ATP}, c, 1), \textit{substrate}(\text{glucose}, c, 1), \\ &\textit{produce}(\text{glucose6P}, c, 1), \textit{produce}(\text{ADP}, c, 1), \\ &\textit{catalyzed}(\text{hexokinase}, c), \textit{inhibited}(\text{factor}, c)\}. \end{aligned}$$

où

$$\begin{aligned} Sub(r) &= \{\textit{substrate}(\text{ATP}, c, 1), \textit{substrate}(\text{glucose}, c, 1)\} \\ Prod(r) &= \{\textit{produce}(\text{glucose6P}, c, 1), \textit{produce}(\text{ADP}, c, 1)\} \\ Catz(r) &= \{\textit{catalyzed}(\text{hexokinase}, c)\} \\ Inh(r) &= \{\textit{inhibited}(\text{factor}, c)\} \end{aligned}$$

2.2.2 Modèle métabolique

Notons $\mathfrak{M}$ l'ensemble de tous les modèles métaboliques et $\mathfrak{G}$ l'ensemble de tous les gènes. Alors un modèle métabolique $M \in \mathfrak{M}$ est défini comme un triplet $M = \langle R, E, C \rangle$ où : R est l'ensemble des réactions du modèle M, $C \subseteq \mathfrak{C}$ est son ensemble de compartiments et $E = \{(g, c) | \, g \in \textit{EPNs}, c \in C\}$ est son ensemble d'espèces, par notation (g, c) on note que l'espèce moléculaire g est situé dans le compartiment c .

Pour un modèle donné $M = \langle R, E, C \rangle$ et une réaction $(r \in R)$ nous notons *EPNs(r)*

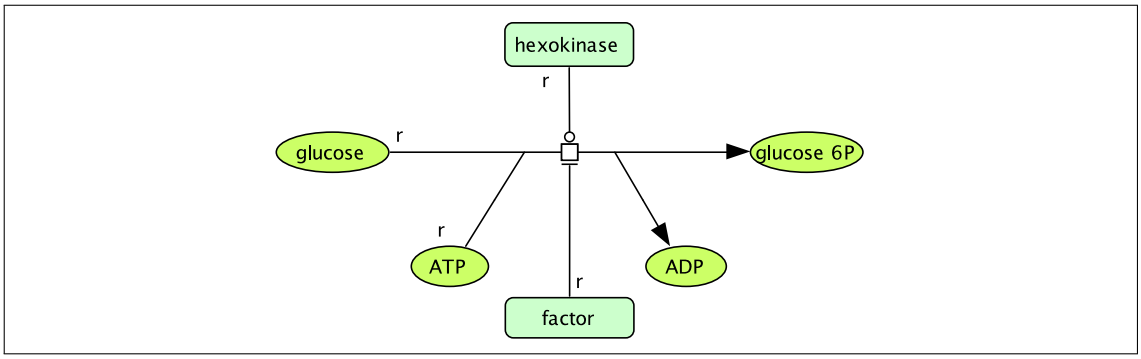


FIGURE 2.12 – Exemple d'une réaction r où l'enzyme hexokinase convertit glucose en glucose 6P inhibée par un facteur non nommé dans un compartiment c.

l'ensemble des espèces participant dans cette réaction. Formellement

$$\begin{aligned} \textit{EPNs}(r) = \{(g,c) \in E | \, \exists n \in \mathbb{N}^*, \\ \textit{substrate}(\mathrm{g},c,n) \quad &\in r \quad \vee \\ \textit{produce}(\mathrm{g},c,n) \quad &\in r \quad \vee \\ \textit{catalyzed}(\mathrm{g},c) \quad &\in r \quad \vee \\ \textit{inhibited}(\mathrm{g},c) \quad &\in r \quad \vee \\ \textit{modulated}(\mathrm{g},c) \quad &\in r \quad \vee \\ \textit{stimulated}(\mathrm{g},c) \quad &\in r \quad \} \end{aligned}$$

Par conséquent l'ensemble des espèces de toutes les réactions R du modèle M est noté comme $\textit{EPNs}(R) = \bigcup_{r \in R} \textit{EPNs}(r)$. De façon similaire, notons $\textit{EPNs}(c) = \{(g,c) \in E\}$ pour l'ensemble des espèces présentes dans le compartiment $(c \in C)$ et $\textit{EPNs}(C) = \bigcup_{c \in C} \textit{EPNs}(c)$ l'ensemble de toutes les espèces présentes dans les compartiments de M.

Une réaction $r \in R$ peut se dérouler dans un compartiment $c \in C$, noté (r,c), si toute espèce participant à la réaction r est présente dans le compartiment c (i.e., $\{(g,c) \in \textit{EPNs}(r)\} \subseteq E$). Nous noterons $R_c = \{r \in R | (r,c)\}$ l'ensemble des réactions pouvant se dérouler dans le compartiment $c \in C$.

L'ensemble de toutes les espèces d'un modèle M, noté $\textit{EPNs}(M)$, est défini par $\textit{EPNs}(M) = \textit{EPNs}(C)$.

On considère deux compartiments c_1 et c_2 tels que c_1 est inclus dans c_2, noté $c_1 \subset c_2$. Comme évoqué précédemment, une réaction de transport est une réaction qui transporte des espèces moléculaires entre deux compartiments différents. Les substrats d'une réaction de transport appartiennent à l'un des compartiments et ses produits à l'autre et représentent le même ensemble d'espèces. Etant donnée une réaction de transport de c_1 vers c_2, notée $r_{c_1 \to c_2}$, du point de vue de c_1, cette réaction est sortante puisqu'elle produit, à partir de molécules présentes dans c_1, des molécules similaires dans un autre compartiment (i.e., c_2). Cette même réaction, du point de vue de c_2, est entrante puisqu'elle produit

des molécules dans c_2 à partir de molécules similaires présentes dans un autre compartiment (i.e., c_1). Formellement, $r_{c_1 \to c_2} = \{\textit{substrate}(\mathrm{g}, n, c_1), \textit{produce}(\mathrm{g}, n', c_2) | (g, c_1) \in E, n, n' \in \mathbb{N}^*\}$. On notera, $R_T(c_1, c_2)$ l'ensemble des réactions de transports entre les compartiments c_1 et c_2. Formellement, $R_T(c_1, c_2) = R_T(c_2, c_1) = \{r_{c_1 \to c_2}\} \cup \{r_{c_2 \to c_1}\}$.

2.2.3 Opérations d'inférence sur les modèles métaboliques

Dans cette section, nous allons décrire, dans le formalisme logique, les opérations qui permettent d'inférer un modèle métabolique à partir d'un modèle de référence. Rappelons, que l'inférence de modèles basée sur un modèle de référence annoté, utilise une représentation abstraite des réactions possibles entre les espèces moléculaires. Le processus infère des réactions, à la fois individuellement et en groupe, jusqu'à ce que le réseau de réactions soit suffisamment complet pour être autonome. Les opérations d'inférence peuvent être de deux grandes familles : celles s'exercant à une échelle locale et celles à une échelle plus globale. Les opérations d'inférence locale, comme le "gap filling", ajoutent une ou deux réactions à la fois. Tandis que les opérations d'inférence, telle que la construction de compartiment, peuvent ajouter des dizaines ou des centaines de réactions.

Une fois les étapes d'inférence terminées, le résultat est obtenu en nettoyant le modèle intermédiaire, en supprimant les réactions et les espèces ou compartiments du modèle de référence qui ne sont pas connectés à des réactions instanciées. Bien que l'utilisation d'un modèle de référence ne garantit pas que le modèle résultant soit complet, connecté ou fonctionnel, il garantit, cependant, que les réactions qui apparaissent dans le modèle sont compatibles avec les règles implicites dans le modèle de référence.

En se basant sur l'analyse de cas d'utilisation, nous pouvons identifier un ensemble d'opérations de petite à grande échelle sur les modèles que nous allons définir sur un modèle métabolique en utilisant la représentation logique définie précédemment.

Soit $M = (R, E, C)$ un modèle. Nous commençons par définir les opérations nécessaires pour ajouter des éléments au modèle ou en supprimer : i.e., espèces, réactions, et compartiments. Toute opération peut être définie par sa condition d'application et l'état du modèle résultant de son application.

Ajouter un ensemble d'espèces $S = \{(g_i, c_i) | 1 \leq i \leq |S|\}$ dans des compartiments du modèle M n'est possible que si M contient tous les compartiments où l'on souhaite ajouter les nouvelles espèces (i.e., $\{c_i | 1 \leq i \leq |S|\} \subseteq C$) et produit un nouveau modèle M' dont l'ensemble des espèces inclut S. Formellement, $\textit{EPNs}(M') = \textit{EPNs}(M) \cup S$.

Retirer un ensemble d'espèces $S = \{(g_i, c_i) | 1 \leq i \leq |S|\}$ dans des compartiments du modèle M n'est possible que si M contient tous les compartiments où l'on souhaite retirer les espèces (i.e., $\{c_i | 1 \leq i \leq |S|\} \subseteq C$) et si les espèces correspondantes ne sont ni consommées ni produites par l'ensemble des réactions du modèle (i.e. R) et produit un nouveau modèle M' dont l'ensemble résultant du procédé de suppression d'un ensemble S d'espèces a un ensemble des espèces qui ne contient pas cet ensemble S. Formellement, $\textit{EPNs}(M') = \textit{EPNs}(M) \setminus S$.

Ajouter un ensemble de réactions R_{new} au modèle M n'est possible que si toutes les espèces participant à ces réactions sont présentes dans l'ensemble des espèces du

modèle et produit un nouveau modèle M' incluant l'ensemble des réactions de R_{new}. Formellement, $R' = R \cup R_{new}$.

Enlever un ensemble de réactions R_{del} du modèle M induit la suppression des espèces participant uniquement à ces réactions mais n'est pas assujettie à une condition d'application autre que la présence des réactions à supprimer et produit un nouveau modèle M' où $R' = R \setminus R_{del}$ et $\mathit{EPNs}(M') = \mathit{EPNs}(R')$.

Etant donné un compartiment c_{out}, ajouter un compartiment c_{in} inclus dans c_{out} et contenant le sous-ensemble d'espèces $E_{toadd} \subseteq EPNs(c_{out})$ produit un nouveau modèle M' incluant ce nouveau compartiment, où les espèces de E_{toadd} ont été déplacées de c_{out} vers c_{in}, les réactions entre deux espèces de E_{toadd} ont été également déplacées de c_{out} vers c_{in}, des réactions de transport entrantes du point de vue de c_{in} ont été ajoutées pour toute réaction produisant une espèce maintenant présente dans c_{in} à partir d'une espèce toujours présente dans c_{out} et, symétriquement, des réactions de transport sortantes du point de vue de c_{in} ont été ajoutées pour toute réaction produisant une espèce toujours présente dans c_{out} à partir d'une espèce maintenant présente dans c_{in}. Formellement,

$$E_{c_{in}} = \{(g, c_{in}) | (g, c_{out}) \in E_{toadd}\},$$

$$R_{c_{in}} = \{r \in R | EPNs(r) \subseteq E_{toadd}\},$$

$$R'_{c_{out}} = R_{c_{out}} \setminus R_{c_{in}},$$

$$RT = \{r_{c_{out} \to c_{in}} | \{substrate(g_1, n, c_{out}), produce(g, n', c_{out})\} \subseteq r\}$$

$$\bigcup \{r_{c_{in} \to c_{out}} | \{substrate(g, n, c_{out}), produce(g_1, n', c_{out})\} \subseteq r\}$$

où $r \in R_{c_{out}}, g_1 \notin E_{toadd}, g \in E_{toadd}, r_{c_{out} \to c_{in}} = \{substrate(g, 1, c_{out}), produce(g, 1, c_{in})\}$ et $r_{c_{in} \to c_{out}} = \{substrate(g, 1, c_{in}), produce(g, 1, c_{out})\}$

$$R' = R \setminus R_{c_{out}} \bigcup R_{c_{in}} \bigcup R'_{c_{out}} \bigcup RT$$

$$C' = C \cup \{c_0\}$$

Etant donné un compartiment c_{out}, supprimer un compartiment c_{in} inclut dans c_{out} produit un nouveau modèle M' excluant ce compartiment, où les espèces de $E_{c_{in}}$ ont été déplacées de c_{in} vers c_{out}, les réactions de c_{in} ont été également déplacées vers c_{out}, les réactions de transport entre c_{in} et c_{out} ont été supprimées après avoir fusionné les espèces impliquées.

La Figure 2.13, (a) présente un modèle $M = \langle R, E, C \rangle$, où $R = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$, $E = \{a_{out}, b_{out}, d_{out}, e_{out}, f_{out}, p_{out}\}$ et $C = \{c_{out}\}$. L'ajout d'un compartiment (c_{in}) contenant le sous-ensemble des espèces $\mathrm{E}_{toadd} = \{a, e, p\}$, notées içi ($a_{out}$), ($e_{out}$) et ($p_{out}$) pour indiquer qu'ils se situent dans le compartiment (c_{out}), produit un nouveau modèle M' présenté en Figure 2.13 (b). Le nouveau modèle inclut le compartiment (c_{in}), où les espèces de l'ensemble E_{toadd} ont été déplacées de c_{out} vers c_{in}, notées (a_{in}), (e_{in}) et (p_{in}), ainsi que les réactions r_5, r_6 et r_7. Il est à noter la duplication des espèces a et e participant aux deux compartiments, et l'ajout de deux réactions de transport

r_{t1} et r_{t2}. Formellement, $M' = \{R', E', C'\}$, où $R' = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_{t1}, r_{t2}\}$, $E' = \{a_{out}, b_{out}, d_{out}, e_{out}, f_{out}, p_{out}, a_{in}, e_{in}, p_{in}\}$ et $C' = \{c_{out}, c_{in}\}$.

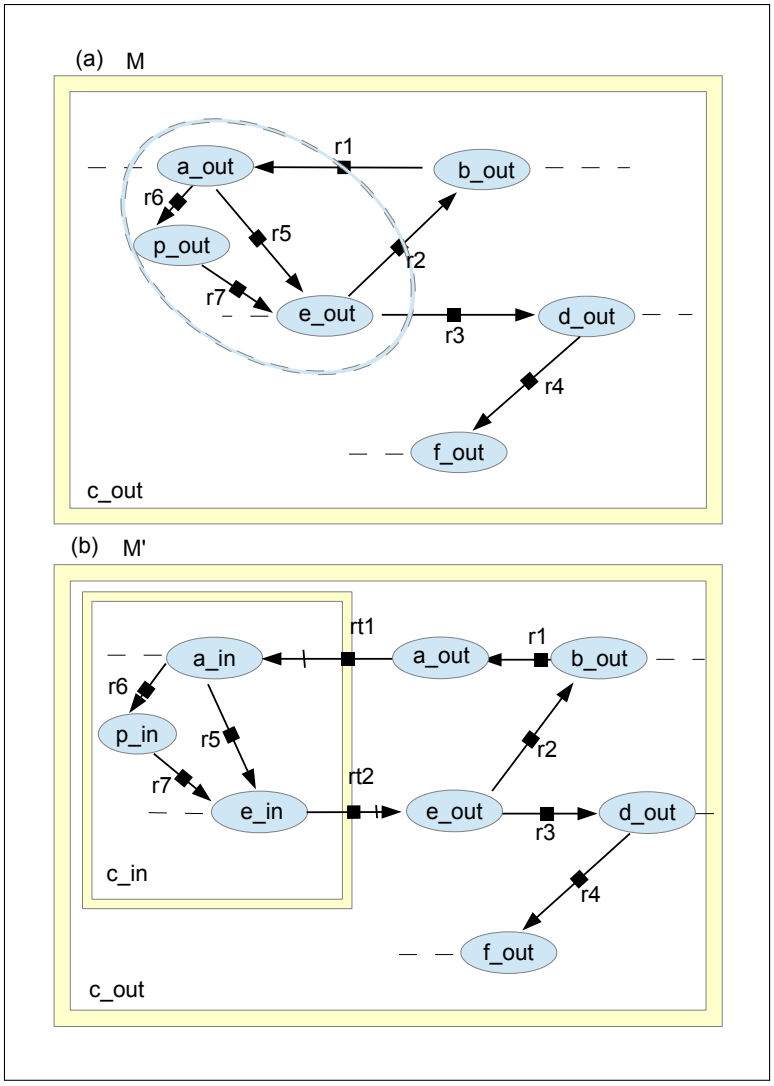


FIGURE 2.13 – Illustration de l'ajout d'un compartiment.

Cette représentation est symbolique du fait que l'on manipule les connaissances par la logique. La projection d'un modèle complet SBML (ou SBGN) dans notre notation est une simple transformation syntaxique.

Dans notre base de connaissances les faits sont de la forme :

— compartiment : compartiment `(C)`, où `C` est l'identifiant de ce compartiment.

— espèce : espèce `(ID,N,C)` où `ID` l'identifiant, `N` le nombre stoechiométrique et `C` le compartiment qui contient cet espèce.

— réaction : `r(C,P,E,M,GA)` où `C`,`P`, `E` et `M` sont respectivement les ensembles de substrats, de produits, d'enzymes et de modificateurs pour cette réaction, et enfin `AG` est l'association de gènes correspondante.

Notre formalisme permet également de représenter les " réactions régulées" (*regulated reactions*), introduites par Blavy *et al.* [BGL$^+$14]. Un formalisme est proposé pour unifier les informations sur les réactions (transformations biochimiques) et les effets de régulation (effet de la variation de la quantité d'une molécule sur la variation de la quantité d'une autre molécule). En d'autres termes les réactions biochimiques et les réactions de régulation ont été fusionnées, dans un graphe orienté.

$$G = \langle V, E, \lambda_V, \lambda_E \rangle$$

Où :

— $V = \textit{EPNs} \cup R'$ pour R' est un ensemble dénombrable fini où il existe une bijection $b : R \to R'$ qui relie chaque réaction $r \in R$ à un seul sommet $v(r) \in R'$.

— $E = \{(g, r) | r \in R \,\&\, g \in \textit{EPNs}(r) \,\&\, g$ not produce(g,n) $\in r\} \cup$
$\{(r, g) | r \in R \,\&$ produce(g,n) $\in r\}$

— $\lambda_V : V \to \{g, irr(r), rev(r)\}$

$$\lambda_V(v \in V) = \begin{cases} g & \text{if} \quad g \in \textit{EPNs} \\ rev(r) & \text{if} \quad b^{-1}(v) = r \,\&\, \rho(r) = T \\ irr(r) & \quad Otherwise \end{cases}$$

— λ_E : E $\to$ { substrat , produce , catalyzed, inhibited, modulated, stimulated }

où :

$$\lambda_E(e \in E) = \begin{cases} substrat & \text{if} \quad e = (g, r) \,\&\, substrat(g, n) \in r \\ produce & \text{if} \quad e = (r, g) \,\&\, produce(g, n) \in r \\ catalyzed & \text{if} \quad e = (g, r) \,\&\, catalyzed(g) \in r \\ inhibited & \text{if} \quad e = (g, r) \,\&\, inhibited(g) \in r \\ modulated & \text{if} \quad e = (g, r) \,\&\, modulated(g) \in r \\ stimulated & \text{if} \quad e = (g, r) \,\&\, stimulated(g) \in r \end{cases}$$

En supposant que la vitesse d'une réaction est croissante en fonction de la disponibilité de chacun des substrats et de la quantité de catalyseurs, décroissante en fonction de la quantité d'inhibiteurs et monotone en fonction de la quantité de modulateurs, chaque "réaction régulée" peut être modélisée sous la forme d'un graphe de causalité pour exprimer les séquences de changement dans la quantité de molécules ainsi que la vitesse de la réaction :

$$G_c = \langle V_c, E_c \rangle$$

où :

— $V_c =$ { *Rate*(r) | r $\in$ R } $\cup$
{ *Dispo*(e, r) | e $\in$ *EPNs* $\&$ substrat(e,n) $\in$ r $\forall$ r $\in$ R } $\cup$

{ *Quant*(e) | e ∈ *EPNs*(R) }

— E_c : l'ensemble des arêtes du graphe. Ces arêtes sont définies pour chaque type de réaction

1. $\forall r \in R$ telle que r soit une réaction irréversible sans régulateur :

   Definition 1 *Une réaction r sans régulateur est une réaction où* $\forall x \in r$, $x \in$ { *substrat(g,n),produce(g,n)* }

   (a) *Dispo*(s, r) $\xrightarrow{+}$ *Rate*(r) ; substrat(s,n) $\in r$.

   (b) *Rate*(r) $\xrightarrow{+}$ *Quant*(p) ; producep(g,n) $\in r$.

   (c) *Dispo*(s_1, r) $\xrightarrow{-}$ *Quant*(s2) pour $s_1, s_2 \in$*EPNs*(R)
   où $s_1 \neq s_2$ & substrat(s_1,n), substrat(s_2,m) $\in r$.

2. $\forall r \in R$ telle que r soit une réaction irréversible avec régulateur :

   Definition 2 *Une réaction avec régulateur est une réaction où* $\exists x \in r | x \in$ { *catalyzed(g), inhibited(g), modulated(g), stimulated(g)* }

   $\forall s \in$ *EPNs*(R) & substrat(s,n) $\in r$
   $\forall p \in$ *EPNs*(R) & produce(p,m) $\in r$

   (a) *Dispo*(s, r) $\xrightarrow{+}$ *Rate*(r)

   (b) *Rate*(r) $\xrightarrow{-}$ *Quant*(s)

   (c) *Rate*(r) $\xrightarrow{+}$ *Quant*(p)

   Et $\forall a, i, m \in$ *EPNs*| catalyzed(a),inhibited(i),modulated(m) $\in r$

   (a) *Quant*(a) $\xrightarrow{+}$ *Rate*(r)

   (b) *Quant*(i) $\xrightarrow{-}$ *Rate*(r)

   (c) *Quant*(m) $\xrightarrow{?}$ *Rate*(r)

3. $\forall r \in R$ telle que r soit une réaction réversible :
   $\forall s \in$ *EPNs*(R) & substrat(s,n) $\in r$
   $\forall p \in$ *EPNs*(R) & produce(p,m) $\in r$

   Soient r_1 la réaction directe et r_2 la réaction réversible :

   (a) *Dispo*(s, r) $\xrightarrow{+}$ *Rate*(r_1)

   (b) *Dispo*(p, r) $\xrightarrow{+}$ *Rate*(r_2)

   (c) *Rate*(r_1) $\xrightarrow{-}$ *Quant*(s)

(d) $\mathit{Rate}(r_2) \xrightarrow{-} \mathit{Quant}(\mathrm{p})$

(e) $\mathit{Rate}(r_1) \xrightarrow{+} \mathit{Quant}(\mathrm{p})$

(f) $\mathit{Rate}(r_2) \xrightarrow{+} \mathit{Quant}(\mathrm{s})$

$\forall o \in \mathit{EPNs}$ où catalyzed(o) or inhibited(o) or modulated(o) $\in r$

(a) $\mathit{Quant}(\mathrm{o}) \xrightarrow{?} \mathit{Rate}(r_1)$

(b) $\mathit{Quant}(\mathrm{o}) \xrightarrow{?} \mathit{Rate}(r_2)$

Nous voyons ainsi que la représentation logique que nous avons définie permet d'incorporer d'autres modèles du métabolisme que le réseau de réactions biochimiques, et de les intégrer dans une démarche d'inférence logique.

Chapitre 3

Inférence des associations de gènes

Les modèles complets d'organismes, qui contiennent les réactions enzymatiques mais aussi celles de la transcription, la modulation et la signalisation, n'existent que très rarement et pour quelques organismes modèles très étudiés. Au même temps, nous n'avons pas forcément besoin d'un modèle complet, pour la plupart des applications. Notre objectif dans ce chapitre est de définir un compromis, un modèle enzymatique annoté par connaissances supplémentaires, qui permet d'inférer de nouveaux modèles à partir de la comparaison de génomes. Précisément, à partir d'un modèle métabolique complet d'un organisme dit de référence, il s'agit de dériver un modèle minimal enzymatique explicite accompagné *d'associations de gènes*. En reprenant le schéma de la page 3, cette étape se résume à :

$$M_0 \longrightarrow \left\langle m_0, a_0^{\{G_0\}} \right\rangle$$

Les associations de gènes $a_0^{\{G_0\}}$, qui sont des formules booléennes comme nous le détaillerons par la suite, représentent des connaissances implicites sur le modèle métabolique complet et permettront de prendre de décisions pendant l'inférence des réactions du modèle métabolique de l'organisme cible. On peut décrire le modèle comme un ensemble de réactions ainsi qu'un ensemble d'associations de gènes par réaction. Ces données représente le modèle métabolique de notre organisme de référence.

La démarche que nous suivons dans ce chapitre est de considérer des exemples biologiques et des applications concrètes, afin de garantir que l'approche que nous définissons est valable. En considérant différents exemples réels, nous montrons que la représentation sous formule booléenne doit être différente en fonction de l'application. Cette étude permet de définir les associations de gènes adaptées pour chaque application.

Nous présentons d'abord un état de l'art des associations de gènes, utilisées dans d'autres méthodes. Nous considérons quatre applications et les notations logiques correspondants à notre démarche : l'inférence de modèles, l'analyse de flux à l'équilibre, la modélisation hiérarchique et enfin la simulation dynamique. Pour ancrer l'étude dans la réalité biologique nous traitons trois exemples qui ont chacun un défi particulier : le

système de la ribonucléotide réductase, où on doit tenir compte des phases du cycle cellulaire ; le système de ferrocytochrome-c, où on tenir compte des compartiments ; et finalement le système de la carboxylase du 3-methyl-2-oxopentanoate, où on doit tenir compte de la régulation des gènes. En confrontant chacun de ces exemples avec les quatre applications, nous définissons une stratégie de déduction qui projette $\langle m_0, a_0^{\{G_0\}} \rangle$ du modèle complet M_0.

3.1 Définitions et notations

La relation entre les gènes, les protéines et les réactions est généralement représentée en utilisant des règles logiques, appelées associations de gènes ou relations Gene-Protein-Reaction (GPR). Ces règles représentent ces relations en utilisant les opérateurs "*AND*" et "*OR* sur deux niveaux : des gènes aux protéines puis des protéines aux réactions. Au premier niveau, les règles expriment comment les protéines sont codées par leurs gènes, tandis qu'au second niveau, elles expriment les dépendances des réactions envers les enzymes (c.f. Figure 3.14). Identifier les associations GPR est primordiale pour la reconstruction du réseau métabolique [FP08, FST05, FHT$^+$09] et une des étapes les plus critiques de la reconstruction .

Les associations de gènes d'un réseau métabolique peuvent être identifiées en utilisant soit la méthode de curation manuelle, ce qui nécessite de nombreuses informations expérimentales primaires [FGAT09] (e.g UniProtDB [C$^+$11]), ou par méthodes automatisées. Ces dernières sont fondées sur les connaissances, et principalement basées sur la similarité de séquences [KDWV13, WLT05, AJW$^+$08]. En raison de la disponibilité limitée des données expérimentales, le développement de méthodes automatisées efficaces pour déchiffrer les annotations correctes est nécessaire [RCO$^+$13].

Les associations de gènes sont stockées dans de nombreuses bases de données publiques comme KEGG [KG00] ou MetaCyc [CFF$^+$06].

Les connaissances représentées dans le modèle par des associations de gènes sont indispensables pour permettre la prédiction du phénotype de la cellule dans des conditions génétiques comme les invalidations géniques (genes knockouts) dans les analyses de flux à l'équilibre FBA [JLP11] et pour tenir compte des effets de la régulation de gènes dans la simulation dynamique [KSM12].

Malheureusement il n'existe pas à ce jour de définition suffisamment rigoureuse des associations de gènes et par conséquent pas de moyen d'y associer une sémantique qui permettrait de démontrer leur utilisation correcte dans l'évaluation d'un réseau métabolique.

Dans ce chapitre, nous allons démontrer que différentes utilisations des associations de gènes peuvent conduire à différentes définitions de ces dernières. Nous identifions quatre utilisations complémentaires d'associations de gènes, et montrons que chacune requiert des assertions différentes concernant la réaction, et donc des associations de gènes de types différents : l'inférence de modèles, l'analyse de flux à l'équilibre, la modélisation hiérarchique et la simulation dynamique.

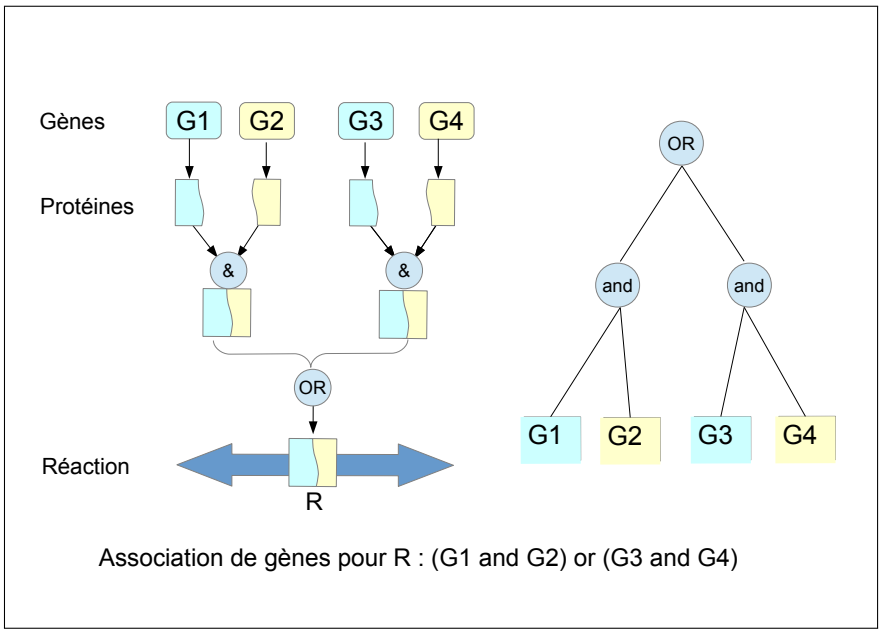


FIGURE 3.14 – (gauche) Une association de gènes peut être décrite comme une formule logique entre les gènes qui codent pour les protéines qui exécutent la réaction R. (droite) pour la réaction R, c'est un arbre booléen représentant la relation logique entre ces gènes.

L'inférence de modèles utilise un modèle de référence annoté et des homologies entre les gènes de l'organisme de référence et les gènes du nouvel organisme afin d'inférer un modèle pour ce dernier. L'assertion qui doit être évaluée pour chaque réaction du modèle de référence, pour être intégrée dans le nouveau modèle, est que le génome de l'organisme dit « cible » contient les gènes nécessaires (i.e., ceux dérivés des associations de gènes), par exemple tous ceux qui codent les enzymes catalysant la réaction.

L'analyse de flux à l'équilibre (FBA) calcule les flux à travers les réactions biochimiques qui maximisent ensemble une fonction objective, sous une supposition d'état stable [**?**]. Le modèle est représenté par un système d'équations linéaires. Une FBA peut être utilisée pour simuler l'effet des invalidations géniques (knockouts genes) sur les flux métaboliques. Afin de filtrer les équations linéaires correspondant à l'état stable, l'assertion qui doit être évaluée pour chaque réaction est qu'il faut enlever cette dernière si la fonction booléenne de son association de gènes est fausse (i.e. suite à la désactivation de certains de ses gènes nécessaires).

La modélisation hiérarchique décompose le système cellulaire complexe en une hiérarchie de systèmes de transition stochastiques modélisés indépendamment, qui communiquent par des flux de valeurs et qui peuvent effectuer des transitions synchrones d'états en réponse à des événements globaux [AGS+11]. On peut voir cette modélisation comme un ensemble de modélisations de parties du modèle d'origine. L'assertion qui doit être évaluée est que la *garde* Booléenne sur une transition, définie en termes des variables d'état telles que l'expression de protéines ou la concentration de métabolites, lui permet d'être exécutée.

La simulation dynamique convertit le modèle métabolique en un système d'équations différentielles qui définit l'évolution dans le temps de la concentration des molécules. Les associations de gène dans une simulation dynamique imposent des dépendances entre variables qui, ultimement, peuvent porter sur n'importe quelle espèce moléculaire. Dans ce cas général, où il n'existe pas des critères d'arrêt, les connaissances peuvent être inférées jusqu'à un nombre d'étapes de réécriture donné.

Pour chacune des utilisations décrites ci-dessus, nous définissons les associations de gènes appropriées et montrons comment l'on pourrait les dériver d'un modèle métabolique complet. Afin d'illustrer les différences entre les quatre utilisations et les associations de gènes correspondantes, nous illustrerons ce procédé sur trois exemples inspirés du modèle consensuel Yeastnet 4.31 [1] de *Saccharomyces cerevisiae* :

— Ribonuclease reductase, reaction r_{0008} ;
— Ferrocytochrome-C oxidoreductase, reaction r_{0469} ;
— 3-methyl-2-oxopentanoate decarboxylase, reaction r_{0067}.

Il est à noter, que dans certains cas, une lecture fine de la littérature a permis de dériver un réseau de réactions amélioré, qui ne correspond pas précisément à ce qui est écrit dans la version 4.31 de Yeastnet.

En montrant comment les associations de gènes appropriées peuvent être dérivées d'un modèle complet, nous fournissons les moyens de définir une norme importante pour l'annotation d'un modèle métabolique. Cependant, dans les applications pratiques, nous avons rarement un modèle complet pour lequel les associations de gènes pourraient être tirées. Plutôt, les associations de gènes sont définies en même temps que la définition ou l'inférence des réactions biochimiques dans dans le modèle. La question intéressante dans ces applications est de savoir comment améliorer la qualité des réactions et leurs annotations. Nous montrons comment la définition précise de l'utilisation des associations de gènes fournir un contrôle de cohérence sur le modèle étant défini.

Avant de présenter en détails notre démarche, nous introduisons rapidement les deux types de descriptions de modèle que nous utiliserons : à savoir, la description logique et BioRica. En effet, notre démarche commence par la mise en oeuvre manuelle d'une description logique ou BioRica de la représentation graphique du modèle.

Définissions tout d'abord la représentation logique employée dans notre démarche pour documenter un état de connaissances sur l'ensemble des réactions dans un modèle et des relations gène-protéine-réaction. Dans cette représentation, les gènes et les protéines sont appelées *atomes*, et les relations entre ces objets et des états du système des *clauses*. Par exemple, la clause **"A :B :- A, B"** veut dire que pour former la complexe **A :B** nous devons disposer des protéines **A** et **B** au préalable. On définit quelques règles d'inférence adaptées qui combinent et réécrivent les clauses et ainsi dérivent les formules dont on a besoin pour chaque utilisation.

Nous définissons un ensemble de notations pour les assertions suivantes dans notre représentation logique :

— $(\mathbf{A}, \mathbf{c})$ - la molécule **A** est présente dans le compartiment **c**

1. `http://comp-sys-bio.org/yeastnet/`

— **A :B** - un complexe macromoléculaire est formé de **A** et **B**
— **DEGRAD(A)** - la molécule **A** est dégradée par la cellule
— **DIS(A :B)** - le complexe **A :B** est dissocié par la cellule
— **OX(A)** - la molécule **A** est oxydée
— **RED(A)** - la molécule **A** est réduite
— **OX-RED(A)** - le cycle d'oxydation de la molécule **A** est présent

Cette représentation logique permet d'écrire des clauses du type **C, D** $\vdash$ **A, B** où **A** et **B** impliquent **C** et **D**, par exemple en raison d'une réaction **A + B** $\rightarrow$ **C + D**. Nous verrons, par la suite, que la formule désirée pour chaque utilisation peut être obtenue en réécrivant les clauses jusqu'à un critère d'arrêt, qui sera différent pour chaque type d'utilisation.

BioRica [AGS+11] est un cadre de modélisation de haut niveau qui intègre des comportements discrets, continus, stochastiques, non-déterministes et datés de manière non ambiguë, permettant la composition de modèles, de traduction à partir de modèles SBML, et les relations hiérarchiques. BioRica étend les idées de AltaRica [ABCC94] [APGR99] à des systèmes biologiques, permettant de résoudre des équations différentielles et de simuler des comportements stochastiques et non déterministes, avec la réutilisation et la combinaison de modèles existants.

Chaque noeud de BioRica est composé par la spécification de ses champs : *state, flow, sub, event, trans, assert, init* et *extern*. Dans le champ *state*, on déclare les variables d'état et leurs domaines. Dans le champ *flow*, on inclue les entrées depuis (ou sortie vers) d'autres noeuds de BioRica tandis que le champ *sub* liste d'autres noeuds que le noeud courant utilise (notion de hiérarchie). Dans le champ *event*, on déclare les noms des actions/évènements possibles, et les transitions provoquées par ces événements sont décrites dans le champ *trans*. Le champ *assert* permet d'imposer les relations entre les flots et/ou les états tandis que dans le champ *init*, on définit les valeurs initiales des variables d'état. Enfin, le champ *extern* permet l'inclusion de directives externes sur les distributions de retards d'évènements et la priorité entre les évènements.

3.2 Des stratégies pour dériver les associations de gènes

Nous proposons ci-après des stratégies de dérivation des associations de gènes pour les quatre cas d'utilisation présentés précédemment et basées sur un comportement commun : les règles de réécriture pour la substitution et la simplification sont appliquées du haut vers le bas jusqu'à ce qu'un critère d'arrêt soit atteint. C'est ce critère d'arrêt à utiliser qui dépend de la nature de l'application en cours d'examen. Dans le cas de l'inférence de modèle, pour déterminer si une réaction peut avoir lieu dans un organisme, il suffit de savoir si les catalyseurs sont présents ou pas. Ainsi, la réécriture peut s'arrêter dès que l'on observe les gènes qui codent pour les protéines catalysant cette réaction. L'analyse de flux à l'équilibre suppose, quant à elle, qu'une réaction peut porter tout flux physiologiquement possible, tant que les gènes nécessaires ont été exprimés. A l'état

d'équilibre, qui est supposé dans l'analyse de flux à l'équilibre, la réécriture des associations de gènes peut s'arrêter lorsque toute substitution correspond à un changement d'état (e.g. comme l'expression des gènes). Dans le cas de la modélisation hiérarchique, les noeuds de la représentation BioRica correspondant au modèle définissent une limite naturelle de la connaissance, donc, la réécriture doit s'arrêter au contenu du noeud (qui peut être identifié syntaxiquement depuis le modèle BioRica). Finalement, pour la simulation dynamique, vu que toute réaction peut contribuer à la simulation, la réécriture doit se poursuivre aussi longtemps que les clauses peuvent être réécrites et ne permet donc pas d'éviter de considérer l'ensemble du modèle. Dans ce dernier cas, nous ne fournirons donc pas d'illustration sur les trois exemples considérés.

Les trois exemples à suivre ont été choisis pour illustrer le processus de dérivation d'associations de gènes dans les réseaux métaboliques. Chaque exemple s'amorce sur une réaction particulière, mais peut inclure jusqu'à 15 réactions associées, nécessaires pour comprendre les conditions préalables de la réaction particulière :

— Le système de la ribonucléotide réductase **RNR** par **Trx1p**, précisément la réaction catalysée par le bisulfite **Trx1p** (r_8 dans Yeastnet)
— Le complexe **Cyc7p :Ccp1p** qui catalyse la transformation de ferricytochrome-c en ferrocytochrome-c (r_{0469} dans Yeastnet).
— Le complexe du facteur de transcription **Thi3p :Pdc2p** ou **Thi3p :Thi2p** qui induit les quatre paralogues de **Pcd1p**, qui catalysent la transformation de 3-méthyl-2-oxopentanoate en 2-méthyl butanol (r_{0067} dans Yeastnet)

3.3 Système de la ribonucléotide réductase (RNR)

La ribonucléotide réductase **(RNR)** convertit des ribonucléotides en deoxiribonucléotides pendant la phase **S** du cycle cellulaire. Rappelons que le cycle cellulaire correspond à l'ensemble des étapes constituant et délimitant la vie d'une cellule et est composé de plusieurs phases de croissance : la phase **Mitose/Méiose** où la cellule se divise, la phase **Gap1** où la cellule grossit jusqu'à atteindre une taille critique qui va donner le signal pour passer à la phase **S** où l'ADN est répliqué avant de passer à la phase **Gap2**. Pendant les phases **Gap1**, **Gap2** et **Mitose/Méiose**, où la réduction de ribonucléotides n'est pas nécessaire, le complexe hétérodimère **Rnr2p :Rnr4p** est importé dans le noyau sous l'action de la protéine d'adressage **Dif1p**, qui amène le complexe **Rnr2p :Rnr4p** à proximité du noyau, puis de son import dans le noyau via l'action de la protéine **Kap122p**. Le complexe **Rnr2p :Rnr4p** est retenu dans le noyau par la protéine d'ancrage **Wtm1p**.

Pendant la phase **S** du cycle cellulaire, le complexe **Rnr2p :Rnr4p** se déplace du noyau vers le cytoplasme, suite à deux mécanismes coopérants :

— La dégradation de **Dif1p** qui prévient l'entrée dans le noyau de nouveaux complexes **Rnr2p :Rnr4p**.
— L'action de la protéine d'export **Crm1p**, qui transporte le complexe **Rnr2p :Rnr4p** du noyau vers le cytoplasme.

Le complexe **Rnr2p :Rnr4p** s'associe avec le complexe homodimère **Rnr1p :Rnr1p**

pour former le nouveau complexe **Rnr1p :Rnr1p :Rnr2p :Rnr4p**, qui est ensuite réduit par l'un ou l'autre des isozymes de thiorédoxine **Trx1p** ou **Trx2p**. Ces protéines sont codées respectivement par les gènes **TRX1** et **TRX2**.

Le complexe réduit **Rnr1p :Rnr1p :Rnr2p :Rnr4p** catalyse à son tour la réduction du diphosphate ribonucléosidique **(NDP)** en diphosphate déoxyribonucléoside **(dNDP)** et il s'oxyde ainsi. La représentation graphique du modèle utilisé est présentée en Figure 3.15. A partir de cette représentation, nous avons déterminé une représentation logique et une description BioRica correspondant à la réaction $\mathrm{NDP} \wedge \mathrm{O_2} \wedge 2\mathrm{H_2} \rightarrow \mathrm{dNDP} \wedge 2\mathrm{H_2O}$.

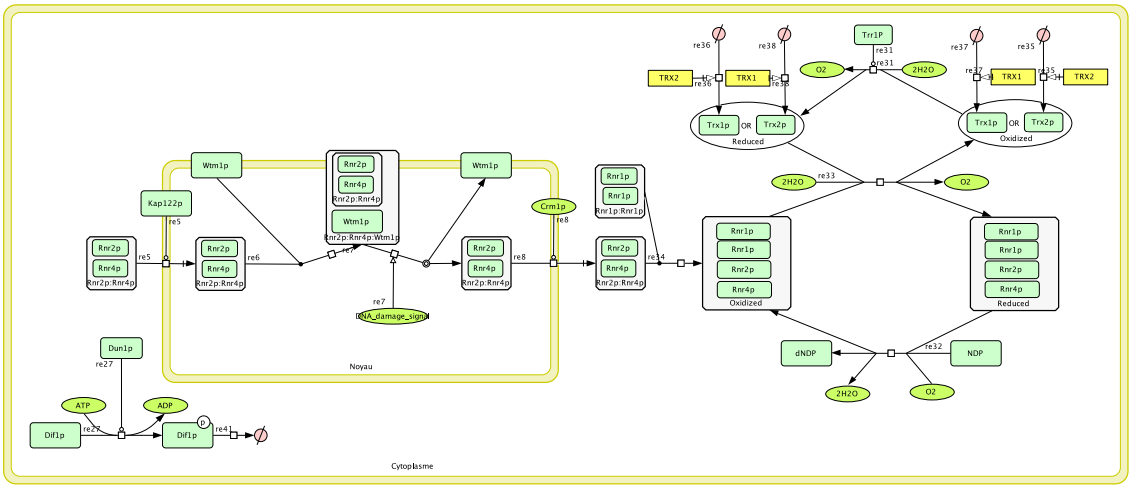


FIGURE 3.15 – Modèle explicite pour la réaction de la ribonucléotide réductase (RNR)

3.3.1 Description logique

$$\mathrm{NDP} \wedge \mathrm{O_2} \wedge \mathrm{2H_2} \rightarrow \mathrm{dNDP} \wedge \mathrm{2H_2O} \vdash \text{OX-RED}(\mathrm{Rnr1p}:\mathrm{Rnr1p}:\mathrm{Rnr2p}:\mathrm{Rnr4p}) \quad (3.3)$$

$$\mathrm{dNDP} \vdash \mathrm{NDP} \wedge \text{OX-RED}(\mathrm{Rnr1p}:\mathrm{Rnr1p}:\mathrm{Rnr2p}:\mathrm{Rnr4p}) \quad (3.4)$$

$$\text{OX-RED}(\mathrm{Rnr1p}:\mathrm{Rnr1p}:\mathrm{Rnr2p}:\mathrm{Rnr4p}) \vdash \mathrm{Rnr1p}:\mathrm{Rnr1p}:\mathrm{Rnr2p}:\mathrm{Rnr4p} \wedge (\text{OX-RED}(\mathrm{Trx1p}) \vee \text{OX-RED}(\mathrm{Trx2p})) \quad (3.5)$$

$$\text{OX-RED}(\mathrm{Trx1p}) \vee \text{OX-RED}(\mathrm{Trx2p}) \vdash (\mathrm{Trx1p} \vee \mathrm{Trx2p}) \wedge \mathrm{Trr1p} \quad (3.6)$$

$$\mathrm{Rnr1p}:\mathrm{Rnr1p}:\mathrm{Rnr2p}:\mathrm{Rnr4p} \vdash \mathrm{Rnr1p}:\mathrm{Rnr1p} \wedge \mathrm{Rnr2p}:\mathrm{Rnr4p} \quad (3.7)$$

$$\mathrm{Rnr1p}:\mathrm{Rnr1p} \vdash \mathrm{Rnr1p} \wedge \mathrm{Rnr1p} \quad (3.8)$$

$$\mathrm{Rnr2p}:\mathrm{Rnr4p} \vdash \mathrm{Rnr2p} \wedge \mathrm{Rnr4p} \quad (3.9)$$

$$(\mathrm{Rnr1p}:\mathrm{Rnr1p}:\mathrm{Rnr2p}:\mathrm{Rnr4p}, \mathrm{c}) \vdash (\mathrm{Rnr1p}:\mathrm{Rnr1p}, \mathrm{c}) \wedge (\mathrm{Rnr2p}:\mathrm{Rnr4p}, \mathrm{c}) \quad (3.10)$$

$$(\mathrm{Rnr1p}:\mathrm{Rnr1p}, \mathrm{c}) \vdash (\mathrm{Rnr1p}:\mathrm{Rnr1p}, \mathrm{n}) \wedge \mathrm{Crm1p} \wedge \mathrm{Phase(S)} \quad (3.11)$$

$$(\mathrm{Rnr1p}:\mathrm{Rnr1p}, \mathrm{n}) \vdash (\mathrm{Rnr1p}:\mathrm{Rnr1p}, \mathrm{c}) \wedge \mathrm{Kap122p} \wedge \mathrm{Dif1p} \wedge \neg\mathrm{Phase(S)} \quad (3.12)$$

$$\mathrm{Dif1p} \vdash \neg\mathrm{Dun1p} \wedge \neg\mathrm{Phase(S)} \quad (3.13)$$

$$\mathrm{Rnr2p}:\mathrm{Rnr4p}:\mathrm{Wrm1p} \vdash \mathrm{Rnr2p} \wedge \mathrm{Rnr4p} \wedge \mathrm{Wrm1p} \quad (3.14)$$

$$\mathrm{Crm1p} \vdash \mathrm{CRM1} \quad (3.15)$$

(3.16)

$$\mathrm{Dif1p} \vdash \mathrm{DIF1} \quad (3.17)$$

$$\mathrm{Dun1p} \vdash \mathrm{DUN1} \quad (3.18)$$

$$\mathrm{Kap122p} \vdash \mathrm{KAP122} \quad (3.19)$$

$$\mathrm{Rnr1p} \vdash \mathrm{RNR1} \quad (3.20)$$

$$\mathrm{Rnr2p} \vdash \mathrm{RNR2} \quad (3.21)$$

$$\mathrm{Rnr4p} \vdash \mathrm{RNR4} \quad (3.22)$$

$$\mathrm{Trr1p} \vdash \mathrm{TRR1} \quad (3.23)$$

$$\mathrm{Trx1p} \vdash \mathrm{TRX1} \quad (3.24)$$

$$\mathrm{Trx2p} \vdash \mathrm{TRX2} \quad (3.25)$$

$$\mathrm{Wrm1p} \vdash \mathrm{WRM1} \quad (3.26)$$

3.3.2 Description BioRica

Le modèle hiérarchique du système RNR est composé de trois sous-noeuds :

— **Subcomplex** qui décrit la formation de l'hétérodimère **Rnr2p :Rnr4p** et son importation dans le noyau. On suppose que la formation du complexe est la limitation de la réaction ; les cinétiques des réactions d'adressage et de l'importation sont supposées être beaucoup plus rapides, et elles ne sont pas modélisées explicitement.
— **Complex** qui décrit la formation au cours de la phase **S** de l'hétérotétramère **Rnr1p :Rnr1p :Rnr2p :Rnr4p** dans le cytoplasme. On suppose que cette formation est seulement limitée par la présence de **Rnr2p :Rnr4p** dans le cytoplasme, qui est à son tour contrôlée par la transition vers la phase **S** ; le transporteur **Crm1** n'est pas modélisé explicitement.
— **Reaction** qui décrit le système de réactions autour de la réaction de la ribonucléotide réductase. Cela inclut les cycles de réactions réduction-oxydation de réductase hétérotétramère et les thiorédoxines **Trx1** et **Trx2** à la fois.

Le noeud **Main** relie les trois sous-noeuds par des variables booléennes indiquant la présence dans le cytoplasme, de l'hétérodimère **Rnr2p :Rnr4p** et de l'hétérotéramère **Rnr1p :Rnr1p :Rnr2p :Rnr4p**, respectivement. Il synchronise également les transitions vers la phase **S** par les sous-noeuds **Subcomplex** et **Complex**. La description BioRica complète finale est la suivante.

node Subcomplex

state
$\mathrm{T}_c, \mathrm{T}_n$: $\{0,1\}$;
ε : $\{0,1\}$;
flow
(Rnr2p :Rnr4p, c) : FLOAT ;
init
(Rnr2p :Rnr4p, c_i) ;
diff
$d(\text{Rnr2p :Rnr4p, n}) = \mathrm{T}_n.(\text{cell}.V1.(1+1.\ln(\frac{(\text{Rnr2p :Rnr4p,c})}{ic(\text{Rnr2p :Rnr4p,c})})-\ln(\frac{(\text{Rnr2p :Rnr4p,n})}{ic((\text{Rnr2p :Rnr4p,n})})))$
$+\varepsilon(\text{Rnr2p :Rnr4p}, c_i)$;
event
phaseS ;
sortieS ;
trans
true $\vdash$ phaseS $\rightarrow$ $\mathrm{T}_n := 0$, $\mathrm{T}_c := 1$, $\varepsilon := 1$;
true $\vdash$ sortieS $\rightarrow$ $\mathrm{T}_n := 1$, $\mathrm{T}_c := 0$, $\varepsilon := 0$;
edon

node Complex

state
(Rnr1p :Rnr1p, c) = BOOL ;
flow

(Rnr2p :Rnr4p, c) = BOOL ;
(Rnr1p :Rnr1p :Rnr2p :Rnr4p, c) = BOOL ;

event

phaseS ;
associate ;

trans

true $\vdash$ phaseS $\rightarrow$ (Rnr2p :Rnr4p, c) := true ;
(Rnr2p :Rnr4p, c), (Rnr1p :Rnr1p, c) $\vdash$ associate $\rightarrow$ (Rnr1p :Rnr1p :Rnr2p :Rnr4p, c) := true ;

edon

node Reaction

state

TRX1 : BOOL ;
TRX2 : BOOL ;
Trx1p : BOOL ;
Trx2p : BOOL ;
Trr1p : BOOL ;
OX(Trx1p or Trx2p) = BOOL ;
D : $\{0, 1\}$;
NDP : FLOAT ;
dNDP : FLOAT ;
dNDP : FLOAT ;
O_2 : FLOAT ;
H_2O : FLOAT ;
OX(Rnr1p :Rnr1p :Rnr2p :Rnr4p, c) : BOOL ;

flow

(Rnr1p :Rnr1p :Rnr2p :Rnr4p, c) : BOOL ;
OX(Rnr1p :Rnr1p :Rnr2p :Rnr4p, c) : BOOL ;

event

OX(TRX) ;
OX(RNR) ;
G-expr(TRX1) ;
G-expr(TRX2) ;
active ;

constants

iC(Ndp) ;
iC(dNdp) ;
iC(O_2) ;
iC(H_2O) ;

diff

$$\text{d(dNDP)} = \text{D.(cell .V0.}(1 + 1.\ln(\tfrac{C(NDP)}{ic(NDP)}) + \ln(\tfrac{C(O_2)}{ic(O_2)}) - 2\ln(\tfrac{C(H_2O)}{ic(H_2O)}) - \ln(\tfrac{C(dNDP)}{ic(dNDP)})));$$

trans

```
                                                TRX1 ⊢   G-expr(TRX1) →          Trx1p := true ;
                                                TRX2 ⊢   G-expr(TRX2) →          Trx2p := true ;
                                                Trr1p ⊢      OX(TRX) →   OX(Trx1p | Trx2p) := true ;
(Rnr1p :Rnr1p :Rnr2p :Rnr4p, c),OX(Trx1p | Trx2p) ⊢      OX(RNR) →
        OX(Rnr1p :Rnr1p :Rnr2p :Rnr4p, c) := true ;
                OX(Rnr1p :Rnr1p :Rnr2p :Rnr4p, c) ⊢      active →          D = 1 ;
edon

node Main

    sub
        N   :   Subcomplex ;
        C   :   Complex ;
        R   :   Reaction ;
    assert
        C.(Rnr1p :Rnr1p :Rnr2p :Rnr4p, c)   =   R.(Rnr1p :Rnr1p :Rnr2p :Rnr4p, c) ;
                N.(Rnr2p :Rnr4p, c)         =   C.(Rnr2p :Rnr4p, c) ;
    sync
        <N.phaseS, C.phaseS> ;
edon
```

3.3.3 L'association de gènes pour l'inférence de modèle

La réaction de ribonucléotide de réductase peut se produire dans l'organisme cible si il y a assez d'évidences pour réécrire son association de gènes. Cette réaction est catalysée par le cycle d'oxydation du complexe **Rnr1p :Rnr1p :Rnr2p :Rnr4p**, donc pour instancier la réaction nous avons besoin d'assurer que les gènes codants les protéines participant à ce complexe sont présents dans l'organisme cible (Figure 3.16).

Formellement, l'association de gènes de la réaction de « ribonucléotide de réductase » est donnée par RNR1 ∧ RNR2 ∧ RNR4. Formellement, comme indiqué précédemment, cette dernière peut être dérivée depuis la présence du cycle d'oxydation du complexe **Rnr1p :Rnr1p :Rnr2p :Rnr4p** :

OX-RED(Rnr1p :Rnr1p :Rnr2p :Rnr4p)

⊢	Rnr1p :Rnr1p :Rnr2p :Rnr4p ∧ (OX-RED(Trx1p) ∨ OX-RED(Trx2p))	eq(3.5)
⊢	Rnr1p :Rnr1p ∧ Rnr2p :Rnr4p	eq(3.7)
⊢	Rnr1p ∧ Rnr2p ∧ Rnr4p	eqs(3.8) (3.9)
⊢	RNR1 ∧ RNR2 ∧ RNR4	eqs(3.20) (3.21) (3.22)

3.3.4 L'association de gènes pour l'analyse de flux à l'équilibre

L'analyse du flux à l'équilibre prend en compte les flux au travers des réactions, en considérant l'hypothèse selon laquelle aucun changement d'état ne se produit. Par conséquent, on ne peut pas considérer les règles qui conduisent à l'expression de gènes (e.g.

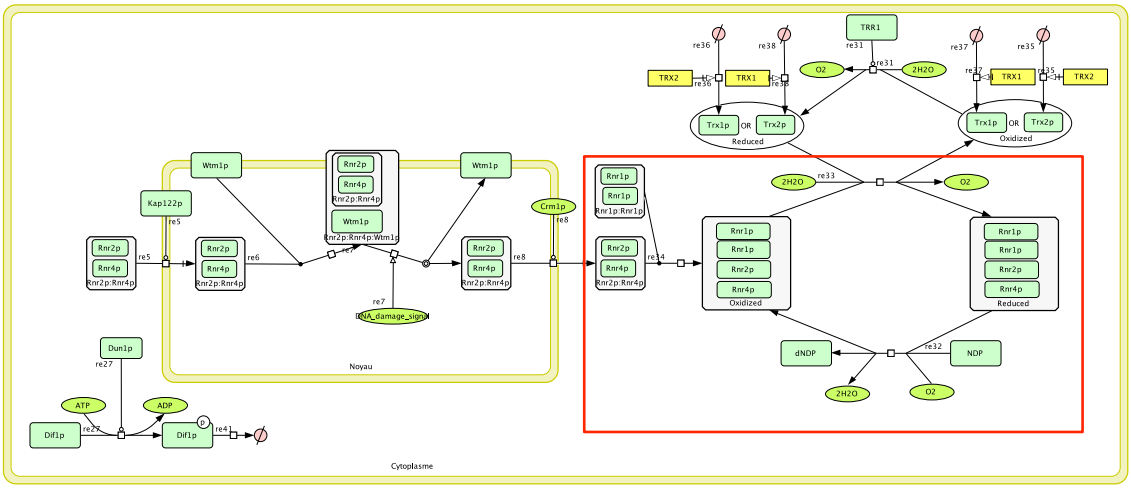


FIGURE 3.16 – Modèle explicite pour la réaction de la ribonucléotide réductase, le rectangle rouge représente la partie considérée par l'application d'inférence du système RNR.

eq(3.24)) ou à la fusion/fission de complexes (e.g. eq(3.7)). Dans ce cas précis, nous supposons que les enzymes et les complexes enzymatiques sont présents, et nous ne nous préoccupons que des cycles de réactions de réduction (Figure 3.17).

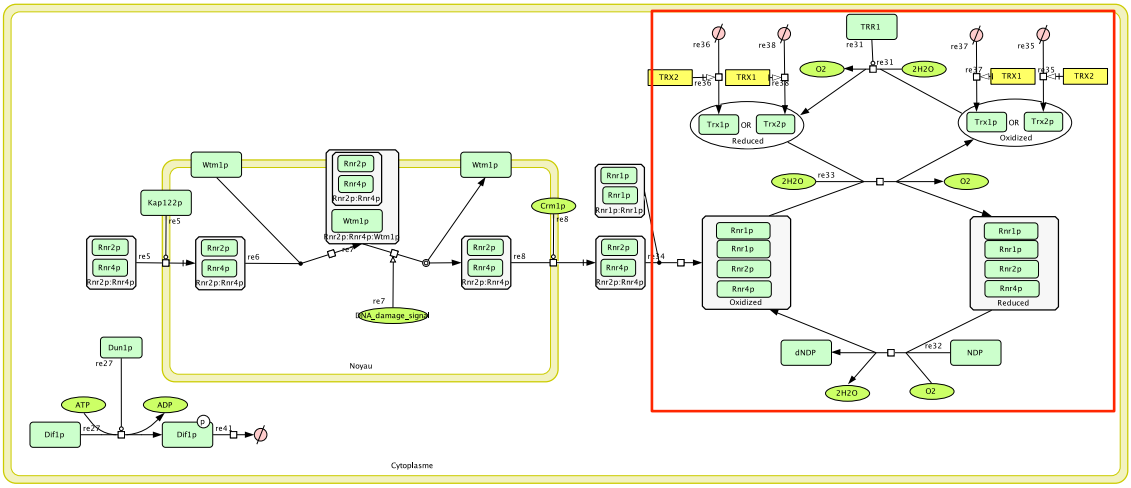


FIGURE 3.17 – Modèle explicite pour la réaction de la ribonucléotide réductase, la partie considérée par l'analyse de flux à l'équilibre du système RNR est représentée dans le rectangle rouge.

Formellement, l'association de gènes de la réaction de « ribonucléotide de réductase

» est donnée par (Trx1p ∨ Trx2p) ∧ Trr1p. Formellement, comme indiqué précédemment, cette dernière peut être dérivée depuis la présence du cycle d'oxydation du complexe **Rnr1p :Rnr1p :Rnr2p :Rnr4p** en utilisant que des règles n'impliquant pas une expression de gènes ou une fusion de complexe :

OX-RED(Rnr1p :Rnr1p :Rnr2p :Rnr4p)
⊢ Rnr1p :Rnr1p :Rnr2p :Rnr4p ∧ OX-RED(Trx1p) ∨ OX-RED(Trx2p) eq(3.5)
⊢ Rnr1p :Rnr1p :Rnr2p :Rnr4p ∧ (Trx1p ∨ Trx2p) ∧ Trr1p eq(3.6)

3.3.5 L'association de gènes pour la modélisation hiérarchique

La réduction de la ribonucléotide en désoxyribonucléotide est effectuée par le complexe de reductase hétérotétramérique **Rnr1p :Rnr1p :Rnr2p :Rnr4p**, qui est formé par l'association de l'homodimère **Rnr1p :Rnr1p** avec l'hétérodimère **Rnr2p :Rnr4p** libéré du noyau. Le complexe de la réductase est activé par la thiorédoxine **Trx1p** ou **Trx2p**, qui est à son tour actionné par la thiorédoxine réductase **Trr1p**. Ces réactions se produisent en même temps : chaque réduction de ribonucléotide oxyde le complexe de reductase, qui est réduit par la thiorédoxine, lui-même réduit par la thiorédoxine réductase. Donc, pour décrire ces dépendances, nous avons besoin de gènes codant le complexe de reductase hétérotétramérique **Rnr1p :Rnr1p :Rnr2p :Rnr4p**, la thiorédoxine qui effectue la réduction, et la thiorédoxine réductase (Figure 3.18).

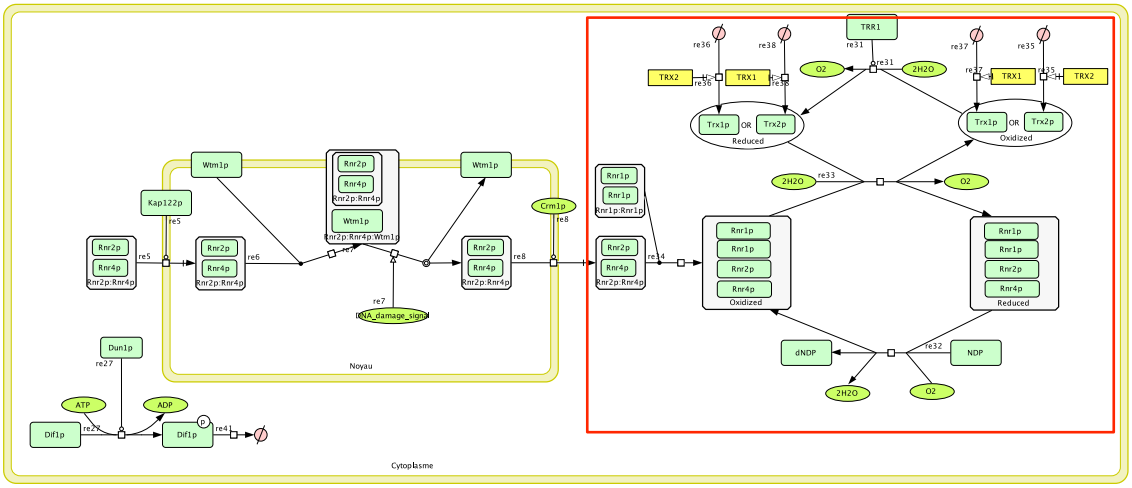


FIGURE 3.18 – Modèle explicite pour la réaction de la ribonucléotide réductase, le rectangle rouge représente la partie modélisée par l'application hierarchique du système RNR.

Rnr1p :Rnr1p :Rnr2p :Rnr4p ∧ (Trx1p ∨ Trx2p) ∧ Trr1p

$\vdash$ Rnr1p :Rnr1p $\wedge$ Rnr2p :Rnr4p $\wedge$ (Trx1p $\vee$ Trx2p) $\wedge$ Trr1p eqs(3.6)(3.7)
$\vdash$ Rnr1p $\wedge$ Rnr2p $\wedge$ Rnr4p $\wedge$ (Trx1p $\vee$ Trx2p) $\wedge$ Trr1p eqs(3.8)(3.9)
$\vdash$ RNR1 $\wedge$ RNR2 $\wedge$ RNR4 $\wedge$ (TRX1 $\vee$ TRX2) $\wedge$ TRR1 eqs(3.20) (3.21) (3.22) (3.24) (3.25) (3.23)

3.4 Système de Ferrocytochrome-c (CYC)

Les Cytochrome-c isoformes **Cyc1p** et **Cyc7p** sont des protéines de transport d'électrons dans la chaîne de transport d'électrons mitochondriale. Dans ce système, la protéine **Cyc1p** ou **Cyc7p** forme d'abord un complexe avec Cytochrome-c peroxidase **Ccp1p**, et dans ce complexe, **Ccp1p** catalyse l'oxydation du 2-ferrocytochrome-c, qui est la forme réduite de **Cyc1p** or **Cyc7p** (Figure 3.19).

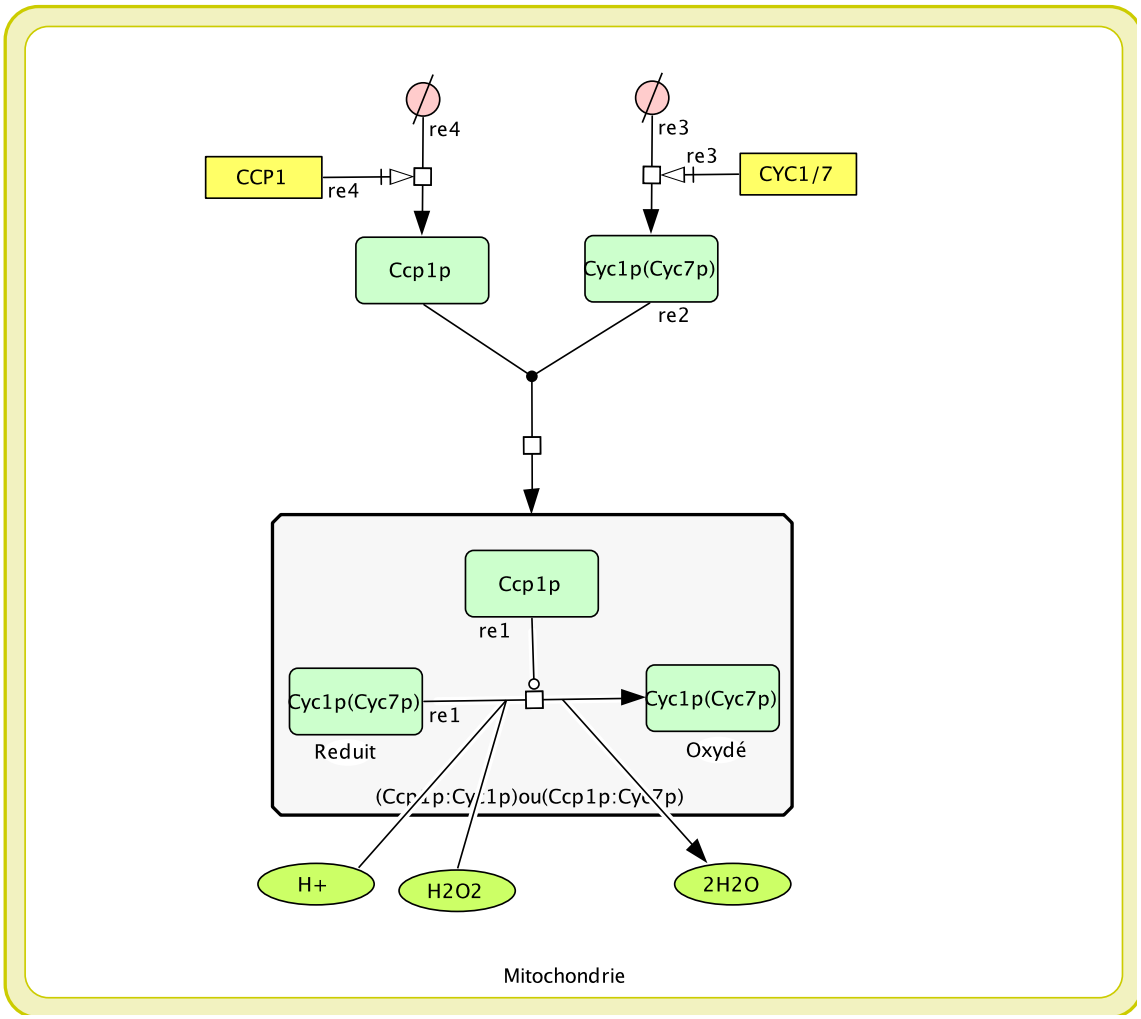


FIGURE 3.19 – Modèle explicite de la réaction de ferrocytochrome-c

3.4.1 Description logique

$$R = \text{RED}(\text{Cyc1p}) \wedge \text{H}^+ \wedge \text{H}_2\text{O}_2 \rightarrow \text{OX}(\text{Cyc1p}) \wedge 2\text{H}_2\text{O}$$

$$\begin{aligned} \text{R} &\vdash \text{Ccp1p} \wedge \text{Cyc1p :Ccp1p} && (3.27)\\ \text{Cyc1p :Ccp1p} &\vdash \text{Cyc1p} \wedge \text{Ccp1p} && (3.28)\\ \text{Cyc1p} &\vdash \text{CYC1} && (3.29)\\ \text{Ccp1p} &\vdash \text{CCP1} && (3.30) \end{aligned}$$

$$R' = \text{RED}(\text{Cyc7p}) \wedge \text{H}^+ \wedge \text{H}_2\text{O}_2 \rightarrow \text{OX}(\text{Cyc7p}) \wedge 2\text{H}_2\text{O}$$

$$\begin{aligned} \text{R'} &\vdash \text{Ccp7p} \wedge \text{Cyc7p :Ccp1p} && (3.31)\\ \text{Cyc7p :Ccp1p} &\vdash \text{Cyc7p} \wedge \text{Ccp1p} && (3.32)\\ \text{Cyc7p} &\vdash \text{CYC7} && (3.33)\\ \text{Ccp1p} &\vdash \text{CCP1} && (3.34) \end{aligned}$$

3.4.2 Description BioRica

Le modèle hiérarchique du système **CYC** est constitué d'un seul noeud hybride **Main**, composé de transitions discrètes de l'expression des gènes et de la formation du complexe **Ccp1p :Cytochrome-c**, et d'une seule équation continue pour l'oxydation du Cytochrome-c par **Ccp1** une fois que le complexe est formé.

node Main

state

CCP1 : BOOL ;
CYC1 : BOOL ;
CYC7 : BOOL ;
Ccp1p : BOOL ;
Cyc1p : BOOL ;
Cyc7p : BOOL ;
Ccp1p :(Cyc1p|Cyc7p) : BOOL ;
R_0 = $\{0, 1\}$;
(Cyc1p|Cyc7p)$_O$: FLOAT ;
(Cyc1p|Cyc7p)$_R$: FLOAT ;

diff

$d((Cyc1p|Cyc7p)_O) = R_0(\text{cell}.V_0.(1 + 1.\ln(\frac{(Cyc1p|Cyc7p)_R}{ic(Cyc1p|Cyc7p)_R}) + \ln(\frac{H^+}{ic(H^+)})$
$+ \ln(\frac{H_2O_2}{ic(H_2O_2)}) - \ln(\frac{(Cyc1p|Cyc7p)_O}{ic(Cyc1p|Cyc7p)_O}) - \ln(\frac{H_2O}{ic(H_2O)})))$;

event

G-expr(CCP1) ; G-expr(CYC1) ; G-expr(CYC7) ;

associateCOM ; Oxydate ;

trans

CCP1	⊢	G-expr(CCP1)	→	Ccp1p := true ;
CYC1	⊢	G-expr(CYC1)	→	Cyc1p := true ;
CYC7	⊢	G-expr(CYC7)	→	Cyc7p := true ;
Ccp1p, (Cyc1p | Cyc7p)	⊢	associateCOM	→	Ccp1p :(Cyc1p | Cyc7p) := true ;
Ccp1 :(Cyc1p | Cycp7)	⊢	Oxydate	→	R_0 := 1 ;

edon

3.4.3 L'association de gènes pour l'inférence du modèle

Les gènes nécessaires pour que cette réaction puisse avoir lieu dans l'organisme cible sont les gènes codant les protéines du catalyseur **Cc1p** et du complexe **Cyc1p|Cyc7p :Ccp1p** (Figure 3.20)

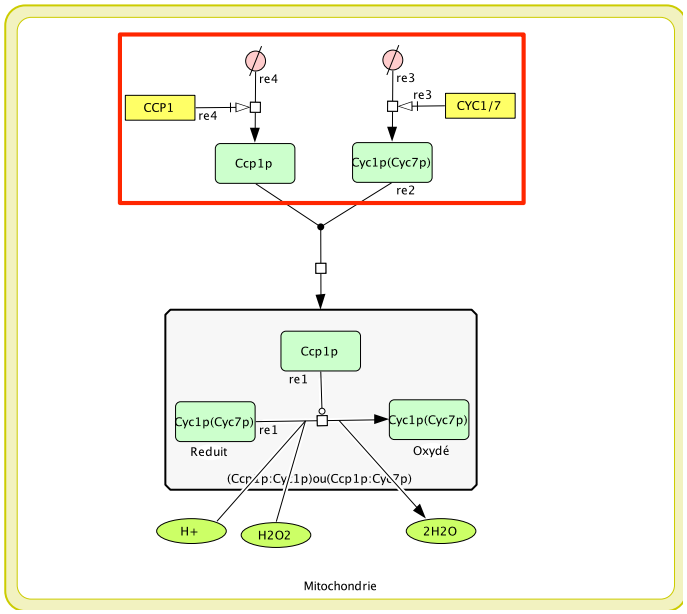


FIGURE 3.20 – Modèle explicite pour le système de Ferrocytochrome-c ,la partie considérée par l'inférence du système CYC est représentée dans le rectangle rouge.

R	⊢	Ccp1p ∧ Cyc1p :Ccp1p	eq(3.27)
	⊢	Ccp1p ∧ Cyc1p	eq(3.28)
	⊢	CCP1 ∧ CYC1	eqs(3.30) (3.29)
R'	⊢	Ccp1p ∧ Cyc7p :Ccp1p	eq(3.31)
	⊢	Ccp1p ∧ Cyc7p	eq(3.32)
	⊢	CCP1 ∧ CYC7	eqs(3.34) (3.33)

3.4.4 L'association de gènes pour l'analyse de flux à l'équilibre

Dans ce cas aucun changement d'état ne se produit, donc la réaction sera instanciée si l'enzyme et le complexe sont déjà présents (Figure 3.21).

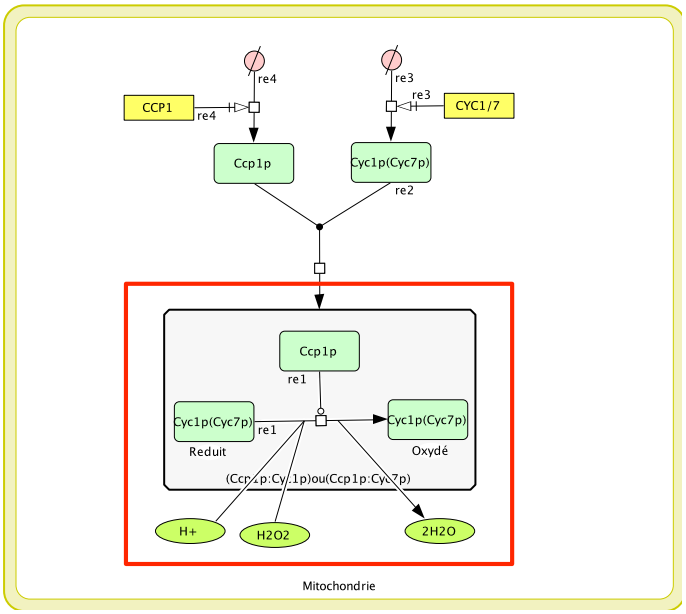


FIGURE 3.21 – Modèle explicite pour le système de Ferrocytochrome-c,l a partie considérée par l'analyse de flux à l'équilibre du système CYC est représentée dans le rectangle rouge.

$$\begin{array}{rcll} R & \vdash & Ccp1p \wedge Cyc1p : Ccp1p & \text{eq(3.27)} \\ R' & \vdash & Ccp1p \wedge Cyc7p : Ccp1p & \text{eq(3.31)} \end{array}$$

3.4.5 L'association de gènes pour la modélisation hiérarchique

L'oxydation de Cytochrome-c **Cyc1p** ou **Cyc7p** par **Ccp1p** a lieu à l'intérieur du complexe **Cyc1p :Ccp1p** (ou le complexe **Cyc7p :Ccp1p**), qui associe la Cytochrome-c avec peroxydase Cytochrome-c (Figure 3.22).

$$\begin{array}{rcll} R & \vdash & Ccp1p \wedge Cyc1p : Ccp1p & \text{eq(3.27)} \\ & \vdash & Ccp1p \wedge Cyc1p & \text{eq(3.28)} \\ & \vdash & CCP1 \wedge CYC1 & \text{eqs(3.30) (3.29)} \\ R' & \vdash & Ccp1p \wedge Cyc7p : Ccp1p & \text{eq(3.31)} \\ & \vdash & Ccp1p \wedge Cyc1p & \text{eq(3.32)} \\ & \vdash & CCP1 \wedge CYC7 & \text{eqs(3.34) (3.33)} \end{array}$$

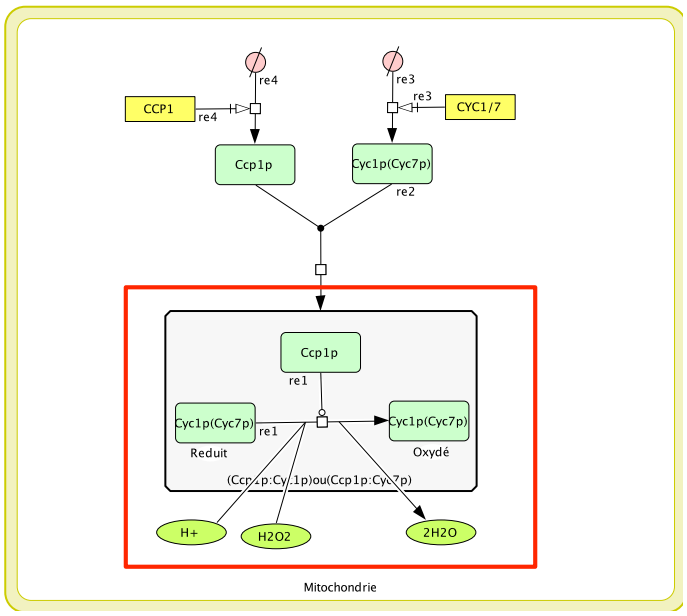


FIGURE 3.22 – Modèle explicite pour le système de Ferrocytochrome-c ,la partie considérée par la modélisation hiérarchique du système CYC est représentée dans le rectangle rouge.

3.5 Système de la decarboxylase du 3-methyl-2-oxopentanoate (PDC)

Les enzymes de la decarboxylase du 3-methyl-2-oxopentanoate, **Pdc1p**, **Pdc5p**, **Pdc6p**, et **ARO10**, réagissent avec les 2-oxo-acides comme le 3-methyl-2-oxopentanoate, qui est un intermédiaire dans la synthèse de l'isoleucine, (Figure 3.23).

Concernant cette réaction, la **Thi3p** est une protéine de régulation qui se lie aux facteurs de transcription **Thi2p** ou **Pdc2p**. En présence de thiamine, les complexes **Pdc2p :Thi3p** et **Thi2P :Thi3p** se dissocient, désactivant ainsi les facteurs de transcriptions. Les gènes **PDC1** et **PDC5**, qui codent respectivement les protéines **Pdc1p** et **Pdc5p**, sont régulés par le complexe **Pdc2p :Thia3p** ou par le complexe **Thi2p :Thi3p**, et par l'absence de la protéine **Pdc1p**. Mais les gènes **PDC6** et **ARO10**, codant respectivement les protéines **Pdc6p** et **Aro10p** ne sont pas régulés par ces mécanismes.

3.5.1 Description logique

$$\text{R(PDC)} = \text{3-metyl-2-oxepentanal} \wedge \text{H}^{+} \rightarrow \text{2-metylbutanal} \wedge \text{CO2}$$

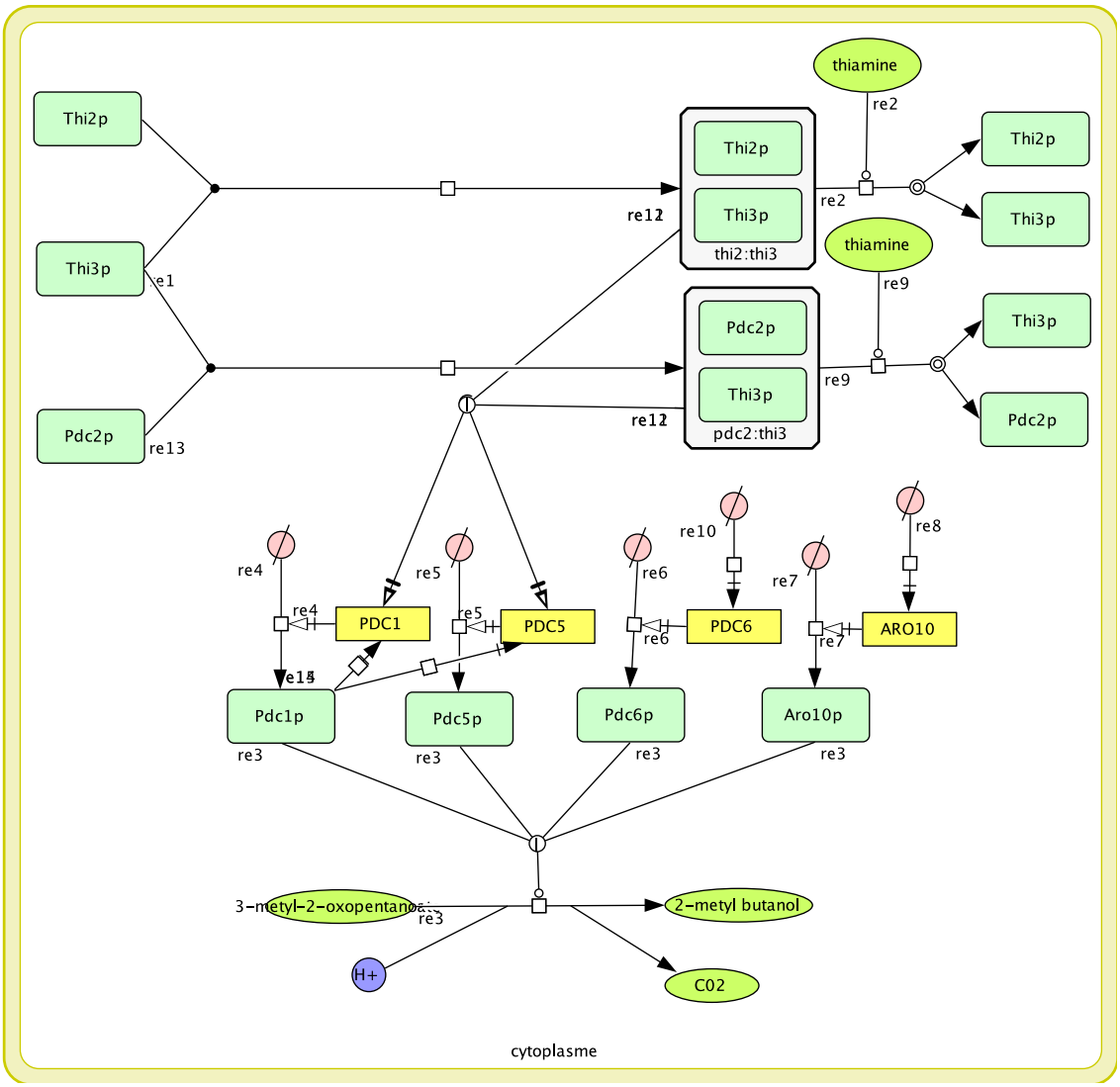


FIGURE 3.23 – Modèle explicite pour la réaction de 3-méthyl-2-oxopentanoate décarboxylase

$$\text{R(PDC)} \vdash \text{Pdc1p} \vee \text{Pdc5p} \vee \text{Pdc6p} \vee \text{Aro10} \quad (3.35)$$

$$\text{Pdc1p} \vdash \text{PDC1} \wedge \text{Thi3p} : (\text{Thi2p} \vee \text{Pdc2p}) \wedge \neg\text{Pdc1p} \quad (3.36)$$

$$\text{Pdc5p} \vdash \text{PDC5} \wedge \text{Thi3p} : (\text{Thi2p} \vee \text{Pdc2p}) \wedge \neg\text{Pdc1p} \quad (3.37)$$

$$\text{Thi3p :Pdc2p} \vdash \text{Thi3p} \wedge \text{Pdc2p} \wedge \neg\text{Thiamine} \quad (3.38)$$

$$\text{Thi3p :Thi2p} \vdash \text{Thi3p} \wedge \text{Thi2p} \wedge \neg\text{Thiamine} \quad (3.39)$$

$$\text{DIS(Thi3p :Pdc2p)} \vdash \text{Thi3p :Pdc2p} \wedge \text{Thiamine} \quad (3.40)$$

$$\text{DIS(Thi3p :Thi2p)} \vdash \text{Thi3p :Thi2p} \wedge \text{Thiamine} \quad (3.41)$$

$$\text{Pdc6p} \vdash \text{PDC6} \quad (3.42)$$

$$\text{Aro10p} \vdash \text{ARO10} \quad (3.43)$$

3.5.2 Description BioRica

Le modèle hiérarchique du système **PDC** est composé de deux sous-noeuds :

— **Complex** qui décrit l'association de la **Thi3p** soit avec la **Thi2p**, soit avec la **Pdc2p** pour former le facteur de transcription qui régule l'expression des gènes **PDC1** et **PDC5**.

— **Factor** qui est un noeud hybride initialement composé de transitions discrètes qui décrivent l'expression des gènes **PDC1**, **PDC5**, **PDC6** ou **ARO10**, les deux premiers dépendant du facteur de transcription, et d'une équation continue décrivant la réaction de décarboxylase.

Le noeud **Main** se connecte à deux sous-noeuds au travers de variables booléennes indiquant la présence de l'un ou l'autre des complexes de facteurs de transcription.

node Complex

state
Thi3p : BOOL ;
Thi2p : BOOL ;
Pdc2p : BOOL ;
Thiamin : BOOL ;
flow
Thi2p :Thi3p : BOOL ;
Pdc2p :Thi3p : BOOL ;
event
associate(Thi2p :Thi3p) ; associate(Pdc2p :Thi3p) ;
trans
Thi3p , Thi2p , ¬Thiamin ⊢ associate(Thi2p :Thi3p) → Thi2p :Thi3p := true ;
Thi3p , Pdc2p, ¬Thiamin ⊢ associate(Pdc2p :Thi3p) → Pdc2p :Thi3p := true ;
edon

node Factor

state

PDC1 : BOOL ;
PDC5 : BOOL ;
PDC6 : BOOL ;
ARO10 : BOOL ;
Pcd1p : BOOL ;
Pcd5p : BOOL ;
Pcd6p : BOOL ;
Aro10p : BOOL ;
3-metyl-2-oxepentanal : FLOAT ;
R_0 : $\{0, 1\}$;

flow
Thi2p :Thi3p : BOOL ;
Pdc2p :Thi3p : BOOL ;

const
ic(3-metyl-2-exepontanal)

diff
d(3-metyl-2-oxepentanal) = $\mathrm{R}_0(\mathrm{cell}.V0.(1 + 1.\ln(\frac{3-metyl-2-oxepentanal}{ic(3-metyl-2-exepontanal)})$
$+ \ln(\frac{H^+}{ic(H^+)}) - \ln(\frac{2-metylbutanal}{ic(2-metylbutanal)}) - \ln(\frac{CO2}{ic(CO2)})))$

event
G-expr(PDC1) ; G-expr(PDC5) ;
G-expr(PDC6) ; G-expr(ARO10) ;
React ;

trans
PDC1, (Thi2p :Thi3p | Pdc2p :Thi3p), ¬ Pdc1p ⊢ G-expr(PDC1) → Pdc1p := true ;
PDC5, (Thi2p :Thi3p | Pdc2p :Thi3p), ¬ Pdc5p ⊢ G-expr(PDC5) → Pdc5p := true ;
PDC6 ⊢ G-expr(PDC6) → Pdc6p := true ;
ARO10 ⊢ G-expr(ARO10) → Aro10p := true ;
Pdc1p | Pdc5p | Pdc6p | Aro10p ⊢ React → R_0 :=1 ;

edon

node Main

sub
F : Factor ;
C : complex ;

assert
C.Thi2p :Thi3p = F.Thi2p :Thi3p ;
C.Pdc2p :Thi3p = F.Pdc2p :Thi3p ;

edon

3.5.3 L'associations de gènes pour l'inférence de modèle

La réaction est catalysée par une des enzymes **Pdc1p**, **Pdc5p**, **Pdc6p** ou **Aro10p** codantes par les gènes **PDC1**, **PDC5**, **PDC6**, **ARO10**, donc l'assurance qu'on peut demander ou inférer cette réaction repose sur l'existence d'un de ces gènes (Figure 3.24).

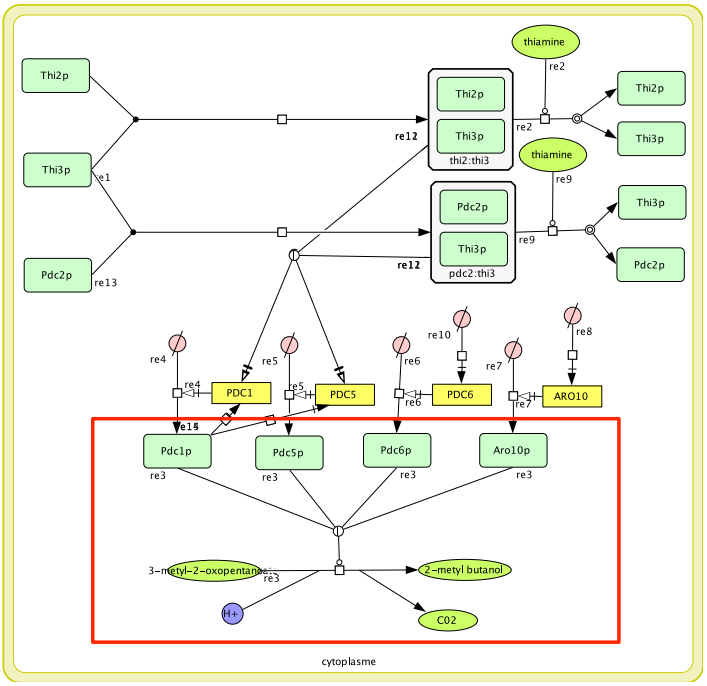


FIGURE 3.24 – Modèle explicite pour le Système de la decarboxylase du 3-methyl-2-oxopentanoate, la partie considérée par l'inférence du système est représentée dans le rectangle rouge.

$$\begin{array}{lll} \text{R(PDC)} & \vdash & \text{Pdc1p} \vee \text{Pdc5p} \vee \text{Pdc6p} \vee \text{Aro10p} \qquad \text{eq(3.35)} \\ & \vdash & \text{PDC1} \vee \text{PDC5} \vee \text{PDC6} \vee \text{ARO10} \qquad \text{eqs(3.36) (3.37) (3.42) (3.43)} \end{array}$$

3.5.4 L'associations de gènes pour l'analyse de flux à l'équilibre

Dans ce cas précis, nous supposons que les enzymes et les complexes enzymatiques sont présents (ou pas), et nous nous occupons seulement des cycles de réactions de réduction (Figure 3.25).

$$\text{R(PDC)} \vdash \text{Pdc1p} \vee \text{Pdc5p} \vee \text{Pdc6p} \vee \text{Aro10p} \qquad \text{eq(3.35)}$$

3.5.5 L'associations de gènes pour la modélisation hiérarchique

La transformation de 3-méthyl- 2-oxopentanoate en 2-méthyl butanol est effectuée par une des enzymes **Pdc1p**, **Pdc5p**, **Pdc6p**, ou **Aro10p** (Figure 3.26) .

$$\begin{array}{lll} \text{R(PDC)} & \vdash & \text{Pdc1p} \vee \text{Pdc5p} \vee \text{Pdc6p} \vee \text{Aro10p} \qquad \text{eq(3.35)} \\ & \vdash & \text{PDC1} \vee \text{PDC5} \vee \text{PDC6} \vee \text{ARO10} \qquad \text{eqs(3.36),(3.37),(3.42),(3.43)} \end{array}$$

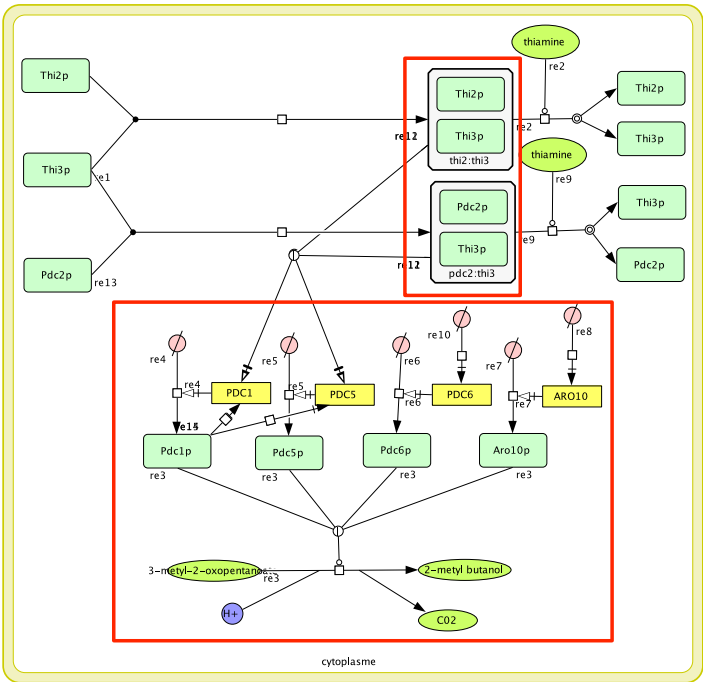


FIGURE 3.25 – Modèle explicite pour le Système de la decarboxylase du 3-methyl-2-oxopentanoate, la partie considérée par l'analyse de flux à l'équilibre du système est représentée dans le rectangle rouge.

3.6 Discussion

En général, ici nous ne disposons pas du modèle complet, et ce document ne décrit pas une méthode, mais un critère formel pour l'exactitude des associations de gènes.

En se basant sur la définition des associations de gènes, l'exigence d'une application, les influences possibles d'une variable sur le résultat, nous pouvons définir les règles de simplification qui produisent les associations de gènes les plus pertinentes pour une réaction. Et nous pouvons l'utiliser pour fournir des lignes directrices de curation pour les conservateurs.

Ce critère peut être utilisé pour définir des procédures pour trouver des associations de gènes à partir des résultats expérimentaux, et aussi pour mieux formaliser les formules d'association de gènes, de sorte que la connaissance peut être plus facilement extraite par d'autres processus d'inférence.

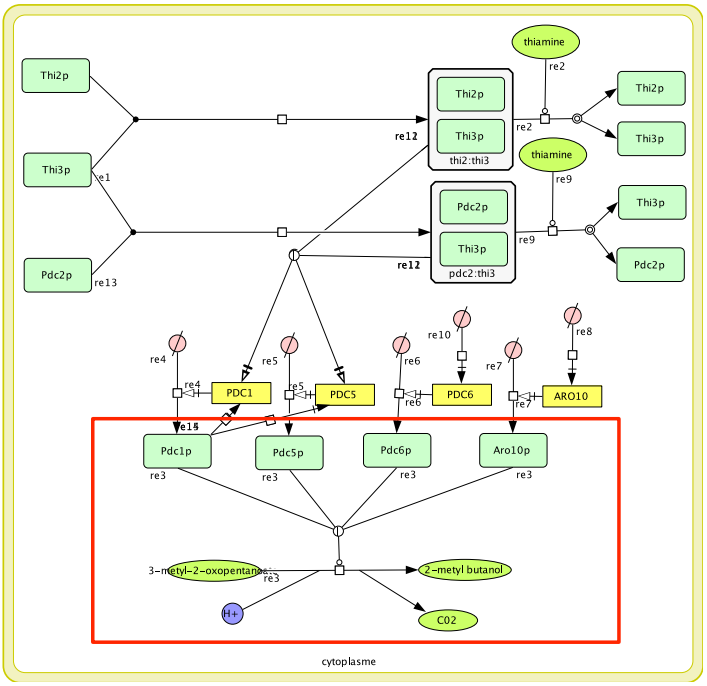


FIGURE 3.26 – Modèle explicite pour le Système de la decarboxylase du 3-methyl-2-oxopentanoate, la partie considérée par la modélisation hiérarchique du système est représentée dans le rectangle rouge.

Chapitre 4

Méthode ab-Pantograph pour l'inférence de modèles

L'idée principale de la plupart des algorithmes de reconstruction de modèles métaboliques est de rechercher la présence de réactions enzymatiques dans le génome annoté de l'organisme à modéliser, et de créer un réseau de ces réactions, représentant l'interconnection de la production et la consommation de métabolites.

Dans ce chapitre nous etudions la question, comment inférer un modèle d'un nouveau organisme cible à partir d'un modèle de référence et d'informations sur l'homologie entre les gènes du nouveau organisme et les gènes de référence. L'originalité de notre approche est de considérer les gènes comme des observations qui doivent être expliqués, et les réactions comme des hypothèses qui doivent être émises pour les expliquer. Ces hypothèses seront trouvées par l'abduction, introduit dans 1.4.2 (page 26) et détaillé dans la suite (page 76).

4.1 Introduction et motivation

L'inférence des réseaux métaboliques devient un challenge important avec l'augmentation de la masse de données, car la construction d'un réseau métabolique et très coûteuse et longue à effectuer. C'est la raison pour laquelle on cherche à en automatiser la construction. L'une des méthodes pour construire un réseau métabolique est *Pantograph* [LZS15]. Dans *Pantograph* nous disposons d'un modèle de référence, où les réactions sont annotées par des associations de gènes et de différentes déterminations d'orthologie entre les gènes du modèle de référence et ceux de l'organisme cible. Son fonctionnement peut être résumé par un fragment de notre diagramme paru sur la page 3 :

$$\left\langle m_0, a_0^{\{G_0\}} \right\rangle \xrightarrow{G_0 \overset{h}{\cdots} G_1} \left\langle \widetilde{m}_1, \widetilde{a}_1^{\{G_1\}} \right\rangle$$

Pantograph traite chaque réaction dans le modèle de référence une après l'autre : il essaie d'abord de réécrire l'association de gènes grâce aux orthologies puis, en cas de succès,

instancie la réaction dans le modèle cible. La méthode comptabilise les orthologies dans un *tally map* pour décider s'il est possible de réécrire un gèn ou une disjonction de gènes paralogues.

Dans ce chapitre nous présentons une nouvelle stratégie, ab-Pantograph, qui utilise également la réécriture d'associations de gènes pour décider si une réaction est possible, mais qui base l'inférence des nouveaux modèles sur la nécessité d'expliquer des gènes dans le génome cible.

Le principe de base de notre stratégie est que les gènes enzymatiques dans le génome le sont parce qu'elle servent à une réaction. D'un point de vue biologique ceci est une simplification grossière, mais elle permet de réfléchir différemment à la question et remettre les gènes, et pas les réaction, au coeur de l'inférence. Nous considérons alors le modèle à construire comme un ensemble d'hypothèses qui expliquent (c'est-à-dire, qui justifient) les gènes observés dans le génome cible. En d'autres termes, partant des gènes présents dans l'organisme cible, nous allons considérer qu'une réaction doit être présente dans le modèle cible si les gènes responsables de son fonctionnement sont présents. Nous utilisos la programmation logique abductive qui, quant à elle, est une technique bien connue pour raisonner avec des connaissances incomplètes. Dans la programmation logique abductive, nous voulons seulement expliquer un fait en termes de cause, mais sans reposer sur d'autres faits. Le programme d'abduction peut ainsi générer différentes versions spécifiques d'une réaction référence si le génome cible contient des gènes spécifiques, ce que ne peut pas faire *Pantograph*.
Dans la suite nous :

- Donnons un aperçu du fonctionnement de la méthode Pantograph ;
- Expliquons comment réécrire les associations de gènes en traduisant [LZS15] en termes logiques ;
- Rappelons les principes de la programmation logique abductive selon Christiansen *et al.* [CD05]
- Définissons une procédure abductive pour inférer les réactions dans un modèle ;
- Décrivons notre implémentation et sa validation.

La validation de la méthode combinée avec les prédicats dans le programme logique, nous permettra d'argumenter la congruence entre le modèle cible généré et le modèle "idéale".

4.2 Méthode *Pantograph*

Dans cette section, nous présentons la méthode *Pantograph* pour construire un réseau métabolique. *Pantograph* [1] [Loi12, LZS15] est une méthode d'inférence par instanciation, elle utilise des modèles métaboliques déjà connus pour commencer son processus de construction, où les décisions pour instancier des éléments sont prises par tentative de satisfaction des formules booléennes d'associations de gènes.

Cette méthode utilise comme entrées un modèle métabolique d'un organisme proche, et les relations d'orthologies entre les gènes de l'organisme utilisé comme référence puis

1. Ce paquetage "open source" peut être obtenu à partir de : http ://pathtastic.gforge.inria.fr.

ceux de l'organisme qu'on le souhaite modéliser. Le modèle produit est bien adapté pour de la correction manuelle éventuelle et pour de la validation par preuve expérimentale.

Les données nécessaires à *Pantograph* sont constituées d'un modèle de référence, de preuves d'orthologies, des informations de phénotypage et de possibles éditions manuelles. Le modèle de référence doit être décrit sous forme d'un fichier SBML rendant compte des réactions, des espèces moléculaires, et des compartements. Les réactions enzymatiques doivent être accompagnées de leurs associations de gènes et de leur possible propriété de réversibilité. Afin de produire un résultat utile, le modèle de référence doit être aussi proche que possible de l'organisme cible, en termes de métabolisme. Les preuves d'orthologie sont fournies sous forme de tableaux contenant les résultats de méthodes de prédiction complémentaires ou concurrentes et déterminant les correspondances entre les gènes du modèle référence et les gènes du modèle cible. Les informations de phénotypage sont fournies sous forme d'un tableau contenant les conditions du milieu, de la désactivation génétique, et les phénotypes de croissance. Les éditions manuelles sont fournies sous forme d'un tableau (qui peut être vide) de réactions, d'espèces moléculaires, et de compartements qui doivent exister dans le résultat.

Une fois le modèle construit, *Pantograph* accepte des modifications manuelles, enregistrées par des experts humains au cours de la phase de correction, qui remplacent les décisions prises par la procédure de construction de modèle. Une edition manuelle peut consister par exemple en un remplacement d'une association de gènes pour une réaction, une suppression d'éléments du modèle, ou une création de compartements, d'espèces moléculaires ou de réactions.

Toute réaction dans le modèle de référence qui n'a pas d'associations de gènes est considérée comme spontanée et est automatiquement choisie pour l'instanciation.

Le processus de construction d'un modèle débute par l'extraction des éléments annotés du modèle référence (i.e., réactions, espèces moléculaires, ou compartements). A partir des preuves d'orthologie, une structure de données appelée *tally map* est produite. Cette dernière enregistre, de manière unique, les relations d'orthologies et comptabilise, pour chacune d'elles, le nombre de méthodes de détections d'orthologie la mettant en évidence. Ces mesures sont utilisées par la fonction de déclenchement pour déterminer si et comment doit être instancier un élément du modèle référence dans le modèle cible. Le processus se poursuit par l'évaluation de la fonction de déclenchement qui détermine l'éventuelle instanciation de chaque élément du modèle référence dans le modèle cible et leur instanciation, le cas échéant.

Les sorties de l'étape de construction du modèle sont, un modèle annoté au format *SBML* pour l'organisme cible, un tableau des réactions du modèle de reference qui n'ont pas été instanciées.

4.2.1 Réecriture des associations de gènes

Dans cette section nous étendons la stratégie développée dans [LZS15] qui a été suivie dans *Pantograph* pour réecrire les associations de gènes. Une décision essentielle dans la procedure de reconstruction est de savoir si une réaction donnée r_s du modèle de référence

appelé S possède une réaction analogue r_t dans le modèle cible appelé T. Pour cela, nous devons déterminer si les gènes associés à r_s issus de l'ensemble de gènes du modèle de référence nommé G_S, ont des analogues dans l'ensemble de gènes du modèle cible nommé G_T.

Si cette condition ne peut être validée expérimentalement, il faut se reposer sur les relations d'orthologies entres gènes pour inférer la fonction d'analogie. Une méthode de détection d'orthologie produit une correspondance entre gène paralogues en se basant sur des similarités de séquence entre les gènes considérés. En combinant plusieurs méthodes complémentaires de détection d'orthologies, une plus grande diversité de critères de similarité peut être utilisée et conduire à une meilleure inférence de la fonction d'analogie entre paralogues. Etant donné que chaque décision sera prise de manière indépendante, il est possible d'utiliser des règles de prépondérance sur les différentes méthodes plutôt que de calculer un consensus (comme dans Nikolski and Sherman [NS07]).

Soient G_S et G_T des ensembles de gènes des deux organismes. Une *gene map* $\mathcal{G}$ est définie comme un ensemble de correspondances entre gènes paralogues de G_S et G_T. Un ensemble de *gene map* $\mathcal{H}$ est défini comme l'ensemble des $\mathcal{G}_i$ calculé par différentes méthodes de détection d'orthologie. Une *tally map* $\mathscr{T}$ correspond à une synthèse de $\mathcal{H}$ qui enregistre de manière unique chaque correspondance présente dans $\mathcal{H}$ et le nombre de gene map où cette dernière apparait. Formellement :

$$\begin{aligned}
\mathcal{G} &\subseteq 2^{\{G_S\}} \times 2^{\{G_T\}} \\
\mathcal{H} &= \{\mathcal{G}_i\} \\
\mathscr{T} &\subseteq \mathbb{N} \times 2^{\{G_S\}} \times 2^{\{G_T\}} \\
\mathscr{T} &= \{\langle |\{\mathcal{G} \in \mathcal{H} \mid \langle p_a, p_b\rangle \in \mathcal{G}\}|, p_a, p_b\rangle\}.
\end{aligned}$$

Une *tally map* $\mathscr{T}$ est utilisée pour enregistrer et évaluer l'évidence d'une relation d'orthologie dans $\mathcal{H}$. Par simplicité d'écriture, *tally*, *gscaf*, et *gtarg* représentent respectivement le compteur, le gène de S et le gène orthologue de T pour un tuple de $\mathscr{T}$. On définit un seuil t^* pour indiquer la prépondérance d'une relation d'orthologie (typiquement $t^* = \frac{1}{2}|\mathcal{H}|$). Pour traduire un ensemble de gènes $L \subseteq G_S$ à $L_T \subseteq G_T$, il faut considérer, dans l'ordre, deux cas : *simple* et *combiné*. Autrement la traduction de l'ensemble L échoue et nous concluons que la réaction correspondante n'est pas conservée dans le cible.

Dans le cas simple, il existe un relation d'orthologie prépondérante pour $a \in L$ maximisant t et minimisant $|p_b|$. Formellement,

$$\begin{aligned}
K_1 &= \min_{|p_b|} \max_t \{\langle t, p_a, p_b\rangle \in \mathscr{T} \mid t \geq t^*\} \\
K_2 &= \bigcup_{a \in L} \left\{ \min_{|p_b|} \max_t \{\langle t, \{a\}, p_a\rangle \in \mathscr{T} \mid t \geq t^*\} \right\}. \\
L_B &= \begin{cases} \text{gtarg}(K_1) & \text{if} \quad \text{gscaf}(K_1) = L \\ \cup_{k_2 \in K_2} \text{gtarg}(k_2) & \text{else if} \quad \cup_{k_2 \in K_2} \text{gscaf}(k_2) = L \end{cases}
\end{aligned}$$

Dans le cas K_1, L correspond exactement à l'ensemble des paralogues de a de la relation d'orthologie optimale de $\mathscr{T}$. Dans le cas K_2, L correspond exactement à l'union des relations d'orthologie optimales impliquant uniquement $a \in L$ (i.e. sans ses paralogues dans S).

Dans le cas combiné, nous avons à choisir une combinaison de relations d'orthologie pour couvrir L. Ce choix est pris en préférant la couverture la plus proche de L avec les relations d'orthologies les plus prépondérantes en se basant sur une fonction de score σ comparant le bénéfice (i.e. taux de couverture) au coût (nombre de gènes cibles) d'une relation d'orthologie pondéré par la prépondérance et la précision de la couverture. Formellement,

$$K = \begin{cases} K_3 = \{\langle t, p_a, p_b\rangle \in \mathscr{T} \mid p_a = L \wedge t \geq t^*\} & \text{if } K_3 \neq \emptyset \\ K_4 = \{\langle t, p_a, p_b\rangle \in \mathscr{T} \mid p_a \cap L \neq \emptyset \wedge t \geq t^*\} & \text{else if } K_4 \neq \emptyset \\ K_5 = \{\langle t, p_a, p_b\rangle \in \mathscr{T} \mid p_a = L\} & \text{else if } K_5 \neq \emptyset \\ K_6 = \{\langle t, p_a, p_b\rangle \in \mathscr{T} \mid p_a \cap L \neq \emptyset\} & \text{else if } K_6 \neq \emptyset \end{cases}$$

$$L_B = \text{gtarg}(\max_{\sigma(k,L)} k \in K)$$

et

$$\sigma(k, L) = \sum_{\langle t, p_a, p_b\rangle \in K} \frac{|p_a \cap L|}{p_b} \cdot t \cdot \begin{cases} 0.5 & p_a \not\subseteq L \\ 1.0 & \text{otherwise} \end{cases}$$

Intuitivement, l'ensemble K correspond préférentiellement à (1) des relations d'orthologies couvrant totalement L et atteignant le seuil t^* (cas K_3), (2) des relations d'orthologies couvrant partiellement L et atteignant le seuil t^* (cas K_4), (3) des relations d'orthologies couvrant totalement L mais n'atteignant pas le seuil t^* (cas K_5) ou (4) des relations d'orthologies couvrant partiellement L mais n'atteignant pas le seuil t^* (cas K_6).

Les relations d'orthologie prépondérantes sont utilisées pour réécrire les associations de gènes en remplaçant les gènes de L par leurs orthologues caclulés dans L_B.

4.2.2 Réécrire ces associations en programmation logique

Réécrire l'association de gènes de l'organisme de référence revient à trouver les correspondant de ses gènes dans l'ensemble des gènes de l'organisme cible (c'est-à-dire les homologues). Dans la suite de cette section, nous allons expliquer comment ces différents cas ont été traités et intégrés dans le programme proposé dans cette thèse.

La prédiction d'homologie entre les gènes de référence et les gène de cible est indépendante de la façon dont on prévoit l'association entre la réaction et les gènes, et c'est donnée comme paramètre pour le processus de reconstruction. Pour la réécriture des associations de gènes dans ces travaux nous suivons la stratégie adaptée par *Pantograph* expliquée précedement dans la section 4.2. Convertir cette stratégie à la logique et l'abduction nous permet d'avoir la possibilité de vérifier la cohérence et la validité de ces règles facilement et de les analyser par des biologistes.

La réécriture des associations de gènes nécessite de considérer plusieurs cas. Nous présentons ici quelques cas à traiter : (1) quand l'association de gènes correspond à un seul gène, (2) quand plusieurs gènes sont nécessaires ensemble pour l'activité d'une réaction où l'association de gènes va être écrite sous forme d'une conjonction de gènes, (3) dans le cas où plusieurs gènes se ressemblent dans leur fonction (c'est-à-dire paralogues) où l'association de gènes va être présentée comme une disjonction des gènes, (4) et enfin le cas le plus compliqué, quand l'association de gènes de référence contient plusieurs gènes équivalents en activité, et d'autres gènes complémentaires où l'association de gènes est écrite comme une conjonction d'une ou plusieurs disjonction(s) ou une disjonction d'une ou plusieurs conjonction(s).

Rappelons que la relation d'homologie entre les gènes références et cibles est représentée par

```
homolog([...ref...],[...targ...],method)
```

où `[...ref...]` sont les gènes de référence, `[...targ...]` sont les gènes de cible homologues avec les gènes de référence, et `method` est la méthode qui prédit cette relation d'homologie.

Dans l'exemple :

```
homolog([yal030w],[yali0a03113g],best).
homolog([yal030w,ylr093c,yor327c],
        [yali0a03113g,yali0b04026g,yali0e00594g],dom).
homolog([yal030w],[yali0a03113g],inp).
homolog([yal030w,ylr093c,yor327c],
        [yali0e00594g,yali0a03113g,yali0b04026g],sons)
```

le gène `yal030w` est trouvé homologue avec le gène `yali0a03113g` par la méthode `best` et homologue avec le gène `yali0a03113g` par la méthode `inp`. Cependant ce gène avec deux autres gènes orthologues (i.e, `yal030w,ylr093c,yor327c`) sont prédit homologues avec `yali0a03113g,yali0b04026g,yali0e00594g` par `dom` et avec `yali0e00594g,yali0a03113g,yali0b04026g` par `son`.

Dans le programme nous cherchons à regrouper les homologues, pour chaque liste de gènes, prédit par toutes les méthodes et donner le nombre de fois que cette liste a été prédit et la longeur de cette liste, prenons par exemple le gène `yal030w`, appliquer *tally map* donne :

```
tally([yal030w],T).
T = [[[yali0a03113g], 2|1]].
```

où (2) indique le nombre des méthodes qui a trouvé cet homologue, et le (1) est la longeur de la liste de gènes cible `[yali0a03113g]` de la même manière :

```
?- tally([yal030w,ylr093c,yor327c],T).
T = [[[yali0a03113g, yali0b04026g, yali0e00594g], 2|3]].
```

Réécrire l'association de gènes de l'organisme de référence revient à trouver les correspondant de ses gènes dans l'ensemble des gènes de l'organisme cible (c'est-à-dire les homologues).

Tout d'abord, quand une association de gènes de référence contient un unique gène (i.e. `AG = G`), nous pouvons avoir les deux possibilités suivante :

Soit le gène de référence (i.e. `G`) a un seul gène homologue cible (disons `Gt`) et, dans ce cas, nous allons avoir une relation `(1:1)` et l'association de gènes va être réécrite comme le gène homologue de cible `AG' = Gt`.

Soit il existe plusieurs gènes homologues dans l'organisme cible `(Gt1,...,Gtn)` pour le gène `G` et, dans ce cas, nous allons avoir une relation `1:n` et l'association de gènes va être réécrite comme une disjonction des gènes `(Gt1,...,Gtn)` dans le cible : `AG'= OR([Gt1,...,Gtn])`.

Pour une association de gènes contenant plusieurs gènes complémentaires donnée comme `AG=AND[G1,...,Gn]` on cherche l'homologue correspondant pour chaque gène dans l'association. La réécriture de cette association de gènes réussit si l'on trouve au moins un gène homologue dans l'organisme cible pour chacun de ses gènes, et l'association prédite sera une conjunction de homolgues de tous les gènes de l'association, c'est-à-dire de la forme `AG'=AND([OR([G11,..,G1k]),...,OR([Gn1,..,Gnl])])` où
`k,l>=1` et `OR([G11,..,G1k]) et OR([Gn1,..,Gnl])` sont les homologues des gènes `G1 et Gn`.

La réécriture d'une association de gènes contenant plusieurs gènes paralogues donnée comme `AG = OR([G1,...,Gn])`, n'est pas forcement une disjonction des gènes homologue. Pour cette association nous cherchons d'abord une relation d'homologie où la partie de référence correspond à la liste `[G1,...,Gn]` c'est-à-dire :

```
homolog([G1,...,Gn],[Gt1,...,Gtk])
```

Dans le cas positif la réécriture de `AG` est donné par la partie cible de la relation `[Gt1,...,Gtk]`.

Dans l'absence de toute relation de ce type, nous cherchons l'homologue pour chaque gène dans `AG` et sa réécriture réussit si au moins un de ses gènes a au moins un gène homologue dans l'organisme cible : il existe `Gt` dans l'organisme cible où `Gt` homologue avec `G`.
Une fois nous avons le résultats, nous appliquons des règle de simplification de la forme :

```
and(and(cluse)) --> and(cluse)
and(or(cluse))  --> or(cluse)
or(and(cluse))  --> and(cluse)
or(or(cluse))   --> or(cluse)
```

et pour garantir cette simplification nous ajoutons des contraintes d'intégrité qui exagèrent respecter telles règles.

Les associations de gène sont réécrites à la volée par le processus abductif décrite dans la suite.

4.3 Méthode *ab-Pantograph*

La méthode ab-Pantograph permet de reconstruire un modèle par application de la logique abductive à des observations, qui sont, dans notre cas, les gènes de l'organisme cible, le modèle de référence et les relations d'homologie (également appelés des faits) pour dériver des explications possibles de l'utilité de ces observations, qui sont, dans notre cas, des réactions qui sont présentes dans le modèle de référence et contrôlées par des gènes de l'organisme de référence et homologues à des gènes de l'organisme cible. Avant de présenter formellement la logique abductive et la méthode proposée, nous présentons brièvement l'intérêt de cette approche.

L'abduction nous offre la possibilité de créer de nouvelles réactions n'existant pas dans le modèle de référence et de les ajouter dans le modèle à inférer. Reste à savoir ou à décider quand et pourquoi nous allons avoir besoin de créer ces réactions (i.e., la stratégie de prise de décision), et comment l'ajout de ces réactions doit être effectué. Illustrons la première interrogation par deux situations nécessitant l'ajout de réactions : (1) la présence d'une famille multigénique, (2) l'application d'opérations au niveau des compartements.

Dans le cas d'une famille multigénique, si l'on considère une association de gènes impliquant des gènes paralogues (i.e., ayant la même fonction), la majorité des méthodes de reconstruction vont inférer une unique réaction (commune à l'ensemble de ces gènes paralogues) et une association de gènes complexe composée de l'ensemble des orthologues de ces gènes. Mais dans de nombreux cas, ces gènes sont issus d'un gène ancestral ayant subi une duplication puis éventuellement des spécialisations. Il serait intéressant dans ce type de configurations, d'inférer des réactions spécifiques pour chacun de ces gènes plutôt qu'une réaction complexe comme évoquée précédement.

Dans le cas de l'application d'opérations au niveau des compartements, il est courant qu'une création de réactions de transport soit requise pour assurer la présence des éspeces nécessaires aux réactions présentes initialement dans le compartement, et préserver l'intégrité du modèle par le maintien des connections entre ses différentes parties.

Par la suite, nous allons présenter une méthode qui peut répondre à la deuxième interrogation – à savoir "comment créer de nouvelles réactions ?" : la programmation logique abductive.

4.3.1 Programmation Logique Abductive et Hyprolog

Face aux limites de la programmation logique standard dans le cadre de la résolution de problèmes liés à l'intelligence artificielle, des chercheurs ont proposé d'élargir la programmation logique à l'abduction. C'est à la fin des années 80, que l'étude de l'inférence abductive a commencée dans le cadre de la programmation logique. La programmation logique abductive [KKT92], [DK02] correspond à l'intégration de l'abduction dans la

programmation logique. Elle est un cadre de représentation des connaissances de haut niveau qui nous permet de résoudre les problèmes de manière déclarative en se basant sur le raisonnement abductif.

La programmation logique abductive est une technique bien connue pour raisonner avec des connaissances incomplètes. Elle élargie la programmation logique normale en permettant à certains prédicats d'être incomplètement définis. Ces prédicats sont déclarés comme *abductibles*. La vérité de ces prédicats ne peut être prouvée, mais elle peut être supposée. La résolution des problèmes est effectuée en formulant des hypothèses sur ces prédicats (hypothèses abductives) alors admises comme des solutions. Toutefois, dans des applications typiques, les hypothèses n'ont pas toutes de sens. Certaines suppositions peuvent être exclues au préalable selon certaines données. Pour cette raison, dans la programmation logique abductive, nous pouvons définir un ensemble de règles qu'on appelle *contraintes d'intégrité* qui doivent être satisfaites par l'ensemble des hypothèses.

La programmation logique abductive suit la stratégie de moindre engagement ("least-commitment" en anglais) qui consisite à reporter les décisions aussi longtemps que possible, de sorte que quand elles sont prises la probabilité de leur exactitude est maximisée. Cette stratégie nous permet de réutiliser les hypothèses existantes, c-à-dire de construire le modèle cible dans un premier temps avec les réactions présentes dans le modèle de réference, et retarder la création de nouvelles réactions jusqu'à l'obtention de suffisament d'informations sur les réactions à ajouter. Grâce à la possible représentation haut-niveau du domaine d'application et à la modélisation directe de certaines propriétés du problème, la programmation logique abductive fournit un certain contrôle sur les calculs aboutissant à une amélioration de l'efficacité du calcul ou de la qualité des solutions.

Nous pouvons séparer les deux objectifs de validité et d'optimalité de la solution. On peut soit améliorer la qualité de la solution par le raffinement incrémentale du modèle du problème représenté dans le programme $\mathbf{P}$ de la théorie de la programmation logique abductive sans affecter la validité de la solution qui est assurée par les contraintes d'intégrité $\mathbf{IC}$ de la théorie. Nous pouvons également expérimenter avec différentes alternatives de conception en adoptant différentes stratégies visant à améliorer l'optimalité des solutions. D'autre part, nous pouvons modifier les exigences sur la solution, exprimées par les contraintes d'intégrité $\mathbf{IC}$, sans affecter la façon dont le problème est modélisé dans le programme $\mathbf{P}$.

Le modèle de base du problème en $\mathbf{P}$ peut être affiné de manière incrémentielle le long de deux directions différentes mais liées entre elles. La première possibilité est d'exploiter les caractéristiques naturelles de manière incrémentielle du problème afin de construire un modèle en $\mathbf{P}$ plus détaillé. Cela peut entraîner non seulement des améliorations relatives à la qualité de la solution, mais aussi sur l'efficacité de calcul de la solution.

La seconde possibilité se réfère au haut niveau manipulation que nous avons sur la solution à travers les prédicats abductibles et le choix des abductibles que nous construisons dans le modèle $\mathbf{P}$ de notre problème. Ce choix peut être facilement mis en oeuvre pour suivre soit des heuristiques, des priorités, ou un algorithme d'optimalité, contrôlant ainsi directement la qualité des solutions. Avec cela, nous pouvons également contrôler

la façon dont une solution est trouvée ou construite. Par exemple, on peut forcer les abductibles à suivre un principe de moindre engagement, ou que ce choix soit implicite à travers quelques contraintes réduites sur les hypothèses de la solution.

Une caractéristique majeure des applications de la programmation logique abductive est la souplesse qu'il offre en vertu des exigences nouvelles ou changeantes dynamiquement. Une fois la représentation complète du problème obtenue, nous pouvons facilement expérimenter cette dernière avec différentes exigences sur la solution, en changeant les contraintes d'intégrité qui spécialisent le modèle général pour les besoins et les préférences d'un cas particulier.

La capacité de l'abduction à raisonner avec des hypothèses ou solutions existantes facilite la tâche de calcul d'une solution, ou plus précisément la tâche d'adaptation d'une solution existante pour répondre aux nouvelles exigences qui apparaissent dans le temps. Ces adaptations peuvent être effectuées d'une manière naturelle avec des changements minimes à la solution existante, en préservant des parties de la solution qui ne devraient pas être affectées. Egalement, des exigences naturelles et des stratégies pour le recalcul de la solution peuvent facilement être logées au sein de l'environnement de modélisation de haut niveau de la programmation logique abductive.

Donc, abduction est gènéralement définit comme le processus de raisonnement à des explications pour un objectif donné (ou observation) selon une théorie gènèralz qui décrit le domaine du problème de l'application. Le problème est représenté par une thèorie abductive. Une abductive théorie est un triplet $(\mathbf{P}, \mathbf{A}, \mathbf{IC})$ constitué d'un programme logique classique $\mathbf{P}$, d'un ensemble $\mathbf{A}$ de noms de prédicats, appelés *abductibles*, qui ne sont pas définis (ou ils sont partiallement définis) dans $\mathbf{P}$, et d'un ensemble de contraintes d'intégrité $\mathbf{IC}$ qui limite les hypothèses acceptables.

Les clauses dans le programme $\mathbf{P}$ définissent un ensemble de prédicats non-abductibles et fournissent donc une description du domaine du problème. Les contraintes d'intégrité de $\mathbf{IC}$, quant-à-elles, précisent les propriétés générales du domaine du problème qui doivent être respectées dans toute solution. Ces contraintes sont souvent exprimées sous forme négative, c'est-à-dire, à des clauses de la forme :

$$\textbf{false} :- \mathbf{A1}, \ldots, \mathbf{An}, \neg\mathbf{B1}, \ldots, \neg\mathbf{Bm}$$

Une telle contrainte implique qu'il n'est pas possible que tous les prédicats $\mathbf{A1}, \ldots, \mathbf{An}$ soient vrais et que, simultanément, tous les prédicats $\mathbf{B1}, \ldots, \mathbf{Bm}$ soient faux.

Un problème, $\mathbf{G}$, qui exprime une observation qui doit être expliquée ou un objectif que l'on cherche à atteindre, est représenté par une combinaison de littéraux positifs et négatifs. Une explication par abduction $\mathbf{E}$ d'un problème $\mathbf{G}$, représente un ensemble non vide de littéraux positifs (et parfois négatifs), constituant les instances de base des prédicats abducTibles, de telle sorte que lorsque ceux-ci sont ajoutés au programme logique $\mathbf{P}$, le problème $\mathbf{G}$ et les contraintes d'intégrité $\mathbf{IC}$ à la fois sont respectés.

Les prédicats abductibles incomplètement définis constituent le support nécessaire à la résolution des problèmes et les explications abductives sont les solutions aux problèmes, et sont conformes à la description du domaine de problème dans $\mathbf{P}$ et $\mathbf{IC}$. Une explication ou une solution abductive pour un objectif $\mathbf{G}$ est un ensemble de abductibles formules

colse $\mathbf{E}$ qui, lorsqu'il est ajouté au programme $\mathbf{P}$ implique l'objectif $\mathbf{G}$ et satisfait les contraines d'intégrité $\mathbf{IC}$ i.e.

$$\mathbf{P} \cup \mathbf{E} \models \mathbf{G} \text{ et } \mathbf{P} \cup \mathbf{E} \models \mathbf{IC}$$

Le raisonnement en programmation logique abductive combine le raisonnement logique normal (pour réduire la problématique des sous-problèmes) avec une sorte de vérification de l'intégrité pour montrer que les explications déductives satisfont les contraintes d'intégrité.

Prenons l'exemple suivant illustrant un programme de logique abductive qui décrit un modèle simple de métabolisme du lactose de la bactérie E. Coli [RK06].

— Connaissance de domaine ($\mathbf{P}$)

$$\begin{aligned}
feed(lactose) &:- make(permease), make(galactosidase)\\
make(Enzyme) &:- code(Gene, Enzyme), express(Gene)\\
express(lac(X)) &:- amount(glucose, low), amount(lactose, hi)\\
express(lac(X)) &:- amount(glucose, medium), amount(lactose, medium)\\
code(lac(y), permease)&\\
code(lac(z), galactosidase)&\\
temperature(low) &:- amount(glucose, low)
\end{aligned}$$

— Contraintes d'intégrité ($\mathbf{IC}$)

$$false :- amount(S, V1), amount(S, V2), V1 \neq V2$$

— Abductibles ($\mathbf{A}$)

$$\textbf{abductible} : amount \setminus 2$$

Le programme $\mathbf{P}$ décrit le fait que *E. coli* peut se nourrir sur le lactose si elle produit deux enzymes permease et galactosidase. Ces enzymes sont codées par deux gènes (respectivement *lac*(y) et lac(z)) dans un cluster de gènes (*lac*(X)), qu'on appellera un opéron, et qui s'exprime lorsque la quantité (amount) de glucose est faible et celle de lactose est élevée, ou quand ils sont à un niveau moyen tous les deux à la fois. Les abductibles, $\mathbf{A}$, déclarent toutes les instances de base de prédicats (amount) comme pouvant être utlisés dans la logique abductive. Cela reflète le fait que, dans notre modèle, nous ne pouvons pas savoir quelles sont les quantités à tout moment des différentes substances. C'est une information incomplète que nous voudrions obtenir dans chacun des cas du problème que nous étudions. Les contraintes d'intégrité indiquent que la quantité d'une substance (S) ne peut avoir qu'une seule valeur.

Notre problème est $\mathbf{G}$ = feed(lactose). Il peut nous apparaître, soit comme l'observation d'une expérience que nous avons réalisée et dont nous voulons alors comprendre le processus menant à cette observation, soit comme un état que nous voulons réaliser et pour lequel nous voulons trouver une manière d'y arriver. Cet objectif a deux explications

abductives :

$$\mathbf{E}_1 = \{amount(lactose, hi), amount(glucose, low)\}$$

$$\mathbf{E}_2 = \{amount(lactose, medium), amount(glucose, medium)\}$$

La décision d'adopter l'explication $\mathbf{E}_1$ ou $\mathbf{E}_2$ pourrait, de plus, dépendre des informations dont nous disposons. Par exemple, nous pouvons savoir que lorsque le niveau de glucose est faible, l'organisme présente un certain comportement et dans notre modèle, c'est la température de l'organisme qui devient faible. En observant la véracité ou non de cette information, nous pouvons choisir la première ou la deuxième explication.

Une fois qu'une explication est choisie, elle devient partie intégrante de notre théorie et nous pouvons tirer de cette théorie de nouvelles conclusions. L'explication, et de manière plus générale ces nouvelles conclusions, constituent la solution à notre problème.

Considérons maintenant un autre exemple issu de [Ray05] sur les fast-food.

— Connaissance de domaine ($\mathbf{P}$)

$$\begin{aligned} meal(x) &:- fries(x), burger(x) \\ burger(x) &:- fries(x), offer(x) \\ offer(mcDonalds) & \end{aligned}$$

— Contraintes d'intégrité ($\mathbf{IC}$)

$$false :- fries(x), \neg bistro(x)$$

— Abductibles ($\mathbf{A}$)

$$\textbf{abductible} : fries \setminus 1, bistro \setminus 1$$

Le programme $\mathbf{P}$ contient trois clauses qui décrivent partiellement un domaine relatif aux établissements de restauration rapide, ou bistrot. Les deux premières clauses sont des règles indiquant respectivement que pour prendre un repas $(meal)$ dans un restaurant rapide nommé x, il suffit d'avoir un hamburger $(burger)$ et des frites $(fries)$ et qu'un hamburger gratuit est fourni avec chaque commande de frites dans les restaurants avec une offre spéciale. Le fait constituant la troisième clause indique que McDonalds participe actuellement à une telle offre.

Les abductibles $fries\setminus 1$, et $bistro\setminus 1$ permettent des suppositions de la forme $fries(t)$ et $bistro(t)$, où t est un « atome de base », mais les contraintes d'intégrité $\mathbf{IC}$ nécessitent que les frites ne soient servies que dans les restaurants qui sont des bistrots. Notre problème est $\mathbf{G} = \{meal(mcDonalds)\}$ et peut être étudié dans l'objectif de trouver une explication au fait "$meal(mcDonalds)$".

Nous pouvons trouver les explications suivantes.

$$\mathbf{E}_1 = \{fries(mcDonalds)\}$$

$$\mathbf{E}_2 = \{fries(mcDonalds), bistro(mcDonalds), bistro(burgerKing)\}$$

$$\mathbf{E}_3 = \{fries(mcDonalds), bistro(mcDonalds)\}$$

Alors que l'explication $\mathbf{E}_1$ ne respecte pas les contraintes d'intégrité, l'explication $\mathbf{E}_2$ est cohérente avec la théorie et les contraintes mais seule l'explication $\mathbf{E}_3$ est minimale et cohérente.

HYPROLOG : Un Langage de Programmation Logique avec Abduction et Hypothèses. HYPROLOG a été créé par Henning Christiansen et Veronica Dahl [CD05, Chr09]. C'est une extension de Prolog et CHR avec la prise en compte de la logique abductive et des hypothèses, qui offre l'une des implémentations les plus efficaces de la programmation logique abductive, même peut-être la plus efficace. Cette efficacité vient du fait que les programmes sont exécutés directement par les systèmes sous-jacents de Prolog et CHR , et par des mécanismes supplémentaires pour l'abduction.

HYPROLOG est mis en oeuvre directement comme une sur-couche de Prolog et CHR et peut utiliser la pleine puissance de ces langages, y compris la notation de la grammaire, les prédicats de base, et tous les solveurs de contraintes disponibles.

HYPROLOG diffère de Prolog et CHR en ayant ses propres déclarations de abductibles et des prédicats pour les hypothèses et quelques autres extensions supplémentaires. Pour autant, il est toujours possible d'exécuter n'importe quel programme existant de Prolog ou CHR sous HYPROLOG.

4.3.2 Inférence de réactions

Muni de ces éléments nous pourrons maintenant développer un programme logique abductive qui infère les réactions sur la base des gènes cibles et les associations de gènes réécrites. Pour rappel le principe de base de *ab-Pantograph* est de considérer le modèle à construire comme un ensemble d'hypothèses qui expliquent la présence des gènes dans le génome cible. En d'autres termes, partant des gènes présents dans l'organisme cible, nous allons considérer qu'une réaction doit être présente dans le modèle cible si les gènes responsables de sa fonctionalié sont présents. Le programme codé en HYPROLOG consiste en trois parties : les faits, les contraintes et les abductibles.

Pour la reconstruction d'un modèle, on dispose de données relatives au modèle de référence (i.e. les réactions et leurs associations de gènes ainsi que l'ensemble des gènes de l'organisme de référence), une fonction d'homologie entre les gènes de l'organisme de référence et ceux de l'organisme cible et l'ensemble des gènes de l'organisme cible.

Nous commençons avec les gènes de l'organisme cible à justifier (i.e. nos observations G), donnés par l'ensemble `Gene_target`, et pour cela nous proposons une théorie abductive (P, A, IC). Le programme P contient l'ensemble des faits et des règles : les faits sont issus du modèle de référence, de la fonction d'homologie et des associations de gènes. Pour chaque réaction r du modèle de référence, on ajoute un fait au programme P décrivant la réaction et incluant l'association de gènes correspondant à cette réaction. Ce

fait est de la forme `r(ref,C,P,E,M,AG)`, où `ref` indique que cette réaction appartient au modèle de référence, `C`, `P`, `E` et `M` sont respectivement les ensembles de substrats, de produits, d'enzymes et de modificateurs pour cette réaction, et enfin `AG` est l'association de gènes correspondante. Par exemple :

```
r(ref,[substrat(s494,1,c_02),substrat(s1434,1,c_02)],
[produce(s369,1,c_02),produce(s495,1,c_01),produce(s764,1,c_02)],
[modifies(s2213,1,c_02),modifies(s2214,b,1,c_02)],[],
ga_or([ylr307w,ylr308w]))
```

représente la réaction *"Id=r-0296 Name=chitin deacetylase"* suivante :

$$\text{s494 + s1434} \xrightarrow{\text{s2213, s2214}} \text{s369 + s495+ s764}$$

avec l'association de gènes `(ylr307w OR ylr308w)`.

L'ensemble des gènes présent dans le modèle de référence correspond à un fait dans P donné comme `Gene_ref`. La fonction d'homologie est définie comme une relation binaire entre chaque liste de gènes de l'organisme de référence et sa liste d'homologues dans l'organisme cible, avec la méthode qui trouve cette relation. Cette relation prend la forme

```
homolog([...ref...],[...targ...],method).
```

par exemple

```
homolog([yhr042w,ypr048w],[yali0c09460g,yali0d04422g],dom).
```

indique que les gènes de référence `yhr042w,ypr048w` ont deux gènes homologues de cibles `yali0c09460g,yali0d04422g` et que cette relation a été prédit par la méthode *dom*.

Les règles dans le programme P correspondent à des règles génériques (i.e. indépendantes du modèle de référence considéré) portant sur les connaissance du domaine.

Par exemple, la règle suivante :

```
rewrite_ga(X1,X2) :- setof(H,homolog(X1,H),[X2]).
```

donne la liste des gènes de l'organisme cible `[X2]` qui sont homologues avec un gène donné `X1` de l'organisme de référence. En effet, elle retourne la réponse `yes` pour deux gènes donnés homologues, et `fail` pour deux gènes donnés mais qui ne sont pas homologues.

Une explication abductive pour un objectif `G` est un ensemble de formules abductibles sans variables, ce qui permet de focaliser les hypothèses émises sur un sous-ensemble d'hypothèses : les hypothèses abductibles. Dans le cadre de la reconstruction de réseaux métaboliques, nos hypothèses aductibles sont de la forme :

```
reaction_in/1 et reaction_instan/1
```

où la solution va prendre la forme

`reaction-in(r(target,C,P,E,M,Ga_target))`

pour les réactions qui satisfont le contraintes données et déduites par réécriture de leurs associations de gènes, ou de la forme

`reaction-instan(r(target,C,P,E,M,true))`

qui donne les réactions déduites directement du fait de l'absence de leurs associations de gènes.

Par conséquent, les espèces moléculaires participant à cette réaction sont ajoutées au modèle cible, ainsi que les compartements dans lesquels ces espèces existent. Quand le modèle de référence présente un ensemble des compartements indépendants, l'instanciation d'un compartement dans le nouveau modèle dépend de l'instanciation de ses espèces, où un compartement sera instancié si une de ces espèces est instancié.

Malgré cette focalisation, il peut y avoir plusieurs explications différentes pour le même objectif. La décision d'adapter certaines explications depend des informations dont nous disposons. Ces informations peuvent être intégrées grace aux contraintes d'intégrité IC et aux contraintes de CHR qui rendent la résolution des IC plus efficace à l'aide de ces trois types de contraintes forcant des relations plus fines entre les hypothèses abductibles.

Par exemple la contrainte :

```
reaction_in(R1),reaction_in(R2) ==> R1= R2 | fail.
```

interdit d'avoir la même réaction plus d'une fois, même si elle a été déduite plusieurs fois en expliquant plusieurs gènes.

A l'issue d'expériences, une liste de réactions peut être donnée par des experts à la méthode de reconstruction pour forcer l'ajout de ces réactions au nouveau modèle sans tenir compte des règles de reconstruction. Cette liste de réactions est ajoutée aux faits de notre programme et nous garantissons leur intégration dans le nouveau modèle par des contraintes.

Dans les cas où une spécialisation des réactions est nécessaire, une réaction du modèle de référence va être de la forme `r(ref,C,P,E,M,ga_or[G1,...,Gn]))` où l'association de gènes est une liste de gènes paralogues (c'est-à-dire de la forme OR[G1,...,Gn]). L'objectif est alors expliquer la liste de gènes [G1,...,Gn] et pour cela nous gènèrons autant d'hypothése que de gènes en considéront chaque gène comme une nouvelle association de gènes à réécrir, cette réécriture nous permet de choisir entre les gènes homologues et donc les hypothèses. Pour chque gène choisit nous génerons une réaction differente avec un e association de gène spécifique. Supposons par exemple la réaction `r(ref,C,P,E,M,ga_or[G1,...,Gn]))` et que les gènes `G1,G2,Gn` sont réécrit dans le cible par les gènes `G1t,G2t,Gnt`, respectivement. Les réactions spécifiques génerées vont être

```
r(target,C,P,E,M,G1t)), r(target,C,P,E,M,G2t))
et r(target,C,P,E,M,Gnt))
```

Cette spécialisation arrête dès il n'existe plus de gènes, dans la format généralisé, à expliquer.

La spécialisation des réactions nécessite peut être avoir des substrats différents pour chaque réaction. Cela nous conduit à une autre question, est-ce que ces substrats sont dans le modèle, où faut-il les ajouter aussi dans ce modèle par abduction ?

4.3.3 Validation

Dans cette section nous argumentons que le modèle inféré est congruent au modèle ideal qu'on pourrait avoir si on dispose au modèle complet M_2, page 3.

Pour cela nous passons sur plusieurs points :

1. Les associations de gènes sont réécrites d'un façon à transférer les connaissances biologiques d'un organisme à un autre.
2. Chaque réaction r du modèle explique au moins un gène présente dans le génome.
3. Le modèle prédit est coherent grâce aux règles logiques.

Les deux derniers points découlent directement des définitions exposées précedemment dans ce chapitre, mais le premier mérite une validation plus approfondie.

Nous avons procédé donc à une validation par analyse de cas, extraits des associations de gène inférées pour la levure oléagineuse *Yarrowia lipolytica* et expertisée par les biologistes [LDNS12]. Nous avons extrait l'ensemble de pairs d'associations de gène entre *S. cerevisiae* et *Y. lipolytica* et "anonymisée" les formules en replaçant les noms de gènes par un marqueur `G`, pour en garder que la structure syntaxique des formules. Nous avons ensuite extrait un ensemble d'exemples typiques, où la structure de la formule a changé, et les convertit en teste unitaires. Chaque exemple définit une réécriture pour l'association de gènes d'une réaction. Ces exemples sont donnés sous la forme

```
sace : formule logique1.
yali : formule logique2.
```

où la formule logique1 indique l'association de gènes, pour une réaction donnée, dans le modèle de référence, et la formule logique2 indique la réécriture de cette association déduite par les règles d'écriture.Par exemple, `unit_rwga14` dans la réaction `r_0114`

```
sace: ga_and([G,G,G,G,G,G])
yali: ga_and([G,G,G,ga_or([G,G]),G,G])
```

Cette association de gènes dans le modèle de référence est une conjonction de six gènes. Cette association est écrite dans les gènes de cible par `ga_and([G,G,G,ga_or([G,G]),G,G]`

La validation de notre méthode par les tests unitaires suivants nous permet de conclure que les associations de gènes sont en effet réécrites d'un façon à transférer les connaissances biologiques d'un organisme à un autre, et donc que $\langle \widetilde{m}_1, \widetilde{a}_1^{\{G_1\}} \rangle \cong \langle m_1, a_1^{\{G_1\}} \rangle$.

```
:- begin_tests(rewrite_ga).
%% :- consult(rewrite_ga_data).

%% Homology data for test tally map
%  ... moved to rewrite_ga_data.pl

%% Unit tests
```

```
% Sanity checks
%
test(unit_rwga0a, [fail]) :-
rewrite_ga( ga_or([]), _ ).
test(unit_rwga0b, [fail]) :-
rewrite_ga( ga_and([]), _ ).
test(unit_rwga0c) :-
rewrite_ga( true, true ).

% Unit test unit_rwga1 gene association rewriting
% from Sace-Yali reaction r_0249
%   sace: true
%   yali: G
%
test(unit_rwga1a, [blocked(what_does_this_mean)]) :-
rewrite_ga( true,
            yali0b22066g ).

test(unit_rwga1b) :-
rewrite_ga( yil160c,
            yali0e18568g ).

% Unit test unit_rwga2 gene association rewriting
% from Sace-Yali reaction r_1143
%   sace: true
%   yali: ga_or([G,G])
%
test(unit_rwga2, [blocked(what_does_this_mean)]) :-
rewrite_ga( true,
            ga_or([yali0d07392g,yali0f30569g]) ).

% Unit test unit_rwga3 gene association rewriting
% from Sace-Yali reaction r_0554
%   sace: G
%   yali: ga_or([G,G,G])
%
test(unit_rwga3a) :-
rewrite_ga( ynl202w,
            ga_or([yali0c03003g, yali0e14322g, yali0d01694g]) ).

test(unit_rwga3b) :-
rewrite_ga( yfl025c,
            ga_or([yali0b10043g,yali0f03927g,yali0f30767g]) ).
```

```
% Unit test unit_rwga4 gene association rewriting
% from Sace-Yali reaction r_29_bh
%   sace: ga_or([ga_and([G,G]),ga_and([G,G])])
%   yali:
%
test(unit_rwga4) :-
rewrite_ga( ga_or([ga_and([ybr036c,ypl057c]),
             ga_and([ybr036c,ybr161w])]), true ).

% Unit test unit_rwga5 gene association rewriting
% from Sace-Yali reaction r_0216
%   sace: ga_or([G,G,G])
%   yali:
%
test(unit_rwga5) :-
rewrite_ga( ga_or([ybr299w,ygr287c,ygr292w]),
            true ).

% Unit test unit_rwga6 gene association rewriting
% from Sace-Yali reaction r_0859
%   sace: ga_and([G,G])
%   yali: G
%
test(unit_rwga6) :-
rewrite_ga( ga_and([ygr240c,ymr205c]),
            yali0d16357g ).

% Unit test unit_rwga7 gene association rewriting
% from Sace-Yali reaction r_0065
%   sace: ga_or([G,G,G])
%   yali: G
%
test(unit_rwga7) :-
rewrite_ga( ga_or([ygr087c,ylr044c,ylr134w]),
            yali0d10131g ).

% Unit test unit_rwga8 gene association rewriting
% from Sace-Yali reaction r_0590
%   sace: ga_and([ga_or([ga_and([G,G]),ga_and([G,G])]),G])
%   yali: G
%
test(unit_rwga8) :-
```

```
rewrite_ga( ga_and([ga_or([ga_and([ydr453c,ygr209c]),
               ga_and([ydr453c,ylr043c])]),ylr043c]), yali0b15125g ).

% Unit test unit_rwga9 gene association rewriting
% from Sace-Yali reaction r_31a_bh
%   sace: ga_or([ga_and([G,G]),ga_and([G,G])])
%   yali: G
%
test(unit_rwga9) :-
rewrite_ga( ga_or([ga_and([ybr036c,ypl057c]),
                ga_and([ybr036c,ybr161w])]), yaliunk4 ).

% Unit test unit_rwga10 gene association rewriting
% from Sace-Yali reaction r_1024
%   sace: ga_and([ga_or([G,ga_and([G,G]),ga_and([G,G])]),G])
%   yali: G
%
test(unit_rwga10) :-
rewrite_ga( ga_and([ga_or([ydr353w,ga_and([ydr353w,ygr209c]),
                        ga_and([ydr353w,ylr043c])]),ylr043c]), yali0

% Unit test unit_rwga11 gene association rewriting
% from Sace-Yali reaction r_0213
%   sace: ga_and([G,G,G])
%   yali: ga_and([G,G])
%
test(unit_rwga11) :-
rewrite_ga( ga_and([ybr126c,yml100w,ymr261c]),
            ga_and([yali0e14685g,yali0e31086g])).

% Unit test unit_rwga12 gene association rewriting
% from Sace-Yali reaction r_0337
%   sace: ga_or([G,G])
%   yali: ga_or([G,G,G])
%
test(unit_rwga12) :-
rewrite_ga( ga_or([yfr055w,ygl184c]),
            ga_or([yali0c22088g,yali0d00605g,yali0f05874g])).

% Unit test unit_rwga13 gene association rewriting
% from Sace-Yali reaction r_1407
%   sace: ga_or([G,G,G,G])
%   yali: ga_or([G,G])
```

```
%
test(unit_rwga13) :-
rewrite_ga( ga_or([ybr298c,ydl247w,ygr289c,yjr160c]),
            ga_or([yali0a14212g,yali0b00396g]) ).

% Unit test unit_rwga14 gene association rewriting
% from Sace-Yali reaction r_0114
%   sace: ga_and([G,G,G,G,G,G])
%   yali: ga_and([G,G,G,ga_or([G,G]),G,G])
%
test(unit_rwga14) :-
rewrite_ga(ga_and([ybr026c,yer061c,yhr067w,ykl055c,ykl192c,yor221c]),
            ga_and([yali0a19096g,yali0c19624g,yali0f30679g,
            ga_or([yali0d14850g,yali0d24629g]),yali0e18590g,
            yali0f29975g]) ).

% Unit test unit_rwga15 gene association rewriting
% from Sace-Yali reaction r_0407
%   sace: ga_or([ga_and([G,G,G,G,G]),ga_and([G,G,G,G,G,G]),
      ga_and([G,G,G,G,G])])
%   yali: ga_or([G,G,G,G,G,G])
%
test(unit_rwga15) :-
rewrite_ga(
ga_or([ga_and([ygl205w,yil160c,ykr009c,ylr284c,ynl202w]),
ga_and([ygl205w,yil160c,ykr009c,ylr284c,ynl202w,yor180c]),
ga_and([ygl205w,yil160c,ykr009c,ynl202w,yor180c])]),
ga_or([yali0c23859g,yali0d24750g,yali0e06567g,yali0e27654g,
yali0e32835g,yali0f10857g]) ).

% Unit test unit_rwga16 gene association rewriting
% from Sace-Yali reaction r_1001
%   sace: ga_or([ga_and([G,G,G,G]),ga_and([G,G,G,G])])
%   yali: ga_and([G,G,G,G])
%
test(unit_rwga16) :-
rewrite_ga( ga_or([ga_and([ydr178w,yjl045w,ykl141w,yll041c]),
ga_and([ydr178w,ykl141w,ykl148c,yll041c])]),
            ga_and([yali0a14784g,yali0d11374g,yali0d23397g,
            yali0e29667g]) ).

% Unit test unit_rwga17 gene association rewriting
% from Sace-Yali reaction r_0347
```

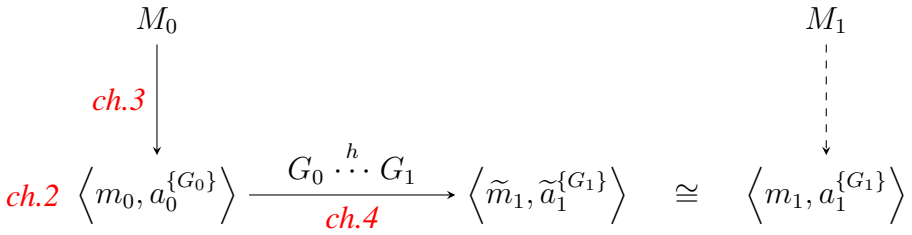
```
%   sace: ga_or([ga_and([G,G,G]),ga_and([G,G,G])])
%   yali: ga_and([G,ga_or([G,G]),G])
%
test(unit_rwga17) :-
rewrite_ga( ga_or([ga_and([yhr007c,yil043c,ynl111c]),
ga_and([yhr007c,ykl150w,ynl111c])]),
           ga_and([yali0b05126g,
           ga_or([yali0d04983g,yali0d11330g]),yali0d12122g])).

%% End of unit tests

:- end_tests(rewrite_ga).

% :- run_tests.
```

Conclusion et perspectives

Conclusion

Dans cette thèse nous avons cherché à définir un cadre logique extensible pour analyser et inférer des modèles métaboliques. Le choix de la logique s'est imposé pour deux raisons. Premièrement, elle permet de définir les règles utilisées pour l'inférence dans un langage de haut niveau, règles dont nous pouvons discuter directement avec les biologistes et faire valider. Cela enlève également une source d'erreurs, la traduction en Python de ces règles. Deuxièmement, elle donne un accès naturel à des méthodes d'inférence sophistiquées, comme l'abduction, sans nous obliger à repenser l'architecture du logiciel.

Au cours de cette thèse, nous avons développé ce cadre logique dans trois chapitres, dont les rôles sont indiqués dans le diagramme suivant :



où M est un modèle complet, et un uplet $\langle m, a^{\{G\}} \rangle$ est sa projection sur un modèle métabolique m annoté par des formules booléennes a définies sur un ensemble de variables G. Ce diagramme montre notre objectif, i.e. le modèle $\langle \widetilde{m}_1, \widetilde{a}_1^{\{G_1\}} \rangle$ que nous inférons est congruent au modèle théorique $\langle m_1, a_1^{\{G_1\}} \rangle$ que nous aurions obtenu si nous avions disposé d'un modèle complet M_1.

Nous avons donc proposé dans le deuxième chapitre une représentation logique de modèles métaboliques annotés, convenable pour les manipulations symboliques que nous voulions faire. Une réaction est définie comme un ensemble de relations avec ses subtrats, produits et molécules modificatrices. On peut considérer que ceci défini implicitement un graphe bipartite, où chaque noeud de type réaction est un ensemble de demi-arcs, le reliant à des noeuds de type substrat, bien que notre manipulation de la structure la traite comme un ensemble de relations et non comme un graphe. La projection d'un modèle complet SBML (ou SBGN) dans notre notation est une simple transformation syntaxique. Cette simplification de M en $\langle m, a^{\{G\}} \rangle$ réduit la complexité et fournit un objet manipulable.

Elle est aussi nécessaire, parce que dans les cas qui nous intéressent nous *n'avons pas* le modèle complet M.

Dans le troisèeme chapitre, nous avons traité la question de l'extraction des connaissances de modèles, au travers de trois exemples et quatre applications. Le modèle simplifié m est annoté avec des formules $a^{\{G\}}$ qui représentent les décisions que nous aurions pu prendre si nous avions disposé de M. En considérant différents exemples réels, nous montrons que la représentation sous formule booléenne doit être différente en fonction de l'application. Cette étude permet de définir les associations de gènes adaptées pour chaque application. Dans le cas de l'inférence ab-Pantograph dans le chapitre suivant, il s'agit de formules booléennes définies sur les gènes dans les génomes des organismes : elles permettent de dire si une réaction peut ou non être présente dans le métabolisme.

Nous avons ensuite dans le quatrième chapitre étudié la question de comment inférer un modèle d'un nouveau organisme cible à partir d'un modèle de référence et d'informations sur l'homologie entre les gènes du nouvel organisme et les gènes de référence. Ce problème est difficile car les relations d'homologie peuvent être complexes, n–m dans les organismes eucaryotes, et les méthodes de prédiction ne sont pas toujours en accord. Une autre difficulté est que les différences entre les génomes engendrent souvent de nouvelles réactions, absentes dans le modèle de référence, qui doivent être inventées. La plupart des méthodes existantes se contentent de recopier des réactions de la référence vers la cible, et de faire appel aux experts humains pendant la curation pour identifier les réactions manquantes. Nous avons considéré en particulier le cas d'expansions d'une famille de gènes, qui suggère qu'une réaction générique doit être déclinée en plusieurs réactions spécialisées. Afin de pouvoir inventer de nouvelles réactions et pour tirer profit de l'approche symbolique adaptée dans cette thèse, nous avons défini une approche par la logique abductive. L'originalité de notre approche est de considérer les gènes comme des observations qui doivent être expliqués, et les réactions comme des hypothèses qui doivent être émises pour les expliquer.

Le système Hyprolog de *Henning Christiansen* est utilisé pour générer des hypothèses à partir d'une théorie de base définie par le modèle métabolique de référence, des contraintes d'intégrité et des observations fournies par les gènes et leurs relations d'homologie multiple. Un programme Prolog avec CHR et Hyprolog a été écrit pour réaliser ces inférences, et appliqué à des génomes complets de levures.

Nous prétendons que le modèle annoté par ses associations de gènes inferées par notre méthode est très proche de ce que nous aurions obtenu dans le cas idéal d'information complète. Ceci est indiqué dans la figure par la congruence $\langle \widetilde{m}_1, \widetilde{a}_1^{\{G_1\}} \rangle \cong \langle m_1, a_1^{\{G_1\}} \rangle$. Nous argumentons cette congruence dans le chapitre 4 en trois points :

1. Les associations de gènes sont réécrites par des règles validées par les biologistes, vérifiées par des tests unitaires extraits de modèles expertisés.
2. Chaque réaction est justifiée parce qu'elle explique au moins un gène, parfois elle en explique plusieurs.
3. Les contraintes d'intégrité écrites dans Hyprolog garantissent que les réactions induites comme hypotheses, ensemble, forment bien un modèle.

Cette méthode ab-Pantograph est au moins aussi bonne que Pantograph, avec la correction en plus qui n'a pas été démontrée dans [Loi12]. Si nous nous en satisfaisons, notre méthode et son programme Prolog ont été définis de manière à permettre leur amélioration par l'ajout des modules d'inférence plus élaborés. Quelques pistes sont présentées dans la suite.

Perspectives

Les contributions de cette thèse conduisent à un certain nombre de perspectives intéressantes à explorer.

Les annotations $a^{\{G\}}$ que nous avons développées ici ont été directement inspirées des associations de gènes couramment utilisés dans les modèles de référence, de *S. cerevisiae* en particulier, et exploitées par [Loi12]. Elles permettent de prendre les décisions nécessaires à partir desquelles nous voulons inférer un modèle pour une nouvelle espèce, où les gènes qui sont différents. Cependant, la méthode n'est pas adaptée quand les différences sont plus petites. Dans le cas de la modélisation de souches, les différences entre les organismes peuvent être seulement quelques SNP, ou un réarrangement génomique qui perturbe l'expression d'un gène critique. Il est donc très important de considérer comment la représentation des connaissances, codée dans les formules, pourrait être améliorée. S'il est facile de voir comment l'ensemble de variables $\{G\}$ peut être étendu, par exemple pour inclure les SNP, l'expression de gènes ou des mesures expérimentales de phénotype, il l'est moins de voir comment étendre les formules $a^{\{G\}}$. La démarche que nous avons suivie dans le chapitre 3 montre comment procéder : il faut analyser des cas réels pour déterminer quelles fonctions de décision doivent être associées à chaque réaction, et comment les représenter par des formules qui peuvent être manipulées symboliquement. Des cas réels comme ceux-ci doivent être étudiés en collaboration avec des biologistes, experts dans les organismes considérés.

En général, des modules logiciels plus sophistiqués peuvent être utilisé dans plusieurs endroits dans le programme. Par exemple, les différentes prédictions d'homologie sont reconciliées en utilisant le *tally map* de Pantograph. C'est un problème qui se prête à l'apprentissage supervisé, si une grande collection d'exemples validés peut être construite. Pour un autre exemple, les formules logiques peuvent être analysées plus profondement, en considérant les ensembles de valeurs que peuvent prendre les variables.

Une dernière piste, que nous étudions actuellement en collaboration avec Anna Zhukova et Nicolas Loira, porte sur l'utilisation de modèles généralisés [ZS14]. Le modèle de référence dans nos travaux ici est un modèle fonctionnel d'un organisme particulier. La généralisation permet de combiner un ensemble de modèles similaires dans une structure où certaines réactions similaires sont regroupées en une réaction plus générique qui est une borne supérieure dans un treillis de réactions. La généralisation est calculée par une factorisation du graphe bipartite de réactions. Utiliser la généralisation comme référence est intéressant parce que cela permet d'éviter un *overfitting* dû au choix de l'organisme utilisé comme référence. L'idée que nous développons est d'utiliser ab-Pantograph pour

"dérouler" la généralisation en fonction des gènes observés. La réaction générique est spécialisée pour expliquer l'ensemble de gènes enzymatiques. La piste explorée est de dérouler librement la réaction générique tant qu'il reste des gènes à expliquer dans le groupe de paralogues, et de définiro des contraintes d'intégrité qui arrêtent le déroulement quand cela introduira de la redondance dans le modèle.

Bibliographie

[ABCC94] André Arnold, Didier Bégay, Paul Crubillé, and P Crubille. *Construction and analysis of transition systems with MEC*. Number 3 in AMAST series in computing. World Scientific, 1994.

[AC01] Slim Abdennadher and Henning Christiansen. An experimental clp platform for integrity constraints and abduction. In *Flexible Query Answering Systems*, pages 141–152. Springer, 2001.

[ACDL+07] Iliana Avila-Campillo, Kevin Drew, John Lin, David J Reiss, and Richard Bonneau. Bionetbuilder : automatic integration of biological networks. *Bioinformatics*, 23(3) :392–393, 2007.

[AGS+11] Rodrigo Assar, Alice Garcia, David James Sherman, et al. Modeling stochastic switched systems with biorica. In *Journées Ouvertes en Biologie, Informatique et Mathématiques JOBIM 2011*, pages 297–304, 2011.

[AJW+08] Michiel E Adriaens, Magali Jaillard, Andra Waagmeester, Susan LM Coort, Alex R Pico, and Chris TA Evelo. The public road to high-quality curated biological pathways. *Drug discovery today*, 13(19) :856–862, 2008.

[AKV+04] E_ Almaas, B Kovacs, T Vicsek, ZN Oltvai, and A-L Barabási. Global organization of metabolic fluxes in the bacterium escherichia coli. *Nature*, 427(6977) :839–843, 2004.

[ALS+13] Rasmus Agren, Liming Liu, Saeed Shoaie, Wanwipa Vongsangnak, Intawat Nookaew, and Jens Nielsen. The raven toolbox and its use for generating a genome-scale metabolic model for penicillium chrysogenum. *PLoS computational biology*, 9(3) :e1002980, 2013.

[APGR99] André Arnold, Gérald Point, Alain Griffault, and Antoine Rauzy. The altarica formalism for describing concurrent systems. *Fundam. Inf.*, 40(2,3) :109–124, August 1999.

[Bai00] Amos Bairoch. The enzyme database in 2000. *Nucleic acids research*, 28(1) :304–305, 2000.

[BBV07] Kevin Bleakley, Gérard Biau, and Jean-Philippe Vert. Supervised reconstruction of biological networks with local models. *Bioinformatics*, 23(13) :i57–i65, 2007.

[BGL+14] Pierre Blavy, Florence Gondret, Sandrine Lagarrigue, Jaap Van Milgen, and Anne Siegel. Using a large-scale knowledge database on reactions and

regulations to propose key upstream regulators of various sets of molecules participating in cell metabolism. *BMC systems biology*, 8(1) :32, 2014.

[BHRP05] Christian L Barrett, Christopher D Herring, Jennifer L Reed, and Bernhard O Palsson. The global transcriptional regulatory network for metabolism in escherichia coli exhibits few dominant functional states. *Proceedings of the National Academy of Sciences of the United States of America*, 102(52) :19103–19108, 2005.

[Boy04] Frédéric Boyer. *Reconstruction ab initio de voies métaboliques-Formalisation et approches combinatoires*. PhD thesis, Université Joseph-Fourier-Grenoble I, 2004.

[C+11] UniProt Consortium et al. Ongoing and future developments at the universal protein resource. *Nucleic acids research*, 39(suppl 1) :D214–D219, 2011.

[CD05] Henning Christiansen and Veronica Dahl. Hyprolog : A new logic programming language with assumptions and abduction. In *Logic Programming*, pages 159–173. Springer, 2005.

[CdVK+12] Ning Chen, Ioscani Jimenez del Val, Sarantos Kyriakopoulos, Karen M Polizzi, and Cleo Kontoravdi. Metabolic network reconstruction : advances in in silico interpretation of analytical information. *Current opinion in biotechnology*, 23(1) :77–82, 2012.

[CFF+06] Ron Caspi, Hartmut Foerster, Carol A Fulcher, Rebecca Hopkinson, John Ingraham, Pallavi Kaipa, Markus Krummenacker, Suzanne Paley, John Pick, Seung Y Rhee, et al. Metacyc : a multiorganism database of metabolic pathways and enzymes. *Nucleic acids research*, 34(suppl 1) :D511–D516, 2006.

[Chr09] Henning Christiansen. Executable specifications for hypothesis-based reasoning with prolog and constraint handling rules. *Journal of Applied Logic*, 7(3) :341–362, 2009.

[CKR+04] Markus W Covert, Eric M Knight, Jennifer L Reed, Markus J Herrgard, and Bernhard O Palsson. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987) :92–96, 2004.

[CR14] Alain Colmerauer and Philippe Roussel. *La naissance de Prolog*. Éditions universitaires européennes, 2014.

[DFB+07] Matthew DeJongh, Kevin Formsma, Paul Boillot, John Gould, Matthew Rycenga, and Aaron Best. Toward the automated generation of genome-scale metabolic networks in the seed. *BMC bioinformatics*, 8(1) :139, 2007.

[DK02] Marc Denecker and Antonis Kakas. Abduction in logic programming. In *Computational Logic : Logic Programming and Beyond*, pages 402–436. Springer, 2002.

[DOD+13] Scott Devoid, Ross Overbeek, Matthew DeJongh, Veronika Vonstein, Aaron A Best, and Christopher Henry. Automated genome annotation and metabolic model reconstruction in the seed and model seed. In *Systems Metabolic Engineering*, pages 17–45. Springer, 2013.

[FA03] Thom Frühwirth and Slim Abdennadher. *Essentials of constraint programming*. Springer, 2003.

[Fan70] Kuang Tih Fann. *Peirce's theory of abduction*. Springer, 1970.

[FGAT09] Nicholas Furnham, John S Garavelli, Rolf Apweiler, and Janet M Thornton. Missing in action : enzyme functional annotations in biological databases. *Nature chemical biology*, 5(8) :521–525, 2009.

[FHK+92] Thom Frühwirth, Alexander Herold, Volker Küchenhoff, Thierry Le Provost, Pierre Lim, Eric Monfroy, and Mark Wallace. *Constraint logic programming*. Springer, 1992.

[FHT+09] Adam M Feist, Markus J Herrgård, Ines Thiele, Jennie L Reed, and Bernhard Ø Palsson. Reconstruction of biochemical networks in microorganisms. *Nature Reviews Microbiology*, 7(2) :129–143, 2009.

[FP08] Adam M Feist and Bernhard Ø Palsson. The growing scope of applications of genome-scale metabolic reconstructions using escherichia coli. *Nature biotechnology*, 26(6) :659–667, 2008.

[FR09] Thom Frühwirth and Frank Raiser. *Constraint handling rules*, volume 3. Springer, 2009.

[Frü98] Thom Frühwirth. Theory and practice of constraint handling rules. *The Journal of Logic Programming*, 37(1-3) :95–138, 1998.

[FST05] Christof Francke, Roland J Siezen, and Bas Teusink. Reconstructing the metabolic network of a bacterium from its genome. *Trends in microbiology*, 13(11) :550–558, 2005.

[FXCT12] Xueyang Feng, You Xu, Yixin Chen, and Yinjie J Tang. Microbesflux : a web platform for drafting metabolic models from the kegg database. *BMC systems biology*, 6(1) :94, 2012.

[HFS+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. *Bioinformatics*, 19(4) :524–531, 2003.

[JLP11] Paul A Jensen, Kyla A Lutz, and Jason A Papin. Tiger : Toolbox for integrating genome-scale metabolic models, expression data, and transcriptional regulatory networks. *BMC systems biology*, 5(1) :147, 2011.

[KDWV13] S Krishnakumar, Dilip A Durai, Pramod P Wangikar, and Ganesh A Viswanathan. Sharp : genome-scale identification of gene–protein–reaction associations in cyanobacteria. *Photosynthesis research*, 118(1-2) :181–190, 2013.

[KG00] Minoru Kanehisa and Susumu Goto. Kegg : kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1) :27–30, 2000.

[KKT92] Antonis C. Kakas, Robert A. Kowalski, and Francesca Toni. Abductive logic programming. *Journal of logic and computation*, 2(6) :719–770, 1992.

[KPR02] Peter D Karp, Suzanne Paley, and Pedro Romero. The pathway tools software. *Bioinformatics*, 18(suppl 1) :S225–S232, 2002.

[KSK+12] Tae Yong Kim, Seung Bum Sohn, Yu Bin Kim, Won Jun Kim, and Sang Yup Lee. Recent advances in reconstruction and applications of genome-scale metabolic models. *Current opinion in biotechnology*, 23(4) :617–623, 2012.

[KSM12] Akhil Kumar, Patrick F Suthers, and Costas D Maranas. Metrxn : a knowledgebase of metabolites and reactions spanning metabolic models and databases. *BMC bioinformatics*, 13(1) :6, 2012.

[LDNS12] Nicolas Loira, Thierry Dulermo, Jean-Marc Nicaud, and David Sherman, James. A genome-scale metabolic model of the lipid-accumulating yeast Yarrowia lipolytica. *BMC Systems Biology*, 6(1) :35, 2012.

[LNHM+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. *Nature biotechnology*, 27(8) :735–741, 2009.

[LNMS+08] Nicolas Le Novere, Stuart Moodie, Anatoly Sorokin, Michael Hucka, Falk Schreiber, Emek Demir, Huaiyu Mi, Yukiko Matsuoka, Katja Wegner, and Hiroaki Kitano. Systems biology graphical notation : process diagram level 1. *Nature Precedings*, 2008.

[Loi12] Nicolas Loira. *Scaffold-based Reconstruction Method for Genome-Scale Metabolic Models*. PhD thesis, Université Sciences et Technologies-Bordeaux I, 2012.

[LZS15] Nicolás Loira, Anna Zhukova, and David James Sherman. Pantograph : A template-based method for genome-scale metabolic model reconstruction. *J Bioinform Comput Biol.*, 13(2) :1550006, 2015. doi :10.1142/S0219720015500067.

[MM97] Alan D McNaught and Alan D McNaught. *Compendium of chemical terminology*, volume 1669. Blackwell Science Oxford, 1997.

[MS98] Kim Marriott and Peter J Stuckey. *Programming with constraints : an introduction*. MIT press, 1998.

[NS07] Macha Nikolski and David James Sherman. Family relationships : should consensus reign ?- consensus clustering for protein families. *Bioinformatics*, 23 :e71–e76, 2007.

[ORS05] Kevin P O'Brien, Maido Remm, and Erik LL Sonnhammer. Inparanoid : a comprehensive database of eukaryotic orthologs. *Nucleic acids research*, 33(suppl 1) :D476–D480, 2005.

[PAR+08] Esa Pitkänen, Arto Akerlund, Ari Rantanen, Paula Jouhten, and Esko Ukkonen. Rematch : a web-based tool to construct, store and share stoichiometric metabolic models with carbon maps for metabolic flux analysis. *J Integr Bioinform*, 5(2) :2008–2102, 2008.

[Pei55] Charles Sanders Peirce. *Philosophical writings of Peirce*. Courier Dover Publications, 1955.

[Pei74] Charles Sanders Peirce. *Collected papers of charles sanders peirce*, volume 5. Harvard University Press, 1974.

[PK02] Suzanne M Paley and Peter D Karp. Evaluation of computational metabolic-pathway predictions for helicobacter pylori. *Bioinformatics*, 18(5) :715–724, 2002.

[PPL+06] Csaba Pál, Balázs Papp, Martin J Lercher, Péter Csermely, Stephen G Oliver, and Laurence D Hurst. Chance and necessity in the evolution of minimal metabolic networks. *Nature*, 440(7084) :667–670, 2006.

[PSMW05] John W Pinney, Martin W Shirley, Glenn A McConkey, and David R Westhead. metashark : software for automated metabolic network prediction from dna sequence and its application to the genomes of plasmodium falciparum and eimeria tenella. *Nucleic acids research*, 33(4) :1399–1409, 2005.

[Ray05] Oliver Ray. *Hybrid abductive inductive learning*. PhD thesis, Citeseer, 2005.

[RCO+13] Predrag Radivojac, Wyatt T Clark, Tal Ronnen Oron, Alexandra M Schnoes, Tobias Wittkop, Artem Sokolov, Kiley Graim, Christopher Funk, Karin Verspoor, Asa Ben-Hur, et al. A large-scale evaluation of computational protein function prediction. *Nature methods*, 10(3) :221–227, 2013.

[RK06] Oliver Ray and Antonis Kakas. Prologica : a practical system for abductive logic programming. In *Proceedings of the 11th International Workshop on Non-monotonic Reasoning*, pages 304–312. Citeseer, 2006.

[SCM14] Rajib Saha, Anupam Chowdhury, and Costas D Maranas. Recent advances in the reconstruction of metabolic models and integration of omics data. *Current opinion in biotechnology*, 29 :39–45, 2014.

[SZ04] Jibin Sun and An-Ping Zeng. Identics–identification of coding sequence and in silico reconstruction of the metabolic network directly from unannotated low-coverage bacterial genome sequence. *BMC bioinformatics*, 5(1) :112, 2004.

[TPVP05] Ines Thiele, Nathan D Price, Thuy D Vo, and Bernhard Ø Palsson. Candidate metabolic network states in human mitochondria impact of diabetes,

ischemia, and diet. *Journal of Biological Chemistry*, 280(12) :11683–11695, 2005.

[VH91] Pascal Van Hentenryck. Constraint logic programming. *The Knowledge Engineering Review*, 6(03) :151–194, 1991.

[vHWGW02] Jacques van Helden, Lorenz Wernisch, David Gilbert, and SJ Wodak. Graph-based analysis of metabolic networks. In *Bioinformatics and genome analysis*, pages 245–274. Springer, 2002.

[WLT05] James D Watson, Roman A Laskowski, and Janet M Thornton. Predicting protein function from sequence and structural data. *Current opinion in structural biology*, 15(3) :275–284, 2005.

[ZS14] Anna Zhukova and David James Sherman. Knowledge-based generalization of metabolic models. *Journal of Computational Biology*, 21(7) :534–547, 2014.

[ZSRM12] Ali R Zomorrodi, Patrick F Suthers, Sridhar Ranganathan, and Costas D Maranas. Mathematical optimization applications in metabolic networks. *Metabolic engineering*, 14(6) :672–686, 2012.