

Thèse de doctorat

Spécialité : Informatique

Anasthasie Joëlle COMPAORÉ

Vers un environnement générique pour la prise en compte de la topologie des systèmes cellulaires dans les modèles de processus biologiques.

Soutenu le 07 mars 2012 à l'Université d'Évry Val d'Essonne devant le jury composé de :

Mme. SERENA CERRITO, Professeur des universités Université d'Evry Val d'Essonne	Président du jury
M. DAVID CAZIER, Maître de conférence à l'Université de Strasbourg 1, LSIIT	Rapporteur du jury
M. JEAN-PAUL COMET, Professeur des universités Université de NICE	Rapporteur du jury
Mme. AGNES ARNOULD, Maître de conférences Université de Poitiers	Membre du jury
M. XAVIER URBAIN, Maître de conférences ENSIIE	Membre du jury
Mme. PASCALE LE GALL, Professeur des universités Université d'Évry, École Centrale Paris	Directrice de thèse
M. BLAISE SOME, Professeur des universités Université de Ouagadougou	Co-directeur de thèse

Résumé et abstract

RÉSUMÉ

Le fonctionnement des systèmes cellulaires est largement conditionné par leurs topologies, c'est-à-dire les compartiments (caractérisés chacun par un type) qui les composent et les dispositions relatives les uns par rapport aux autres de ces compartiments. Cependant, on constate généralement une relative faiblesse dans la prise en compte de la topologie dans certains outils à base de règles pour la modélisation des processus biologiques, ces outils intégrant par ailleurs de bonnes capacités de simulation et d'analyse de modèles. Les travaux présentés dans cette thèse proposent une approche de modélisation qui consiste à définir les modèles en considérant d'une part un ensemble de règles génériques traduisant les comportements des types de compartiments en fonction des molécules et d'autre part l'abstraction d'une compartimentation cellulaire par un graphe d'échanges qui donne en particulier les voisinages existant entre les différents compartiments. Ces deux éléments sont couplés, ce qui permet d'obtenir un modèle intermédiaire qui est le résultat de l'instanciation des règles génériques en fonction des contraintes issues du graphe d'échanges. La traduction de ce modèle intermédiaire vers le langage de règles d'un outil cible (choisi entre BIOCHAM et PATHWAY LOGIC) constitue la dernière étape de modélisation et permet d'utiliser les capacités de simulation et d'analyse de modèles de cet outil.

ABSTRACT

The functioning of cellular systems is widely conditioned by their topology, that is compartments (characterized each by a type) which compose them and the relative position of these compartments with regard to the others. However, we notice generally a relative weakness in the consideration of the topology in certain tools for the rule-based modelling of biological processes, these tools integrating besides good capacities of models simulation and analysis. The works presented in this thesis propose an approach which consists in defining the models by considering on one hand a set of generic rules translating the behavior of the types of compartments according to molecules and on the other hand the abstraction of a cellular compartmentation by an exchanges graph which gives in particular the neighborhood existing between the various compartments. These two elements are coupled, what allows the obtention of an intermediate model which is the result of the instanciation of the generic rules according to the constraints stemming from the exchanges graph. The translation of this intermediate model towards the language of rules of a target tool (chosen between BIOCHAM and PATHWAY LOGIC) constitutes the last stage of the modelling and allows to use the capacities of simulation and analysis of the chosen tool.

Avant-Propos

Tant de personnes de bonne volonté, tant morales que physiques ont contribué chacune à son niveau à l'élaboration de ce mémoire de thèse. Je ne suis pas sûre de trouver les mots justes pour leur témoigner ma reconnaissance ; aussi, je voudrais tout simplement, à travers ces quelques lignes, leur dire merci pour leur engagement et pour tout ce qu'elles ont fait. En écrivant cette page, je pense tout particulièrement à la coopération Française qui à travers le Service de Coopération et d'Action Culturelle (SCAC) de l'ambassade de France au Burkina Faso, a financièrement soutenu pour une grande part ces travaux de recherche en m'octroyant une bourse de mobilité avec tous les avantages (prise en charge des frais d'inscription, séjours de quatre mois en France chaque année pendant trois ans, frais de reprographie de la thèse etc) que cela suppose. Je pense également à la région Île-de-France qui dans le cadre de cette thèse m'a fait bénéficier d'une subvention qui m'a permis d'effectuer deux voyages en France en plus de ceux initialement prévus. Je pense enfin au Centre Muraz de Bobo, la structure qui m'emploie depuis dix ans maintenant qui m'a ménagé du temps libre pour travailler à cette thèse.

A côté de ces institutions, je tiens également à remercier Mme Martinez et le docteur Philippe MSehatti qui m'ont apporté leur soutien pour la préparation du dossier de demande de la bourse.

Dédicace

A papa

A maman

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué à la réalisation de cette thèse de doctorat.

En premier lieu, je remercie Mme Pascale LE GALL, ma directrice de thèse qui est une personne exceptionnelle. Elle m'a donné l'occasion de traiter d'un sujet intéressant, à l'intersection de plusieurs disciplines. Elle s'est toujours montrée disponible pour m'orienter et répondre à mes questions. Elle a également fait preuve d'une grande compréhension quand le travail n'avancait pas selon le planning prévu. Pour finir, elle m'a donné l'occasion de rencontrer sa charmante famille que je remercie pour le chaleureux accueil qu'elle m'a toujours réservé lors de mes séjours en France.

Je remercie Mme Agnès ARNOULD pour sa grande disponibilité, sa réactivité et sa gentillesse. Elle a une très bonne maîtrise de la modélisation géométrique et m'a enseignée tout ce que je sais en la matière. Aujourd'hui grâce à elle, la modélisation géométrique est un domaine informatique que j'aimerais étudier un peu plus.

Je remercie M. Xavier URBAIN qui est un maître de la réécriture et de la compilation. J'ai beaucoup appris dans ces domaines grâce à lui. Il s'est toujours montré disponible et a su orienter mes réflexions dans la mise en œuvre des outils d'analyse syntaxique.

Je remercie M. Blaise SOMÉ de l'Université de Ouagadougou pour sa disponibilité et ses conseils. Il a été mon enseignant il y a 20 ans et je suis contente qu'il ait accepté de co-diriger cette thèse.

Après ces personnes à qui je présente toutes mes excuses pour le manque de communication et de réactivité de ma part, je voudrais dire merci :

- aux deux rapporteurs de cette thèse, M. Jean-Paul COMET et M. David CAZIER, pour avoir accepté ce rôle et surtout pour ce qu'ils ont écrit : leurs rapports témoignent de toute l'attention qu'ils ont mis à les rédiger ;
- à M. François KÉPÈS le responsable du programme d'Épigénomique qui m'a accueillie dans son laboratoire et qui m'a gentiment offert un roman comme souvenir du jour de ma soutenance ;
- à Mlle. MBarka MABROUKI pour sa grande gentillesse et ses encouragements durant ces années de thèse ; elle m'a fait découvrir la vie parisienne et nous avons partagé beaucoup de moments de joies et de fous rires. Elle est devenue pour moi une grande amie ;
- À M. DEVILLERS que j'ai rencontré à l'Église, et à sa famille pour leur sympathie : ils m'ont plusieurs fois invitée à partager leur repas et m'ont deux fois conduite à l'aéroport pour mes départs ;
- à mes amis Burkinabès vivant dans la Région Parisienne :
 - Esther et Serge
 - Haby et Karim
 - Mariam et Oussoumany

-
- Ibrahim
 - Daouda et Alizéta

Mes remerciements vont enfin :

- à M. GAUDEAU de la coopération Française qui nous a gentiment reçu à l’ambassade et a accordé une oreille favorable à notre demande de complément de bourse ;
- à Mme Annick GIRAUDEAU et Mme Nathalie GAUCHY qui ont géré efficacement chacune à son niveau toutes les questions administratives liées à la bourse ;
- à Mme Florence HAMON et Mme Carole TROUSSIER qui ont fait montre d’une gentillesse et d’un grand professionnalisme dans la gestion des formalités pédagogiques et administratives à l’Université d’Évry ;
- au professeur Jean-Bosco OUEDRAOGO, directeur général du Centre MURAZ : il m’a toujours encouragée et m’a laissé beaucoup de temps pour la thèse ;
- au docteur Nicolas MEDA, directeur scientifique du Centre MURAZ qui a donné son accord pour mon inscription au DEA, ce qui m’a ouvert la voie vers la thèse ;
- à mes collègues de travail et amis : Jérémie, Emmanuel, Serge et Ibrahim Ballo, Biba et Françoise, Olga, Florence, Marie-Jeanne, Antoinette, Diane, Sophie, Thérèse, Éric et Bernadette pour leur soutien et leurs encouragements ;
- à mes parents pour leurs prières, leurs encouragements et la confiance qu’ils ont toujours eu en mes capacités ;
- à mes frères et sœurs pour leur soutien et leur intéressement à mon travail :
 - Nathalie et Salif, qui n’auraient pas pu s’investir plus ;
 - Estelle, pour ses encouragements et sa bonne humeur ;
 - Gaëtan, pour ses conseils pour la présentation ;
 - Bertrand pour les avis qu’il m’a donnés en matière de Biologie et pour avoir toujours été disponible pour les formalités administratives à l’Université de Ouagadougou ;
 - Serge pour ses mots d’encouragements ;
 - Nadia pour tous ses coups de fils à travers lesquels elle me transmettait son optimisme et sa joie de vivre ;
- à ma belle-mère qui s’est bien occupée de mes enfants pendant mes séjours en France ;
- à Seddik et Sarah, mes chers enfants : j’espère que le bénéfice de cette thèse compensera un tant soit peu, les absences répétées de leur mère, qu’ils ont dû, très jeunes, vivre ;
- à Balla pour m’avoir donné la permission d’effectuer cette thèse, . . . et pour tellement d’autres choses que je ne détaillerai pas.

Table des matières

1	Introduction	1
2	Approche suivie	11
2.1	BIOCHAM	13
2.1.1	Les objets formels manipulés	14
2.1.2	Les composantes des règles biochimiques	16
2.1.3	Prise en compte de la localisation	19
2.2	PATHWAY LOGIC	20
2.2.1	Aperçu de la spécification algébrique : Les objets manipulés	21
2.2.2	Les composantes des règles de réaction	23
2.2.3	Prise en compte de la compartimentation cellulaire	25
2.3	Analyse des besoins	26
2.3.1	Exemple d'illustration de notre approche	29
2.3.2	Graphe d'échanges et compartimentation cellulaire	31
2.3.3	Langage des règles génériques	31
3	Modélisation des compartiments cellulaires	37
3.1	Modélisation géométrique	38
3.1.1	Quelques modèles	39
3.1.2	Modèles géométriques à base topologique	42
3.1.3	Graphes d'incidence, d'adjacence et arbres d'inclusion	43
3.1.4	Les cartes généralisées	46
3.2	Graphe d'échanges	52

3.2.1	Rappels sur les graphes	53
3.2.2	Principales caractéristiques et définition du graphe d'échanges	56
3.2.3	Opérations applicables dans un graphe d'échanges	58
3.3	Modélisation des compartiments cellulaires	60
3.3.1	Le modèle bio-géométrique	60
3.3.2	Opérations de création des compartiments cellulaires	63
3.3.3	Extraction du graphe d'échange	64
3.4	Implémentation	65
3.4.1	Présentation de MOKA	66
3.4.2	Construction d'une structure cellulaire	68
3.4.3	Extraction d'un graphe d'échanges à partir d'un modèle bio-géométrique	75
3.5	Application	78
3.6	Conclusion et perspectives	82
4	Langage de règles génériques	83
4.1	Les règles génériques	84
4.1.1	Composantes d'une règle générique	86
4.1.2	Le modèle générique	93
4.2	Règles intermédiaires	95
4.2.1	Notions de substitution de termes	96
4.2.2	Les données d'illustration	100
4.2.3	La procédure de couplage règles génériques et graphe d'échanges	101
4.3	Traduction des règles intermédiaires	107
4.3.1	Traduction vers BIOCHAM	108
4.3.2	Traduction vers PATHWAY LOGIC	110
4.3.3	Exemple : traduction du modèle intermédiaire donné par <i>fichInterm3</i>	114
4.4	Implémentation	116
4.4.1	Le modèle générique	116
4.4.2	Implantation de l'instanciation et de la traduction	118

5	Validation à partir de premiers exemples	121
5.1	Validation à partir de modèles BIOCHAM	122
5.1.1	Les processus biologiques modélisés	123
5.1.2	Compartimentation cellulaire et graphe d'échanges	124
5.1.3	Règles génériques	125
5.1.4	Application de l'approche de couplage/traduction	128
5.1.5	Bilan	135
5.2	Validation à partir d'un modèle PATHWAY LOGIC	136
5.2.1	Processus modélisé : premières étapes de l'activation de Ras	136
5.2.2	compartimentation cellulaire et graphe d'échanges	137
5.2.3	Les règles génériques	138
5.2.4	Application de l'approche de couplage/traduction	140
5.2.5	Commentaires	142
5.3	Perspectives	142
A	Éléments de syntaxe concrète	149
A.1	Le modèle générique	149
A.2	Syntaxe des règles génériques	150
B	Parties immuables des fichiers d'un modèle PL	153
B.1	theops.maude	153
B.2	modOtherOps.maude	155
B.3	modCompnents.maude	156
B.4	modRules.maude	156
C	Modèles pour la validation de l'approche	157
C.1	Modèles initiaux BIOCHAM	157
C.2	Modèle initial PATHWAY LOGIC	161

D Quelques fichiers entêtes de classes	167
D.1 Classe CCBio	167
D.2 Classe CExchangeGraph	168
D.3 Classe CModeleGenerique	170
D.4 Classe CRegleGenerique	172
D.5 Classe CMolecule	174
E Fichier d'extraction du graphe d'échanges : informations topologiques	177

Chapitre 1

Introduction

Si la bio-informatique a permis entre autres, d'acquérir et d'organiser dans des bases de données, d'importantes quantités de données biologiques, le recours à l'informatique (et aux mathématiques) est plus que jamais nécessaire pour l'élucidation des phénomènes biologiques. En effet, malgré les (ou en raison des) grandes masses de données biologiques disponibles, les comportements des systèmes biologiques restent mal maîtrisés et leur compréhension constitue un pôle de recherche très actif pour les biologistes. Cela est imputable au fait que les systèmes biologiques font partie des systèmes dits *systèmes complexes* [?, ?] qui sont principalement caractérisés par un grand nombre d'entités en interaction et un comportement global émergent, donc, *non naturellement prédictible* à partir des comportements isolés des composantes de base du système. L'existence de propriétés émergentes est l'une des caractérisations possibles pour les systèmes complexes, y compris pour les systèmes industriels ou informatiques [?]. L'absence d'un cadre unanimement reconnu pour expliquer les phénomènes biologiques, amène les biologistes à s'aider d'outils de simulation et d'analyse pour appréhender le comportement de ces systèmes. Cela nécessite une démarche préalable de modélisation.

Dans le présent document, nous nous intéressons aux modèles de processus biologiques qui tiennent compte des structures topologiques des systèmes sous étude. En cela, nous nous inscrivons dans le domaine de la *biologie des systèmes*, et plus précisément, dans le cadre de la *modélisation à base de règles des interactions de molécules dans les systèmes biologiques cellulaires*. Dans ce chapitre, nous commencerons par présenter de façon succincte les éléments biologiques qui font partie du contexte des modélisations discutées dans le manuscrit. Ensuite, nous évoquerons les principales options adoptées dans les modélisations.

Principaux éléments constitutifs des cellules eucaryotes

L'organisation structurale des cellules biologiques est aujourd'hui assez bien connue. Il est unanimement admis que la cellule biologique est un *cloisonnement délimité par une membrane* dite *membrane plasmique*. À la différence des cellules procaryotes, êtres unicellulaires sans noyau (par exemple, la bactérie *Escherichia Coli*), les cellules eucaryotes sont constituées d'un noyau délimité par une membrane et d'un cytoplasme contenant de nombreuses entités, appelées organites (ou organelles) également séparées du reste de la cellule par une membrane (nous utiliserons le terme *bio-membrane* comme terme générique pour désigner toute membrane d'un système cellulaire). Ces organites jouent des rôles précis : l'appareil de Golgi permet

la maturation et l'excrétion des protéines ; les mitochondries participent à la transformation du glucose en énergie disponible pour la cellule (sous la forme d'ATP), ...

Les réactions chimiques ayant lieu au sein des cellules sont aussi appelées réactions biochimiques. Ces réactions qui peuvent être catalysées (par des enzymes) permettent la production d'énergie (ATP), la synthèse de métabolites (protéines) à partir des éléments de base fournis par l'alimentation (réactions d'anabolisme), ou au contraire, la dégradation de constituants moléculaires (catabolisme). Le métabolisme regroupe les réactions de synthèse et de dégradation de molécules. Les réactions biochimiques sont des réactions rapides, souvent réversibles en lien avec les taux de concentration des molécules concernées (substrats et enzymes). Elles sont régies par le principe de complémentarité moléculaire : les liaisons covalentes, en permettant à deux atomes de partager les mêmes électrons, expliquent la constitution de molécules stables à partir d'entités séparées, tandis que les liaisons plus faibles (Forces électrostatiques, liaison hydrogène, interactions de van der Waals) expliquent l'importance des structures tridimensionnelles des molécules en présence et les grandes possibilités de transformations de molécules (souplesse de création et rupture de ces liaisons). Les relations faibles sont typiques des interactions biochimiques : à titre d'exemples représentatifs, l'appariement complémentaire des bases entre deux brins d'ADN, ou le repliement des protéines sont expliqués par la présence de telles liaisons faibles.

L'eau joue un rôle prépondérant : elle est le principal solvant, notamment dans le cytosol, milieu liquide constituant le cytoplasme, pour les molécules en jeu dans les réactions biochimiques, et elle régit la configuration des bio-membranes qui sont des bicouches lipidiques formées de deux feuilletts distincts, (avec des compositions en lipides différentes selon [?]), avec les parties hydrophobes prises à l'intérieur de la (double) membrane.

Dans la cellule, des milliers de molécules de types différents sont susceptibles d'être modifiées par les réactions biochimiques. Comme la plupart des fonctions cellulaires (régulation, structure, transport, signalisation, mouvement) sont assurées par les protéines (macromolécules constituées de chaînes d'acides aminés), elles sont les principaux éléments en jeu dans les réactions biochimiques intracellulaires. L'hypothèse, ou dogme de la biologie moléculaire, formulée par Francis Crick [?], postule que l'ADN contenu dans les noyaux constitue l'information dite génétique qui gouverne les principales fonctions de la cellule (reproduction, nutrition, excrétion, etc.). Dans un premier temps, l'ADN est transcrit en ARN messenger dans le noyau (étape de transcription). Dans un deuxième temps, l'ARN messenger est traduit en une protéine dans le réticulum endoplasmique (étape de traduction).

Lorsque les molécules produites par une réaction biochimique apparaissent comme substrats d'une autre réaction, il y a formation de chaîne (ou voie) métabolique. Ces chaînes sont régulées afin d'optimiser les taux de concentration des différentes molécules y participant. Par exemple, l'activité de catalyse d'une des enzymes de la chaîne peut être inhibée. Les réactions biochimiques sont locales puisqu'elles ne peuvent avoir lieu que lorsque les substrats en jeu, ainsi que les catalyseurs, sont en situation de proximité. Une fois produites, les molécules résultantes peuvent migrer au sein de la cellule ou à l'extérieur de la cellule pour être utilisées par d'autres cellules. On peut donc en déduire que chaque cellule est dotée d'un comportement autonome et d'un comportement coordonné avec les autres, ce qui suppose une nécessité de communication (échanges de molécules) entre les cellules et entre les compartiments d'une même cellule.

Les macromolécules biologiques sont souvent véhiculées par le biais de vésicules membranaires. L'exocytose est le transport de molécules par excrétion d'une vésicule suivie d'une fusion de la vésicule avec une membrane : s'il s'agit de la membrane de la cellule, les molécules sont alors évacuées vers le milieu extérieur. L'opération inverse, ou réception de molécules venant de l'extérieur, s'appelle l'endocytose (la membrane se referme pour constituer un espace fermé, appelé à se détacher sous forme de vésicule).

Rôle de la compartimentation cellulaire

Tout organisme biologique multicellulaire est constitué de cellules biologiques organisées en tissus, puis en organes et enfin en appareils et systèmes, ce qui en fait un système totalement hiérarchisé. Cette organisation compartimentée est déterminante pour le fonctionnement des systèmes biologiques, dans la mesure où l'effectivité des réactions biochimiques, qui régissent les grandes fonctions biologiques, en dépend. A l'échelle de la cellule, les types de compartiments conditionnent la fonction : le noyau contient le génome, le cytoplasme contient les organites, le cytosol permet la synthèse de protéines, les mitochondries permettent la transformation des glucides en ATP, le réticulum endoplasmique stocke le calcium (Ca^{++}) contrôlant les mécanismes de prolifération des cellules (apoptose), les lysosomes sont le lieu de la dégradation de molécules, les vésicules assurent le transport des molécules d'un compartiment à l'autre . . . C'est dire que la compartimentation cellulaire permet de délimiter des unités fonctionnelles spécialisées, et garantit pour chaque compartiment son *identité moléculaire*, ce qui constitue un mécanisme clé de régulation dans de nombreux systèmes biologiques [?]. Du reste, on admet que le degré de compartimentation d'une cellule est d'autant plus élevé que celle-ci est évoluée, spécialisée. Le rôle de la compartimentation cellulaire est donc d'assurer un fonctionnement plus efficace de l'organisme en délimitant des unités fonctionnelles spécialisées. Par exemple, une cellule qui a un grand besoin d'énergie contiendra beaucoup de mitochondries.

Au regard des réactions biochimiques, les bio-membranes jouent un rôle essentiel :

- Elles permettent de localiser les réactions biochimiques dans un des deux espaces qu'elles délimitent (en pratique, pour une membrane considérée, l'intérieur ou l'extérieur du compartiment délimité). Ainsi, les réactions biochimiques sont conditionnées par les densités des éléments biochimiques présents dans ces espaces séparés par la membrane. En quelque sorte, ces réactions biochimiques s'apparentent à des réactions effectuées en vase clos. Ainsi, chaque organite met en place un espace local qui lui est propre, et favorable aux réactions biochimiques en lien avec sa fonction cellulaire.
- Elles organisent le transport sélectif de molécules à travers la membrane : n'entrent dans un compartiment et n'en sortent que les molécules agréées pour circuler dans un sens ou dans l'autre. Ce transport se fait soit à l'aide de protéines membranaires (transport des petites molécules assuré de façon passive par simple diffusion, ou de façon active en consommant de l'énergie par exemple pour lutter contre le gradient de concentration) soit à l'aide de formation ou absorption de vésicules (exocytose et endocytose).
- Les bio-membranes constituent aussi un lieu de forte activité biochimique : leur surface varie, et elles contiennent en plus des phospholipides déjà mentionnées de nombreuses autres molécules (notamment des protéines constituant environ la moitié de la masse de la membrane [?]) susceptibles de se mouvoir, d'agir comme catalyseur ou de se modifier et qui sont responsables d'une grande partie des fonctions membranaires. La voie sécrétoire, décrite ci-dessous de façon succincte, met en jeu des protéines membranaires actives.

Il est admis que l'activité métabolique de la cellule (par exemple taux d'activité des réactions biochimiques "en vase clos") est proportionnelle au volume tandis que les échanges de molécules sont proportionnels à la surface des membranes. L'activité métabolique dépend donc des échanges de matière permis par les endomembranes (i.e. membranes intérieures à la cellule).

La voie sécrétoire est un exemple de processus cellulaire dans lequel les réactions chimiques et la conformation compartimentale sont fortement imbriquées afin de réaliser la fonction, à savoir la maturation, le tri et l'excrétion des protéines. Il s'agit du processus qui permet la sécrétion de protéines synthétisées et maturées. Commencant dans le réticulum endoplasmique et passant par l'appareil de Golgi, il se termine

par une phase de transport dans des vésicules ou des grains de sécrétion [?, ?]. Diverses réactions biochimiques au sein des saccules de l'appareil de Golgi permettent la maturation des protéines en assurant la perte de séquences peptidiques, l'ajout de sucres (glycosylation) ou l'ajout de sulphates (sulphatation). Ainsi, la maturation et le tri des protéines résultent de réactions biochimiques et de plusieurs étapes de transport (empaquetage et de dépaquetage dans les compartiments – saccules de l'appareil de Golgi, vésicules et grains de sécrétion – de la voie sécrétoire). Il existe aussi un transport rétrograde, à contre courant de la voie sécrétoire qui ramène les enzymes de maturation de la face « trans » de l'appareil de Golgi tournée vers la membrane plasmique vers la face « cis » de l'appareil de Golgi accolée au réticulum endoplasmique.

Biologie des systèmes : modélisation des processus cellulaires

La *biologie des systèmes* poursuit, par modélisation, la compréhension du fonctionnement des systèmes biologiques. Selon l'encyclopédie libre Wikipédia [?, ?], la biologie des systèmes peut être défini comme suit :

” un domaine académique qui cherche à intégrer différents niveaux d'informations pour comprendre comment fonctionnent des systèmes biologiques. En étudiant les relations et les interactions entre différentes parties du système biologique (organites - organelle -, cellules, systèmes physiologiques, réseaux de gènes et de protéines permettant la communication des cellules), le chercheur tente de découvrir un modèle de fonctionnement de la totalité du système. ”

De nombreux modèles ont été définis et utilisés avec plus ou moins de succès dans l'étude de certains phénomènes biologiques tels que les interactions de protéines, la régulation des gènes etc [?]. Certains de ces modèles sont issus de travaux qui ont cherché à modéliser l'activité biochimique cellulaire, en lien avec la présence des compartiments. L'ambition commune est l'étude des modèles par simulation ou analyse afin de reproduire les phénomènes observés ou de formuler des hypothèses sur les systèmes biologiques modélisés. Ces hypothèses, appelées aussi prédictions, sont destinées à être confrontées aux expériences des biologistes. En fonction des résultats de la confrontation, les modèles proposés pourront être aménagés, pour refléter au mieux les observations, tout en restant exploitables vis-à-vis des outils de simulation ou d'analyse. Ainsi, cette démarche correspond à un cycle « modélisation / simulation ou analyse / résultats expérimentaux / raffinement de modèle ». Les modèles proposés par les mathématiciens ou bioinformaticiens nécessitent des compromis de calibration (taille du modèle, nature et précision des concepts biologiques pris en compte) pour donner lieu à des prédictions susceptibles d'être jugées intéressantes par les biologistes.

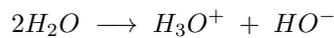
Modélisation des réactions biochimiques

Une large majorité des premiers modèles définis aux fins d'élucidation des processus biologiques par modélisation des réactions biochimiques, tels que [?, ?] s'appuient sur les *systèmes d'équations différentielles* tant il est vrai que la *théorie des systèmes dynamiques*, déjà bien établie, offre une multitude de techniques mathématiques pour raisonner sur de tels systèmes. [?, ?] énoncent plusieurs raisons expliquant pourquoi certains modélisateurs ont été amenés à s'affranchir des systèmes d'équations différentielles pour modéliser autrement les processus biologiques :

- les équations différentielles sont définies sur les paramètres cinétiques qui quantifient les réactions ; cependant, nombre de ceux-ci ne sont pas connus et leur estimation automatique qui fait l’objet de quelques initiatives de recherches [?, ?] est un problème difficile ;
- le nombre d’équations à définir peut être très élevé puisque croissant avec le nombre d’états possibles de l’ensemble des molécules en jeu. Par exemple pour une molécule comportant 8 sites de modification, 256 états sont possibles. L’ajout d’une molécule peut entraîner une *explosion combinatoire* du nombre d’équations.

Une autre approche pour la modélisation des interactions de molécules est celle à base de règles, formalisme initialement introduit pour modéliser les systèmes dynamiques concurrents. En ce qui concerne la modélisation des réactions biochimiques cellulaires, les approches à base de règles ont fait leurs preuves, et sont, de fait, largement acceptées. En effet :

- Ces règles s’inspirent fortement de la notion classique des règles ou équations chimiques. Les usages des règles sont inspirées des règles chimiques (synthèse, oxydo-réduction, combustion, ...). Par exemple, la dissociation de l’eau s’écrit classiquement :



Cette réaction indique que deux molécules d’eau peuvent donner lieu à deux ions, respectivement H_3O^+ et HO^- . Les réactifs (ou substrats) sont placés à gauche de la flèche et représentent la situation avant réaction. Les produits, quant à eux, sont placés à droite de la flèche et donnent la situation à l’issue de la réaction. Le signe « + » permet de séparer les molécules en jeu. Les réactions chimiques peuvent être annotées d’informations : l’énergie libérée, les catalyseurs favorisant la réaction, la vitesse de réaction, ... Ces règles sont très familières aux scientifiques : les réactions biochimiques sont donc souvent modélisées à partir des usages acceptés en chimie ;

- Les langages à base de règles permettent de séparer les connaissances, modélisées à l’aide de règles, des mécanismes d’agencement des connaissances, modélisés comme une stratégie d’application des règles à partir d’une configuration initiale, l’application d’une règle se ramenant à une réécriture [?, ?] de l’état du système. La granularité de la configuration initiale et les mécanismes d’application des règles définissent le dynamique du système, et donc le modèle.

Les molécules en jeu dans les règles biochimiques sont en général de grande taille, comportant quelquefois des informations particulières comme les sites de modification de forme ou les sites de liaison. Pour simplifier l’écriture des règles biochimiques, ces molécules ne sont en général pas décrites à l’aide des éléments chimiques les constituant mais plutôt par un simple identificateur. Afin de faciliter le suivi des modifications des molécules et d’assurer une certaine forme d’équilibre des réactions biochimiques, des manipulations simples sur les identificateurs sont mises en place. Par exemple, la molécule résultant de la complexation de deux molécules notées respectivement par les identificateurs A et B est souvent simplement notée $A - B$. Cette dernière notation explicite que la molécule résultante est constituée des éléments des molécules A et B .

Par défaut, une règle biochimique s’applique pour des substrats co-localisés dans un même espace, en pratique, l’intérieur d’un compartiment. S’il est nécessaire de localiser précisément les molécules, des annotations de localisation (types de compartiments) sont ajoutées aux règles. En particulier une règle de transport de molécules indiquera des localisations différentes à droite et à gauche.

Classification des réactions biochimiques

Nous détaillons dans ce paragraphe les différents types de réactions biochimiques présentes dans la cellule, et qui donnent en général lieu à différents modes de transcriptions dans les langages à base de

règles. Les principales réactions biochimiques élémentaires que l'on peut distinguer dans la cellule sont peu nombreuses et se résument en la *synthèse* et la *dégradation* d'une molécule, la formation et la dissociation d'un *complexe*, la *modification de forme* d'une molécule et le *transport* d'une molécule d'un compartiment vers un autre.

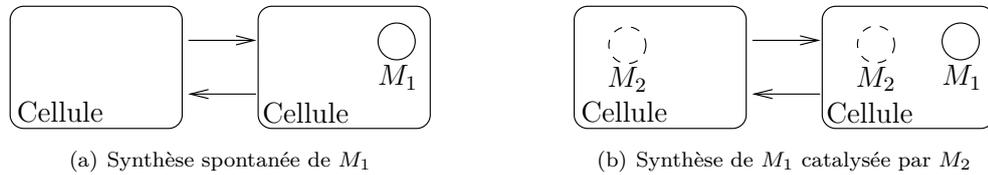


FIGURE 1.1 – Synthèse (\rightarrow) et dégradation (\leftarrow) de molécules

La synthèse et la dégradation de molécule consistent respectivement en une production et une élimination d'une molécule. Il s'agit d'interactions ayant lieu dans un même compartiment comme le montre la figure 1.1 où le compartiment concerné est *Cellule*. Elles peuvent être spontanées (figure 1.1(a)) ou résulter de réactions faisant intervenir d'autres molécules en tant que catalyseurs (molécule M_2 dans la figure 1.1(b)).

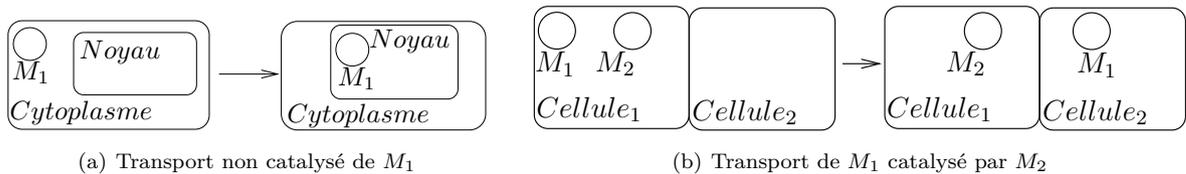


FIGURE 1.2 – Transport de molécules

La figure 1.2 schématise le transport non catalysé (1.2(a)) et catalysé (1.2(b)) d'une molécule M_1 . Dans le premier cas (1.2(a)), les compartiments impliqués sont tels que l'un (*Noyau*) est inclus dans l'autre (*Cytoplasme*). Le second cas montre deux compartiments *Cellule₁* et *Cellule₂* qui adhèrent l'un à l'autre. Ces deux dispositions spatiales relatives de compartiments les uns par rapport aux autres, sont les deux cas de *voisinages entre compartiments* dont nous traiterons dans la suite de ce document.

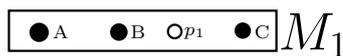


FIGURE 1.3 – Représentation d'une molécule (M_1) avec ses domaines fonctionnels.

En dehors des réactions de synthèse, de dégradation et de transport de molécules, les autres interactions élémentaires font intervenir la notion de *domaines fonctionnels* qui regroupent les *sites de liaison* et les *sites de modification* des molécules. La figure 1.3 schématise la molécule M_1 comportant les sites de liaison (points noirs) A , B et C et le site de modification (cercle) p_1 .

La liaison entre un site de liaison d'une molécule M_1 et un site de liaison d'une molécule M_2 , encore appelée *complexation*, résulte en la formation d'une autre molécule plus précisément d'une chaîne de molécules) appelée *complexe* qui a des propriétés différentes de celles des molécules de départ. La complexation et son pendant, la *décomplexation* permettent d'avoir des molécules de tailles différentes, allant des plus petites comme les éléments chimiques de base, les acides aminés à des molécules de grande taille,

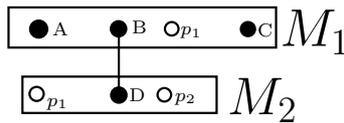


FIGURE 1.4 – Complexation entre les molécules M_1 et M_2 par leurs sites de liaison respectifs B et D .

telles que les protéines et les gènes. La figure 1.4 montre une complexation entre une molécule M_1 et une molécule M_2 , la liaison engageant les sites B de M_1 et D de M_2 .

Les modifications de formes de molécules sont des réactions *co-translationnelles* ou *post-translationnelles* (selon qu'elles interviennent pendant ou après la synthèse de la protéine) qui, en affectant la structure chimique de la protéine concernée, lui confèrent des propriétés particulières, généralement un changement de fonction. Les modifications de protéines ont des buts très diversifiés tels que l'activation des protéines, la régulation de l'activité des protéines, le marquage des protéines afin de permettre leur reconnaissance par d'autres molécules ou des systèmes de dégradation, l'ancrage des protéines dans une membrane ou encore leur intégration à des cascades de signalisation etc. La *phosphorylation* qui consiste en l'ajout d'un groupe phosphate sur un site de modification d'une protéine ou d'une petite molécule est l'une des modifications de forme les plus mentionnées dans la littérature.

Modélisation de la compartimentation cellulaire

L'organisation structurale de la cellule (topologie qui est l'ensemble des compartiments de la cellule et leurs dispositions spatiales relatives les uns par rapport aux autres) suit une structure d'arbre : la racine de l'arbre est le compartiment « cellule » tout entier avec sa membrane plasmique et avec le cytosol comme intérieur, les arcs entre les nœuds compartiments représentent la relation d'inclusion directe entre compartiments. Cette structure arborescente est commune, en particulier, elle est adoptée dans le standard SBML (Systems Biology Markup Language) [?, ?]. Elle permet en particulier de modéliser l'emboîtement des compartiments.

La modélisation de la structure topologique de la cellule diffère selon les paramètres suivants :

- les membranes sont considérées comme un simple séparateur d'espace, ou au contraire, au même titre que les intérieurs des compartiments, elles sont le lieu de réactions biochimiques ;
- la structure topologique peut être plus ou moins abstraite, selon qu'elle comporte un compartiment générique représentant toutes les instances possibles (par exemple une vésicule générique) ou plusieurs instances représentatives du compartiment (par exemple quelques vésicules dans une cellule simplifiée) ;
- la structure topologique peut être statique ou dynamique selon que la topologie est fixée pour toutes les simulations ou qu'au contraire, elle subit des transformations au cours d'une simulation ;
- les éléments topologiques (compartiments, membranes) peuvent être munies de données géométriques et biomoléculaires (volumes, dimensions, positions, concentrations de molécules). Ces données peuvent aussi simplement être partielles (par exemple ne concerner que les concentrations des molécules).

Il est évident que la modélisation de la structure spatiale des cellules nécessite des compromis drastiques pour simplifier la complexité intrinsèque des cellules biologiques. Ces compromis diffèrent selon les approches de modélisation suivies.

Couplage entre réactions biochimiques et compartimentation cellulaire

Dans les paragraphes précédents, nous avons présenté de façon découpée les éléments qui sont pris en compte respectivement pour modéliser les réactions biochimiques et la structure spatiale des cellules. Selon les aspects privilégiés, et en lien avec les outils de simulation ou d'analyse utilisés, le couplage peut privilégier l'une ou l'autre composante, en particulier faire abstraction de l'existence des compartiments.

- À l'une des extrémités, les approches à base de règles peuvent ne considérer qu'un ensemble de règles biochimiques, s'appliquant implicitement dans une soupe (par exemple, le cytosol), contenant toutes les molécules en jeu. Par exemple, les premiers cas d'étude menés dans l'environnement BIOCHAM [?] ne contiennent que des règles, sans faire mention d'une compartimentation.
- À l'autre extrémité, les modélisations orientées « dynamiques spatio-temporelles » mettent l'accent sur les modifications de la structure topologique et géométrique étudiée en lien avec la fonction biologique sous étude. Dans [?], l'appareil de Golgi est étudié sous le seul angle de la dynamique de sa structure topologique. Trois hypothèses, différant par la structure topologique, ont été modélisées. En quelques mots, la forme 3D de l'appareil de Golgi est souvent assimilée à une pile d'assiettes entourée de vésicules ou de grains de sécrétion. Cependant, sa dynamique en lien avec la fonction de tri et d'excrétion des molécules n'est pas vraiment connue. Dans [?], il est montré qu'une structure topologique continue (assiettes connectées par des tubules) est plus en accord avec les observations faites par les biologistes qu'une structure topologique déconnectée (simples assiettes) : en particulier, une étude des flux de matières transportées (quantité de surfaces membranaires et de volumes) plaide pour une structure connectée en raison d'un principe d'équilibre des flux entrants et sortants en régime stationnaire.

Il existe des approches de modélisation qui ont intégré des possibilités de modification des compartiments en jeu et des possibilités d'analyse. En particulier, [?, ?] proposent des cadres de modélisation, « Brane calculi » et « BioAmbients », dédiés à la biologie cellulaire, fondé sur le π -calcul [?], une algèbre de processus modélisant des systèmes informatiques concurrents et distribués, pour lesquels la topologie des communications entre processus peut varier au cours du temps. Dans [?], l'accent est mis sur la modélisation de l'activité des membranes. Les principales modifications (exocytose, phagocytose, endocytose) des compartiments peuvent être capturées par des transformations de configurations membranaires. Comme les membranes apparaissent explicitement dans les termes du calcul des membranes, il devient possible d'exprimer des réactions qui modifient les membranes, et donc la structure topologique de la cellule.

Dans ce document, nous nous focaliserons sur une modélisation d'une structure topologique statique des systèmes biologiques en jeu : nous n'évoquerons donc plus les formalismes à base de π -calcul. Notre objectif est de combiner une description précise des aspects compartiments et des règles biochimiques, afin de modéliser les systèmes biologiques avec une forte préoccupation orientée « réalisme et cohérence de la structuration en compartiments ». Nous pensons qu'il s'agit d'une étape préalable avant d'intégrer des mécanismes de transformations topologiques.

Notre approche : un environnement générique

De manière générale, les outils de modélisation de processus biologiques sont caractérisés par une certaine faiblesse au niveau de la prise en compte de la structure topologique des systèmes biologiques étudiés. Notre motivation principale dans la conduite de ce travail est d'offrir un environnement générique

(sur les compartiments et sur les réactions) de modélisation par des règles, de processus biologiques avec la garantie que toutes les réactions exprimées sont autorisées eu regard à la compartimentation de la structure biologique étudiée : toutes les réactions engagent les compartiments de types appropriés et tous les échanges de molécules sont en accord avec les voisinages entre les compartiments impliqués. Par ailleurs, bon nombre des outils à base de règles existant pour la modélisation de ces phénomènes offrent d'intéressantes capacités de simulation et d'analyse que nous souhaitons pouvoir exploiter.

Parmi les approches de modélisation des processus biologiques existantes, certaines intègrent déjà une généralité pouvant porter aussi bien sur les molécules que sur les localisations. Cette généralité est mise en œuvre à travers l'utilisation de variables. [?] introduit un langage qui utilise des variables de molécules et des variables de sites de modification pour modéliser des réactions génériques. En ce qui concerne BIOCHAM [?, ?], les variables peuvent être définies sur toutes les composantes des règles. Les modèles définis dans ces langages peuvent être interprétés selon plusieurs sémantiques, en fonction des éléments considérés dans l'expression des réactions : ainsi dans BIOCHAM, les réactions peuvent être interprétées en ne considérant uniquement que la présence des molécules ; il est également possible de tenir compte des paramètres cinétiques pour effectuer des simulations qui tiennent compte des concentrations ou des quantités de molécules.

Dans ce paragraphe nous donnons les principales options pour la définition de notre environnement de modélisation : d'un point de vue structurel, les modèles qui seront définis auront deux composantes indépendantes : les règles modélisant les réactions biochimiques et la topologie des systèmes sous étude. Cette indépendance entraîne une généralité des règles de réactions qui ne portent de fait sur aucun compartiment concret. En termes de modélisation des compartimentations cellulaires, nous préconisons dans un premier temps de considérer des structures topologiques statiques, ce qui, eu égard au caractère dynamique des systèmes biologiques est restrictif. L'objectif à terme est d'intégrer une prise en compte des modifications de topologie. Dans un souci de réalisme de la représentation et également pour permettre de modifier les modèles tout en conservant la cohérence, nous choisissons d'utiliser la modélisation géométrique pour la représentation des compartimentations cellulaires. Enfin, les modèles seront simulés et analysés selon les sémantiques d'outils de modélisation externes, et pour ce faire, nous avons retenus deux outils cibles BIOCHAM et PATHWAY LOGIC dont les sémantiques en ce qui concerne la prise en compte des localisations de molécules peuvent être fidèlement rendues par un couplage entre les règles génériques et une compartimentation cellulaire.

Organisation du document

Le manuscrit sera organisé autour des chapitres suivants :

Le chapitre 2 sera consacré à la présentation de notre approche couplant règles de réaction génériques et compartimentation cellulaire pour la modélisation des processus biologiques cellulaires. Le chapitre nous permettra de développer notre analyse des besoins en identifiant les éléments à intégrer côté représentation de compartimentations cellulaires et côté règles génériques. Nous commencerons par présenter les deux outils cibles que nous avons retenus.

Dans le chapitre 3, nous présentons le *graphe d'échanges* qui est notre abstraction de la compartimentation cellulaire. Afin de fournir une base réaliste aux graphes d'échanges, ces derniers seront extraits à partir d'objets 3D construits à l'aide d'un *modèle géométrique* dédié. Nous allons en effet spécialiser un

modeleur géométrique afin de fournir à l'utilisateur des primitives d'aide à la construction de structures compartimentées.

Dans le chapitre 4, nous présentons notre langage de règles génériques, incluant des variables typées appelées à être substituées par des compartiments issus de la compartimentation biologique cible. Nous introduisons aussi notre langage de règles intermédiaires, issues du couplage entre règles génériques et graphe d'échanges. Enfin, nous décrivons nos mécanismes de traduction des règles intermédiaires vers BIOCHAM et vers PATHWAY LOGIC.

Dans le chapitre 5, nous présentons quelques éléments de validation de notre approche. Nous y discutons de la qualité des modèles PATHWAY LOGIC et BIOCHAM construits, et nous illustrerons l'intérêt de notre approche à l'aide d'exemples issus de la littérature comportant des éléments de compartimentation.

Chapitre 2

Couplage entre règles biochimiques et compartiments : Approche suivie

Sommaire

2.1 BIOCHAM	13
2.1.1 Les objets formels manipulés	14
2.1.2 Les composantes des règles biochimiques	16
2.1.3 Prise en compte de la localisation	19
2.2 PATHWAY LOGIC	20
2.2.1 Aperçu de la spécification algébrique : Les objets manipulés	21
2.2.2 Les composantes des règles de réaction	23
2.2.3 Prise en compte de la compartimentation cellulaire	25
2.3 Analyse des besoins	26
2.3.1 Exemple d'illustration de notre approche	29
2.3.2 Graphe d'échanges et compartimentation cellulaire	31
2.3.3 Langage des règles génériques	31

Dans ce chapitre, nous présentons notre approche pour coupler des règles biochimiques et une compartimentation cellulaire donnée : l'objectif visé est l'obtention d'un modèle de processus biologique, décrit dans le langage de règles de l'un des deux outils cibles choisis (BIOCHAM ou PATHWAY LOGIC - PL - en l'occurrence), et tenant compte d'une compartimentation cellulaire. Cette dernière indique à la fois quels sont les types des compartiments en jeu (noyau, cytoplasme, vésicule, ...) et quelles sont les dispositions relatives des compartiments composant le système cellulaire, c'est-à-dire quelles sont les relations de voisinage immédiat entre les compartiments.

Comme évoqué précédemment dans l'introduction de ce manuscrit, le choix des environnements cibles BIOCHAM ou PATHWAY LOGIC a été en partie guidé par le fait que ces deux environnements de modélisation, reconnus par l'un comme par l'autre assez similaires, fournissent des langages de règles relativement faciles à appréhender, par la volonté même affichée par leurs concepteurs respectifs de proposer aux biologistes-modélisateurs un langage qui leur est accessible. Une étude comparative succincte des deux outils nous a permis de nous rendre compte que les deux environnements ont pour objet la modélisation de

phénomènes de même nature, et en particulier, fournissent des analyses qualitatives permettant de mettre en évidence des propriétés biologiques similaires (exprimées à l'aide de formules temporelles). Nous tenons à préciser que même si la partie analyse des modèles n'entre pas à proprement parlé dans le cadre de notre travail, il est intéressant, voire même rassurant, de remarquer que le fait d'exhiber pour les modèles BIOCHAM et PATHWAY LOGIC d'un même processus, les mêmes propriétés biologiques, renforce la présomption que les points fondamentalement communs aux deux outils sont nombreux.

Ce point a son importance : l'idée est qu'*a priori*, avec deux outils réputés relativement semblables sur la modélisation de phénomènes biologiques, on a plus de chances de capturer sans effort excessif en adaptations diverses, un large panel des aspects abordés dans le domaine considéré, et donc potentiellement d'être assez complet.

Nous considérons même que les divergences entre ces deux outils constituent des points forts spécifiques à chacun pour la modélisation de processus biologiques. La première différence entre BIOCHAM et PATHWAY LOGIC est celle du contexte de définition des deux environnements : en effet, d'un côté, BIOCHAM a été conçu pour modéliser uniquement les phénomènes biologiques cellulaires et de l'autre, PATHWAY LOGIC est fondé sur un langage initialement défini pour modéliser les processus concurrents. Il en résulte que le langage de règles de BIOCHAM est dédié à l'expression de réactions biologiques dans le cadre des analyses prévues, et n'intègre pas de ce fait des éléments externes, tandis que celui de PATHWAY LOGIC, de par la possibilité qu'a l'utilisateur de définir ses propres types algébriques de données, offre une plus grande flexibilité dans l'expression des éléments de règles biologiques.

Pour ces raisons, nous avons, dans un premier temps, étudié les deux environnements de modélisation, en particulier leurs langages de règles, afin de mettre à jour les aspects des processus biologiques pris en compte par l'un et/ou par l'autre et de proposer un langage de règles génériques exprimant les mêmes aspects (communs ou spécifiques à l'un ou à l'autre) de modélisation de processus biologiques. L'idée *a priori* est de pouvoir retranscrire les modèles écrits dans notre langage de règles en modèles BIOCHAM et PATHWAY LOGIC. Cette étude comparative confirme que BIOCHAM et PATHWAY LOGIC présentent des similitudes marquées¹, qui vont fonder la définition d'un langage de règles génériques compatible avec ceux des deux outils. Par ailleurs, nous avons également relevé que les divergences constatées ont plutôt tendance à s'amenuiser si l'on se fie aux perspectives d'évolution affichées par les concepteurs des deux environnements. Nous comptons donc nous inspirer des deux langages de règles de ces outils, pour proposer le nôtre avec la même qualité de description intuitive, simple et familière aux biologistes.

Le chapitre a principalement pour objet de motiver notre approche de couplage entre règles biochimiques et compartimentation cellulaire. La première partie du chapitre consistera en la présentation des environnements BIOCHAM et PATHWAY LOGIC. Plus précisément, le chapitre va s'articuler autour des sections suivantes :

- Dans les sections 2.1 et 2.2, nous présentons respectivement les outils BIOCHAM et PATHWAY LOGIC à travers notamment les objets manipulés dans les règles de réaction. Nous insistons sur la prise en compte des informations de localisation, attendu que notre travail est axé sur l'idée que la répartition des molécules dans les compartiments des cellules et l'agencement de ces compartiments sont déterminants pour les phénomènes biochimiques observés.
- La section 2.3 est consacrée à la problématique développée dans ce manuscrit, et constitue plus précisément une analyse des besoins. Nous identifions ce qui nous sera utile en termes d'éléments à considérer. Ainsi, nous y évoquons la problématique de l'abstraction des compartimentations cellulaires, et nous synthétisons à partir des deux sections précédentes concernant BIOCHAM et

1. Nous ne considérons à ce niveau que la sémantique booléenne de BIOCHAM

PATHWAY LOGIC les éléments clés qui apparaîtront dans notre langage de règles. L'objectif que nous poursuivons dans cette section est de délimiter notre domaine d'étude en ce qui concerne les contours de notre langage de règles génériques. En particulier, nous identifions les aspects que nous prendrons en compte dans notre langage de règles afin de modéliser les processus biologiques ciblés.

2.1 BIOCHAM

BIOCHAM [?, ?, ?, ?, ?, ?, ?] est un environnement logiciel destiné aux biologistes-modélisateurs, qui offre deux langages formels :

- un langage à base de règles, pour la modélisation des interactions biochimiques ;
- un langage fondé sur la logique temporelle, pour la formalisation des propriétés biologiques du système.

BIOCHAM offre la possibilité de décrire des modèles selon plusieurs angles :

- purement qualitatifs, simulés et analysés selon une sémantique dite *booléenne*,
- quantitatifs, interprétés selon une sémantique dite *des concentrations*, une autre dite *des populations* et une autre dite *discrète*.

Pionnier dans l'adjonction aux modèles, de propriétés biologiques établies de manière expérimentale, ou inférées, BIOCHAM offre une méthode de validation automatique et de révision des modèles par utilisation des techniques d'apprentissage de règles biochimiques à partir de propriétés en logique temporelle [?]. Il est également possible de rechercher automatiquement les valeurs des paramètres cinétiques correspondant à un comportement donné.

Les processus modélisés se résumaient initialement aux réseaux d'interactions entre protéines (et gènes) au niveau intracellulaire. De plus en plus, BIOCHAM s'intéresse à la modélisation des interactions dans des populations de cellules.

Les règles de réactions biochimiques constituent le cœur de tout modèle BIOCHAM. Elles sont données sous la forme d'un fichier ".bc" contenant, outre la section des règles de réaction qui est obligatoire, plusieurs autres parties qui peuvent être déclarées de manière optionnelle :

- Les objets formels sur lesquels portent les règles de réactions, sont des composés chimiques ou biochimiques (molécules telles que les protéines, les gènes, etc) assortis d'information de localisation ou non ;
- une section d'initialisation des identificateurs de paramètres cinétiques ;
- une section de déclaration de macros qui sont des opérations définies par l'utilisateur à des fins d'observation. Par exemple si A désigne une molécule, l'utilisateur peut souhaiter connaître à un instant donné, la quantité totale de A présente dans la cellule étudiée. Il définira donc une macro, qui lui permettra de calculer cette valeur en se fondant sur toutes les formes de A ;
- une section de définition des concentrations initiales des molécules. Lorsque cette section est absente, toutes les molécules sont considérées avoir une concentration nulle à l'état initial ;
- une section répertoriant les propriétés biologiques du système sous étude. Cette section contient des formules de logique temporelle et peut faire l'objet d'un fichier séparé.

Les logiques temporelles utilisées pour l'analyse des modèles sont respectivement

- la logique temporelle arborescente, ou Computation Tree Logic, (CTL) pour l'établissement des propriétés booléennes comme les propriétés d'accessibilité, d'existence d'états stables ou des points de passage obligés ...

- la logique temporelle linéaire, ou Linear Time Logic, (LTL) pour les propriétés temporelles quantitatives : les atomes portent sur les niveaux de concentrations des éléments biochimiques considérés via des opérateurs de comparaison ou de dérivation.

L'existence d'une interface graphique qui donne accès aux principales commandes facilite grandement les interrogations des modèles et offre également la possibilité de comparer des résultats de requêtes.

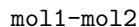
L'environnement BIOCHAM dont le développement a débuté en 2002 en est actuellement à sa version 3.2. C'est un logiciel libre, téléchargeable à l'adresse <http://contraintes.inria.fr/BIOCHAM/>. Le téléchargement inclut le logiciel lui-même et des exemples de modèles classés selon les aspects de modélisation qui y sont mis en exergue (kinetics, locations) ou les processus modélisés (cell cycle, MAPK, etc).

Dans les paragraphes suivants, nous allons présenter le langage des règles de BIOCHAM à travers notamment les objets biochimiques manipulés, la syntaxe des règles de réaction, la prise en compte des compartiments cellulaires et l'association de paramètres cinétiques aux règles de réaction.

2.1.1 Les objets formels manipulés

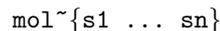
En tant qu'environnement de modélisation de processus biologiques, BIOCHAM manipule des *objets* qui désignent des molécules localisées ou non. Les molécules simples dites "*molécules de base*" sont identifiées par leurs noms tandis que les molécules plus complexes sont obtenues en appliquant les deux opérations essentielles suivantes :

- l'opération de complexation utilisée comme suit :



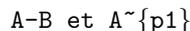
où mol1 et mol2 sont les molécules liées par l'opérateur « - » ;

- l'opérateur de modification de forme noté



où les *si* sont des sites de modification de la molécule mol

Par exemple, si A et B sont des molécules,



sont des molécules qui désignent respectivement le complexe formé par A et B et la forme phosphorylée de A sur le site p1.

On notera que la complexation dans BIOCHAM est gérée de façon implicite. Par ailleurs, la modification de forme précise uniquement les sites de modification, sans nommer la modification elle-même. Il est possible mais pas indispensable de déclarer avant utilisation les sites de modification d'une molécule. Par exemple

```
declare MEK~parts_of({p1,p2}).
```

déclare que les sites possibles de modification pour la molécule MEK sont p1 et p2. On pourra avoir dans le modèle les formes suivantes pour MEK :

```
MEK, MEK~{p1}, MEK~{p2}, MEK~{p1, p2}
```

L'opérateur `::` permet d'associer des localisations sous forme d'étiquettes (labels) aux molécules. On obtient des objets localisés tels que

```
A::noy
```

avec `noy` pour noyau. Cette dernière expression désigne la molécule **A** localisée dans un compartiment dénommé `noy`.

Il est possible d'utiliser des *schémas d'objets formels* pour spécifier des ensembles d'objets de manière concise. Un schéma d'objet (molécule ou molécule localisée) est obtenu par utilisation dans l'expression d'un objet, du caractère `<<? >>` ou d'une variable qui est un *mot* commençant par le caractère `$`. Par exemple :

```
A~? et A~?~?
```

sont des schémas d'objets qui spécifient respectivement l'ensemble des formes de la molécule **A**, comprenant **A** et les formes phosphorylées de **A**, et l'ensemble des formes de **A** complexées ou non.

Les variables ou le caractère `<<? >>` peuvent apparaître dans n'importe quelle partie des objets formels. Ainsi dans un schéma de molécules localisées, on peut les retrouver au niveau de la partie molécule ou au niveau de la localisation. Par exemple,

```
A-?::?
```

est un schéma qui définit l'ensemble contenant la molécule **A** localisée ou non et tous les complexes engageant **A** éventuellement localisés.

2.1.2 Les composantes des règles biochimiques

Les règles biochimiques dans BIOCHAM sont essentiellement composées de *solutions* et de *paramètres cinétiques*. La forme générale en est

$$\text{kinetics for MG} \Rightarrow \text{MD}.$$

où *kinetics* est un paramètre cinétique et où MG et MD sont des solutions. La partie réaction, MG=>MD, traduit le passage du système sous étude de l'état caractérisé par MG à l'état caractérisé par MD. Des abréviations ont été prévues pour plus de concision et une meilleure lisibilité des modèles. Au titre de ces abréviations, on peut lister

$$\text{MG} \Leftrightarrow \text{MD}.$$

$$\text{MG}=[\text{Object}]\Rightarrow\text{MD}.$$

qui permettent de définir respectivement une réaction réversible et une réaction catalysée, les catalyseurs étant donnés entre crochets sous forme de solutions.

2.1.2.1 Les solutions

Un ensemble quelconque d'objets formels est appelé « *solution* ». Elle est notée en séparant les objets la composant par des « + ». Une solution est qualifiée de *vide* lorsque cet ensemble est vide : elle est alors notée `_`. Ainsi,

$$\begin{array}{l} \text{A} \\ \text{A+B} \\ \text{A+B+A-B}::\text{Cytoplasme} \end{array}$$

sont des exemples de solutions contenant respectivement la molécule A, les molécules A et B, les molécules A, B et le complexe A-B localisé dans le `Cytoplasme`.

2.1.2.2 Les paramètres cinétiques

Ils précisent les contextes d'application des règles de réaction. Il peut s'agir d'objets formels comme définis à la section 2.1.1, de valeurs réelles, d'expressions arithmétiques ou d'expressions conditionnelles. Leur utilisation permet dans une certaine mesure de définir des règles conditionnelles. Notons toutefois que les paramètres cinétiques, qui constituent une *partie optionnelle* (la loi d'action de masse des molécules

de MG, avec 1 comme paramètre, est considérée par défaut) des règles de réaction, ne sont pas pris en compte au niveau de la sémantique booléenne. Par exemple, si on considère deux modèles se résumant respectivement aux règles de réaction suivantes :

$$\begin{aligned} & _ \Rightarrow B \\ & (\text{if } [A] > 1 \text{ then } 1 \text{ else } 0) \text{ for } _ \Rightarrow B \end{aligned}$$

ils se comportent de la même manière en simulation booléenne quelle que soit la concentration de A dans l'état initial. Cependant, il convient de noter que les paramètres cinétiques constituent un élément très important dans BIOCHAM et qu'ils expriment parfois des conditions engageant des voisinages entre localisations. C'est le cas par exemple lorsque les contenus des compartiments voisins d'un compartiment C activent une réaction dans C.

2.1.2.3 Les schémas de règles biochimiques

Lorsqu'une règle de réaction contient des schémas d'objets, elle devient un schéma de règle, donc susceptible d'être instanciée en un certain nombre de règles de réaction. Si ces schémas d'objets sont exprimés à l'aide de variables, les valeurs de celles-ci doivent être contraintes soit par les déclarations préalables, soit par utilisation de la clause **where** suivie d'une contrainte qui détermine les valeurs autorisées ou les valeurs interdites. Par exemple, les règles suivantes

$$A + \$V \Rightarrow A - \$V \text{ where } \$V \text{ in } \{B, C\}$$
$$A + \$V \Rightarrow A - \$V \text{ where } \$V \text{ not in } \{B, C\}$$

sont des schémas de règles qui vont donner lieu respectivement aux deux règles de réaction suivantes

$$\begin{aligned} A+B & \Rightarrow A-B. \\ A+C & \Rightarrow A-C. \end{aligned}$$

et à toutes les règles de complexation entre A et toutes les autres molécules différentes de B et de C. Notons que la deuxième règle peut également s'exprimer par

$$A + \$V \Rightarrow A - \$V \text{ where } \$V \text{ diff } \{B, C\}.$$

Comme on peut le constater, tout comme « **in** » et « **not in** » ... il existe dans BIOCHAM d'autres opérateurs tels que « **in all** », « **diff** », « **phos_form** », etc. qui permettent de contraindre les valeurs d'une variable. Il est important de noter que les schémas de règles de réaction seront instanciées

différemment en fonction de leur emplacement dans le modèle. Ainsi par exemple, en considérant le modèle comportant les trois règles suivantes :

```
r1 : _=>B.
r2 : C=>_.
r3 : A+? => A-?.
```

le schéma de règle `r3` sera instanciée en

$$A+B \Rightarrow A-B.$$

si les règles apparaissent dans l'ordre `r1`, `r3`, `r2` et en

$$\begin{aligned} A+B &\Rightarrow A-B. \\ A+C &\Rightarrow A-C. \end{aligned}$$

si elles apparaissent dans l'ordre `r1`, `r2`, `r3`. Seuls les éléments (objets formels) « connus » au moment de la définition des schémas de règles peuvent être pris en compte dans l'instanciation de ces schémas.

2.1.2.4 Extrait d'un modèle BIOCHAM : Mapk

Listing 2.1 – Extrait du modèle MAPK de BIOCHAM

```
1 %DECLARATIONS
2 declare MEK~parts_of({p1,p2}).
3 declare MAPK~parts_of({p1,p2}).
4
5 %REGLES DE REACTION
6 (MA(1),MA(0.4)) for RAF + RAFK <=> RAF-RAFK.
7
8 (MA(3.3),MA(0.42)) for MEK~$P + RAF~{p1} <=> MEK~$P-RAF~{p1}
9   where p2 not in $P.
10
11 (MA(10),MA(0.8)) for MEKPH + MEK~{p1}~$P <=> MEK~{p1}~$P-MEKPH.
12
13 MA(0.1) for RAF-RAFK => RAFK + RAF~{p1}.
14
15 MA(0.1) for RAF~{p1}-RAFPH => RAF + RAFPH.
16
17 %ETAT INITIAL
```

```
18 present(MAPK,0.3).
19 present(MAPKPH,0.3).
20 present(MEK,0.2).
21 present(MEKPH,0.2).
22 present(RAF,0.4).
23 present(RAFK,0.1).
24 present(RAFPH,0.3).
25
26 %same as 'make_absent_not_present.'
27 absent({?-?,?~{p1}~?}).
```

Le listing 2.1 présente un extrait du modèle BIOCHAM de la *signalisation mapk* qui est un des processus biologiques relativement bien connu et ayant fait l'objet de plusieurs modélisations.

Nous donnons ci-dessous quelques éléments relatifs au modèle décrit :

- Les lignes précédées par ”%” (lignes 1, 5, 17, 26) représentent des commentaires.
- Les lignes 2 et 3 sont des déclarations donnant les sites de phosphorylation des molécules MEK et MAPK. Cette partie est optionnelle.
- La première règle (ligne 6) concerne une réaction de complexation réversible entre RAF et RAFK. On remarquera l'expression du paramètre cinétique qui définit deux composantes, la première (MA(1)) contextualisant la complexation tandis que la deuxième (MA(0,4)) concerne la décomplexation.
- Les règles 2 et 3 commençant respectivement en lignes 8 et 11 sont des schémas de règles de réaction réversibles. On constate que dans la règle 2 (ligne 8), les valeurs de la variable \$P sont contraintes par une clause **where**, tandis que dans la réaction 3 (ligne 11), c'est la déclaration en ligne 2 qui contraint les valeurs de \$P.
- En lignes 13 et 15, on a deux règles de réaction non réversibles.
- Toutes les règles de ce modèle ont comme paramètres cinétiques la loi d'action de masse appliquée à des valeurs réelles. La règle 4 (ligne 13) aurait pu s'écrire

$$0.1 * [\text{RAF-RAFK}] \text{ for } \text{RAF-RAFK} \Rightarrow \text{RAFK} + \text{RAF}^{\sim\{p1\}}.$$

où [RAF-RAFK] correspond à la concentration de la molécule RAF-RAFK.

- A partir de la ligne 18, l'état initial est spécifié : toutes les formes simples (non complexées, non phosphorylées) sont présentes avec une concentration donnée. Ainsi, on a par exemple MAPK présente avec une concentration de 0.3, MEK avec une concentration de 0.2, etc.

2.1.3 Prise en compte de la localisation

Nous avons introduit en section 2.1.1 l'opérateur `::` qui permet d'associer une localisation aux molécules. Les objets localisés sont utilisés pour matérialiser les transports de molécules. Ainsi, la règle de réaction

$$\text{A}::\text{cyt} \Rightarrow \text{A}::\text{noy}$$

correspond au transport de la molécule **A** du cytoplasme (**cyt**) vers le noyau (**noy**). Si la notion de localisation est bien présente dans BIOCHAM, elle gagnerait à être renforcée. En effet, dans un contexte intracellulaire, on peut raisonnablement supposer que les modèles définis sont en accord avec une certaine compartimentation cellulaire cohérente. En effet, la quasi totalité des organelles membranaires sont accessibles les uns à partir des autres grâce au réseau membranaire interne. Par contre, pour la modélisation d'interactions dans une population de cellules, il est important de pouvoir assurer que toutes les règles de transport sont cohérentes par rapport à la structure topologique de l'ensemble cellulaire sous étude. Une étude menée en 2005 et qui a conduit à la modélisation de la *signalisation Delta-Notch* (*notch4n36c*) adapté de [?] (nous reviendrons sur ce modèle dans le chapitre 5 de ce document) dans une population de trente-six cellules, avait permis de relever la nécessité de tenir compte de la structure topologique des systèmes cellulaires dans les modélisations BIOCHAM.

C'est sans doute pour atténuer les effets de cette relative faiblesse au niveau de la prise en compte de la structure topologique, que les concepteurs de BIOCHAM ont introduit en 2007, une méthode de vérification *a posteriori* de la cohérence topologie/règles [?]. Cette méthode permet d'inférer, pour un modèle donné, une structure topologique et de vérifier la conformité de la topologie inférée par rapport aux informations d'ordre topologique disponibles pour le système cellulaire dans lequel se déroule le processus modélisé. Deux molécules sont considérées comme voisines (comprenons cela par : « peuvent potentiellement interagir ») si elles sont localisées dans un même compartiment ou dans des compartiments voisins. La méthode a permis d'établir que certains modèles BIOCHAM faisant intervenir des localisations, sont en accord avec les informations d'ordre topologique données. Il s'agit généralement des modèles issus de transcriptions de modèles pris dans Biomodels [?] pour lesquels il existe cette catégorie d'informations, et d'autre part du modèle de la signalisation Delta-Notch². Les structures topologiques inférées dans le cadre de toutes ces vérifications ont permis de distinguer deux types de voisinage :

- l'inclusion d'un compartiment dans un autre : les modèles de Biomodels incluent quelquefois l'attribut `outside` dans la définition d'un compartiment `C`, pour spécifier le compartiment contenant `C` ;
- le fait pour deux compartiments d'être « collés ».

En l'absence de telles informations, les structures topologiques inférées ne peuvent être vérifiées et les voisinages qui y sont mentionnés ne peuvent être qualifiés. En effet, si les identificateurs de compartiments ne sont pas assez explicites, on ne pourra pas affirmer pour deux compartiments voisins s'ils sont collés ou si l'un est inclus dans l'autre.

Nous notons également que cette méthode aborde la topologie uniquement en termes de voisinage entre les compartiments, sans se préoccuper de la nature des compartiments en jeu.

L'autre aspect important qui plaide pour un renforcement de la prise en compte de la localisation est la possibilité de modéliser des compartiments imbriqués de façon cohérente.

2.2 PATHWAY LOGIC

Pathway Logic [?, ?, ?, ?, ?, ?, ?] est une approche symbolique pour la modélisation, la simulation et l'analyse des processus biologiques cellulaires. Il est fondé sur la logique de réécriture et est conçu avec le souci de permettre des définitions de modèles correspondant au mieux aux modèles intuitifs (informels) que les biologistes ont du comportement d'un système biologique donné. Jusqu'à présent, la modélisation

2. <http://contraintes.inria.fr/BIOCHAM/EXAMPLES/locations/notch4n36c.bc>

en PATHWAY LOGIC est uniquement qualitative, mais les récents développements prévoient la prise en charge de paramètres cinétiques, et la possibilité de faire des analyses quantitatives [?]. Par ailleurs, le champ des processus modélisés qui ne comprenait que les interactions intracellulaires, s'est élargi et couvre actuellement les interactions inter-cellulaires.

Sur la base des données disponibles dans le domaine étudié, PATHWAY LOGIC permet de modéliser les processus biologiques, à différents niveaux d'abstraction : par exemple, il est possible d'écrire des règles de réaction engageant des protéines prises comme telles (interaction intermoléculaire), ou des règles de réaction portant sur les domaines fonctionnels des molécules, et dans ce cas, celles-ci sont vues comme des agrégats de domaines fonctionnels. Les modèles dans lesquels se côtoient des règles de réaction de différents niveaux d'abstraction sont qualifiés de *modèles multi-échelles*.

Le langage de réécriture *Maude* [?] est utilisé pour le développement des modèles PATHWAY LOGIC, ce qui fait de ces derniers, des *modules systèmes Maude* comprenant une description d'un type algébrique de données et un ensemble de règles de réécriture associé. Étant ainsi bâti sur *Maude*, PATHWAY LOGIC réalise les simulations et les analyses de ses modèles, en utilisant les fonctions de simulation et le model-checker de Maude qui est fondé sur la logique temporelle « Linear Time Logic (LTL) ». Les propriétés biologiques établies par ces analyses sont celles de stabilité, d'accessibilité, d'existence d'états d'équilibre etc. Grâce à l'interface graphique intégrée, le *Pathway Logic Assistant (PLA)* [?], il est possible de représenter un modèle donné sous forme de réseaux de Pétri, d'interroger les modèles, de générer automatiquement des sous-graphes correspondant aux requêtes exprimées et de faire des comparaisons de parties de graphes.

PATHWAY LOGIC est un logiciel libre distribué avec des exemples de modèles, sous licence GPL. Il intègre une version d'exécution en ligne qui permet de l'expérimenter sans avoir à l'installer. La dernière version 3.3 du *PLA*, sortie le 12 juin 2011, est accessible en téléchargement sur le site web du projet <http://www.pl.csl.sri.com/download.html>.

Dans les paragraphes suivants, nous présentons l'outil PATHWAY LOGIC en commençant par un aperçu de sa spécification algébrique des données qui lui permet de représenter toutes les composantes des modèles. La section briefera sur la structure des règles de réaction tandis qu'une analyse de la prise en charge des compartiments viendra boucler la présente section.

2.2.1 Aperçu de la spécification algébrique : Les objets manipulés

Comme tout environnement de modélisation de processus biologiques à base de règles, PATHWAY LOGIC manipule des objets formels représentant des molécules et éventuellement des compartiments cellulaires. Étant fondé sur Maude, PATHWAY LOGIC a spécifié un type algébrique de données qui lui permet de définir l'ensemble de ces éléments. Cette spécification, objet du fichier *"theops.maude"* contient toutes les *sortes* pertinentes pour la modélisation d'un processus biologique, ainsi que les opérations sur ces sortes. L'extrait 2.2 suivant du fichier « *theops.maude* »

Listing 2.2 – extrait 1 de *theops.maude* : déclaration des sortes AminoAcid et Protein

```
1 fmod PROTEIN is pr NAT .
2
3 sorts AminoAcid Protein .
```

```

4  subsort AminoAcid < Protein .
5
6  ops T Y S K P N~: -> AminoAcid .
7  ops pT pY pS~: -> AminoAcid . *** modified amino acids
8
9  endfm

```

spécifie dans un module fonctionnel `PROTEIN` de Maude (ligne 1) utilisant la spécification `NAT` préalablement définie dans Maude, deux sortes : `AminoAcid` et `Protein` (ligne 3) avec l'information de hiérarchisation (ligne 4) qui précise qu'un `AminoAcid` est un `Protein`. Les lignes 6 et 7 permettent de déclarer des exemples particuliers de `AminoAcid`.

De la même manière, les sortes `Thing`, `Family`, `Complex`, `Chemical`, `DNA` ... sont déclarées dans le module fonctionnel `THING` utilisant `PROTEIN`, avec la précision que `Protein`, `Family`, `Complex`, `Chemical`, `DNA` etc. sont des `Thing`. En particulier, on notera dans ce module la présence de l'opération

```
op (.-.): Thing Thing -> Complex
```

qui permet de spécifier l'opération de complexation, gérée de manière implicite. Il est cependant important de signaler que `PATHWAY LOGIC` permet également une gestion explicite de la complexation des molécules. C'est précisément ce qui lui permet de modéliser des processus en niveau d'abstraction 2 (considération des domaines fonctionnels).

La syntaxe de représentation d'une complexation explicitant les domaines fonctionnels est dans le listing 2.3 ci-dessous. On peut y déceler trois molécules différentes `Mol1`, `Mol2` et `Mol3` chacune étant représentée avec ses domaines fonctionnels :

- (S 11), ABC et (S 12) pour `Mol1`;
- (S 21) et (S 22) pour `Mol2`;
- (S 31), (S 32) et (S 33) pour `Mol3`.

Les domaines tels que (S 11), (S 12) ... marqués « bound » sont liés. Les lignes 5 à 7 explicitent les différentes liaisons. Ainsi on voit que toutes les molécules sont liées chacune sur deux domaines

Listing 2.3 – Exemple de liaison explicite

```

1  [Mol1 | (S 11 - bound), ABC, (S 12 - phos - bound)]
2  [Mol2 | (S 21 - bound), (S22 - bound)]
3  [Mol2 | (S 31 - bound), (S32 - bound), S33]
4
5  e((Mol1,(S 11)), (Mol2, (S 21))
6  e((Mol1,(S 12)), (Mol3, (S 31))
7  e((Mol2,(S 22)), (Mol3, (S 32)) .

```

A présent que nous avons donné les types nécessaires à la définition des molécules de base et des complexes, nous introduisons les sortes `Soup`, `Modification` et `ModSet` qui vont servir entre autre pour la définition des formes modifiées de molécules. La sorte `Soup` représente un ensemble quelconque de `Thing`,

le mot-clé « empty » désignant la *Soup* vide. Les sortes *Modification* et *ModSet* servent respectivement, comme leurs noms le laissent supposer, à définir les types de modifications de forme qu'on rencontre dans un processus biologique et un ensemble de *Modification* avec ou sans précision des sites de modification (*Site*). La forme modifiée d'une protéine est donnée par l'opération

$$\text{op } [_-] : \text{Protein ModSet} \Rightarrow \text{Protein}.$$

De la même manière, on spécifie les formes modifiées des autres *Thing* (*Chemical*, *DNA*, ...) à l'exception des *Complex* qui sont considérés comme des *Protein*.

La cellule biologique, vue comme un ensemble de compartiments cellulaires, chacun étant délimité par une membrane et contenant une *Soup*, est au cœur de toute modélisation PATHWAY LOGIC. Aussi, pour la représenter, les sortes *Cell*, *CellType*, *MemType* ont été définies avec *CellType* et *MemType* qui permettent de spécifier respectivement les types de cellules (épithéliale, nerveuse, musculaire...) et les types de membranes (membrane cytoplasmique, nucléaire, ...). Par exemple

$$\begin{aligned} & [\text{Cell} \mid \text{A B C}] \\ & \{\text{CM} \mid \text{A} \{\text{Cyto} \mid \text{B C}\}\} \end{aligned}$$

sont des cellules de type non spécifié (*Cell*) qui contiennent respectivement :

- une *Soup* composée des molécules A, B et C ;
- une *Soup* composée de A au niveau de la membrane cellulaire (*CM*) et une *Soup* composée de B et C dans le cytoplasme (*Cyto*).

On remarquera que dans le premier cas, le compartiment contenant les molécules n'a pas été précisé.

A côté du fichier *theops.maude* dont le contenu vient d'être brossé et qui change peu d'un modèle à un autre, nous avons le fichier *Components.maude* qui, lui, contient les déclarations des molécules de base, spécifiques au modèle en conception. Par exemple, pour la modélisation de la transduction de signal de la kinase MAP, les protéines *Ras* et *Egfr* sont déclarées dans ce fichier, qui peut inclure d'autres déclarations de sortes, à des fins de catégorisation particulière des molécules. Les sortes déclarées à ce niveau seront toujours des sous-sortes (direct ou non) d'une de celles déclarées dans le fichier *theops.maude*

2.2.2 Les composantes des règles de réaction

Dans PATHWAY LOGIC, les règles de réaction sont des règles de réécriture classiques ayant les formes

$$\begin{aligned} \text{r1[label]} & : \text{MG} \Rightarrow \text{MD} \\ \text{crl[label]} & : \text{MG} \Rightarrow \text{MD} \text{ if Condition} \end{aligned}$$

selon qu'elles sont dépourvues de conditions ou non. MG et MD sont des *Soup* donnant des caractérisations partielles d'états cellulaires. Toutes les règles de réaction d'un modèle sont regroupées dans le fichier *rules.maude* qui contient en outre des déclarations de variables.

L'identification de la règle est donnée par *rl[label* ou *crl[label]* où *label* est une composante facultative mais il est conseillé de la définir et il est important d'accorder une certaine attention à sa définition, car elle peut renseigner sur la réaction modélisée.

En plus des *Soup* MG et MD et des informations d'identification de la règle, on a comme autre classe de composantes des règles de réaction, les variables. Elles sont soit déclarées avant utilisation dans les règles, soit déclarées à l'utilisation. Elles permettent de définir

- des schémas de règles de réaction : Par exemple

```
rl[complex] : ?A:A [CellType:CellType | ct {CLm | c1m B}]
=>
[CellType:CellType | ct {CLm | c1m (?A:A : B)}]
```

est un schéma de règle de complexation entre une molécule B localisée sur la membrane cellulaire et une molécule de sorte A (?A:A) située à l'extérieur de la cellule. Elle comporte trois variables :

- *ct* et *c1m* déclarées en amont et qui tiennent lieu respectivement du contenu non décrit de la cellule et du contenu non décrit de la membrane cytoplasmique. Ces deux variables jouent des rôles passifs.
- ?A:A qui est déclarée à la volée et qui représente toute molécule de la sorte A. A et ses composantes de sorte A dans le modèle doivent avoir été déclarées plus tôt. Toute instantiation de ?A:A par une molécule de type A est une règle de réaction valide.
- Des règles conditionnelles : On distingue deux types de conditions selon que la variable considérée est une molécule ou une soupe (*Soup*) :
 - lorsque la variable est une molécule, la condition contraint les valeurs qu'elle peut prendre. Par exemple la règle ci-dessous :

```
crl[phos.A] : {CM | cm A ?V}
=> {CM | cm [A - Yphos] ?V}
if ?V S : Soup := B
```

est une règle de phosphorylation de A sur le site Y (Yphos) catalysée par B ou C, seules valeurs que peut prendre la variable ?V ;

- Lorsque la variable est une *Soup*, la condition précise les molécules qu'elle ne doit pas contenir. Par exemple

```
crl[phos.A] : {CM | cm A} => {CM | cm [A - Yphos]}
if cm has B = false /\ cm has C = false.
```

est une règle de phosphorylation de A sur le site Y à condition que la membrane cytoplasmique ne contienne ni B ni C.

Une règle conditionnelle peut contenir plusieurs conditions. Elles doivent être conjointement vérifiées pour que la règle soit valide.

2.2.3 Prise en compte de la compartimentation cellulaire

Dans PATHWAY LOGIC, la notion de compartimentation cellulaire est présente, mais sa prise en compte n'est pas toujours assurée. En effet, en raison de la spécification des règles qui veut que **MG** et **MD** soient des **Soup**, et attendu qu'une **Soup** est soit un ensemble de **Thing**, soit un ensemble de **Soup** localisées, soit une **Cell**, il est possible de définir des règles de réaction non localisées. Ainsi

$$A B \Rightarrow C$$

$$\{CL_m \mid A\} \{CL_i \mid B\} \Rightarrow \{CL_i \mid C\}$$

$$\{CM \mid A \{Cyto \mid B\}\} \Rightarrow \{Cyto \mid C\}$$

sont des règles de réaction toutes valides. On peut donc distinguer trois niveaux de prise en compte de la compartimentation cellulaire :

- le niveau 0 dans lequel les molécules ne sont pas localisées : C'est le cas de la première règle de réaction ;
- le niveau 1 qui fait apparaître des localisations mais sans information d'ordre topologique, permettant d'affirmer que tel compartiment est inclus dans tel autre : la deuxième règle est une illustration de ce niveau de prise en compte de la compartimentation ;
- le niveau 2 qui fait figurer les compartiments chacun à l'intérieur de son contenant : on peut voir dans le membre gauche de la troisième règle que **Cyto** est inclus dans **CM**.

PATHWAY LOGIC a établi deux conventions de dénomination des différents types de compartiments mis en évidence à nos jours, dans la cellule biologique.

- Dans la première convention, il s'agit d'identifier chaque type de compartiment par deux lettres majuscules : **CL** pour la cellule, **NU** pour le noyau, **GA** pour l'appareil de Golgi et ainsi de suite, on a les identificateurs pour les mitochondries, les lysosomes etc. Pour chacun de ces types de compartiments, quatre localisations sont prédéfinies par ajout de suffixes particuliers aux identificateurs à deux lettres :
 - suffixe « *o* » pour représenter l'*extérieur* du compartiment qui est l'espace délimité situé en dehors de la membrane dudit compartiment ;
 - suffixe « *m* » pour représenter la *membrane* du compartiment qui isole celui-ci de son environnement extérieur immédiat ;
 - suffixe « *i* » pour désigner l'*intérieur* immédiat (face interne) de la membrane du compartiment ;
 - suffixe « *c* » pour désigner l'intérieur du compartiment. Il s'agit de contenu du compartiment à l'exception des compartiments inclus.

Par exemple, on aura

$$CL_o, CL_m, CL_i, CL_c$$

pour représenter respectivement l'extérieur, la membrane cytoplasmique, la face interne de la membrane cytoplasmique et le cytosol.

- La deuxième convention ramène les compartiments à deux composantes : la membrane et le cytosol³. C'est dans cette convention qu'il est possible d'imbriquer des compartiments les uns dans les autres.

3. Nous entendons ici par "cytosol" tout contenu intra-compartiment sans distinction de type de compartiment.

Cependant, la liste des identificateurs de types de compartiment n'est pas exhaustive. On rencontre généralement **CM** pour la membrane cytoplasmique et **Cyto** pour le cytosol. De même **NM** désigne la membrane du noyau.

La prise en compte de la compartimentation dans **PATHWAY LOGIC** gagnerait à être renforcée pour d'une part permettre une gestion efficace des cas où on a affaire à une structure cellulaire comportant plusieurs compartiments de même type et d'autre part intégrer les voisinages par accollement qui sont incontournables dans la perspective qu'à **PATHWAY LOGIC** d'évoluer vers les modélisations intercellulaires.

2.3 Analyse des besoins

Dans les sections 2.1 et 2.2, nous avons mis en lumière une certaine faiblesse de la prise en compte de la compartimentation cellulaire dans les environnements de modélisation **BIOCHAM** et **PATHWAY LOGIC**. C'est un constat qui peut être généralisé à la plupart des outils de modélisation de processus biologiques. En effet, la littérature fait état de nombreux modèles à base de règles pour la catégorie de processus biologiques adressés, mais relativement peu se préoccupent de la compartimentation des structures cellulaires étudiées. Pour illustration, en juin 2007, sur les cent-douze (112) modèles validés que comptait **BioModels** (<http://www.biomodels.net>), trente-cinq (35) comportaient plus d'un compartiment, et seulement sept (7) donnaient des informations relatives à la topologie : tel compartiment est inclus dans tel autre [?].

Si jusqu'à récemment, l'absence de considérations de type « cohérence d'un modèle par rapport à une compartimentation cellulaire » n'a pas été ressentie comme un problème crucial dans les langages à base de règles pour la biologie, c'est sans doute que les processus modélisés par ces environnements se déroulaient dans des structures cellulaires à la topologie évidente. Les processus se déroulent généralement dans un même compartiment d'une même cellule, ou lorsqu'il y en a deux, il s'agit du cas typique du cytoplasme et du noyau. On peut donc supposer que les modèles définis jusque là sont conformes à une compartimentation cellulaire cohérente. Du reste **BIOCHAM** a testé avec succès la consistance de certains de ces modèles [?] en se basant sur les informations de topologie données dans **Biomodels**. Cependant, les récentes évolutions ou les perspectives d'évolution respectivement de **BIOCHAM** et **PATHWAY LOGIC** rendent nécessaire une prise en compte soignée de la compartimentation cellulaire en termes de types de compartiment et de répartition spatiale de ces compartiments.

C'est pour apporter une première réponse à cela, que nous cherchons à offrir au biologiste-modélisateur le moyen de définir simplement, un modèle de processus biologique à base de règles, compilable dans **BIOCHAM** ou **PATHWAY LOGIC**, qui soit compatible avec la topologie de la structure cellulaire sous étude. La compatibilité d'un modèle avec une compartimentation cellulaire se traduisant par la vérification par le modèle, des deux conditions suivantes :

- les règles de réactions s'appliquent dans les compartiments de types appropriés ;
- si une règle de réaction implique plusieurs compartiments, ceux-ci doivent respecter certaines conditions de voisinage.

Nous présentons notre approche en en donnant une schématisation (figure 2.1) dans laquelle les rectangles aux bords arrondis représentent les différentes procédures que nous avons définies, et qui prennent en entrée ou produisent des données schématisées sous formes d'ellipses, tandis que les formes hexagonales

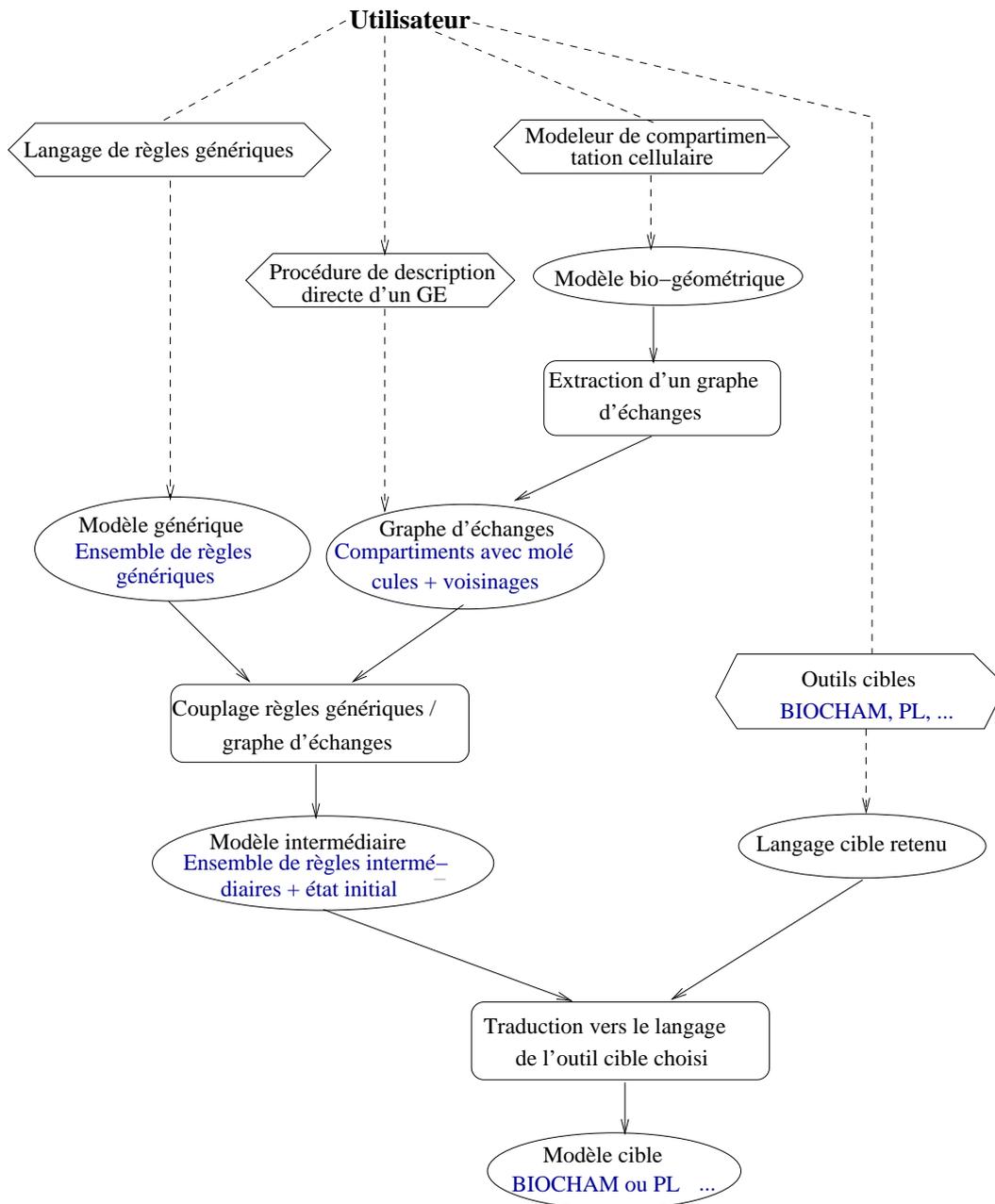


FIGURE 2.1 – Schématisation de l’approche

servent à représenter les moyens (outils ou formalismes) mis à la disposition du modélisateur pour formater et/ou fournir les données en entrée des différentes procédures. Ainsi, l'obtention d'un modèle dans le langage de règles de l'outil choisi par l'utilisateur (BIOCHAM ou PATHWAY LOGIC), qui constitue le cœur et la finalité de notre travail, part de deux éléments distincts, en entrée de la procédure de couplage. Il s'agit :

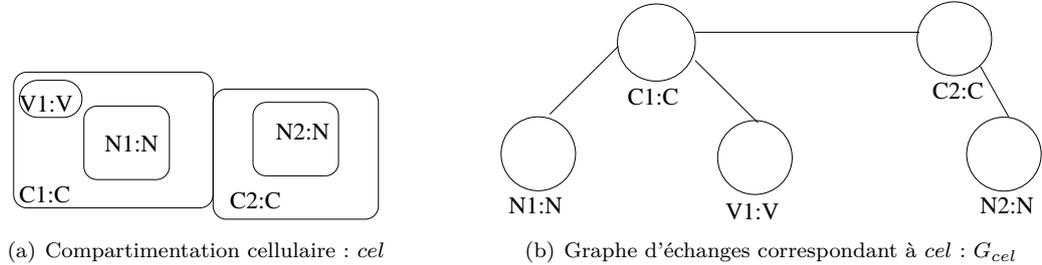


FIGURE 2.2 – Une compartimentation cellulaire et le graphe d'échanges correspondant

- d'un *graphe d'échanges* (cf. section 3.2) qui fournit une abstraction d'une compartimentation cellulaire en termes d'informations topologiques ; il indique l'ensemble des compartiments composant le système cellulaire étudié, et pour chaque compartiment, l'information de son type et ses voisinages avec les autres compartiments. Par exemple, la figure 2.2 montre en 2.2(a) une compartimentation cellulaire *cel*. Chaque compartiment y est représenté par un rectangle aux bords arrondis étiqueté d'un identificateur et d'un type de compartiment séparés par « : ». Ainsi, *cel* comporte deux compartiments de type N , $N1$ et $N2$ contenus respectivement dans deux compartiments de type C adhérant l'un à l'autre $C1$ et $C2$. $C1$ contient également un compartiment $V1$ de type V . Cette compartimentation cellulaire peut être abstraite par le graphe d'échanges G_{cel} donné par la figure 2.2(b) dans lequel les compartiments sont représentés sous forme de cercles avec en dessous leur nom et leur type. On retrouve effectivement dans G_{cel} les mêmes compartiments que dans *cel* et les relations de voisinage existant entre eux : C_1 voisin à $N1$, à $V1$ et à $C2$ qui a également comme autre voisin $N2$.
- d'un *modèle générique* essentiellement composé de *règles génériques* (cf. section 4.1) qui sont des schémas de règles modélisant des réactions biochimiques, dans lesquels les informations de localisation des molécules sont des *types de compartiment* plutôt que des compartiments. Par exemple, une règle générique *rg* pourra se libeller

” transport d'une molécule M_1 d'un compartiment de type cytoplasme vers un compartiment de type noyau, si ces deux compartiments sont voisins. ”

Ces deux éléments, le graphe d'échanges et le modèle générique, sont ensuite couplés sur la base des informations de types de compartiments existant de part et d'autre. En fonction des informations de voisinage recelées par le graphe d'échanges, la procédure de couplage fournit un modèle dit *modèle intermédiaire* (section 4.2) qui comporte des *règles intermédiaires*. Celles-ci sont des règles de réactions biochimiques, instances particulières (car respectant des conditions particulières de voisinage) de règles génériques, dans lesquelles les molécules sont localisées dans les compartiments donnés par le graphe d'échanges. En considérant le graphe d'échanges G_{cel} (convenons que le type C désigne le cytoplasme et N le noyau) et la règle générique *rg*, nous obtenons comme ensemble de règles intermédiaires celui aux deux éléments

$\{ri_1 : \text{transport d'une molécule } M_1 \text{ de } C_1 \text{ vers } N_1\}$
 $\{ri_2 : \text{transport d'une molécule } M_1 \text{ de } C_2 \text{ vers } N_2\}$

L'ultime étape pour l'obtention du modèle BIOCHAM ou PATHWAY LOGIC est celle de traduction des règles intermédiaires vers le langage de l'outil cible choisi par l'utilisateur. Les deux règles intermédiaires donneront en BIOCHAM

$$\begin{aligned}M_1 :: C_1 &\Rightarrow M_1 :: N_1. \\M_1 :: C_2 &\Rightarrow M_1 :: N_2.\end{aligned}$$

et en PATHWAY LOGIC

$$\begin{aligned}rl[ri_1] &: \{C_1 | c_1 M_1\} \{N_1 | n_1\} \Rightarrow \{C_1|c_1\} \{N_1 | n_1 M_1\} . \\rl[ri_2] &: \{C_2 | c_1 M_1\} \{N_2 | n_1\} \Rightarrow \{C_2|c_1\} \{N_2|n_1 M_1\} .\end{aligned}$$

Revenons sur la question de l'acquisition des données en entrée de la procédure de couplage. La figure 2.1 (page 27) montre deux moyens d'obtention du graphe d'échanges qui doit *absolument être cohérent par rapport à la compartimentation cellulaire étudiée*. La première procédure qui consiste à le décrire directement est un raccourci prévu pour permettre à l'utilisateur de passer outre l'étape de modélisation, dans les cas où la compartimentation cellulaire est relativement simple (se limitant par exemple à un cytoplasme contenant un noyau) pour donner lieu à un graphe d'échanges *évident*. La seconde approche, garantissant par construction la cohérence avec une structure cellulaire réaliste consiste à représenter dans un premier temps la compartimentation cellulaire du système sous étude et à ensuite extraire de cette représentation le graphe d'échanges correspondant. Le graphe d'échanges ainsi obtenu est cohérent par rapport à la compartimentation cellulaire étudiée.

Quant au modèle générique, l'utilisateur le définit en se servant d'un langage dit *langage de règles génériques* qui répond en particulier au besoin qu'à l'utilisateur d'exprimer en fonction de types de compartiment, des règles biochimiques qui une fois instanciées traduiraient des réactions interprétables dans BIOCHAM et PATHWAY LOGIC. Nous détaillerons ce langage au niveau de la section 4.1 .

L'objet de la présente section consiste donc à préciser nos besoins au regard des deux entités considérées en entrée de notre approche, à savoir le graphe d'échanges et le langage de règles. Nous sommes dirigés par deux préoccupations :

- préparer le couplage entre nos règles génériques et la structure topologique abstraite par le graphe d'échanges ;
- anticiper une traduction des règles obtenues par la procédure de couplage vers des modèles BIOCHAM ou PATHWAY LOGIC qui soit aussi conforme que possible à l'intuition.

Pour le graphe d'échanges, nous adopterons une représentation simple sous forme de graphe où les informations présentes sont essentiellement celles d'identification et de typage des différents compartiments ainsi que les voisinages entre ces compartiments. A ces données viendront s'ajouter des données biochimiques correspondant aux molécules présentes dans la structure cellulaire abstraite. Quant au niveau d'expressivité du langage des règles génériques, directement lié aux aspects de modélisation de processus biologiques par les règles que nous prendront en compte, il sera discuté à la lumière de ce qui a été écrit dans les sections 2.1 et 2.2. C'est dire que nous chercherons à inclure dans notre langage, les éléments de modélisation pris en compte par BIOCHAM et PATHWAY LOGIC.

Nous détaillons dans la suite de la section les options que nous avons privilégiées dans ce document.

2.3.1 Exemple d'illustration de notre approche

Nous considérons comme éléments en entrée :

- l'ensemble RGs de quatre règles de réaction biochimique, données par le listing 2.4 en langage naturel, où $M1$ et $M2$ sont des molécules et C , E , N et V des types de compartiment ;
- la compartimentation cellulaire cel donnée par la figure 2.2(a).

Listing 2.4 – RGs : Règles génériques en langage naturel pour l'illustration de notre approche

- 1 rg1 : Transport de $M2$ d'un compartiment de type V vers un autre de type C voisin.
- 2 rg2 : Complexation de $M1$ et $M2$ dans un compartiment de type C .
- 3 rg3 : Transport du complexe de $M1$ et $M2$, d'un compartiment de type C vers un autre de type N voisin.
- 4 rg4 : Phosphorylation de $M2$ dans un compartiment de type E sous l'action de $M1$ dans ce compartiment
- 5 rg5 : $M1$ dans un compartiment V_i de type V et $M2$ dans un compartiment C_j de type C donnent un complexe $M1-M2$ dans un compartiment N_k de type N , si V_i et C_j sont voisins et C_j et N_k sont voisins
- 6 rg6 : $M1$ dans un compartiment V_i de type V et $M2$ dans un compartiment C_j de type C donnent un complexe $M1-M2$ dans un compartiment N_k de type N , si V_i et C_j sont voisins, C_j et N_k sont voisins et V_i et N_k sont aussi voisins.

La lecture des éléments de RGs met en évidence l'importance des types de compartiment dans les règles génériques : toutes les règles sont exprimées en fonction des localisations des molécules dans des compartiments identifiés par leur type. On remarquera que la présence simultanée de $M1$ et $M2$ dans un compartiment entraîne des réactions différentes selon le type de celui-ci : une complexation pour un compartiment de type C et une phosphorylation de $M2$ pour un compartiment de type E . On notera également que la règle $rg6$ exprime la même réaction que $rg5$ sous condition d'un voisinage supplémentaire entre N_k et V_i . Les règles sont définies indépendamment de toute structure cellulaire : nous défendons que les règles biochimiques sont essentiellement contextualisées par la nature et les relations de voisinage des compartiments.

A partir des éléments donnés ou calculés RGs et G_{cel} (cf figure 2.2(b)), la procédure de couplage va instancier chaque règle de RGs en respectant notamment les types de compartiment et les voisinages entre ceux-ci (donnés par G_{cel}). Nous obtenons comme réactions intermédiaires celles du listing suivant 2.5 (page 30) suivant.

Listing 2.5 – Règles intermédiaires issues du couplage règles de RGs (listing 2.4) et G_{cel} (figure 2.2(b))

- 1 rg1.1 : Transport de $M2$ de $V1$ vers $C1$
- 2 rg2.1 : Complexation de $M1$ et $M2$ dans $C1$
- 3 rg2.2 : Complexation de $M1$ et $M2$ dans $C2$
- 4 rg3.1 : Transport du complexe de $M1-M2$, de $C1$ vers $N1$
- 5 rg3.2 : Transport du complexe de $M1-M2$, de $C2$ vers $N2$
- 6 rg5.1 : $M1$ dans $V1$ et $M2$ dans $C1$ donne un complexe $M1-M2$ dans $N1$

On constate que $rg4$ n'a pas été instanciée, ce qui est normal parce que G_{cel} ne comporte aucun compartiment de type E . Par ailleurs, $rg1$ n'est pas instanciée pour $C2$ qui n'a pas de voisin de type V .

L'étape suivante est la traduction des règles en BIOCHAM (on obtient le listing 2.6) ou en PATHWAY LOGIC (listing 2.7).

Listing 2.6 – Règles BIOCHAM issues de la traduction des règles intermédiaires du listing 2.5

```

1 rg1_1 : M2::V1 => M2::C1.
2 rg2_1 : M1::C1 + M2::C1 => M1-M2::C1.
3 rg2_2 : M1::C2 + M2::C2 => M1-M2::C2.
4 rg3_1 : M1-M2::C1 => M1-M2::N1.
5 rg3_2 : M1-M2::C2 => M1-M2::N2.
6 rg5_1 : M1::V1 + M2::C1 => M1-M2::N1.

```

Listing 2.7 – Règles PATHWAY LOGIC issues de la traduction des règles intermédiaires du listing 2.5

```

1 rl[rg1_1] : {V1 | v1 M2} {C1 | c1} => {V1 | v1} {C1 | c1 M2} .
2 rl[rg2_1] : {C1 | c1 M1 M2} => {C1 | c1 M1:M2} .
3 rl[rg2_2] : {C2 | c2 M1 M2} => {C2 | c2 M1:M2} .
4 rl[rg3_1] : {C1 | c1 M1:M2} {N1 | n1} => {C1 | c1} {N1 | n1 M1:M2} .
5 rl[rg3_2] : {C2 | c2 M2:M2} {N2 | n2} => {C2 | c2} {N2 | n2 :M2} .
6 rl[rg5_1] : {V1 | v1 M1} {C1 | c1 M2} => {N1 | n1 M1:M2} .

```

2.3.2 Graphe d'échanges et compartimentation cellulaire

Les logiciels de modélisation géométrique sont appelés *modeleurs*. La construction dans un modeleur d'un objet 3D représentant une compartimentation cellulaire présente certaines difficultés. En particulier, il faut s'assurer que deux compartiments quelconques ne s'intersectent pas. Par ailleurs, un compartiment $C1$ inclus dans un autre $C2$ doit être effectivement représenté à l'intérieur de la représentation de $C2$, de même que deux compartiments accolés devront « se toucher » dans la représentation « objet 3D » de la compartimentation. Notre ambition à ce niveau est d'offrir à l'utilisateur les outils facilitant la prise en compte de ces contraintes. Comme ce qui nous importe, ce sont les positions relatives entre compartiments, plus que leurs dimensions géométriques, nous avons choisi de spécialiser un modeleur géométrique en le munissant d'opérations de création de compartiments, de forme parallélépipédique, toutes positionnés selon les mêmes axes. Cela nous permettra de construire des compartimentations simplifiées, car cubiques, dans lesquelles les questions de voisinage (objets accolés, intersection d'objets, inclusion d'objets) seront plus simples à appréhender. Nous reviendrons sur ce parti-pris dans le prochain chapitre.

A partir d'un objet 3D représentant une compartimentation sous étude, nous allons extraire le graphe d'échanges correspondant explicitant les seules informations utiles, i.e. relatives à la topologie de la structure cellulaire étudiée. Ainsi, la procédure d'extraction assurera les faits suivants :

- Tous les compartiments représentés et seulement eux, seront répertoriés dans le graphe d'échanges ;
- Tous les voisinages représentés et seulement eux, seront répertoriés dans le graphe d'échanges.

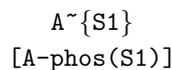
La cohérence géométrique de la représentation initiale garantit celle du graphe d'échanges extrait. Celle-ci se traduit par le fait que tous les arcs du graphe peuvent être catégorisés comme traduisant soit une inclusion d'un compartiment dans un autre, soit une adhérence entre deux compartiments.

2.3.3 Langage des règles génériques

En raison même de la nature des processus modélisés, les molécules (composés chimiques et biochimiques) constituent la première catégorie d'objets formels manipulés dans les deux environnements. On

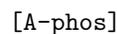
y rencontre les molécules dites de base et les deux principales opérations qui permettent de définir des formes altérées de molécules ou des complexes. Au delà des notions et de ce qu'elles recouvrent, quelques divergences peuvent être observées :

- Au niveau des molécules de base, si les deux environnements s'accordent sur le fait qu'il s'agit de simples identificateurs, PATHWAY LOGIC leur associe quelquefois des domaines fonctionnels. C'est ce qui du reste lui donne la possibilité de gérer la complexation de manière explicite. La modélisation au niveau d'abstraction 2 (domaines fonctionnels) n'étant pas prise en compte dans BIOCHAM et étant rarement utilisée dans PATHWAY LOGIC, nous convenons de nous passer de cette option de précision des domaines fonctionnels.
- En ce qui concerne la complexation, pour les mêmes raisons que précédemment, nous nous en tiendrons à la forme implicite.
- Les différences au niveau des opérations d'altération de forme des molécules sont de trois ordres :
 - Dans BIOCHAM, il est impératif de préciser les sites de modification dans les formes phosphorylées des protéines, tandis que cette information est optionnelle dans PATHWAY LOGIC. Par exemple



correspondent à la forme phosphorylée de la molécule A sur le site S1 respectivement en BIOCHAM et PATHWAY LOGIC. Cette différence peut être minimisée du moment qu'elle n'induit pas une difficulté (voire une impossibilité) de représentation de l'objet.

- Le type de modification n'est pas mentionné dans BIOCHAM qui ne considère que la phosphorylation comme seule modification de forme, alors qu'il est obligatoire dans PATHWAY LOGIC où sont définies d'autres opérations de modification de forme telles que l'acétylation. La molécule

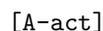


qui désigne en PATHWAY LOGIC n'importe quelle forme phosphorylée de A ne pourra pas être exprimée de façon « naturelle » en BIOCHAM. En effet pour définir cela en BIOCHAM, on pourrait, après avoir associé par déclaration préalable de tous ses sites de modification à A, utiliser le schéma d'objet



On pourra ainsi comme on le souhaite, générer toutes les formes phosphorylées de A. Cependant, ce schéma d'objets génère également A lui-même, ce qui n'est pas le résultat recherché. En forçant ? à ne pas matcher avec `vide`, le problème est résolu, mais cela relève d'une mauvaise pratique et provoque de ce fait un *warning* dans la mesure où BIOCHAM considère qu'une molécule doit exister dans sa forme native (non altérée).

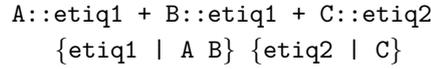
- Le fait pour PATHWAY LOGIC et BIOCHAM de ne pas considérer le même ensemble de modifications de forme, est également source de problème. Par exemple



en PATHWAY LOGIC, désignant la forme active de A ne pourra pas être exprimé en BIOCHAM. Ces difficultés doivent être contournées. En effet, en raison de l'importance des phénomènes de modification de forme dans les processus biologiques, il ne serait pas pertinent d'envisager un langage

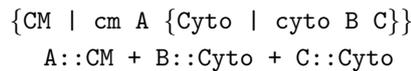
de règles pour les interactions biochimiques qui ne tiennent pas compte de la modification de forme. Dans le chapitre 4, nous exposons la solution que nous proposons pour ce problème.

Les langages de règles de BIOCHAM et de PATHWAY LOGIC incorporent la notion de localisation. La seule différence notable est que lorsqu'une localisation est utilisée, elle concerne une molécule pour BIOCHAM et une Soup (ensemble de molécules) pour PATHWAY LOGIC. Par exemple



désignent les mêmes solutions ou Soup respectivement en BIOCHAM et PATHWAY LOGIC.

Même si PATHWAY LOGIC intègre une hiérarchisation de compartiments dans certains cas, l'information, qui n'a aucune pertinence pour BIOCHAM qui ne gère pas les aspects topologiques sera ignorée. Aussi,



désignent également la même Soup ou solution respectivement en PATHWAY LOGIC et BIOCHAM. Ceci étant, nous pensons qu'il est important d'inclure dans notre langage de règles, la décomposition d'un compartiment en ses parties (membrane, intérieur, ...), ce en raison entre autres du fait qu'elle permet de donner plus de précision sur la localisation. Par ailleurs, la plupart des modèles de processus biologiques à base de règles (159 sur 259 pour la cellule prise comme compartiment, et 32 sur 82 pour les organelles prises comme compartiments [?]) en tient compte.

Nous distinguerons règle de réaction et schéma de règle de réaction qui donne lieu par instanciation à plusieurs règles de réaction construits sur le même modèle.

Les deux types de règles de réaction que nous rencontrons dans les deux langages sont les règles non conditionnelles et les règles conditionnelles. De manière générale, la possibilité de représentation d'une même règle de réaction dans BIOCHAM ou dans PATHWAY LOGIC dépend de la possibilité de représentation de ses composantes qui sont principalement des molécules et des localisations. Les règles de réactions non conditionnelles pourront être représentées sans difficulté notable dans BIOCHAM comme dans PATHWAY LOGIC. Il n'en va pas de même des règles conditionnelles, dans lesquelles interviennent des composantes supplémentaires : les variables et la condition. Pour déterminer quelles formes de règles nous pouvons inclure (il s'agit de celles qui peuvent s'exprimer dans les deux langages), nous considérons les règles conditionnelles de BIOCHAM (respectivement PATHWAY LOGIC) et nous étudions la possibilité de les exprimer en PATHWAY LOGIC (respectivement BIOCHAM).

En BIOCHAM, ce sont les conditions sur paramètres cinétiques qui donnent lieu à des règles conditionnelles. Étant donné que PATHWAY LOGIC est une approche purement qualitative, nous pouvons affirmer que sa syntaxe n'a pas prévu la prise en compte de tels paramètres.

Nous avons vu section 2.2.2 que PATHWAY LOGIC permettait deux classes de conditions. Une règle conditionnelle dans laquelle toutes les conditions relèvent de la première classe (les variables sont des variables de molécules) pourra être exprimée en BIOCHAM. Par exemple

```
[crl[decomp1] : [?V1 : ?V2] => ?V1 ?V2
if ?V1 S1:Soup := A C  $\wedge$  ?V2 S2:Soup := B D
```

sera rendue en BIOCHAM par

```
V1-$V2 => $V1 + $V2 where $V1 in {A,C} and $V2 in {B, D}
```

qui correspond non pas à une règle conditionnelle mais à un schéma de règle de réaction.

Quant aux règles conditionnelles dans lesquelles une au moins des variables est une variable de **Soup**, telles que

```
crl[phos.A] : {CM | cm A} => {CM | cm [A - Yphos]}
if cm has B = false  $\wedge$  cm has C = false
```

elles ne peuvent pas être exprimées naturellement en BIOCHAM où les variables sont de variables de molécules, de sites de phosphorylation, de localisation mais pas de **Solution**.

Les schémas de règles de réaction sont utilisés dans BIOCHAM par le biais de l'utilisation de schémas d'objets. Ils peuvent être exprimés en PATHWAY LOGIC uniquement sous certaines conditions : toutes les variables qui y apparaissent portent sur des variables de molécules et leurs valeurs sont contraintes avec l'opérateur **in**. Par exemple le schéma de règle

```
V1-$V2 => $V1 + $V2 where V1 in {A, C} and $V2 in {B, D}
```

sera rendu en PATHWAY LOGIC par

```
crl[decomp1] : [?V1 : ?V2] => ?V1 ?V2 if ?V1 S1:Soup := A C  $\wedge$  ?V2 S2:Soup:= B D
```

Le tableau 2.1 fournit une synthèse des arguments développés précédemment. Il s'agit d'un tableau à quatre colonnes qui donne pour chaque aspect de modélisation de processus biologique considéré (colonne 1), l'information de sa prise en compte dans BIOCHAM (colonne 2), dans PATHWAY LOGIC (colonne 3), notre choix de l'inclure ou non dans notre langage de règles (colonne 4) et un petit commentaire si nécessaire qui explique notre option (colonne 5).

Éléments	BIOCHAM	PATHWAY LOGIC	Oui/Non	Commentaires
Molécules de base	Identificateurs	Identificateurs + domaines fonctionnels	Oui	en tant qu'identificateurs
Complexation	Implicite	Implicite et explicite	Oui	Implicite
Modification	Oui sans type et avec sites	Oui avec types et possiblement sans sites	Oui	On ne peut pas s'en passer, alors on trouvera le moyen de gérer les divergences
Localisations	Étiquettes	Avec ou sans hiérarchisation des localisations	oui.	Types de compartiments
Paramètres cinétiques	Oui	Non	Oui	Point important de BIOCHAM qui intègre parfois des informations de voisinage et dont la prise en compte ne nuit pas à la traduction vers PATHWAY LOGIC
Règles non conditionnelles	Oui	Oui	Oui	
Règles conditionnelles	Oui	Oui	Oui	Il faut trouver le moyen de ne pas traduire celles qui ne peuvent pas l'être pour un langage considéré
Schémas de règles	oui	Non	Oui	Celles où les variables désignent des molécules

TABLE 2.1 – Tableau récapitulatif des éléments à inclure dans le nouveau langage de règles

Chapitre 3

Modélisation géométrique à base topologique des compartiments cellulaires

Sommaire

3.1	Modélisation géométrique	38
3.1.1	Quelques modèles	39
3.1.2	Modèles géométriques à base topologique	42
3.1.3	Graphes d'incidence, d'adjacence et arbres d'inclusion	43
3.1.4	Les cartes généralisées	46
3.2	Graphe d'échanges	52
3.2.1	Rappels sur les graphes	53
3.2.2	Principales caractéristiques et définition du graphe d'échanges	56
3.2.3	Opérations applicables dans un graphe d'échanges	58
3.3	Modélisation des compartiments cellulaires	60
3.3.1	Le modèle bio-géométrique	60
3.3.2	Opérations de création des compartiments cellulaires	63
3.3.3	Extraction du graphe d'échange	64
3.4	Implémentation	65
3.4.1	Présentation de MOKA	66
3.4.2	Construction d'une structure cellulaire	68
3.4.3	Extraction d'un graphe d'échanges à partir d'un modèle bio-géométrique	75
3.5	Application	78
3.6	Conclusion et perspectives	82

Comme nous l'avons expliqué dans la section 2.3 (page 26) notre but est de spécialiser des règles génériques pour une compartimentation cellulaire donnée. Cette spécialisation s'effectue à partir du graphe d'échanges qui est une carte des échanges possibles entre les compartiments cellulaires considérés. En particulier deux compartiments voisins, par inclusion ou par adhérence, pourront échanger, tandis que deux compartiments distants ne pourront pas le faire. Cependant, la construction manuelle du graphe d'échanges

devient très rapidement complexe lorsque le nombre de compartiments cellulaires croît. Le but de cette section est d'utiliser la modélisation géométrique pour représenter les compartimentations cellulaires, puis extraire automatiquement de ces représentations, les graphes d'échanges. C'est principalement en raison des avancées notables enregistrées ces dernières années par la modélisation géométrique que nous avons investigué de ce côté pour identifier des modèles adaptés pour la représentation des compartimentations cellulaires. En effet, la modélisation géométrique couvre désormais une grande diversité de domaines d'applications et a, dans la foulée, utilisé des structures de données bien adaptées pour représenter les systèmes où les relations de voisinage jouent un rôle important, comme c'est le cas pour les systèmes cellulaires.



FIGURE 3.1 – Deux plongements pour une même structure topologique

Ainsi, notre modélisation des compartiments cellulaires sera fondée sur les *cartes généralisées* qui entrent dans le cadre de la modélisation géométrique à base topologique dont une des principales caractéristiques est la nette distinction qui est faite entre *topologie* et *géométrie*. La topologie d'un objet est sa structuration en terme de sommets, arêtes, faces, volumes, etc. Par opposition, la géométrie de l'objet quant à elle, précise la forme et la position des différentes *cellules topologiques*. Ainsi, deux objets différents peuvent avoir la même structure topologique. Par exemple, la figure 3.1 présente un cube et une sphère qui ont la même topologie constituée d'un volume de 6 faces (dont 3 visibles), etc.

L'organisation du chapitre se fera autour des sections suivantes :

- la section 3.1 nous permet de dresser un bref état de l'art de la modélisation géométrique et des principaux modèles de représentation des objets géométriques, puis nous présenterons plus en détail les modèles géométriques à base topologique et plus particulièrement celui des cartes généralisées ;
- la section 3.2 quant à elle, va nous permettre de détailler les particularités du graphe d'échanges ;
- nous aborderons en section 3.3 la présentation du modèle géométrique et topologique utilisé pour représenter les compartimentations cellulaires à travers ses principes fondamentaux et les informations qu'il recèle ;
- la section 3.4 sera consacrée à l'implémentation du modèleur de compartiments cellulaires. Nous concluons le chapitre par quelques cas d'utilisation de notre modèleur (section 3.5).

3.1 Modélisation géométrique

La modélisation géométrique relève de l'informatique graphique. Elle permet de représenter et de manipuler des objets géométriques sur ordinateur. Ses domaines d'application sont diversifiés. On peut citer :

- la mécanique, avec le Dessin et la Conception Assistée par Ordinateur (DAO/CAO) en aéronautique ou en construction automobile par exemple ;

- l’architecture, avec la réalisation de plans de bâtiments ou la visite virtuelle de villes ;
- l’imagerie médicale, avec la reconstruction d’organes en 3D, par exemple pour la simulation d’opérations chirurgicales endoscopiques ;
- le cinéma et les jeux vidéo, avec la réalisation d’effets spéciaux, la création d’êtres virtuels ;
- la simulation scientifique, notamment dans le domaine biologique, avec la représentation de molécules, de compartimentations cellulaires ;
- etc.

La modélisation géométrique concerne différentes catégories d’objets. Dans le cadre de ce document, nous considérons uniquement la modélisation des objets structurés. Ainsi, n’entrent pas en ligne de compte les modèles de fluides (Computational Fluid Dynamic (CFD)) généralement à base de particules et d’interactions mécaniques entre ces particules.

3.1.1 Quelques modèles

Les modeleurs géométriques utilisent des structures de données différentes selon l’ensemble des objets que l’on peut représenter. Bien que les travaux théoriques portent souvent sur des structures en toute dimension, la plupart des outils existants implémentent des modèles en dimension allant de 2 à 4 (2D, 3D, 4D). La quatrième dimension est généralement utilisée pour l’animation d’objets 3D, on parle alors de 3D plus temps. Le modeleur Catia de Dassault et le modeleur open source Blender sont des outils représentatifs de cette classe de modeleurs.

Les modèles 2D sont des outils de traitement de dessins techniques qui utilisent des points et des lignes. Ils sont bien adaptés pour la production de plans en Dessin Assisté par Ordinateur (DAO) qui utilise les techniques classiques du dessin industriel pour représenter les objets. Ces modèles représentent les objets par leurs contours. La modélisation 2D présente de nombreux avantages tels que :

- la facilité d’apprentissage et d’utilisation ;
- la facilité de modification d’un dessin ;
- le gain de temps dans la construction de plans.

Leur principale limite est qu’ils ne permettent aucune relation entre les différentes vues d’une même pièce ou d’un même mécanisme. On peut citer comme modeleur 2D *Mecaplan*, le premier logiciel à avoir associé mécanique et informatique. La version actuelle *Mecaplan SW* peut être chargée dans *SolidWorks*¹ qui est un logiciel de CAO.

Dans la catégorie de modeleurs permettant des représentations en dimension supérieure à deux, la modélisation tridimensionnelle semble de prime abord la plus naturelle étant donné que les objets du monde réel sont en 3D. Des modèles de nature différente ont été définis et peuvent être regroupés en trois catégories que nous allons détailler dans les paragraphes 3.1.1.1 à 3.1.1.3 suivants. Il s’agit :

- des modèles *filaires ou en fil de fer* ;
- des modèles de la *géométrie de construction de solides* ;
- des modèles de *représentation par les bords*.

3.1.1.1 Le modèle en fil de fer

Les représentations fil de fer constituent le premier niveau de modélisation dans l’espace. Un objet est représenté par un ensemble de sommets, reliés entre eux par des arêtes comme le montre la figure 3.2(a)

1. Logiciel de DAO de Dassault Systèmes

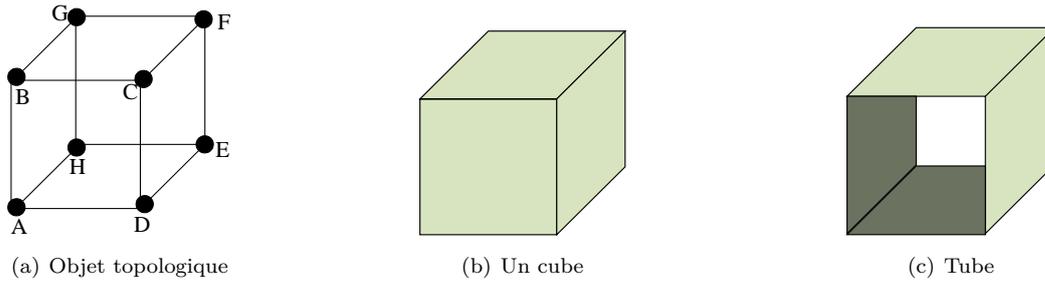


FIGURE 3.2 – Ambiguïté du modèle en fil de fer

représentant une structure topologique : on y distingue les différents sommets de l'objet identifiés par les points A à H, ainsi que les arêtes qui relient chacune deux sommets.

Le principal point faible de ce modèle est l'ambiguïté des représentations due au fait que la notion de face n'est pas prise en compte. Ainsi, une même représentation peut correspondre à des objets différents. Sur la figure 3.2 par exemple, pour la même structure en fil de fer (3.2(a)), on obtient un cube (3.2(b)) ou un tube (3.2(c)) selon que les faces ABCD et HEFG existent ou non.

3.1.1.2 La Géométrie de construction de solides

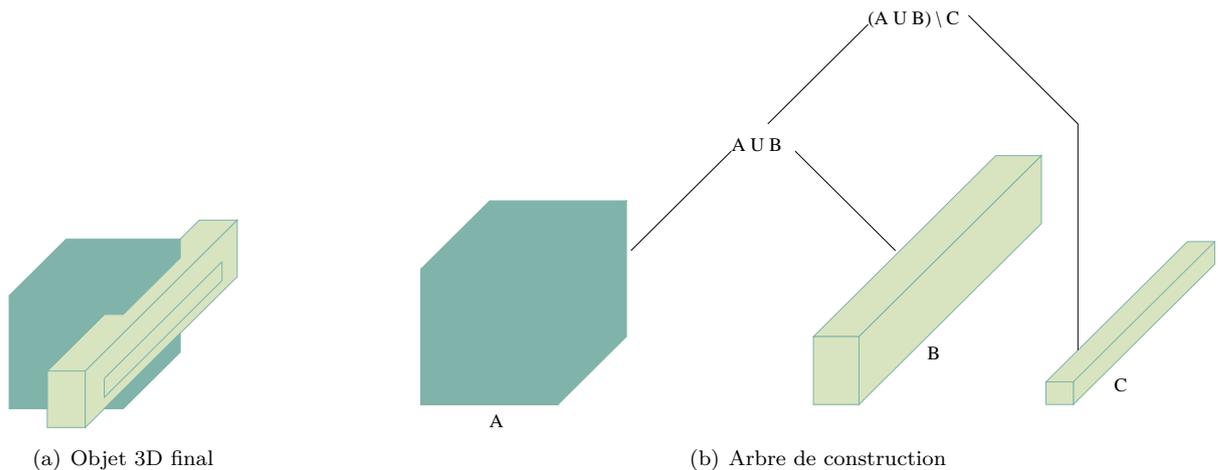


FIGURE 3.3 – Représentation par arbre CSG

Dans la *géométrie de construction de solides* (*Constructive Solid Geometry (CSG) en anglais*) [?], la représentation des objets s'appuie sur un ensemble de solides élémentaires (cube, cylindre, sphère, polyèdre etc), auxquels on applique des opérateurs ensemblistes (union, intersection, différence etc) et des opérations géométriques notamment des rotations et des translations, tout en conservant les étapes de la construction. L'expressivité du modèle est directement liée au nombre de formes de base et à celui des opérations applicables.

La représentation CSG d'un objet est un arbre dont les nœuds internes sont des opérateurs ensemblistes et les feuilles sont des solides élémentaires appelés *primitives*, comme le montre la figure 3.3. L'objet à représenter (figure 3.3(a)) peut être modélisé par l'arbre donné par la figure 3.3(b). On constate que

les nœuds internes de l'arbre correspondent à des opérateurs ensemblistes (\cup , \cap , $/$) et constituent des racines d'arbres CSG d'objets intermédiaires, tandis que les feuilles correspondent aux solides élémentaires utilisés.

Les principaux avantages de la représentation des objets en CSG sont d'une part la rapidité de mise en œuvre, due en partie à la possibilité de réutilisation de tout ou partie d'un modèle dans un autre et à celle d'avoir à disposition une riche bibliothèque de solides élémentaires.

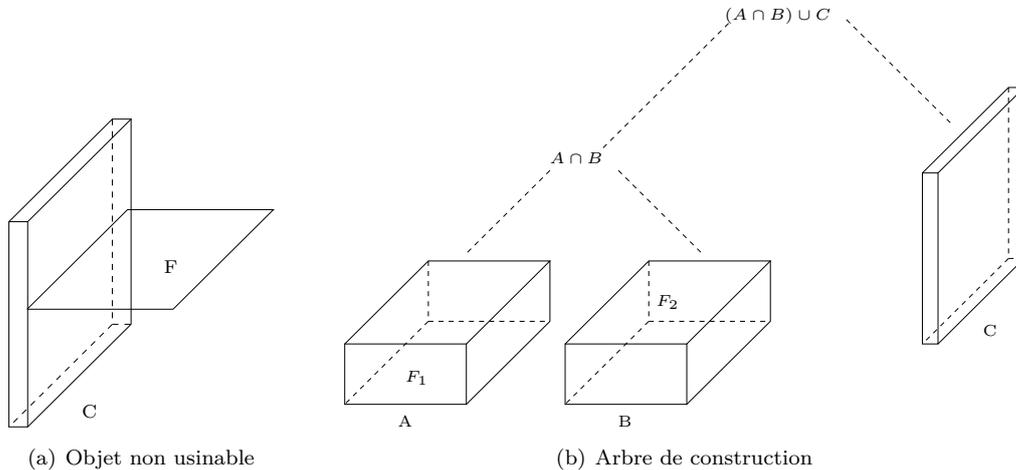


FIGURE 3.4 – Représentation CSG d'un objet non usinable

Il existe également des inconvénients tels que la possibilité de représenter un même objet par des arbres différents (mais avec le même ensemble de primitives). De plus, l'application d'opérateurs ensemblistes à des solides élémentaires peut résulter en un objet impossible à usiner : par exemple, si on considère que les faces F_1 et F_2 des objets respectifs A et B de la figure 3.4(b) sont géométriquement localisées au même endroit, l'objet de la figure 3.4(a), résultant de l'arbre de construction (3.4(b)) est non usinable. Enfin, les représentations par arbre CSG ne permettent pas d'avoir une définition explicite des objets. En effet, l'objet lui-même n'est pas représenté, seul l'agencement des solides élémentaires est donné. Il en résulte des calculs relativement lourds pour l'affichage des objets.

3.1.1.3 Modèles de représentation pas les bords

Les *modèles de représentation par les bords* (*Boundary Representation - B-rep - en anglais*) [?, ?] constituent une classe importante de la modélisation géométrique. L'idée générale de la représentation par les bords est de conserver sur l'objet à représenter l'information de la surface fermée qui le délimite ; celle-ci étant constituée d'un ensemble de faces, elles-mêmes définissables par leurs contours formés d'arêtes, puis de sommets. C'est dire que les représentations par les bords s'appuient sur les notions de décomposition des objets en cellules de dimensions différentes, les cellules de dimension i constituant les bords des cellules de dimension $i + 1$. Ainsi les sommets (dimension zéro) sont les bords des arêtes (dimension un) qui à leur tour sont les bords des faces (dimension deux), etc. Sur la figure 3.5 on peut voir en 3.5(a) un objet 3D dont uniquement deux faces sont visibles. La figure 3.5(b) montre les cinq (5) faces de l'objet, celles visibles étant représentées en traits pleins.

Les cellules obtenues par décomposition sont appelées *cellules topologiques de dimension i* . Selon que

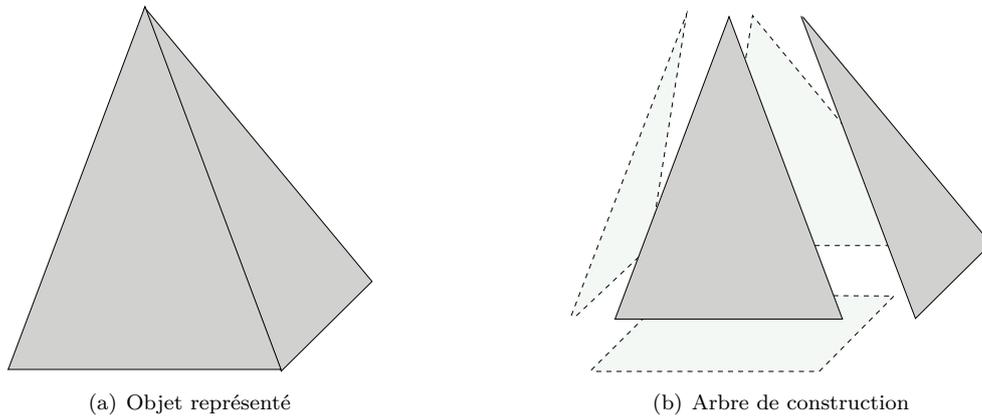


FIGURE 3.5 – Un objet représenté par les bords

ces cellules sont de formes quelconques ou non, on a des *modèles cellulaires généraux* ou des *modèles simpliciaux*.

Un des principaux avantages des représentations par les bords est le fait que l'objet à représenter l'est explicitement. De plus, un objet donné a une unique représentation possible.

3.1.2 Modèles géométriques à base topologique

La modélisation géométrique à base topologique s'appuie sur la décomposition des objets à représenter en cellules de différentes dimensions, reliées entre elles par plusieurs types de relation de voisinage :

- les *relations d'incidence* qui impliquent deux cellules de dimensions différentes ;
- les *relations d'adjacence* qui concernent deux cellules de même dimension, qui sont incidentes à une même cellule de dimension directement inférieure ;
- les *relations d'inclusion* qui concernent aussi deux cellules de même dimension, non contiguës, qui sont telles que l'une est incluse dans l'autre.

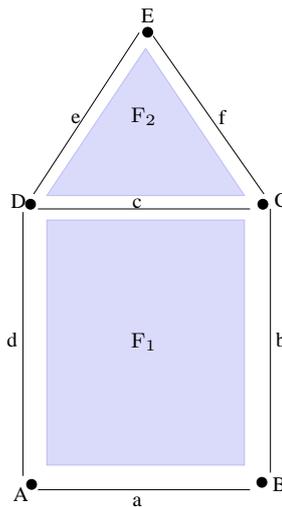


FIGURE 3.6 – Décomposition d'un objet en cellules

La vue éclatée de la figure 3.6 permet d'expliciter la structure topologique de l'objet 2D. On distingue d'une part les cellules topologiques : les sommets (A, B, \dots, E), les arêtes (a, b, \dots, f) et les faces (F_1 et F_2). En termes de relations de voisinage, l'arête a est incidente à la face F_1 , car c 'est une arête du bord de la face et les deux faces F_1 et F_2 sont adjacentes, car elles sont contiguës étant toutes deux incidentes à l'arête c .

Dans le domaine de la simulation biochimique, il est évident que la topologie et donc les relations de voisinage déterminent les échanges moléculaires entre compartiments. Nous allons donc dans la suite de ce chapitre, nous placer dans le cadre de la modélisation à base topologique. Le paragraphe 3.1.3 va ainsi présenter quelques modèles topologiques, en l'occurrence les graphes d'incidence, les graphes d'adjacence et les arbres d'inclusion.

3.1.3 Graphes d'incidence, d'adjacence et arbres d'inclusion

3.1.3.1 Graphe d'incidence

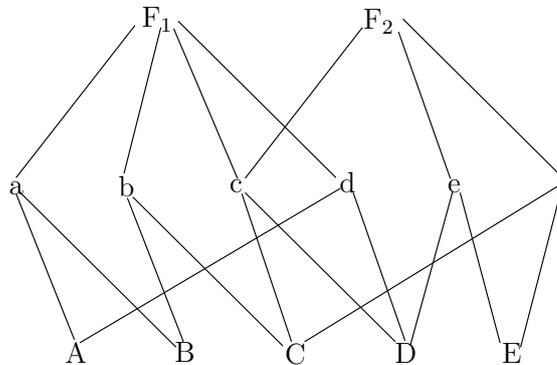


FIGURE 3.7 – Graphe d'incidence de l'objet figure 3.6

Les graphes d'incidence associent un nœud à chaque cellule topologique, avec une arête reliant chaque paire de cellules topologiques incidentes dont les dimensions se suivent. Les autres relations d'incidence se déduisent par simple transitivité. Ainsi dans le graphe d'incidence de la figure 3.7 qui est celui de la structure topologique de la figure 3.6, on peut lire que la face F_1 est incidente à l'arête a qui est incidente au sommet A . Par transitivité donc, F_1 est incidente à A .

Notons que les graphes d'incidence sont ambigus. En particulier ils ne permettent pas toujours de représenter l'orientation relative des cellules topologiques. Le graphe d'incidence de la figure 3.7 permet de retrouver l'orientation relative des faces F_1 et F_2 de l'objet figure 3.6 (page 42), car les deux faces partagent une arête commune c . Par contre, lorsque les faces ne partagent qu'un sommet, comme c'est le cas dans la figure 3.8, l'orientation relative des deux faces des objets $Objet_1$ et $Objet_2$ ne peut être déduite du seul graphe d'incidence.

En outre, les graphes d'incidence ne représentent pas l'inclusion, comme le montre la figure 3.9. Dans le graphe d'incidence figure 3.9, la face C_3 appartient à une composante connexe disjointe des faces C_2 et C_1 . À la seule donnée du graphe d'incidence, il est impossible de déterminer si la face C_3 se trouve bien incluse dans la face C_1 , ou au contraire incluse dans la face C_2 ou à l'extérieur de ces deux faces.

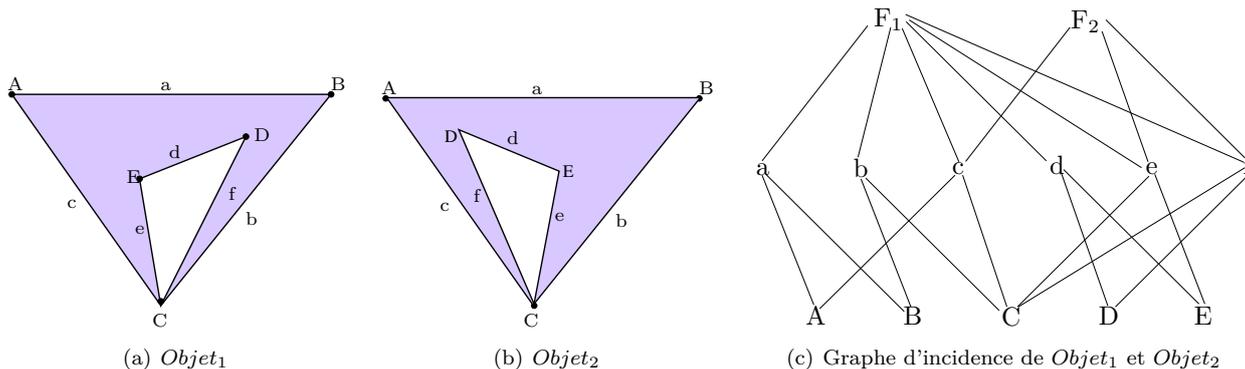


FIGURE 3.8 – Ambiguïté du graphe d'incidence

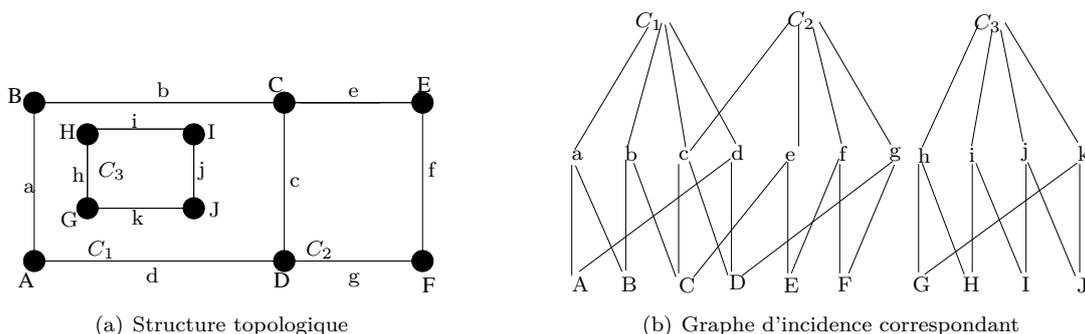


FIGURE 3.9 – Non prise en compte de l'inclusion par les graphes d'incidence

Notons que l'inclusion est une relation topologique cruciale dans les échanges biochimiques cellulaires. Il est donc essentiel pour nous de représenter la relation d'inclusion.

3.1.3.2 Graphe d'adjacence

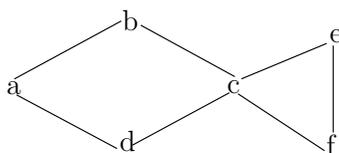


FIGURE 3.10 – Graphe d'adjacence pour les cellules de dimension 1 de l'objet figure 3.6 (page 42)

Les graphes d'adjacence permettent de représenter les relations d'adjacence existant entre les cellules topologiques. Ils sont définis pour une dimension donnée. Étant considérée une dimension, il est associé à chaque cellule topologique un nœud avec des arêtes reliant les paires de cellules adjacentes. La figure 3.10 représente le graphe d'adjacence associé à la structure topologique de la figure 3.6 (page 42) pour les arêtes.

Les graphes d'adjacence sont surtout utilisés en traitement d'images dans les algorithmes de segmentation [?, ?, ?, ?, ?, ?]. On parle de *graphe d'adjacence de régions* (*Region Adjacency Graph en anglais*) où

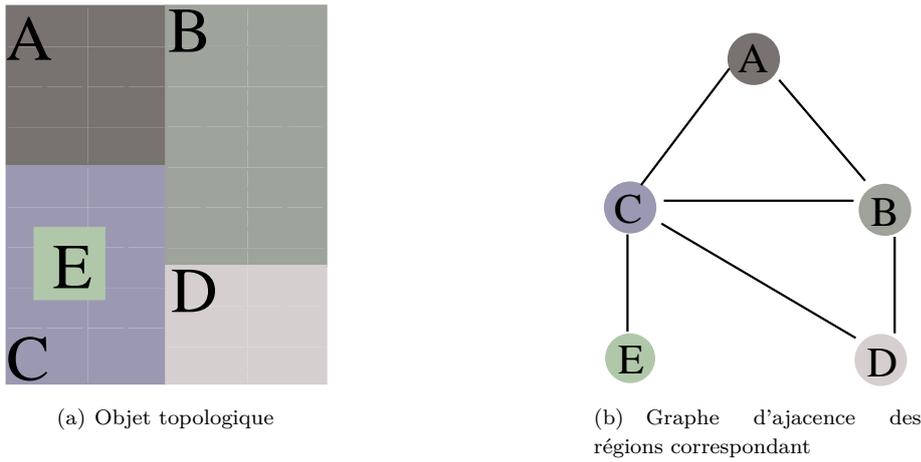


FIGURE 3.11 – Graphe d'Adjacence des régions

les régions correspondent à des cellules topologiques de dimensions 2 ou 3 selon la dimension des images. Les régions et leurs relations d'adjacence sont alors représentées sous la forme d'un graphe. La figure 3.11 montre un objet topologique (figure 3.11(a)) et le graphe d'adjacence des régions correspondant (figure 3.11(b)). Dans ce graphe les nœuds sont dans les mêmes couleurs que les régions qu'ils représentent. La fusion de deux régions connexes, motivée par le fait qu'elles sont de couleurs proches et/ou de faible taille, se fait par la suppression de l'arête matérialisant l'adjacence, dans le graphe. On peut voir sur cette figure que les graphes d'adjacence représentent de la même manière les relations d'inclusion (comme entre les régions C et E) et celle d'accolement (comme entre les régions C et D par exemple).

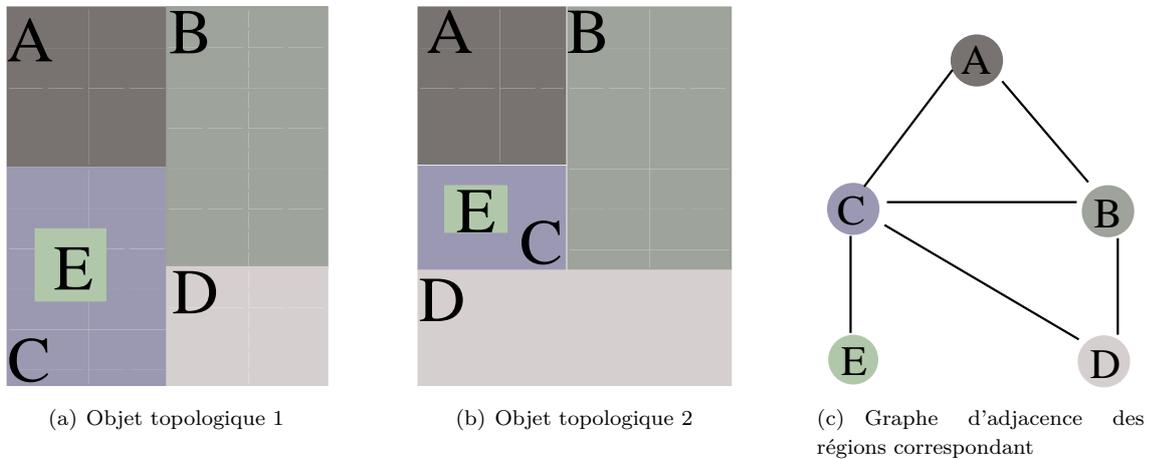


FIGURE 3.12 – Ambiguïté des graphes d'Adjacence des régions

Tout comme les graphes d'incidence, les graphes d'adjacence sont ambigus. En effet, un seul graphe d'adjacence peut dénoter deux objets différents. Cette ambiguïté est illustrée dans la figure 3.12 où les deux objets topologiques (figures 3.12(a) et 3.12(b)) ont le même graphe d'adjacence (3.12(c)).

Dans le cas de la simulation biochimique, nous pouvons utiliser les graphes d'adjacence pour représenter les compartiments contigus qui pourront échanger des molécules via les membranes qui les séparent. En effet, les graphes d'échanges dont nous avons besoin (section 3.2) ne sont rien d'autre que des graphes

d'adjacence de compartiments. Par contre, en raison de leur ambiguïté, ils ne sont pas un modèle adapté à la modélisation de compartimentations cellulaires.

3.1.3.3 Arbre d'inclusion

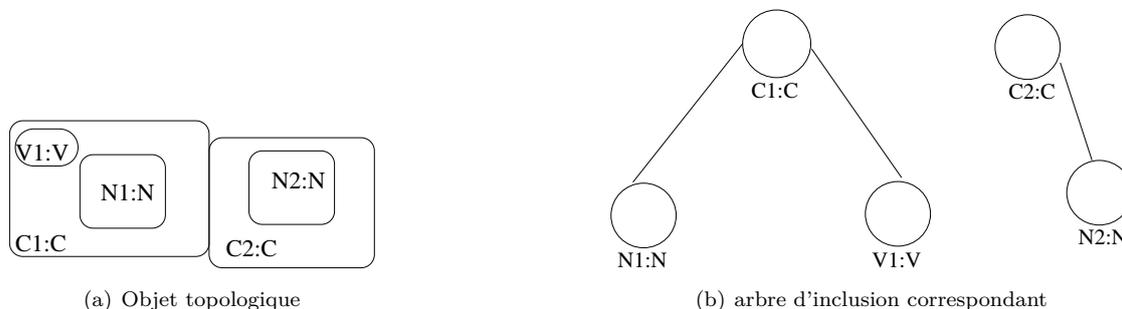


FIGURE 3.13 – Arbre d'inclusion

Les arbres² d'inclusion sont des graphes de hiérarchie entre les différentes composantes d'une structure topologique. Les relations matérialisées par des arcs entre les nœuds, sont des relations d'inclusion. C'est dire que l'arbre d'inclusion décrit la structure topologique représentée à travers sa décomposition en sous-structures topologiques. Les arbres d'inclusion sont définissables pour chaque dimension supérieure à 0. La figure 3.13(b) donne l'arbre d'inclusion correspondant à la compartimentation cellulaire 3.13(a) (qui est celui de l'exemple illustratif de notre approche (figure 2.2(a), page 28). On peut constater que toutes les liaisons traduisent des inclusions de compartiments : $N1$ inclus dans $C1$, $V1$ inclus dans $C1$, $N2$ inclus dans $C2$.

Les arbres d'inclusion ne représentent pas les voisinages par accollement. On remarquera à cet effet que la différence entre l'arbre d'inclusion (figure 3.13(b)) et le graphe d'échanges (figure 2.2(b), page 28) est l'adjacence par accollement reliant les compartiments $C1$ et $C2$. Notons que les arbres d'inclusion sont couramment utilisés en complément d'une autre structure topologique qui ne représente pas les inclusions telles que les graphes d'incidence (sous-section 3.1.3.1) ou des structure topologiques plus complètes telles que les *cartes généralisées* qui feront l'objet de la section suivante.

3.1.4 Les cartes généralisées

Les cartes généralisées [?] font partie de la grande famille des modèles topologiques utilisés pour représenter des objets dont les cellules topologiques sont quelconques, par opposition aux modèles réguliers dans lesquels toutes les faces sont triangulaires ou quadrilatères par exemple. Ce modèle permet de représenter des objets en toute dimension, avec ou sans bord, orientables ou non. En outre il a l'avantage d'avoir une structure très régulière et de représenter de la même manière les relations de voisinage dans toutes les dimensions, ce qui n'est pas le cas dans d'autres modèles comme celui des cartes combinatoires [?, ?] ou des arêtes ailées [?] par exemple. De plus ce modèle est implanté dans le noyau du modèleur open source MOKA [?] que nous présentons plus loin dans le paragraphe 3.4.1.

Le principe de représentation des objets par des cartes généralisées est illustré sur l'objet 3D de la figure 3.14 par les figures 3.15 et 3.16 représentant respectivement les phases de décomposition de l'objet

2. le terme "arbre" est utilisé par abus de langage : on a bien des arborescences mais pouvant partir de plusieurs racines

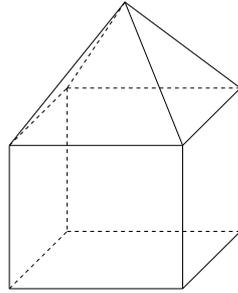


FIGURE 3.14 – Objet 3D à représenter par une 3-G-Carte

initial en cellules topologiques de dimensions successives, et d'établissement des relations d'adjacence entre les brins obtenus pour reconstituer l'objet d'origine.

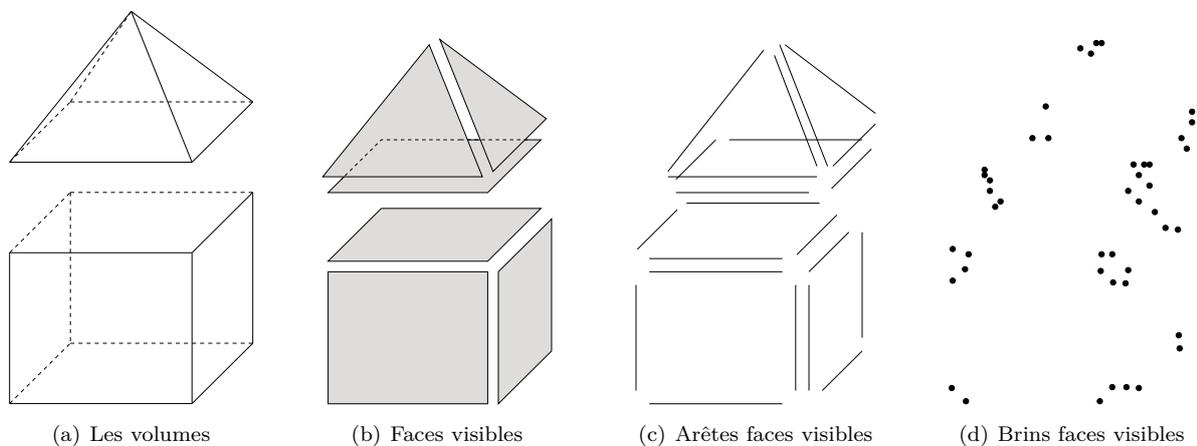


FIGURE 3.15 – Décomposition de l'objet 3D (figure 3.14) en brins

Étant donné que nous avons affaire à un objet 3D, la phase de décomposition se déroule en quatre étapes. Elle consiste en une décomposition de l'objet à représenter en ses composantes de dimension de plus en plus petite jusqu'à l'obtention des *brins*, éléments abstraits qui sont les briques de base des cartes généralisées et qui résultent de la décomposition des arêtes. Ils peuvent de ce fait être pris comme des sommets vus d'une arête, elle-même vue d'une face, vue à son tour d'un volume. Les sous-composantes obtenues par la décomposition de notre objet initial sont dans l'ordre les suivantes :

- deux sous-objets (figure 3.15(a)) qui correspondent aux différents volumes constituant l'objet ;
- ces sous-objets obtenus en figure 3.15(a) sont à leur tour décomposés en leurs faces dont les six (sur onze) visibles sont données par la figure 3.15(b) ;
- une décomposition de ces six faces nous donne en figure 3.15(c) leurs arêtes ;
- enfin, la décomposition des arêtes donne les brins (figure 3.15(d)).

Une fois les brins obtenus, l'étape suivante de la représentation consiste en l'établissement de relations d'adjacence entre les brins. Ces relations sont désignées par $\alpha_0, \dots, \alpha_3$, et chaque α_i permet de lier les brins de deux sous-objets de dimension i pour former des objets de dimension $i + 1$. (les brins sont pris comme étant de dimension 0). On obtient ainsi :

- en figure 3.16(a), la liaison des brins par des relations α_0 qui permet d'obtenir les sous-composantes de dimension 1 ;
- la figure 3.16(b) montre les liaisons α_1 (et α_0) qui permettent d'obtenir les cellules de dimension 2 ;

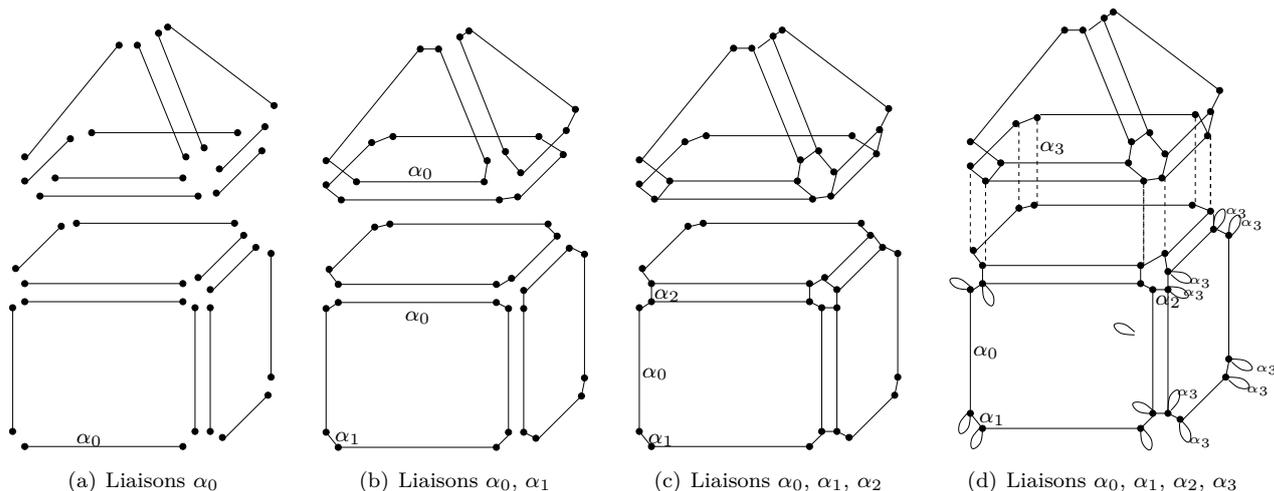


FIGURE 3.16 – Etablissement des liaisons α_0 à α_3 entre les brins de la figure 3.15(d)

- en figure 3.16(c), les liaisons α_2 vont relier ces cellules (obtenues en 3.16(b)) entre elles pour donner les cellules de dimension 3 ;
- enfin, dans la figure 3.16(d), les différentes cellules de dimension 3 sont reliées les unes aux autres par les liaisons α_3 . On obtient ainsi la G-carte de l'objet initial.

On remarque dans la figure 3.16(c) que certains brins n'ont pas de voisin par α_2 . Cela est dû au fait que les α_2 de ces brins sont sur les faces non visibles donc non figurées dans le schéma. Par contre, dans la figure 3.16(d), il y a des brins qui n'ont pas de voisins par α_3 . Dans ce cas, par convention dans les G-Cartes, le brin est son propre voisin par α_3 d'où les boucles figurées sur certains brins (pour des raisons de lisibilité, nous n'avons pas représenté toutes les boucles sur la figure).

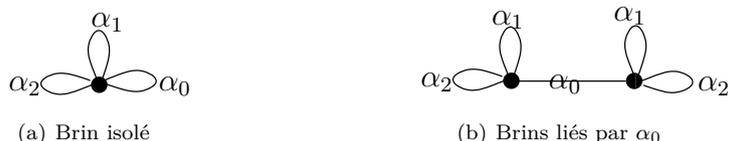


FIGURE 3.17 – Représentation des voisinages de brins

De manière générale, on constate que dans une G-carte de dimension n , ou n -G-carte, on a $n + 1$ types de relations d'adjacence (α_0 à α_n) et tous les brins doivent avoir un unique voisin par chaque α_i . Lorsque qu'un brin n'a pas de voisin par un α_i donné, il est convenu de considérer qu'il est son propre voisin par cet α_i . Ainsi, dans une 2-G-carte par exemple, un brin n'ayant aucun voisin sera représenté par la figure 3.17(a) et une arête isolée sera représentées par deux brins reliés par α_0 et n'ayant pas d'autre voisin comme dans la figure 3.17(b) et ainsi de suite.

Les n -G-cartes sont mathématiquement définies de la manière suivante :

Définition 3.1. (*n*-G-carte).

Une involution est une application $f : E \rightarrow E$ dont la composition par elle-même correspond à l'identité : $f \circ f(a) = a$, pour tout $a \in E$.

Une carte généralisée de dimension n , ou n -G-carte, est une algèbre $G = (B, \alpha_0, \dots, \alpha_n)$ où :

- B est un ensemble fini de brins,
- pour tout $i, 0 \leq i \leq n$, α_i est une involution sur B ,
- pour tout i et j , si $0 \leq i < i + 2 \leq j \leq n$, alors $\alpha_i \circ \alpha_j$ est une involution sur B .

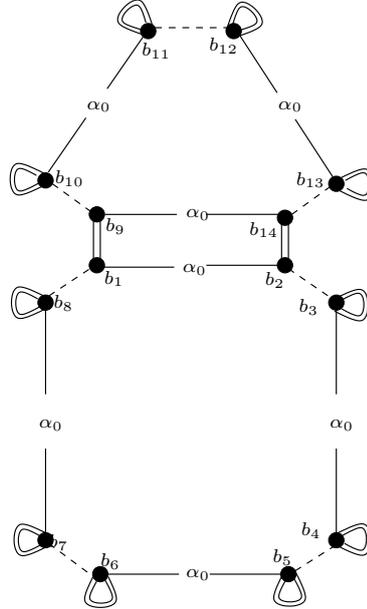


FIGURE 3.18 – Modélisation 2-G-Carte de l’objet de la figure 3.6 (page 42)

La figure 3.18 donne la représentation sous la forme d’une 2-G-carte de l’objet 2D introduit figure 3.6 (page 42). Les traits en pointillés représentent les α_1 tandis que les α_2 sont représentés par des traits doubles. On peut vérifier sur la figure que tous les points de la définition sont respectés :

- l’ensemble des brins (b_1, \dots, b_{14}) est fini ;
- les liaisons $\alpha_0, \dots, \alpha_3$ sont toutes des involutions, représentées graphiquement par des liaisons non orientées ; par exemple $\alpha_1(b_1) = b_8$ et $\alpha_1(b_8) = b_1$, de même $\alpha_2(b_8) = b_8$;
- comme nous sommes en 2D, le troisième point ne concerne que $\alpha_0 \circ \alpha_2$, il permet de garantir le bon collage des 2 faces le long de l’arête centrale, ainsi $\alpha_2(b_1) = b_9$, $\alpha_0(b_9) = b_{14}$, $\alpha_2(b_{14}) = b_2$ et $\alpha_0(b_2) = b_1$; de même ce point est vérifié au bord de l’objet, par exemple, $\alpha_2(b_8) = b_8$, $\alpha_0(b_8) = b_7$, $\alpha_2(b_7) = b_7$ et $\alpha_0(b_7) = b_8$.

Avant de revenir sur la définition 3.1, notamment le dernier point, nous donnons la définition des notions de i -Cellule et plus généralement d’orbite, qui permettent de retrouver les cellules topologiques d’une G-carte. En effet, bien que les G-cartes soient un modèle topologique, elles ne s’appuient pas sur une représentation explicite des cellules topologiques (sommet, arête, faces, etc.).

Définition 3.2. (notion d’orbite).

Soient $G = (B, \alpha_0, \dots, \alpha_n)$ une n -G-carte, $b \in B$ un brin, et β un ensemble de liaisons α_i , ($0 \leq i \leq n$).

On appelle orbite β de b ou orbite β incidente à b , notée $\langle \beta \rangle (b)$, l'ensemble des brins de B accessibles à partir de b en utilisant uniquement les involutions α_i de β . I.e, $\langle \beta \rangle (b)$ est le plus petit ensemble inclus dans B tel que :

- b est un brin de $\langle \beta \rangle (b)$,
- pour tout brin d de $\langle \beta \rangle (b)$ et toute involution α_i de β , $\alpha_i(d)$ est un brin de $\langle \beta \rangle (b)$.

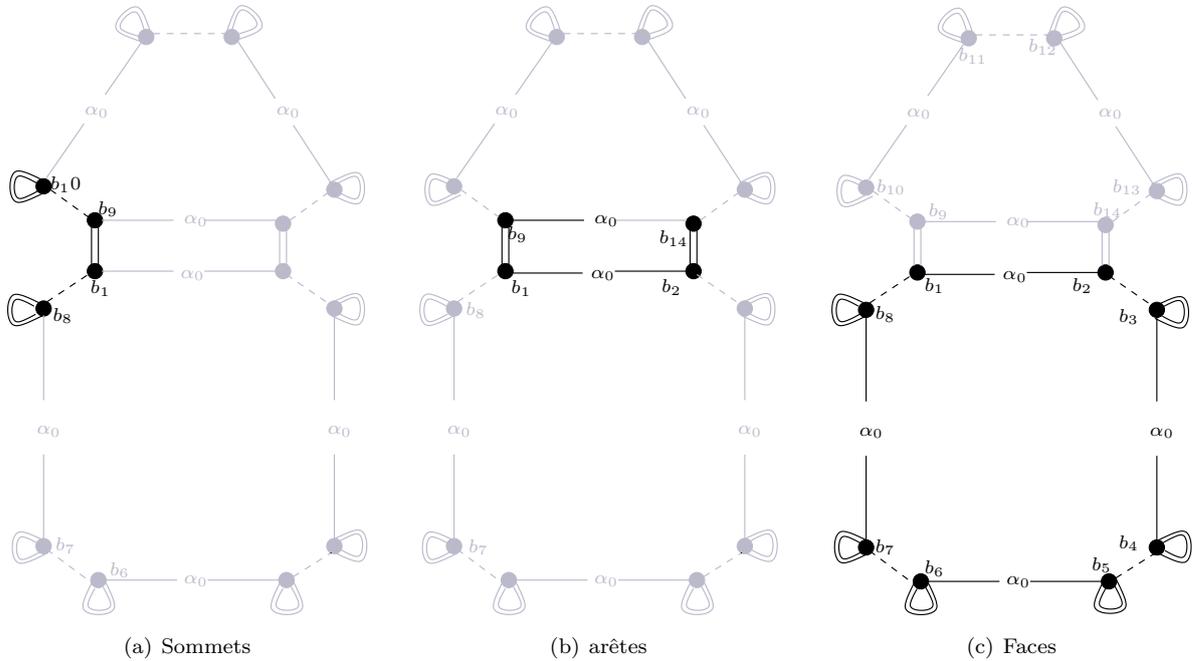


FIGURE 3.19 – Les différentes cellules topologiques dans une 2-G-Carte

Considérons la G-carte figure 3.18. Les cellules des différentes dimensions peuvent être retrouvées à l'aide des orbites, comme le montre la figure 3.19 qui donne en noir les cellules associées à b_1 .

- le sommet (voir figure 3.19(a)) associé à un brin b est donné par l'orbite $\langle \alpha_1 \alpha_2 \rangle (b)$; ainsi en est-il par exemple du sommet incident à b_1 qui est constitué par les brins b_1, b_8, b_9 et b_{10} et du sommet incident à b_7 qui comporte les brins b_7 et b_6 ;
- l'arête (voir figure 3.19(b)) associée à un brin b est donnée par $\langle \alpha_0 \alpha_2 \rangle (b)$; ainsi l'arête incidente à b_7 est constituée par les brins b_7 et b_8 et l'arête associée à b_1 comprend les brins b_1, b_2, b_9 et b_{14} .
- la face (voir figure 3.19(c)) associée à un brin b est donnée par $\langle \alpha_0 \alpha_1 \rangle (b)$; ainsi la face incidente à b_{10} est constituée par les brins b_9 à b_{14} et la face associée à b_1 comprend les brins b_1 à b_8 .

Ainsi, chaque brin est impliqué dans exactement un sommet, une arête et une face.

En s'appuyant sur les commentaires de la figure 3.19, on peut dire que la cellule de dimension i ($i \in [0, 2]$), incidente à un brin b correspond à

$$\langle \alpha_0 \dots \alpha_{i-1} \alpha_{i+1} \dots \alpha_2 \rangle (b).$$

la généralisation de cette observation va nous donner la définition 3.3 qui est celle des cellules de dimension $(i, i \in [0, n])$ incidente à un brin b (encore appelées *i-cellule incidente à un brin*) dans une n-G-Carte,

Définition 3.3. (*i-cellule*).

Soient

- $G = (B, \alpha_0, \dots, \alpha_n)$ une n -G-carte;
- $b \in B$;
- $i \in [0, n]$;

on appelle i -cellule incidente à b l'orbite $\langle \alpha_0 \dots \alpha_{i-1} \alpha_{i+1} \dots \alpha_n \rangle$ de b

La contrainte exprimée par le deuxième point de la définition 3.1 qui n'est pertinente que pour les dimensions de modélisation supérieures à 1, peut s'exprimer en termes des exigences suivantes :

- lier par α_2 deux brins $b1$ et $b2$ impose que soient également liés par α_2 les deux brins $\langle \alpha_0 \rangle$ ($b1$) et $\langle \alpha_0 \rangle$ ($b2$);
- lier par α_3 deux brins $b1$ et $b2$ d'une même arête, nécessite que soient également liés par α_3 les brins $\langle \alpha_1 \rangle$ ($b1$) et $\langle \alpha_1 \rangle$ ($b2$) (valable pour les dimensions 3 et supérieures);
- etc.

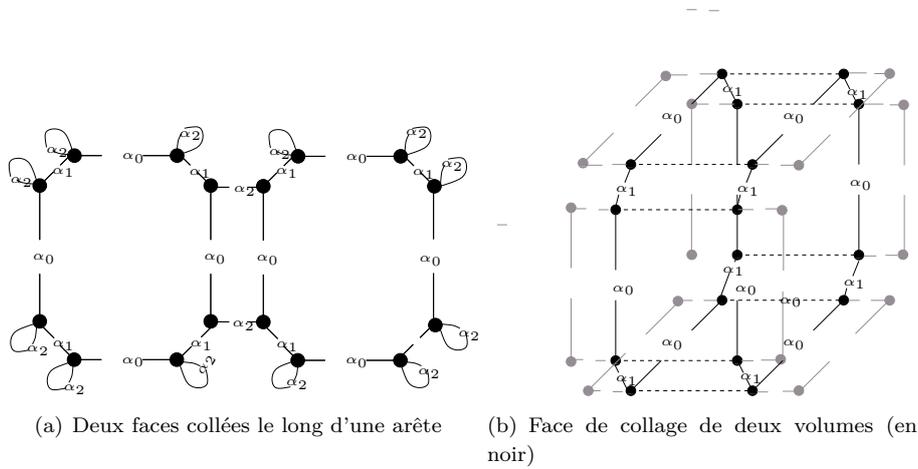


FIGURE 3.20 – Collage de i – cellules le long de $(i - 1)$ – Cellules

Intuitivement, en dimension 2 et 3, le deuxième point de la définition exprime le fait que le collage de deux faces se fait le long d'une arête et celui de deux volumes (dimension 3 uniquement) se fait le long d'une face. Ainsi, ce point précis de la définition impose que le collage d'une i – Cellule se fasse le long d'une $(i - 1)$ – Cellule. Pour illustrer cela, nous donnons à travers la figure 3.20 deux n -G-cartes respectivement une 2-G-carte (figure 3.20(a)) représentant un objet 2D consistant en deux faces collées le long d'une arête et une 3-G-carte (figure 3.20(b)) représentant la face de collage de deux objets 3D.

Sur la base de cette intuition, les modèles de la figure 3.21 qui sont tous en 2D ne sont pas des n -G-cartes. En effet aucun ne respecte le deuxième point de la définition 3.1 puisque pour tous, $\langle \alpha_0 \alpha_2 \rangle$ n'est pas une involution (on a pour les modèles 1 (figure 3.21(a)) et 2 (figure 3.21(b))

$$\langle \alpha_0 \alpha_2 \rangle \circ \langle \alpha_0 \alpha_2 \rangle (b1) = b3$$

et pour le modèle 3 (figure 3.21(c)) qui représente deux faces collées par un sommet, on a

$$\langle \alpha_0 \alpha_2 \rangle \circ \langle \alpha_0 \alpha_2 \rangle (b2) = b15$$

Pour conclure sur la modélisation géométrique, nous rappelons que notre objectif dans cette section est de proposer un modèle de représentation des compartimentations cellulaires, qui fournisse les informations

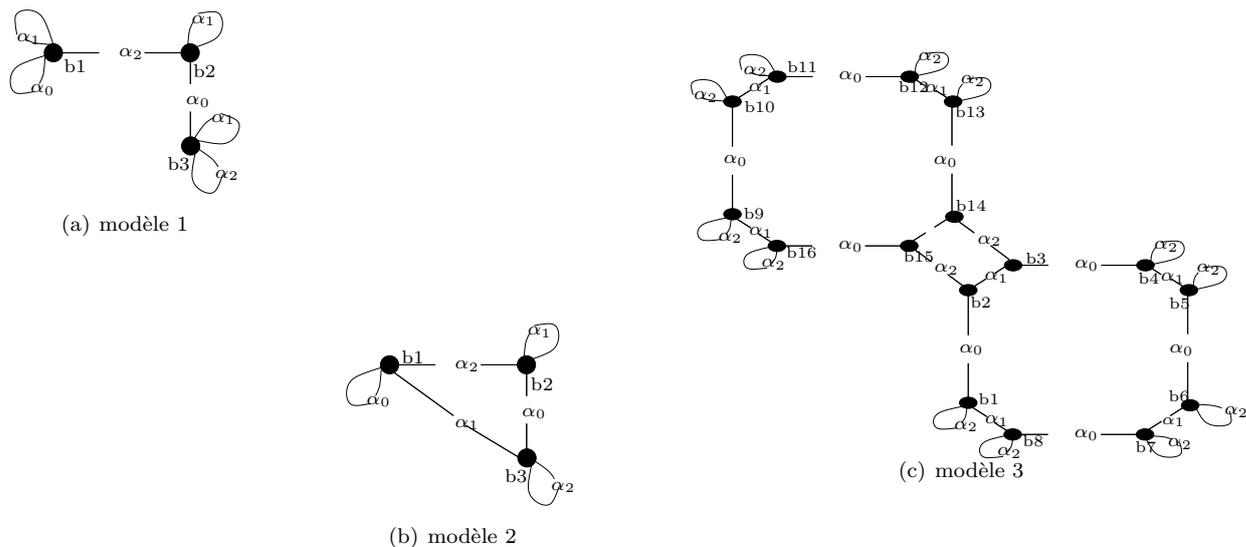


FIGURE 3.21 – Exemples de représentations qui ne sont pas des n-G-Cartes

de voisinage et d'inclusion nécessaires pour l'écriture de règles de réaction cohérentes avec la compartimentation cellulaire du système sous étude. La structure topologique d'une cellule étant déterminante pour le comportement de celle-ci, nous avons besoin de pouvoir représenter les relations d'adjacence et les relations d'inclusion. Or, des modèles topologiques que nous avons étudiés, aucun ne permet de représenter simultanément ces deux types de relations. Aussi, nous allons associer un arbre d'inclusion (la seule structure à représenter uniquement les inclusions) aux n-G-cartes. Le choix des n-G-cartes par rapport aux graphes d'adjacence, est dicté par la possibilité qu'offrent les premières, en plus du fait qu'elles représentent sans ambiguïté toutes les relations d'adjacence, de construire et de visualiser des modèles cohérents.

3.2 Abstraction de la compartimentation cellulaire : le graphe d'échanges

Comme nous l'avons vu en section 2.3, une fois le modèle de compartimentation cellulaire construit, nous allons en extraire le graphe d'échanges qui sera combiné avec les règles génériques pour construire les règles dédiées à la compartimentation cellulaire de départ. Nous venons de retenir, section 3.1, les cartes généralisées associées à un arbre d'inclusions, pour la représentation des compartimentations cellulaires. Cependant, une G-carte contient beaucoup d'informations propres à la modélisation géométrique des compartiments, qui sont inutiles pour la spécialisation des règles. Aussi, les informations nécessaires, c'est-à-dire principalement le graphe d'adjacence des compartiments, doit être extrait avant de pouvoir spécialiser les règles.

Dans cette section, nous rappelons dans un premier paragraphe, quelques éléments de la théorie des graphes, avant d'introduire un type de graphe d'adjacence particulier, le *graphe d'échanges*.

3.2.1 Rappels sur les graphes

Les graphes sont des structures abstraites appropriées pour représenter un ensemble d'entités et les voisinages qui existent entre elles. Les applications de graphes sont diversifiées couvrant classiquement les domaines liés à la notion de réseau (réseaux de télécommunication, réseaux routiers, réseaux de régulation de gènes et autres réseaux biologiques ...)

Intuitivement, un graphe peut être vu comme un ensemble de points appelés *sommets*, certains ayant des liens directs appelés *arcs* entre eux. Mathématiquement, un graphe peut être défini de la manière suivante :

Définition 3.4. (*Grappe*).

Un graphe $G = (V, E)$ est défini par :

- Un ensemble fini de sommets : $V = \{v_1, \dots, v_n\}$.
- Un ensemble fini d'arcs $E \subseteq V \times V$. Pour un arc $e = (v, v')$, le sommet v est appelé source de e et noté $source(e)$ et le sommet v' est appelé cible de e et est noté $target(e)$.

Remarquons que dans la définition précédente, les arcs sont nommés par les couples de noms des sommets qu'ils relient. Il n'est donc pas possible de représenter plusieurs arcs entre deux sommets donnés. Ceci est conforme aux graphes d'adjacence, pour lesquels les arcs représentent la relation de voisinage. En effet, deux compartiments sont voisins ou non, mais il n'y a pas de voisinage multiple.

Cette définition s'intéresse à la structure du graphe sans autre précision. Cependant, les sommets et/ou les arcs d'un graphe peuvent être assortis d'étiquettes prises dans des ensembles d'étiquettes de sommets et/ou d'arcs. Ces étiquettes précisent certaines caractéristiques des sommets ou des arcs. Lorsque certains sommets et/ou certains arcs d'un graphe sont étiquetés, on parle de *graphe partiellement étiqueté* et la définition 3.4 peut alors s'étendre comme suit pour prendre en compte cet/ces étiquetage(s).

Définition 3.5. (*Grappe partiellement étiqueté*).

Soient Σ_V et Σ_E deux ensembles non vides d'étiquettes de sommets et d'étiquettes d'arcs, ne contenant pas l'étiquette $\{\perp\}$.

Un graphe partiellement étiqueté sur Σ_V et Σ_E est un graphe $G = (V, E, l_V, l_E)$ tel que :

- (V, E) est un graphe
- $l_V : V \rightarrow \Sigma_V$ est une application qui associe à chaque sommet v de V une étiquette dans $\Sigma_V \cup \{\perp\}$. Si $l_V(v) = \perp$, on dit que le sommet v est non étiqueté sinon on dit que v est étiqueté par $l_V(v)$.
- $l_E : E \rightarrow \Sigma_E$ est une application qui associe à chaque arc $e = (v, v')$ de E une étiquette dans $\Sigma_E \cup \{\perp\}$. Si $l_E(e) = \perp$, on dit que l'arc e est non étiqueté, sinon on dit que e est étiqueté par $l_E(e)$.

Lorsque tous les sommets et tous les arcs sont étiquetés, on dit que le graphe est totalement étiqueté.

Dans la suite du document, nous convenons de représenter les graphes avec les éléments suivants :

soit $G = (V, E, l_V, l_E)$ un graphe partiellement étiqueté sur Σ_V et Σ_E :

- les sommets sont représentés par des cercles avec en dessous le nom du sommet et à l'intérieur l'éventuelle étiquette ; les sommets non étiquetés sont représentés par des cercles vides ;

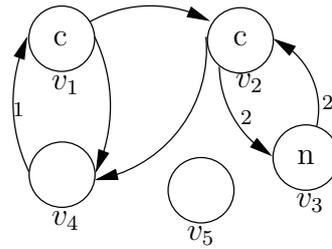


FIGURE 3.22 – Graphe G_1 : exemple de graphe partiellement étiqueté

– les arcs sont des traits entre deux sommets, assortis de flèches orientées vers la cible ; les étiquettes d’arc sont mentionnées à côté des traits.

Ainsi la figure 3.22 représente le graphe $G_1 = (V, E, l_V, l_E)$ partiellement étiqueté sur $\Sigma_V = \{a, \dots, z\}$ et $\Sigma_E = \{1, \dots, 9\}$, tel que :

$$V = \{v_1, v_2, v_3, v_4, v_5\}$$

$$E = \{(v_1, v_2), (v_1, v_4), (v_2, v_3), (v_2, v_4), (v_3, v_2), (v_4, v_1)\}$$

$$l_V(v_1) = c, l_V(v_2) = c, l_V(v_3) = n, l_V(v_4) = \perp, l_V(v_5) = \perp$$

$$l_E((v_1, v_2)) = \perp, l_E((v_1, v_4)) = \perp, l_E((v_2, v_3)) = 2, l_E((v_2, v_4)) = \perp, l_E((v_3, v_2)) = 2, l_E((v_4, v_1)) = 1$$

On constate dans G_1 (figure 3.22) que les sommets v_1 et v_2 , étiquetés par c sont représentés par des cercles contenant la lettre c . De même, l’arc (v_4, v_1) , étiqueté par 1 est représenté par un trait assorti d’une flèche orientée vers v_1 avec à côté le chiffre 1. Aucun arc ne part ni n’aboutit à v_5 . v_4 et v_5 qui sont non étiquetés sont représentés par des cercles identifiés par leurs noms respectifs.

Définition 3.6. (*Chemin, graphe connexe, distance entre deux sommets*).

Un chemin c dans graphe $G = (V, E, l_V, l_E)$ est la donnée d’une séquence d’arcs consécutifs $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)$. On appelle v_1 (resp. v_n) "source" (resp. "cible") de c . La longueur de c est le nombre d’arcs qu’il compte.

G est dit connexe si pour tout couple de sommets (v, v') , il existe un chemin ayant pour source v et pour cible v' .

La distance entre deux sommets notée $d(v, v')$ est la plus petite longueur de chemin entre ces deux sommets. Ainsi pour $v \in V$ et $v' \in V$, on dira qu’ils sont :

- voisins directs si $d(v, v') = 1$ ($(v, v') \in E$);
- voisins indirects si $d(v, v') > 1$.

Par convention, lorsqu’il n’existe aucun chemin entre deux sommets v et v' , $d(v, v') = +\infty$.

Cette définition 3.6 nous permet de dire pour le graphe G_1 (figure 3.22) que la séquence d’arcs

$$(v_1, v_2), (v_2, v_3)$$

est un chemin de longueur 2 ayant pour source v_1 et pour cible v_3 . Par ailleurs on a

$$d(v_1, v_4) = 1$$

d'où v_1 et v_4 sont des voisins directs tandis que v_3 et v_4 sont voisins indirects car $d(v_3, v_4) = 2$. Pour finir, on peut dire que G_1 n'est pas connexe, car il n'existe aucun chemin ayant pour source ou pour cible v_5 .

Dans de nombreux problèmes de la vie courante, il peut s'avérer intéressant d'étudier les parties d'un graphe. Par exemple dans un réseau routier structuré sous forme de graphe, on peut se demander s'il est possible d'aller du point a au point b sans passer par le point c . Cet intérêt pour la caractérisation de parties de graphe a conduit à la définition de notions telles que *graphe partiel*, *graphe généré par un sommet*, etc.

Nous définissons dans ce qui suit la notion de *sous-graphe engendré par un sous-ensemble des sommets d'un graphe*, dont nous aurons besoin dans la suite de notre travail.

Définition 3.7. (*Sous-graphe engendré par un ensemble de sommets*).

Soient $G = (V, E, l_V, l_E)$ un graphe étiqueté sur Σ_V et Σ_E deux ensembles d'étiquettes respectivement de sommets et d'arcs et $V' \subseteq V$ un sous-ensemble des sommets de G .

Le sous-graphe de G engendré par V' est le graphe noté $G|_{V'} = (V', E', l_{V'}, l_{E'})$ tel que :

- $E' = \{(v, v') \mid (v, v') \in E, v \in V', v' \in V'\}$;
- $l_{V'}$ (également noté $l_{V|_{V'}}$) : $V' \rightarrow \Sigma_V$ est la restriction de l_V à V' ($\forall v \in V', l_{V'}(v) = l_V(v)$);
- $l_{E'}$ (également noté $l_{E|_{E'}}$) : $E' \rightarrow \Sigma_E$ est la restriction de l_E à E' ($\forall e \in E', l_{E'}(e) = l_E(e)$).

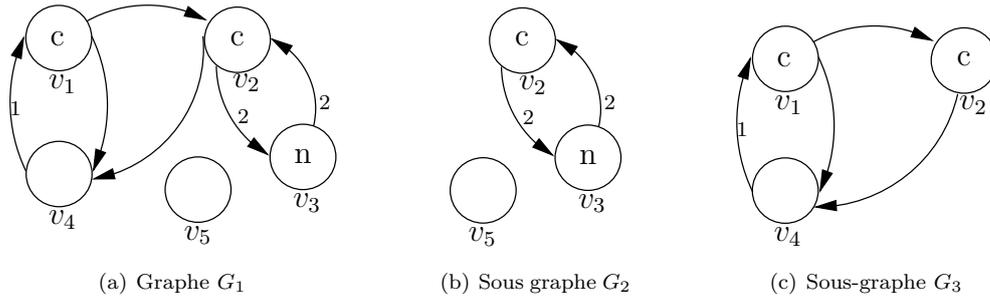


FIGURE 3.23 – Deux sous graphes

La figure 3.23 montre G_2 et G_3 , deux sous-graphes de G_1 (repris de la figure 3.22) engendrés respectivement par $V_2 = \{v_2, v_3, v_5\}$ et $V_3 = \{v_1, v_2, v_4\}$.

Comme les relations de voisinage entre régions sont symétriques, nous utiliserons des *graphes non orientés* que l'on définit de la manière suivante :

Définition 3.8. (*Graphe non orienté*).

Un graphe $G = (V, E, l_V, l_E)$ est dit non orienté si tout arc a un arc symétrique, c'est-à-dire si

$$\forall e = (v, v') \in E, \exists e' = (v', v) \in E, l_E(e) = l_E(e').$$

Le couple d'arcs symétriques $((v, v'), (v', v))$ est appelé arête et se note $\{v, v'\}$.

La représentation graphique des graphes non orientés utilise des *arêtes* (arcs sans flèche) plutôt que des arcs classiques. Pour transformer un graphe non orienté représenté avec des arcs, en un graphe non orienté représenté avec des arêtes, on remplace chaque couple d'arcs symétriques $((v, v'), (v', v))$ ayant la

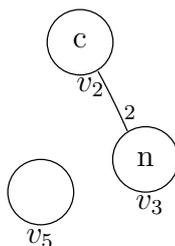


FIGURE 3.24 – Graphe non orienté correspondant au graphe G_2 (figure 3.23(b))

même étiquette $l_E((v, v'))$ par une unique arête $\{v, v'\}$ portant l'étiquette $l_E((v, v'))$. En procédant de la sorte, le graphe G_2 (figure 3.23(b)) est représenté par le graphe G_2 de la figure 3.24.

Notons que le graphe G_3 (figure 3.23(c)) n'est pas un graphe non orienté d'une part parce que les arcs (v_1, v_2) et (v_2, v_4) n'ont pas d'arcs symétriques et d'autre part, parce que $l_E((v_4, v_1)) \neq l_E((v_1, v_4))$.

3.2.2 Principales caractéristiques et définition du graphe d'échanges

Un *graphe d'échanges* est un graphe d'adjacence, qui rend compte des possibilités qu'ont les différents compartiments d'échanger des bio-molécules. En effet les échanges se font entre compartiments voisins soit par inclusion, soit par accollement. Ainsi, d'un point de vue structurel, chaque sommet du graphe doit dénoter un compartiment biologique et chaque arc traduit un voisinage entre les deux compartiments correspondants aux sommets impliqués.

Nous allons passer en revue les différentes caractéristiques des graphes d'échanges avant de les définir mathématiquement.

3.2.2.1 De l'orientation du graphe

Comme dit dans l'introduction, les bio-membranes sont à perméabilité sélective dans les deux sens et la vitesse de passage des molécules peut varier en fonction des molécules et du sens. Ce sont là autant d'éléments qui militent en faveur de l'orientation des *graphes d'échanges*, ce qui donnerait la possibilité à l'utilisateur de préciser par le biais d'étiquettes sur les sommets par exemple, les molécules agréées par tel ou tel autre compartiment, dans tel ou tel sens...

Cependant, les informations sur la perméabilité des membranes à telle molécule ou telle autre, dans un sens ou l'autre est par ailleurs définie dans les règles. Le but du graphe d'échanges n'est donc pas de modifier les règles de comportement données par l'utilisateur, mais seulement de les spécialiser pour une compartimentation cellulaire donnée. C'est pourquoi nous choisissons de représenter les voisinages de manière simple par des arêtes. Nos graphes d'échanges sont donc non orientés.

3.2.2.2 Des étiquetages nécessaires

Parmi les éléments caractéristiques d'un compartiment biologique (type, contenu, volume, surface de la membrane, etc.) le *type de compartiment* et le *contenu du compartiment en termes de molécules*, apparaissent comme ceux qui doivent nécessairement être pris en compte dans le graphe d'échanges.

Pour illustrer la nécessité de prendre en compte les types de compartiments, nous supposons une réaction bio-moléculaire qui a besoin qu'un compartiment de type t_1 et un autre de type t_2 soient voisins. Si le graphe d'échanges ne précise pas les types de compartiments, il ne permettra jamais de vérifier si une telle condition de voisinage est satisfaite. Il ne pourra donc pas rendre compte de la possibilité qu'une telle réaction se produise. Cependant, nous décidons d'introduire les types de compartiments non pas comme un étiquetage mais plutôt comme un typage de sommets. Cela nous permet de typer les compartimentations cellulaires et de faire le lien entre les règles génériques (définies sur les types de compartiments) et les règles intermédiaires et cibles. Ceci étant, ce choix au niveau des définitions mathématiques, est indépendant de celui que nous ferons au niveau de l'implantation. En effet, au niveau logiciel, les types seront attachés aux sommets et représentées comme les étiquettes.

La considération en ce qui concerne les contenus des compartiments est toute autre : en effet, s'il est vrai que les interactions bio-moléculaires dépendent essentiellement des types de compartiments en présence et de leurs contenus respectifs, l'importance de l'aspect contenu peut être relativisée (voire ignorée) dans notre procédure de couplage règles génériques/graphes d'échanges. Cependant, les modèles cibles que nous obtenons sont destinés à être simulés à partir d'un état initial, et nous pensons qu'il est intéressant (nécessaire) que celui-ci puisse être défini dans le graphe d'échanges, qui est, des deux éléments en entrée de la procédure de couplage, celui qui « connaît » les vrais compartiments et qui peut donc en initialiser le contenu. Aussi, proposons-nous d'introduire les contenus des compartiments comme un étiquetage des sommets. Cependant, cet étiquetage des sommets par le contenu des compartiments doit être vu comme une possibilité supplémentaire donnée à l'utilisateur pour définir un modèle BIOCHAM ou PATHWAY LOGIC directement exécutable. Comme nous allons le voir, elles n'interviennent pas dans la traduction des règles.

Les étiquettes d'arcs quand à elles ne sont pas utiles. En effet, comme nous l'avons vu, il n'y a pas lieu de qualifier les relations de voisinage.

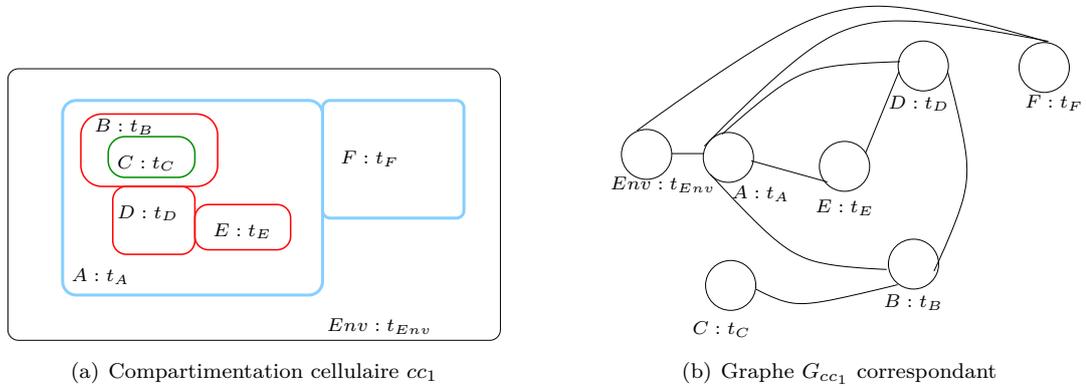


FIGURE 3.25 – Compartimentation cellulaire et le graphe d'échanges l'abstrayant

Pour illustrer les caractéristiques des graphes d'échanges, nous prenons l'exemple, figure 3.25(b), d'une compartimentation cellulaire cc_1 et du graphe d'échanges correspondant G_{cc_1} . Les compartiments sont nommés de A à F et typés de t_A à t_F . Dans cet exemple, nous ne précisons pas les conditions initiales des compartiments. Les sommets du graphe d'échanges ne portent donc aucune étiquette. Remarquons que la compartimentation cellulaire cc_1 et son graphe d'échanges G_{cc_1} contiennent un compartiment particulier pré-défini Env de type par défaut t_{Env} , qui représente l'environnement extérieur de la compartimentation cellulaire considérée.

3.2.2.3 Définition mathématique du graphe d'échanges

Une fois fixées les différentes caractéristiques du *graphe d'échanges*, nous pouvons le définir de la manière suivante.

Définition 3.9. (*Grappe d'échanges*).

Soit T_c un ensemble de types de compartiments contenant le type pré-définie t_{Env} et Σ_V un ensemble d'étiquettes de sommets.

Un graphe d'échanges $GE = (V, E, l_V)$ défini sur T_c et Σ_V est un graphe non orienté, étiqueté aux sommets tel que :

les sommets de V sont typés sur T_c , c'est-à-dire qu'à tout sommet $v \in V$ il est associé un type $t \in T_c$; on dit que v est de type t et on note $v : t$.

3.2.3 Opérations applicables dans un graphe d'échanges

Nous allons distinguer deux grandes catégories d'opérations applicables dans les graphes d'échanges :

- les opérations de création/définition du graphe d'échanges qui modifient le graphe. Elles sont utilisables lors de la définition du graphe et consistent en l'ajout, la suppression d'un sommet ou d'une arête ;
- les opérations d'utilisation qui lisent le graphe pour en tirer des informations mais ne le modifient pas. Entrent dans cette catégorie, en ce qui concerne notre étude, toutes les opérations qui sont nécessaires pour la procédure de couplage règles génériques/graphe d'échanges. On peut citer entre autres la vérification d'un voisinage entre deux sommets, la recherche des voisins d'un sommet, le calcul de sous-graphes, la vérification de la connexité d'une partie de graphe, etc. Cette catégorie d'opérations ne sera pas détaillée dans ce document.

3.2.3.1 Création d'un graphe d'échanges vide

Avant toute chose, il s'agit de *créer le graphe d'échanges vide*, ne contenant donc ni sommet ni arête. Nous allons supposer tout au long de cette sous-section que le graphe créé est

$$GE = (V_{GE}, E_{GE}, l_{V_{GE}})$$

On aura $V_{GE} = \emptyset$ et par conséquent $E_{GE} = \emptyset$ et $l_{V_{GE}} = \emptyset$. Dans la suite de cette section, nous ne nous préoccupons plus de $l_{V_{GE}}$.

3.2.3.2 Création de l'environnement

La première *vraie* opération consiste à *créer l'environnement* : on a à ce niveau la possibilité de préciser le type de l'environnement $type_{env}$ qui est par défaut t_{Env} . On aura après l'opération

$$V_{GE} = \{Env : type_{Env}\}$$

avec E_{GE} inchangé.

Cette opération est utilisée une seule fois pour la définition d'un graphe et n'est applicable que lorsque celui-ci est vide. Une fois que Env est ajouté, les opérations suivantes deviennent applicables :

- ajout d'un sommet v ;
- suppression d'un sommet v ;
- ajout d'une arête $\{v, v'\}$;
- suppression d'une arête $\{v, v'\}$.

3.2.3.3 Ajout d'un sommet autre que Env

Pour l'ajout d'un sommet v au graphe, le sommet v' représentant le compartiment d'inclusion du compartiment représenté par v est donné et est présent dans le graphe. Par défaut, Env est le contenant considéré. Cette opération modifie V_{GE} et E_{GE} . On obtient

$$V_{GE} = V_{GE} \cup \{v\}$$

$$E_{GE} = E_{GE} \cup \{v, v'\}$$

3.2.3.4 Suppression d'un sommet

La suppression d'un sommet v entraîne celle de toutes les arêtes reliant v à un autre sommet. Cette opération modifie donc à la fois V_{GE} et E_{GE} . On obtient

$$V_{GE} = V_{GE} \setminus \{v\}$$

$$E_{GE} = E_{GE} \setminus (\coprod \{v, v'\}).$$

Il est à noter que la suppression de Env vide le graphe d'échanges.

3.2.3.5 Ajout et suppression d'une arête

L'opération d'ajout d'une arête $\{v, v'\}$ au graphe ainsi que celle de suppression d'une arête $\{v, v'\}$ modifient seulement E_{GE} . On a respectivement

$$E_{GE} = E_{GE} \cup \{v, v'\}$$

$$E_{GE} = E_{GE} \setminus \{v, v'\}$$

Notons que toutes ces opérations ne contrôlent pas la cohérence du graphe d'échanges obtenu par rapport à une compartimentation cellulaire. En effet, en considérant Env comme le compartiment le plus externe, on peut toujours retrouver (si le graphe est cohérent) la nature de la relation entre deux sommets (inclusion ou accolement). Ainsi, l'opération de suppression d'un sommet devrait en principe supprimer dans un premier temps tous les compartiments contenus dans celui représenté par le sommet à supprimer. De même l'ajout d'une arête devrait vérifier que les deux compartiments ont le même contenant. Nous n'intégrons pas ces contrôles parce que non seulement ils peuvent se révéler lourds lorsque'on a affaire à des compartimentations cellulaires et par ailleurs, nous offrons à l'utilisateur un moyen sûr d'avoir un graphe d'échanges cohérent avec la compartimentation qu'il souhaite étudier. Nous présentons section 3.3 notre modèleur de compartimentations biologiques et l'extraction du graphe d'échanges à partir des modèles construits.

3.3 Modélisation des compartiments cellulaires

Pour permettre aux utilisateurs de modéliser leurs compartiments cellulaires, nous avons développé un modèleur bio-géométrique présentant les caractéristiques suivantes :

- Il est fondé sur deux structures topologiques mutuellement liées pour assurer la construction de compartimentations cellulaires cohérentes et l'extraction du graphe d'échanges associé : il s'agit d'une part d'une carte généralisée de dimension 3 qui permet de représenter les compartiments et leurs relations de collage (adjacence) et d'autre part d'un arbre d'inclusions qui permet de matérialiser les inclusions entre les compartiments.
- Il permet d'associer des contenus en termes de molécules aux compartiments représentés, donnant ainsi la possibilité à l'utilisateur d'initialiser son modèle directement au niveau de la compartimentation cellulaire. Cet état initial sera traduit dans le langage de règles de l'outil cible via le graphe d'échanges.

Outre le contrôle de cohérence assuré par les opérations de construction, la visualisation des compartiments cellulaires en 3 dimensions est d'un intérêt appréciable pour l'utilisateur.

3.3.1 Le modèle bio-géométrique

3.3.1.1 Structure topologique

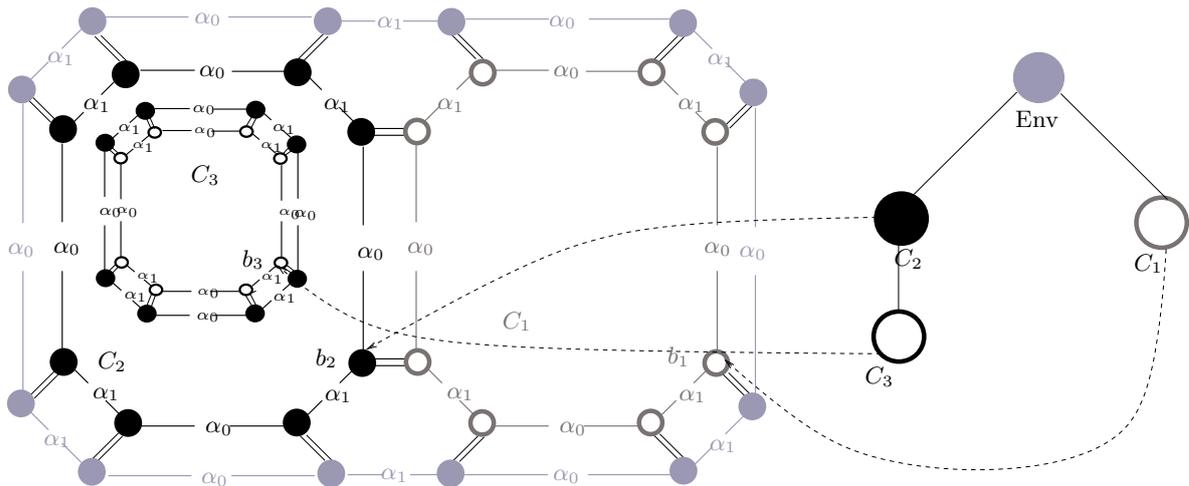


FIGURE 3.26 – Représentation d'une compartimentation cellulaire par un modèle bio-géométrique

Notre modèleur bio-géométrique est basé sur une représentation des objets par un *modèle bio-géométrique*. Comme nous venons de l'expliquer, c'est un modèle basé sur deux structures topologiques qu'il convient de lier pour en assurer la cohérence.

Prenons l'exemple figure 3.26, de représentation d'une compartimentation cellulaire à trois compartiments, C_1 , C_2 et C_3 . Pour une question de lisibilité, nous prenons notre exemple en deux dimensions (2D) sur la figure, même si en pratique les modèles bio-géométriques sont définis en trois dimensions (3D).

Le compartiment par défaut Env est la racine de l'arbre d'inclusion dans tous les modèles bio-géométriques. Il permet d'avoir une carte généralisée fermée, puisque tout l'espace extérieur à la com-

partitionnement cellulaire est occupé par l'environnement qui est explicitement représenté par la G-carte. Dans l'exemple figure 3.26, ce sont les brins gris qui correspondent à cet environnement extérieur.

Le lien entre les deux parties du modèle se fait dans un sens par le plongement de la G-carte. Les volumes de la G-carte sont plongés sur les compartiments représentés par les nœuds de l'arbre d'inclusions. Comme le montre l'exemple figure 3.26 :

- les brins gris qui appartiennent au volume externe sont plongés sur l'environnement,
- les brins gris-blanc qui appartiennent au volume du compartiment C_1 , sont plongés sur le compartiment C_1 ,
- les brins noir-blanc qui appartiennent au volume du compartiment C_3 , sont plongés sur le compartiment C_3 ,
- et les brins noirs qui correspondent aux deux volumes du compartiment C_2 sont plongés sur le compartiment C_2 .

Le premier volume de C_2 correspond à son enveloppe convexe extérieure, et l'autre correspond à son interface avec C_3 qui est le compartiment inclus dans C_2 . Ces deux volumes sont distincts dans la G-carte, mais ils sont bien plongés sur le même compartiment.

Dans l'autre sens, nous associons à chaque compartiment de l'arbre d'inclusions le volume de la G-carte correspondant à l'enveloppe convexe extérieur du compartiment, via un brin représentant (Ces associations sont matérialisées par les flèches en pointillés partant des nœuds de l'arbre vers les brins de la G-carte). Les volumes internes du compartiment sont accessibles via les compartiments fils dans l'arbre d'inclusions. Par exemple, figure 3.26, le compartiment C_2 a un lien vers le brin b_2 et donc vers toute l'enveloppe convexe du compartiment C_2 dans la G-carte. Pour accéder à l'enveloppe interne de C_2 (celle qui entoure C_3) il suffit :

- d'accéder au compartiment fils dans l'arbre d'inclusions, c'est-à-dire au compartiment C_3 ,
- puis d'accéder au brin b_3 représentant l'enveloppe convexe de C_3 ,
- puis d'accéder au brin voisin dans la G-carte $\alpha_2(b_3)$ qui est un représentant de l'enveloppe interne de C_2 , celle qui entoure C_3 .

La cohérence topologique du modèle bio-géométrique est principalement assurée par la cohérence de sa G-carte. Il convient juste de garantir en plus la cohérence entre la G-carte et l'arbre d'inclusions. Comme les relations d'inclusion ne sont pas représentées dans la G-carte, cela revient juste à vérifier que toute liaison entre 2 volumes relie : soit deux compartiments père et fils, soit deux compartiments frères, dans l'arbre d'inclusions. Dans l'exemple en 2 dimension figure 3.26, cela est bien vérifié :

- tous les brins gris de l'environnement sont reliés par α_2 (représenté par un double trait sur la figure) soit à des brins gris-blancs de C_1 , soit à des brins noirs de C_2 , qui sont les deux fils de Env dans l'arbre d'inclusions ;
- tous les brins gris-blancs de C_1 sont reliés par α_2 soit aux brins gris de l'environnement comme nous venons de le voir, soit aux brins noirs de C_2 qui est son frère dans l'arbre d'inclusions ;
- tous les brins noirs de C_2 sont reliés soit à des brins de Env ou C_1 comme nous venons de le voir, soit aux brins noirs-blancs de C_3 qui est son fils dans l'arbre d'inclusion ;
- enfin tous les brins noirs-blancs de C_3 sont uniquement reliés aux brins de C_2 , car C_3 n'a ni fils ni frère dans l'arbre d'inclusions.

3.3.1.2 Plongement géométrique

De manière classique, les sommets de la G-carte sont plongés sur des points 3D pour en assurer la représentation géométrique et ainsi un affichage en 3 dimensions.

Le cohérence géométrique vise à assurer d'une part l'absence de chevauchements entre compartiments frères et d'autre part, l'inclusion géométrique des compartiments fils dans leur père. Pour faciliter ces vérifications lors de la construction d'une compartimentation cellulaire, nous restreignons la forme des compartiments à des volumes convexes, voir à des parallélépipèdes rectangles.

3.3.1.3 Plongement biologique

nom : C_1	nom : C_2
type : $type_{C_1}$	type : $type_{C_2}$
contenu global : $M_1 + M_2$	contenu feuillet externe membrane : M_1
(a) Compartiment C_1	contenu transmembranaire : vide
	contenu feuillet interne membrane : vide
	contenu intérieur compartiment : M_2
	(b) Compartiment C_2

FIGURE 3.27 – Les informations du plongement compartiment sur deux exemples

Comme nous l'avons vu dans la section 3.3.1.1, les volumes de la G-carte sont plongés sur les compartiments. Ces compartiments doivent disposer d'un nom et d'un type pour permettre la spécialisation des règles génériques pour une compartimentation cellulaire.

En outre, il convient d'associer au modèle bio-géométrique la composition bio-chimique initiale de la compartimentation cellulaire. Selon les choix de l'utilisateur, cette composition peut être *globale*, c'est-à-dire que chaque compartiment contient des composants bio-chimiques sans précision de leur localité. Pour cela, nous associons une troisième information de plongement aux compartiments. Ainsi, l'exemple du compartiment C_1 figure 3.27(a) contient sans précision de localisation les molécules M_1 et M_2 .

Au contraire, le contenu d'un compartiment cellulaire peut être *localisé* et dans ce cas on distingue le contenu interne d'un compartiment, de celui du feuillet externe ou interne d'une membrane, ou encore du contenu trans-membranaire. S'il est logique d'associer le contenu global ou intérieur d'un compartiment aux compartiments du modèle bio-topologique (c'est-à-dire aux volumes de la G-carte), la question se pose pour les contenus membranaires. En effet, les membranes sont naturellement représentées par les faces de la G-carte. Cependant, la géométrie polyédrique impose de représenter les membranes par plusieurs faces, alors qu'un compartiment ne contient qu'une seule membrane externe représentée par son enveloppe convexe. C'est pourquoi nous choisissons d'associer l'ensemble des contenus localisés aux compartiments. L'exemple du compartiment C_2 figure 3.27(b) contient sur le feuillet externe de sa membrane la molécule M_1 et à l'intérieur du compartiment la molécule M_2 .

Seulement l'un de ces deux plongements biologiques, contenu global ou localisé, peut être associé à chaque compartiment cellulaire. Par ailleurs, l'ensemble des compartiments cellulaires d'une même compartimentation doivent avoir un contenu biologique de même nature : tous les compartiments d'un même modèle bio-géométrique doivent donc soit avoir un contenu biologique global, soit un contenu localisé.

3.3.2 Opérations de création des compartiments cellulaires

Les principales opérations du modéleur bio-géométrique sont celles qui permettent de créer les compartiments cellulaires. Ce sont des opérations de haut niveau, qui permettent à l'utilisateur de créer directement des compartiments cohérents, sans manipuler directement les modèles bio-géométriques et ses structures topologiques.

La création d'un compartiment nécessite que soit fournis le nom du compartiment à créer, son type ainsi que le compartiment dans lequel il sera inclus. Cela établit de fait un certain ordre dans les opérations de création de compartiments : un compartiment ne peut être créé que lorsque son contenant a déjà été représenté. L'environnement qui n'a pas de contenant et qui contient directement ou indirectement tous les autres compartiments doit donc être créé en premier.

Le collage entre deux compartiments se fait lors de la création du deuxième compartiment et consiste en des liens entre deux faces topologiques, appartenant respectivement aux volumes topologiques représentant le premier compartiment (*compartiment de collage*) déjà représenté, et le second compartiment (*compartiment à coller*). Le choix de la *face de collage* (face du compartiment de collage sur laquelle va s'effectuer le collage) impose le contenant du compartiment à créer.

Pour résumer, nous pouvons écrire qu'en dehors de la création de l'environnement, trois options de créations de compartiment existent :

- création d'un *compartiments isolé*, c'est-à-dire directement inclus dans l'environnement ;
- création d'un compartiments inclus dans un autre : le contenant du compartiment à créer est choisi parmi les compartiments déjà représentés ;
- création d'un compartiment collé à un autre : la face de collage est choisie parmi les *faces libres* du compartiment de collage qui est déjà représenté.

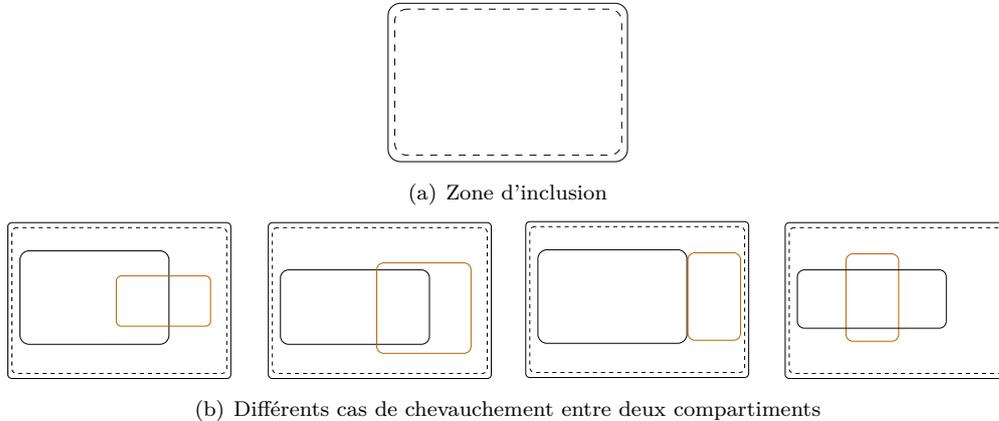


FIGURE 3.28 – Vérifications à effectuer pour le positionnement des compartiments à créer

Notons que la création d'un compartiment isolé est un cas particulier de création d'un compartiment inclus. Pour toutes les créations de compartiment, des vérifications sont faites avec pour objet d'assurer que :

- le nom proposé pour le compartiment à créer n'est pas déjà porté par un autre ;
- l'emplacement choisi par l'utilisateur pour le compartiment à créer respecte les exigences de non-chevauchement et d'inclusion appropriées. La figure 3.28 présente en 3.28(a) la zone d'inclusion en pointillés : tous les sommets de la représentation d'un compartiment inclus dans C_1 doivent être à

l'intérieur de la zone d'inclusion. En 3.28(b), on a les différents cas de chevauchement qui doivent être évités : aucun sommet du compartiment en création ne doit se trouver dans la représentation d'un autre compartiment déjà créé et vice versa ; les deux représentations ne doivent pas être collées, ni "se traverser".

3.3.3 Extraction du graphe d'échange

Il semble de prime abord naturel de se fonder sur la structure topologique du modèle bio-géométrique pour l'extraction du graphe d'échanges à partir de l'arbre d'inclusions. En effet, l'arbre d'inclusions du modèle contient déjà :

- pour sommets les compartiments et les informations biologiques associées,
- pour arcs les liaisons d'inclusions.

Extraire le graphe d'échange consiste donc à reprendre l'arbre d'inclusions tel quel et à y ajouter les liaisons de collage issues de la G-carte.

Du fait de la cohérence géométrique des modèles bio-topologiques, cohérence garantie par les opérations de construction précédentes, ces liaisons de collages ne peuvent exister qu'entre des compartiments frères de l'arbre d'inclusions. Elles peuvent donc aisément être retrouvées en parcourant les brins de l'enveloppe convexe de chaque compartiment.

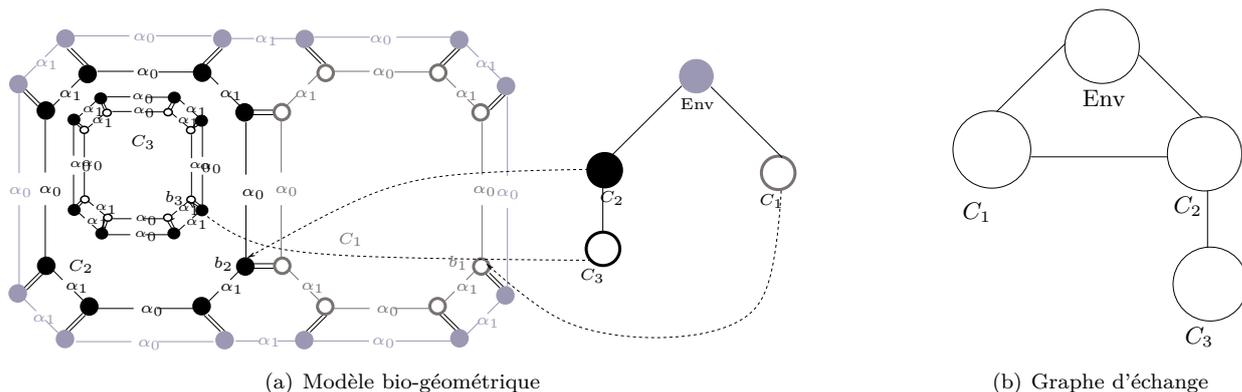


FIGURE 3.29 – Extraction du graphe d'échange

Reprenons le modèle bio-géométrique de la figure 3.26 page 60 reproduit par la figure 3.29(a) et le graphe d'échanges extrait sur la figure 3.29(b). L'environnement n'ayant jamais de frère, il est inutile de parcourir ses brins (en gris sur la figure), ils sont tous voisins de brins gris-blancs de C_1 ou de brins noirs de C_2 ses deux fils dans l'arbre d'inclusions. C_1 et C_2 sont frères dans l'arbre d'inclusions et voisins par collage dans la G-carte. En effet quand on parcourt les brins noirs et gris-blancs des enveloppes convexes respectives de C_1 et C_2 elles sont voisines par α_2 (en double trait sur la figure) soit des brins gris de leur père Env , soit mutuellement voisins. Le compartiment C_3 , quand à lui, n'a pas de frère dans l'arbre d'inclusion et donc les seuls voisins des brins noirs-blancs de son enveloppe convexe dans la G-carte, sont des brins noirs de C_2 .

Nous faisons cependant remarquer que l'extraction du graphe d'échanges s'appuie uniquement sur la G-carte. En effet grâce à l'utilisation de G-cartes fermées, toutes les relations de voisinage, y compris celles dues à l'inclusion, se traduisent par des liaisons α_3 dans la G-carte. De plus, l'arbre d'inclusions peut

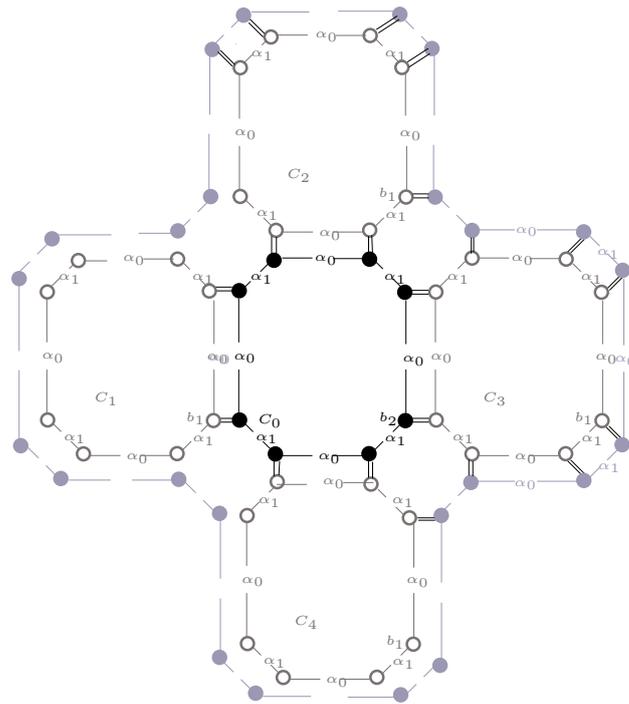


FIGURE 3.30 – Compartiment inclus mais non voisin du contenant

contenir des relations pères-fils qui n'induisent aucun voisinage en termes de possibilité de communication. Considérons par exemple la figure 3.30. Le compartiment C_0 , bien qu'inclus dans l'environnement comme les autres, n'a pas de voisinage avec celui-ci. En effet, on ne voit sur le schéma aucune liaison reliant un brin gris (de l'environnement) à un brin noir (de C_0).

Cet état de fait souligne l'importance d'avoir une représentation réaliste des compartimentations cellulaires. De plus, cela nous conforte dans notre option d'avoir une G-carte fermée permettant la construction du graphe d'échanges sans passer par l'arbre d'inclusions.

3.4 Implémentation

L'implantation de nos différents algorithmes s'est fait en C/C++ dans l'environnement de développement intégré Qt.

Dans cette section, nous présentons les différentes structures de données que nous utilisons pour l'implantation de notre modèleur bio-géométrique et pour celle des graphes d'échanges. Par ailleurs, nous y décrivons la fonction d'extraction d'un graphe d'échanges à partir d'un modèle bio-géométrique. Pour la définition de notre modèleur, nous nous appuyons sur un modèleur existant, MOKA [?] qui implémente les 3-G-Cartes. Une telle option est motivée par les faits suivants :

- MOKA est open-source, distribuable avec une licence GPL ;
- il est structurée en couches autour d'un noyau, ce qui permet la réalisation d'applications spécifiques bâties autour de ce noyau ;
- nous avons travaillé avec une équipe qui avait déjà une bonne connaissance de l'outil d'où une prise en main potentiellement facilitée.

Notre modelleur sera donc une surcouche de MOKA offrant comme fonctionnalités la définition et la manipulation de cartes généralisées représentant des compartimentations cellulaires, la gestion des compartiments et de leurs contenus initiaux, et enfin l'extraction d'un graphe d'échanges.

Pour commencer cette section, nous nous attachons à donner un aperçu de MOKA à travers notamment son architecture globale et les principales classes et structures de données qui y sont définies. Ensuite nous présentons les extensions propres au modelleur bio-géométrique, et donnons les détails de l'implantation de l'algorithme d'extraction d'un graphe d'échanges. Enfin nous concluons la section par des exemples de modèles bio-géométriques et des graphes d'échanges correspondants.

3.4.1 Présentation de MOKA

MOKA est un modelleur géométrique qui implémente les 3-G-cartes. Il a été réalisé en C++ avec une interface Qt, par une équipe de l'Université de Poitiers. Il s'agit d'un outil riche en fonctionnalités, intégrant une grande diversité de formes représentables et de nombreuses fonctions de manipulation des objets telles que les rotations, les translations, les extrusions, etc. Il offre également plusieurs options d'affichage (vue compacte, vues éclatées ou semi-éclatées) ainsi que des options de coloration de faces et de volumes etc.

3.4.1.1 Architecture et principales classes du noyau

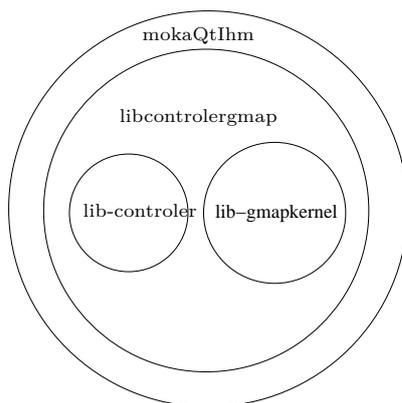


FIGURE 3.31 – Architecture 3 tiers de MOKA

MOKA a une architecture 3/3 classique avec un vue graphique, des contrôleurs et un noyau fonctionnel, comme le montre la figure 3.31. Le noyau de MOKA, *lib-gmapkernel*, contient l'ensemble des principales fonctions de définition et de manipulation d'une G-Carte ainsi que les classes nécessaires pour la définition et l'association de modèles de plongement aux orbites souhaitées. Le noyau MOKA compte environ 170 fichiers (entêtes et sources) totalisant plus de 30000 lignes de code.

La figure 3.32 est une portion du diagramme des classes de MOKA, présentant les principales classes du noyau. En accord avec les notations de UML (Unified Modeling Language) qui représentent :

- les héritages par des flèches « creuses » allant des classes spécialisées vers celles générales ;
- les relations de composition (contenances) par des flèches pleines commençant par un diamant noir, orientées vers la classe dont les objets sont contenus par l'autre,

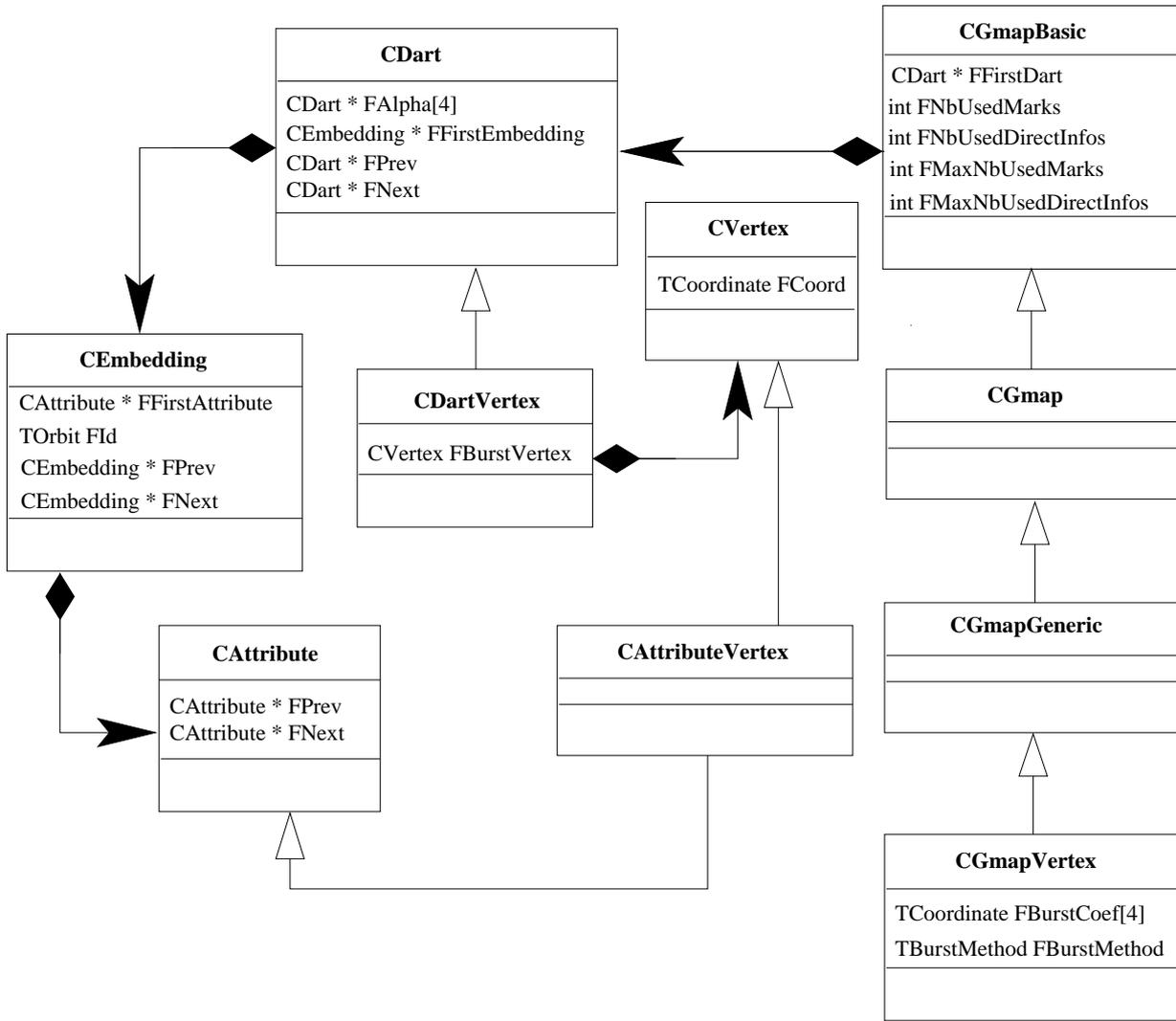


FIGURE 3.32 – Portion du diagramme des classes de MOKA.

nous pouvons lire en particulier, qu'une *CGMapVertex* (un objet polyédrique, i.e. une 3-G-carte dont les sommets sont plongés sur des points 3D) est par transitivité une *CGMapBasic* qui comporte une liste de *CDart* (les brins de la G-carte). Chaque *CDart* contient une liste de *CEmbedding* (qui sont les plongements associés à chaque orbite).

Sur le schéma, nous avons fait figurer les attributs privés et certains attributs protégés des différentes classes. Par contre nous n'avons pas représenté les relations réflexives telles que un *CDart* contient d'autres objets de la même classe qui jouent des rôles précis. Cette relation notamment nous fonde à voir la représentation des G-cartes dans MOKA comme celle d'un graphe dont les nœuds correspondraient aux brins et les arcs aux relations de voisinage entre brins. La classe *CDart* qui représente les brins, et toutes les informations associées est donc au cœur de la structure de données que nous détaillons dans le paragraphe suivant.

3.4.1.2 Structure des brins dans MOKA

Les informations associées à un brin b dans MOKA sont principalement :

- un tableau $FAlpha$ de quatre pointeurs vers les brins voisins $\alpha_0(b)$, $\alpha_1(b)$, $\alpha_2(b)$ et $\alpha_3(b)$;
- un double chaînage des brins dans la liste des brins de la G-carte, i.e. deux pointeurs sur des brins $FPrev$ et $FNext$ correspondant respectivement au brin précédent et au brin suivant b dans la liste des brins du modèle ;
- un pointeur sur la liste des listes des plongements $FFirstEmbedding$, chaque liste de plongements concernant une même orbite (notons que chaque plongement est porté par un unique brin de l'orbite concerné - par exemple, un seul brin de chaque sommet porte le plongement géométrique -) ;
- un tableau de trente-deux *marques* $FMarks$, les marques étant utilisées pour distinguer les brins lors des opérations de parcours, ce qui permet en particulier d'arrêter les parcours du graphe ;
- un tableau de 16 *bitset* $FUsedOrbits$ permettant de retrouver pour une carte donnée, les orbites plongées ;

3.4.2 Construction d'une structure cellulaire

L'implémentation du modèleur bio-géométrique a nécessité trois actions principales :

- la définition d'une bibliothèque de fonctions *lib-gmapbiokernel* utilisant les fonctions de *lib-gmapkernel* et dont les éléments servent pour les créations et les modifications de compartiments en maintenant à jour l'arbre d'inclusions associé à la G-carte ;
- la définition de la bibliothèque *lib-contrôleur-gmapbio*, correspondant à la partie *lib-contrôleur-gmap* du contrôleur et contenant les paramètres à prendre en compte dans les opérations sur les compartiments ;
- la programmation de l'interface utilisateur.

Nous nous limitons dans ce qui suit à décrire certains détails d'implémentation de la bibliothèque *lib-gmapbiokernel*, notamment ceux relatifs à la création et au collage des compartiments, ainsi qu'à l'enregistrement des modèles bio-géométriques. Auparavant, nous donnons dans la section 3.4.2.1 la forme qu'auront nos compartiments ainsi que la représentation d'un compartiment.

3.4.2.1 Forme des compartiments et quelques options de représentation

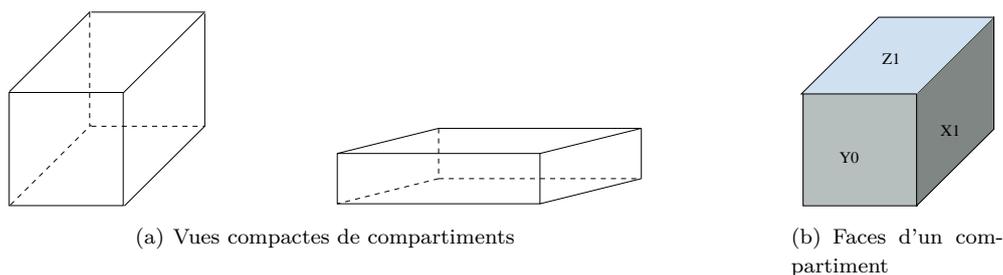


FIGURE 3.33 – Deux vues compactes de compartiments et identification des faces dans la représentation des compartiments

Nous convenons de donner aux compartiments biologiques une forme de parallélépipède rectangle. La figure 3.33(a) montre deux vues compactes de compartiments, la première correspondant à un cube qui est

un cas particulier de parallélépipède rectangle. Cette option est motivée par le fait qu'il s'agit d'une forme géométrique relativement simple ayant des propriétés intéressantes, ce qui est plus pratique pour les calculs liés aux considérations géométriques. Par exemple, en tant que parallélépipède rectangle, un compartiment comporte six faces deux à deux opposées : X_0 et X_1 , Y_0 et Y_1 , Z_0 et Z_1 dont trois X_1 , Y_0 et Z_1 (celles visibles) sont identifiées par la figure 3.33(b). En imposant par ailleurs qu'aucune rotation par rapport à l'un quelconque des trois axes OX , OY et OZ n'est autorisée dans notre modèleur, nous avons que, pour une représentation de compartiment donnée, les faces X_0 , Y_0 et Z_0 sont celles décrites par les quatre sommets ayant les plus petites coordonnées respectivement en X , Y et Z , tandis que les faces X_1 , Y_1 et Z_1 sont celles décrites par les quatre sommets ayant les plus grandes coordonnées respectivement en X , Y et Z . Cet état de fait est largement exploité dans les calculs pour les positionnements des compartiments.

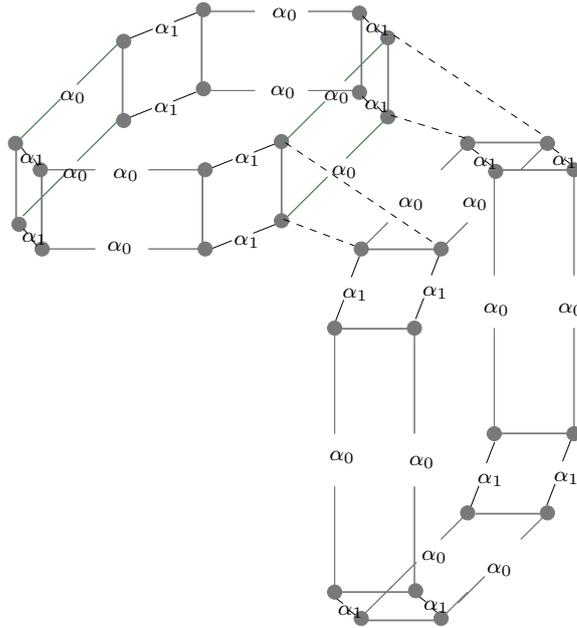


FIGURE 3.34 – Représentation de la double-enveloppe sur deux faces de cube

Une fois fixée la forme à donner aux compartiments, nous construisons sa double enveloppe, comme présenté dans la section 3.3. Nous les représentons comme des associations de deux parallélépipèdes de même taille, géométriquement localisés au même emplacement, liés par des liaisons α_3 . La figure 3.34 donne une vue éclatée partielle représentant les faces X_1 et Z_1 d'un compartiment. On y distingue les liaisons α_0 et α_1 qui sont mentionnées tandis que les liaisons α_2 sont des traits en pointillés ce qui les différencie des liaisons α_3 qui sont des traits pleins gris ne portant pas d'étiquette.

3.4.2.2 création et collage de compartiments

Pour la création et le collage de compartiments, nous avons défini de nouvelles classes d'objets, devant notamment servir pour la mise en œuvre du plongement biologique dont nous avons besoin. La figure 3.35 donne (en noir) les classes définies dans la librairie lib-gmapbiokernel et leurs liens avec les classes préexistantes dans MOKA (grisé). On y lit en particulier qu'une *GMapBio* est une *GMapVertex* et que les brins (*CDartCompartiment*) d'un modèle bio-géométrique, qui héritent de la classe *CDartVertex* ont un plongement compartiment (*CCompartimentBio*). L'annexe D comporte les *fichiers entêtes C++*

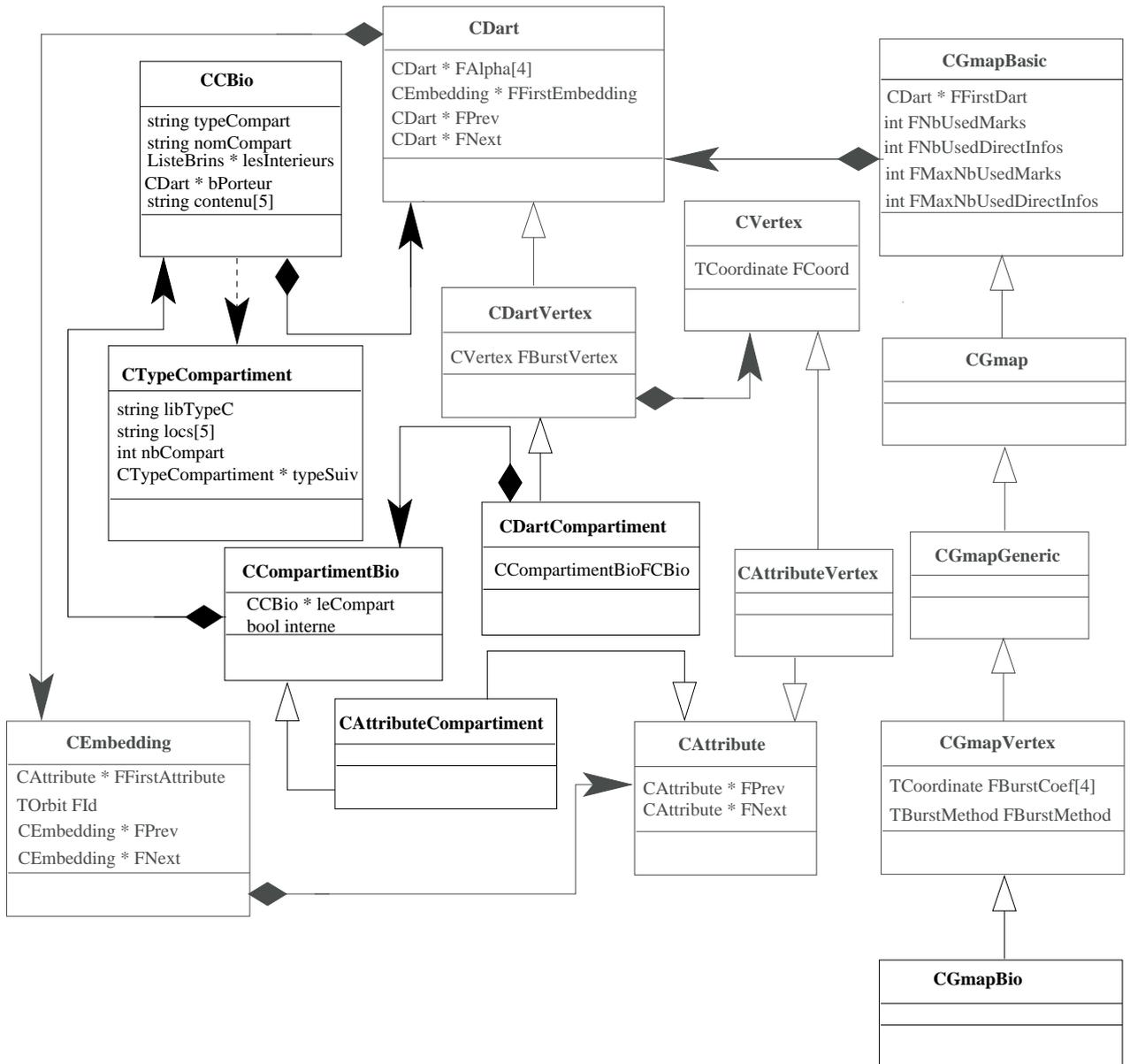
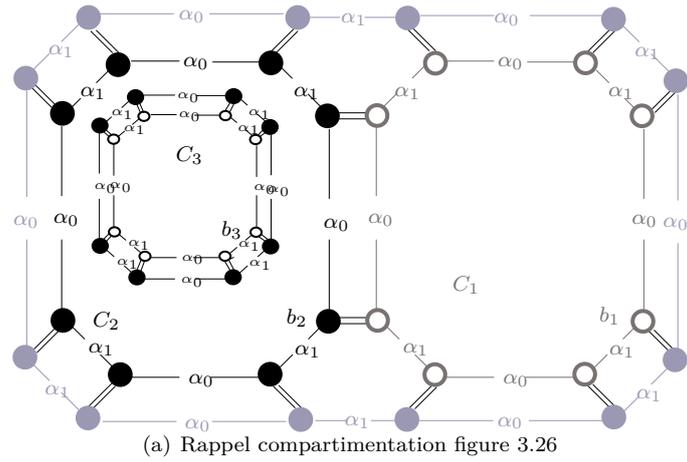


FIGURE 3.35 – Diagramme de classes de lib-gmapbiokernel.

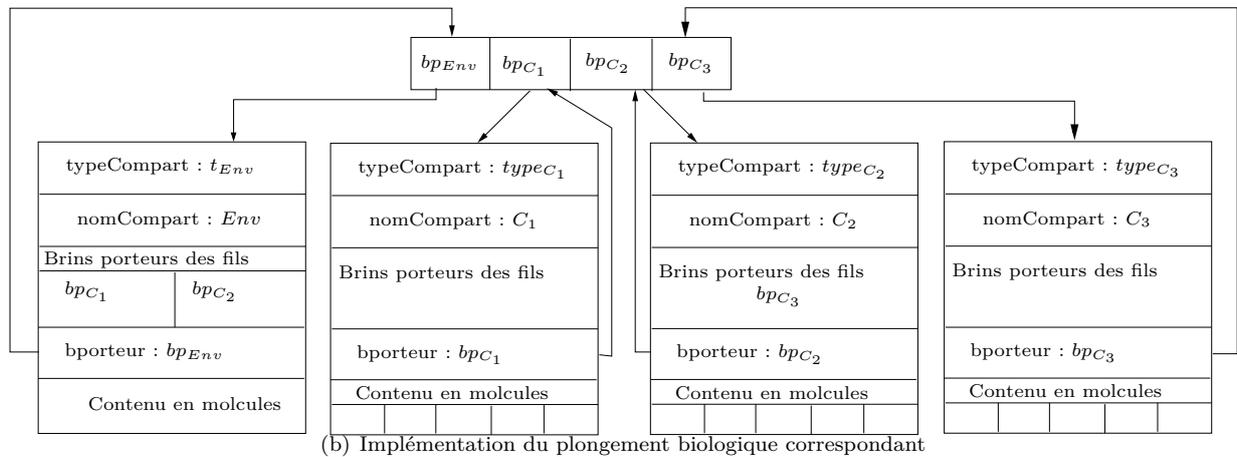
définissant les classes de lib-gmapbiokernel (Pour des raisons de concision, seules quelques méthodes sont données) .

La classe *CCBio* implante le plongement biologique (voir la section 3.3). Pour accélérer les parcours des brins des modèles bio-géométriques, le plongement des volumes consiste en une association d’un objet *CCBio* et d’un booléen qui précise le rôle du volume plongé. Ce booléen indique si le volume est l’enveloppe externe du compartiment ou non. Cette association donne un objet de la classe *CCompartmentBio*.

Concrètement, l’arbre d’inclusions n’est pas représenté en tant que tel. Nous maintenons plutôt à jour une liste de brins représentatifs des compartiments *listeCompartment*. A partir de ses brins représentatifs, on retrouve le plongement biologique (objets *CCompartmentBio* et *CCBio*) et donc l’arbre d’inclusions.



(a) Rappel compartimentation figure 3.26



(b) Implémentation du plongement biologique correspondant

FIGURE 3.36 – Compartimentation et implémentation du plongement biologique correspondant

En effet, chaque compartiment $CCBio$ contient, d'une part le brin représentatif de son enveloppe convexe $bporteur$, et d'autre part la liste des brins porteurs de ses fils (qui permettent à leur tour d'accéder aux compartiments inclus). Ainsi, partant d'un quelconque compartiment $CCBio$, il est possible de parcourir son enveloppe convexe à l'aide de l'orbite $\langle \alpha_0 \alpha_1 \alpha_2 \rangle$ ($bporteur$), et l'enveloppe interne de son contenant qui le contient lui-même et tous ses compartiments collés à l'aide de l'orbite $\langle \alpha_0 \alpha_1 \alpha_2 \rangle$ ($\alpha_3(bporteur)$). Les vérifications de non chevauchement des compartiments est basée sur les parcours de ces enveloppes.

La figure 3.36 reprend (pour rappel) en 3.36(a) la compartimentation cellulaire de la figure 3.26 et donne (3.36(b)) le détail des objets qui l'implémentent : la liste des brins représentatifs et les compartiments ($CCBio$) correspondants.

Comme nous l'avons expliqué (section 3.3) le modeler bio-géométrique offre plusieurs opérations de haut niveau pour créer facilement des compartiments cellulaires. La première opération consiste à créer un compartiment isolé dans l'environnement ou à l'intérieur d'un compartiment donné. Nous en donnons l'algorithme général ci-après.

Algorithme 3.1. (Création d'un compartiment isolé).

1. Effectuer les vérifications sur le nom de compartiment proposé pour s'assurer que le nom proposé n'est pas porté par un compartiment déjà créé : on parcourt pour cela `listeCompart` à la recherche d'un éventuel brin incident à un volume topologique dont le plongement biologique a comme valeur pour `nomCompart`, le nom proposé. Si on ne trouve pas un tel brin dans `listeCompart`, le nom proposé est accepté autrement l'utilisateur en fournit un autre qui subit les mêmes vérifications et ainsi de suite.
2. Vérifier que l'emplacement délimité par l'utilisateur respecte les exigences d'inclusion : Lorsque le contenant est `Env`, aucune vérification d'inclusion n'est faite. Par contre, lorsqu'il s'agit d'un autre compartiment, l'application intègre des contrôles interactifs empêchant l'utilisateur de définir une zone hors de la zone d'inclusion (cf. figure 3.28(a)). Nous avons fixé à 0.02 la distance minimale entre les faces du contenant et les faces des compartiments inclus.
3. Vérifier que l'emplacement délimité par l'utilisateur respecte les exigences d'absence de chevauchements avec les autres compartiments déjà représentés dans le contenant : Cette vérification ne se fait pas interactivement mais plutôt après que l'utilisateur a demandé la création effective du compartiment. Elle s'appuie sur l'intuition que deux compartiments (parallépipède rectangle) ne se chevauchent pas si tous les sommets de l'un sont du même côté des deux plans délimités par deux faces opposées de l'autre. Notons que cette intuition n'est vérifiée qu'en raison de l'absence de rotation imposée.
4. Si toutes les exigences de positionnement sont satisfaites, créer la membrane interne du compartiment en utilisant la fonction de création de cube de MOKA ;
5. Créer la membrane externe en effectuant une fermeture par α_3 du cube créé ;
6. effectuer les plongements volumes (un brin de la membrane interne va porter le plongement compartiment (à créer) consistant au nom du compartiment en création et un brin de la membrane externe porte le plongement compartiment consistant au contenant.

En ce qui concerne la création d'un compartiment collé à une face, les traitements sont quelque peu différents dans la mesure où il faut déterminer la surface de collage qui représente la membrane commune aux deux compartiments. La face de collage est choisie par l'utilisateur, ce qui impose la face à coller dans le compartiment à créer : si la face de collage est $X0$ ou $Y0$ ou $Z0$, la face à coller sera respectivement $X1$ ou $Y1$ ou $Z1$ et vice-versa.

Pour des raisons pratiques, notamment pour éviter des calculs relativement plus lourds dans la détermination de la surface de collage, nous imposons que la face à coller soit totalement incluse dans la face de collage. Cela est quelque peu restrictif pour l'utilisateur qui n'a notamment pas la possibilité de représenter un collage engageant une portion de face de chacun des compartiments concernés : en effet, un collage occupe totalement une des faces du compartiment qui a la plus petite face de collage. Ainsi, quatre possibilités de collage peuvent être distinguées :

- le premier cas est celui où les deux faces sont identiques : aucun traitement n'est nécessaire pour déterminer la surface de collage ;
- les trois autres cas concernent une face à coller plus petite que la face de collage.

Ces trois cas énumérés par la figure 3.37 se distinguent par l'emplacement relatif de la face à coller sur la face de collage. La figure 3.37 montre en 3.37(a) la face de collage F_1 en noir et en 3.37(b) la face à coller F_2 en gris. Les figures 3.37(c), 3.37(d) et 3.37(e) délimitent les surfaces de collage pour un collage respectivement dans un coin de F_1 , sur un bord de F_1 et au milieu de F_1 . On peut voir que selon les cas la topologie à créer est différente. En particulier dans le cas d'un collage en milieu de face, il convient de créer une arête fictive pour représenter l'inclusion des faces (voir figure 3.37(e)).

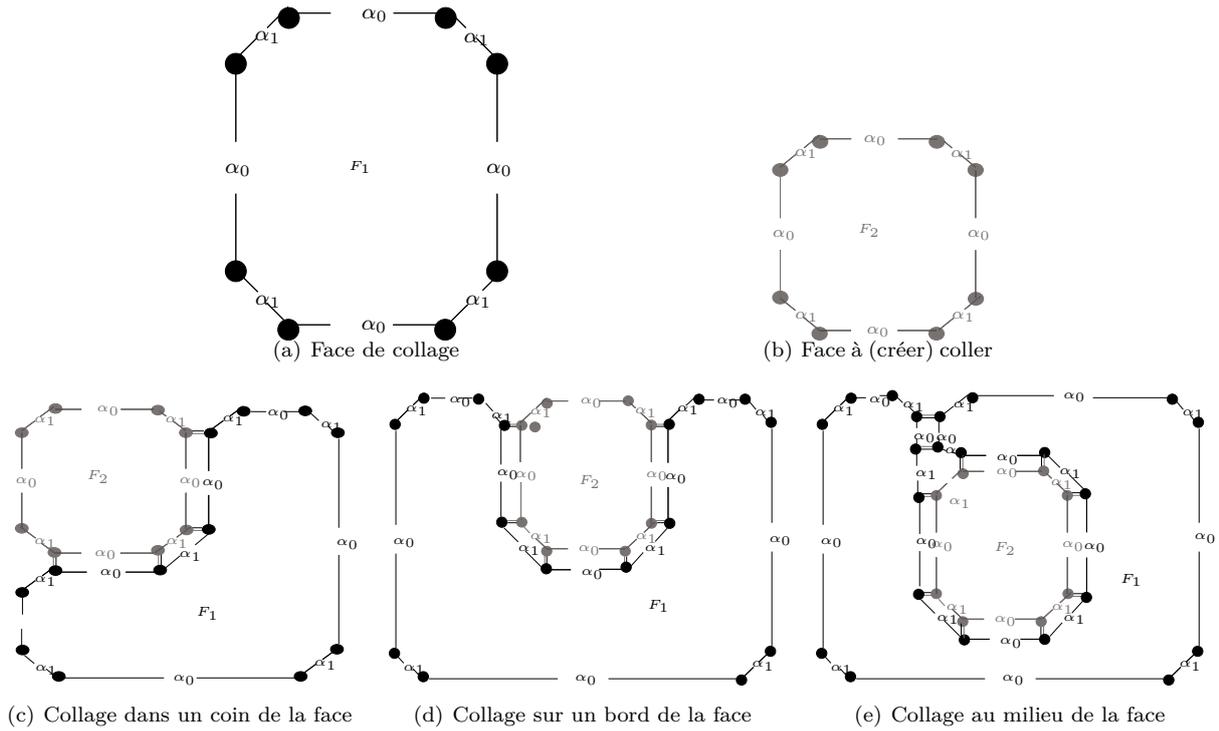


FIGURE 3.37 – Cas de collage de faces de dimensions différentes

Algorithme 3.2. (Création d'un compartiment collé à un autre).

Des vérifications sont faites pour s'assurer que la face est libre de tout collage. Tous les brins de la face doivent être α_3 liées à un volume interne d'un compartiment, et non à une enveloppe convexe externe. Dans ce cas, la face est libre et peut faire l'objet d'un collage.

1. Effectuer les vérifications sur le nom de compartiment proposé.
2. Vérifier que l'emplacement délimité par l'utilisateur respecte les exigences d'inclusion dans l'enveloppe convexe du contenant.
3. Vérifier que l'emplacement délimité par l'utilisateur respecte les exigences d'absence de chevauchements avec les autres compartiments (sauf celui de collage) déjà représentés dans le contenant. La vérification se fait comme dans l'algorithme 3.1. Par ailleurs des contrôles sont intégrés pour que l'utilisateur ne puisse définir son emplacement que géométriquement collé à la face choisie.
4. Créer la membrane du compartiment (son enveloppe convexe) et effectuer le plongements géométriques et biologiques appropriés.
5. Supprimer l'enveloppe englobante de la composante connexe de collage et préparer la surface de collage en parcourant les deux faces simultanément et en traitant les trois cas suivants :
 - (a) pour chaque brin de la face à coller, soit il a un « homologue » (mêmes coordonnées et même sens) sur la face de collage, soit il se trouve sur une demi-arête de la face de collage du compartiment, soit il est au milieu de la face de collage sans lien particulier avec tous les brins de cette face ;
 - (b) pour chaque brin b de la face de collage, soit il a un homologue sur la face à coller soit il se trouve sur une demi-arête de cette face.

A l'issue de cette étape, on doit avoir deux faces isomorphes topologiquement et géométriquement.

6. *Effectuer le collage par une 3-couture et créer si nécessaire une arête fictive.*
7. *Recréer la membrane englobante de la nouvelle composante connexe et effectuer les plongements nécessaires.*

Il peut être utile ou pratique de coller des compartiments déjà créés par l'une des opérations précédentes. Bien entendu cela n'a de sens que si les deux compartiments sont contenus dans le même compartiment père. De plus, si les deux compartiments à coller ne sont pas déjà contigus géométriquement, il convient de les déplacer et donc de refaire les vérifications de non chevauchement appropriés.

Pour cela, nous avons repris le choix effectué par MOKA, qui propose une opération de couture qui étire l'objet « à coller » pour déplacer sa face de collage le long de la face « de collage » de l'autre objet. Dans notre cas, de formes de compartiments simplifiés, cela revient à étirer le parallélepède rectangle de l'objet à coller. Nous détaillons l'algorithme ci-après.

Algorithme 3.3. (Collage de deux compartiments déjà créés).

Les deux compartiments à coller doivent avoir le même contenant. L'utilisateur sélectionne les deux faces à coller. L'un des compartiments (celui dont la face est la plus grande) sera considéré comme le compartiment de collage (C1) et l'autre le compartiment à coller (C2).

1. *On vérifie que les deux faces sont :*
 - (a) *opposées (face X0 de C1 et face X1 de C2, face Y0 de C1 et face Y1 de C2, face Z0 de C1 et face Z1 de C2 et inversement) ;*
 - (b) *libres de tout collage : la vérification se fait comme dans l'algorithme 3.2 ;*
 - (c) *disposées dans le bon ordre : (pour deux faces X0 et X1, la coordonnée en X de la face X1 doit être inférieure à celle en X de la face X0).*
2. *On vérifie que le collage ne va pas créer de chevauchements en déterminant la position de C2 après collage. Si le collage doit engendrer des chevauchements, il est abandonné. Dans le cas contraire, on procède au collage de manière similaire à l'algorithme 3.2.*
3. *Supprimer l'enveloppe englobante de la composante connexe des 2 compartiments à coller. (Les deux compartiments peuvent ou non être dans la même composante connexe.)*
4. *Si besoin subdiviser la surface de collage comme précédemment.*
5. *Effectuer le collage par une 3-couture et créer si nécessaire une arête fictive.*
6. *Recréer la membrane englobante de la nouvelle composante connexe et effectuer les plongements nécessaires.*

3.4.2.3 Initialisation de la structure cellulaire

Nous avons défini une syntaxe pour représenter les ensembles de molécules et avons réalisé un analyseur syntaxique pour vérifier qu'une expression donnée par l'utilisateur respecte cette syntaxe. Le détail de l'implantation de cet analyseur sera donné dans le chapitre 4, car la syntaxe utilisée ici est extraite de celle plus générale des règles génériques.

L'initialisation peut se faire au moment de la création du compartiment ou après. L'utilisateur entre pour un compartiment ou une localisation précise dans un compartiment une chaîne de caractères qui

représente l'ensemble des molécules présentes initialement. Si cette entrée est syntaxiquement correcte, elle est stockée. Nous optons de la garder sous forme de chaîne de caractères pour faciliter son édition par l'utilisateur.

3.4.2.4 Enregistrement des modèles bio-géométriques

Deux fonctions ont été développées pour respectivement l'enregistrement dans un fichier et le chargement à partir d'un fichier des modèles bio-géométriques. Ces fonctions s'appuient sur celles de MOKA pour l'enregistrement et le chargement d'une G-carte, qu'elle complète avec les informations issues des plongements biologiques.

3.4.3 Extraction d'un graphe d'échanges à partir d'un modèle bio-géométrique

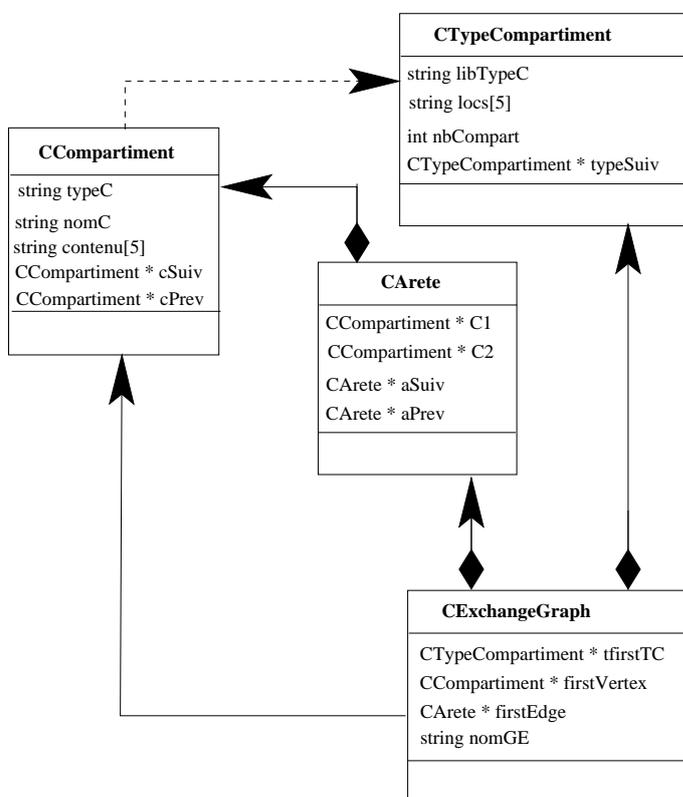


FIGURE 3.38 – Diagramme de classes de libGE.

L'implantation du graphe d'échanges a donné lieu à une bibliothèque statique *libGE* essentiellement bâtie autour de la classe *CCompartment* qui a à peu près les mêmes attributs que la classe *CCBio*. La figure 3.38 représente le diagramme de classes de la bibliothèque *libGE*. On y lit qu'un graphe d'échanges (*CExchangeGraph*) est défini sur un ensemble de types de compartiments (*CTypeCompartment*) et contient un ensemble de sommets (*CCompartment*) et un ensemble d'arêtes (*CArete*).

En plus des constructeurs et des accesseurs classiques, *libGE* offre à travers la classe *CExchangeGraph* (entête en annexe D.2 quelques fonctions de manipulation des graphes d'échanges telles que le calcul de sous-graphes induits par un ensemble de sommets, la détermination de la connexité des graphes, de l'ensemble des sommets d'un type donné, le calcul des arrangements de compartiments sur la base d'un vecteur de types de compartiments etc. Il intègre pour finir une fonction d'enregistrement dans un fichier texte (extension *.ge*) d'un graphe d'échanges et une fonction permettant de charger le contenu d'un fichier de graphe d'échanges pour utilisation. C'est du reste par le biais de tels fichiers que l'utilisateur peut créer directement un graphe d'échanges. Un analyseur syntaxique intégré à celui des règles génériques a été développé pour vérifier que les fichiers qu'on veut charger respectent la syntaxe d'un fichier de graphe d'échanges.

L'algorithme d'extraction du graphe d'échanges dont le code est donné (listing 3.1) utilise deux parcours d'orbites de la G-carte plus un marquage (par *treated* défini en ligne 5) des brins traités.

Le premier parcours (*cov* défini en ligne 4) concerne l'ensemble des brins. Il permet d'inclure dans le graphe d'échanges les compartiments correspondant aux volumes incidents aux brins non marqués, si ceux-ci n'ont pas encore été inclus dans le graphe d'échanges (première boucle *for*, ligne 7) : pour le brin courant dans le parcours (**cov* qu'on affecte à une variable temporaire *b1*), si *b1* est marqué (test ligne 10), on passe au brin suivant. Si *b1* n'est pas marqué :

- on récupère le plongement biologique de *b1* (ligne 12) principalement le nom de compartiment (*nC*, ligne 13) et on vérifie s'il est déjà dans le graphe en extraction *GERes* (ligne 14). Si c'est le cas, on passe au brin suivant dans le parcours *cov*. Sinon, on vérifie (lignes 16 et 17) si le type de compartiment (*tC*) est déjà enregistré et on l'insère (ligne 17) dans le graphe sinon (ligne 18). Ensuite on insère dans *GERes* le compartiment de nom *nC* et de type *tC* (ligne 19).
- C'est alors que commence le second parcours *cov1* qui concerne les brins du volume incident à *b1* (ligne 21) pour récupérer les voisinages traduits par les brins non marqués de ce volume et leurs α_3 . Pour un brin *b1* de ce parcours (ligne 24), s'il n'est pas marqué (ligne 25), on récupère le plongement biologique de son α_3 (ligne 27) et on vérifie si le type de compartiment (*tC1*) et le compartiment lui-même (*nC1*) sont déjà enregistrés dans le graphe. Si ce n'est pas le cas, on procède à leur insertion (lignes 31 à 34). On insère ensuite une arête entre le compartiments *nC* et *nC1* (ligne 36) et on marque les brins de la face incidente à *b1* (lignes 37).

Pour finir, on démarque tous les brins, on libère la marque *treated* et on détruit le parcours *cov* avant de retourner le graphe extrait comme résultat.

Listing 3.1 – Code d'extraction du graphe d'échanges

```

1 CExchangeGraph * CGMapBio::extraireGrapheDEchanges()
2 {
3     CExchangeGraph * GERes = new CExchangeGraph();
4     CStaticCoverageAll cov(this);
5     int treated = getNewMark();
6     CDart * b1; CCBio * ctemp;
7     for (; cov.cont(); ++cov)
8     {
9         b1 = *cov;
10        if (!isMarked(b1,treated))
11        {
12            ctemp = findCompartiment(b1)->getLeCompart();
13            string nC = ctemp->getNomC();
14            if (GERes->findCompartiment(nC) == NULL)
15            {
16                CTypeCompartiment * tC = ctemp->getTypeC();
17                if (GERes->findTypeCompartiment(tC) == NULL)
18                    GERes->insertTypeC(tC);
19                GERes->insertVertex(tC, nC);
20            }
21            CStaticCoverage012 cov1(this, b1);
22            for (; cov1.cont(); ++cov1)
23            {
24                b1=*cov1;
25                if (!isMarked(b1, treated))
26                {
27                    ctemp = findCompartiment(alpha3(b1))->getLeCompart();
28                    string nC1 = ctemp->getNomC();
29                    if (GERes->findCompartiment(nC1) == NULL)
30                    {
31                        CTypeCompartiment * tC1 = ctemp->getTypeC();
32                        if (GERes->findTypeCompartiment(tC1) == NULL)
33                            GERes->insertTypeC(tC1);
34                        GERes->insertVertex(tC1, nC1);
35                    }
36                    GERes->insertEdge(nC, nC1);
37                    markOrbit(b1, 013, treated);
38                }
39            } cov1.~CCoverage();
40        }
41    }
42    unmarkAll(treated); freeMark(treated); cov.~CCoverage();
43    return GERes;
44 }

```

3.5 Application

Nous montrons dans cette section quelques modèles bio-géométriques que nous avons construits comme représentations de structures cellulaires.

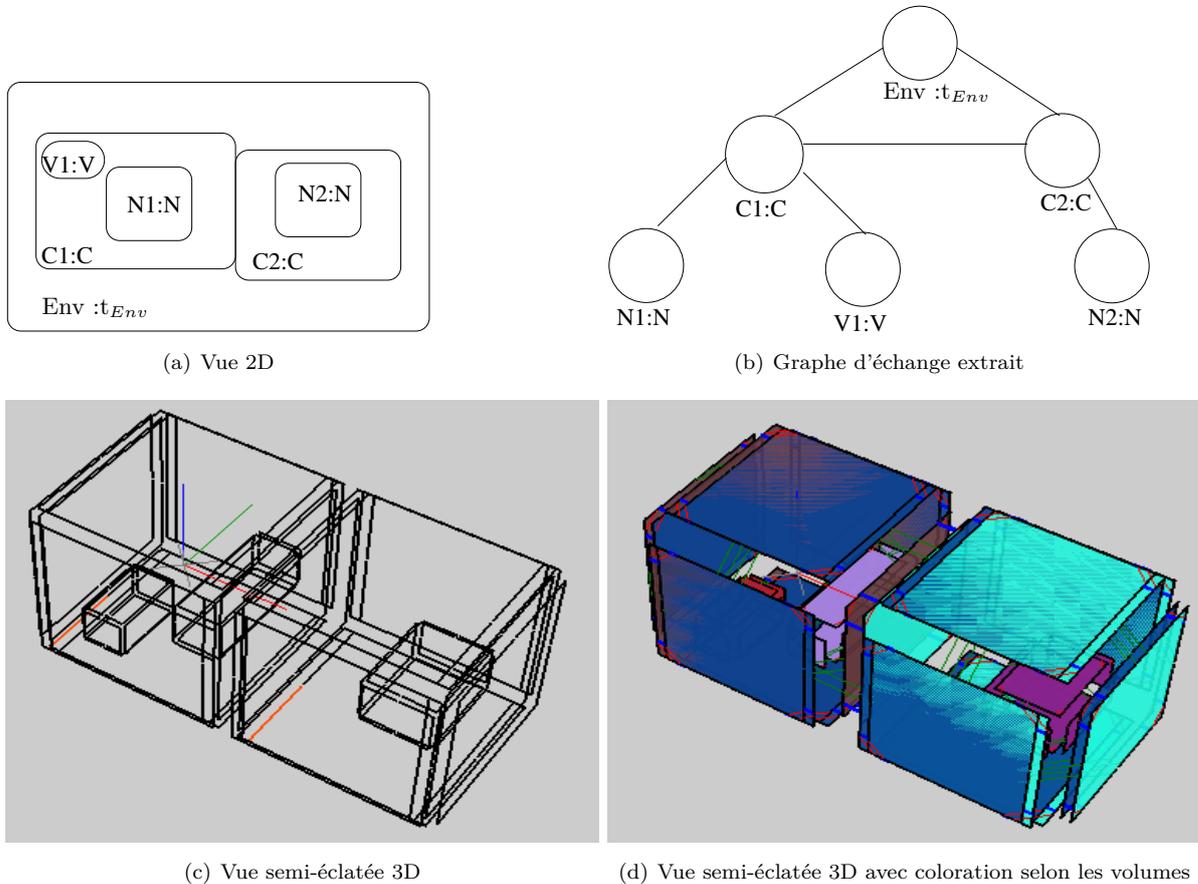


FIGURE 3.39 – Modélisation d'une compartimentation cellulaire et extraction de son graphe d'échange

La figure 3.39 reprend la compartimentation cellulaire de la figure 3.26 page 60. Elle montre, figure 3.39(a), un schéma 2D de cette compartimentation cellulaire, et figures 3.39(c) et 3.39(d), deux copies d'écran de la compartimentation modélisée à l'aide du modèle bio-géométrique. Notons que l'utilisateur dispose de toutes les options de visualisation de MOKA, pour choisir son option d'affichage (vues éclatés ou non, affichage fil de fer ou plein, etc.) et se déplacer dans la compartimentation cellulaire. Ainsi l'utilisateur peut « entrer » dans chaque compartiment pour en afficher l'intérieur. Le graphe d'échanges extrait est donné figure 3.39(b).

Le résultat de l'extraction du graphe d'échanges est donné par le listing 3.2. On y retrouve tous les compartiments de la représentation ainsi que toutes les arêtes.

Listing 3.2 – Fichier d'extraction de graphe d'échanges correspondant

```

1 GRAPHE D'ECHANGES
2 TYPES_COMPARTIMENT
3 N, V, C,
    
```

```

4 ext
5 FIN_TYPES_COMPARTIMENT
6
7 COMPARTIMENTS
8 C1 : C ; C2 : C ; Env : ext ;
9 N1 : N ; N2 : N ; V1 : V
10 FIN_COMPARTIMENTS
11
12 ARETES
13 {C1,C2}; {C1,Env}; {C1,N1};
14 {C1,V1}; {C2,Env}; {C2,N2}
15 FIN_ARETES
    
```

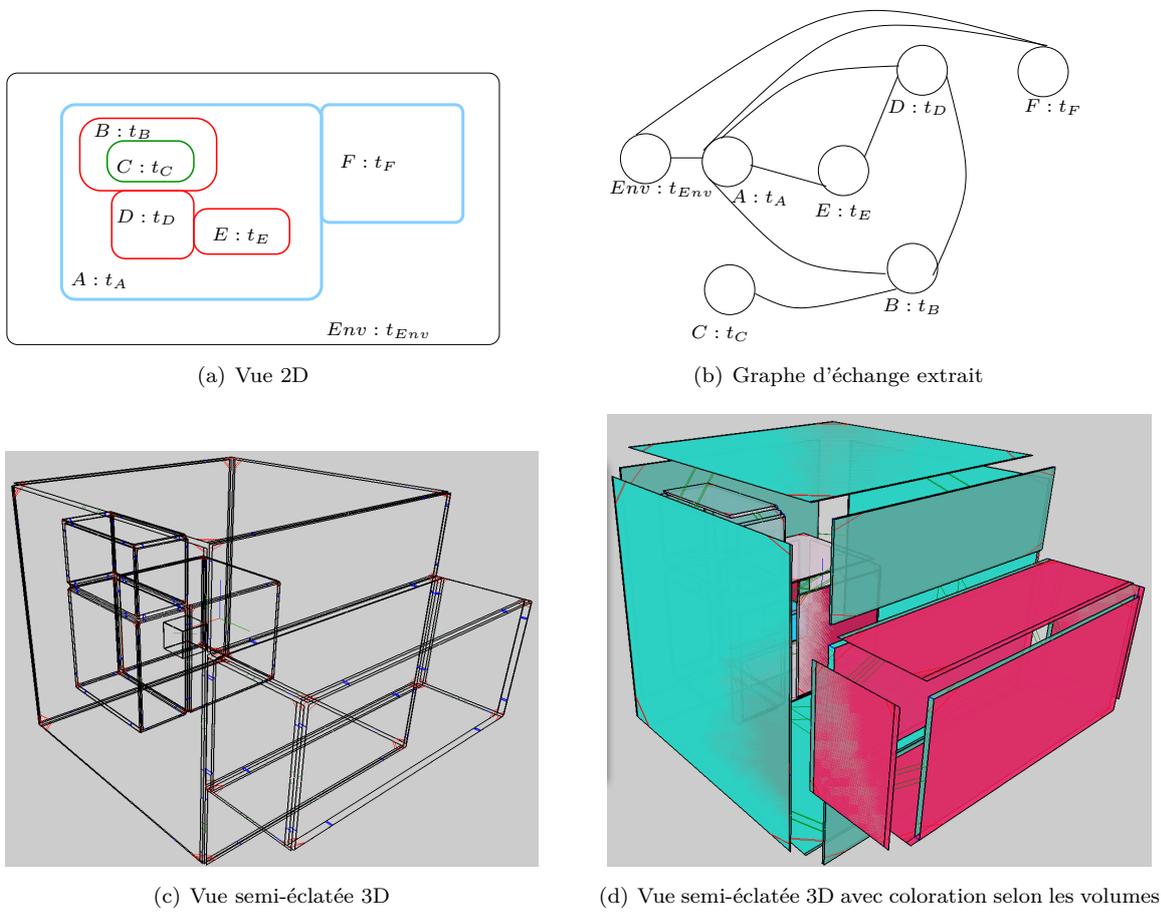


FIGURE 3.40 – Modélisation d'une compartiment cellulaire aux compartiments de taille différents

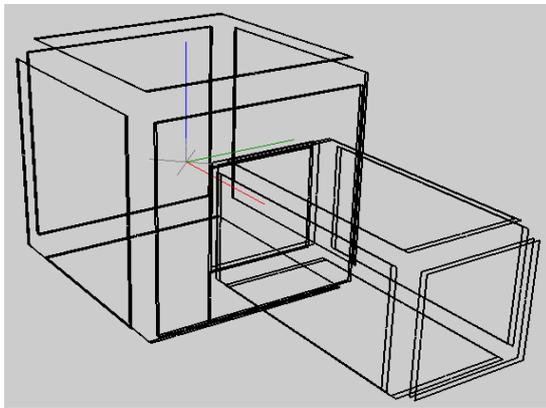
Comme deuxième cas d'illustration, nous avons construit le modèle bio-géométrique correspondant à la compartimentation donnée figure 3.25(a) page 57 qui est rappelée par la figure 3.40(b). Ce cas nous permet de tester le collage de deux compartiments le long de faces n'ayant pas les mêmes dimensions. Les sommets de la « petite face » sont alignés sur les arêtes de l'autre face. On constate que celle-ci a été divisée en trois parties, la partie centrale ayant servi de surface de collage. Le graphe d'échanges extrait

est donné par la figure 3.40(b), et le fichier d'extraction listing 3.3. Ces deux graphes sont bien identiques.

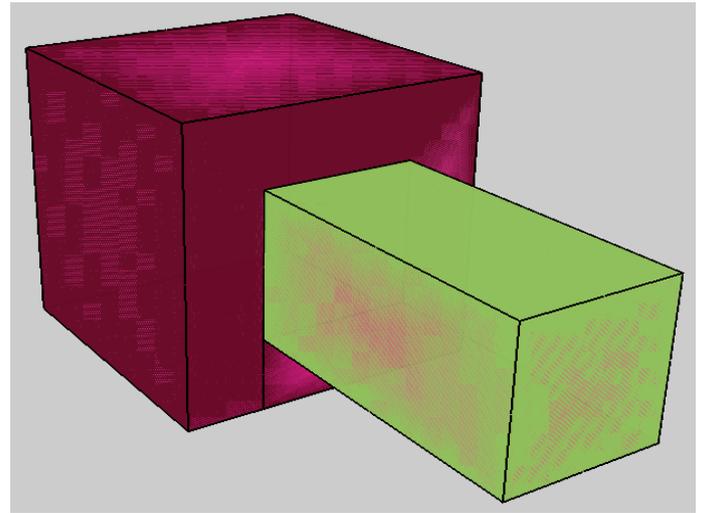
Listing 3.3 – Fichier d'extraction de graphe d'échanges correspondant

```

1 GRAPHE D'ECHANGES
2 TYPES_COMPARTIMENT
3 TC, TE, TD,
4 TB, TF, TA,
5 Ext
6 FIN_TYPES_COMPARTIMENT
7
8 COMPARTIMENTS
9 A : TA ; B : TB ; C : TC ;
10 D : TD ; E : TE ; Env : Ext ;
11 F : TF
12 FIN_COMPARTIMENTS
13
14 ARETES
15 {A,B}; {A,D}; {A,E};
16 {A,Env}; {A,F}; {B,C};
17 {B,D}; {D,E}; {Env,F}
18 FIN_ARETES
    
```



(a) Vue semi-éclatée simple



(b) Vue semi-éclatée avec coloration selon les volumes

FIGURE 3.41 – Modélisation d'un collage avec inclusion de la face de collage

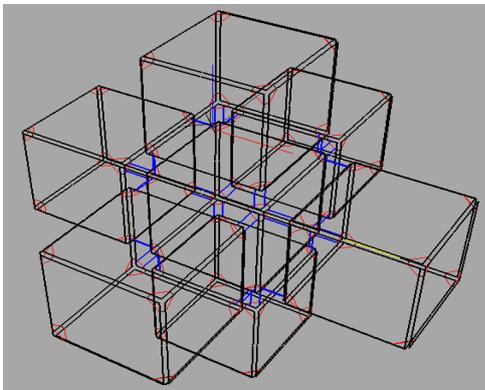
Le troisième cas que nous avons traité est un cas nouveau, relativement simple puisque consistant en une structure cellulaire à deux compartiments (plus l'environnement) adhérant l'un à l'autre. Son principal intérêt est qu'il nous permet de tester le collage de deux compartiments avec création d'une arête fictive. L'extraction du graphe d'échanges ne pose pas de problème et donne le fichier en listing 3.4.

Listing 3.4 – Fichier d'extraction de graphe d'échanges correspondant

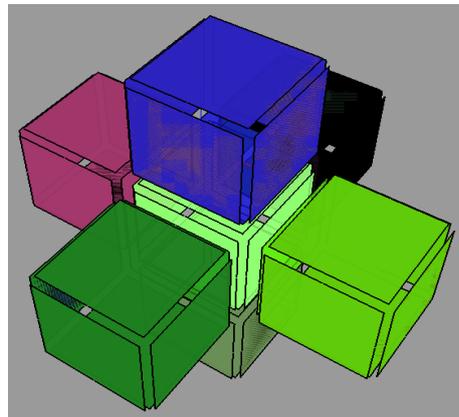
```

1 GRAPHE D'ECHANGES
2 TYPES_COMPARTIMENT
3 TB, TA, ext
4 FIN_TYPES_COMPARTIMENT
5
6 COMPARTIMENTS
7 A : TA ; B : TB ; Env : ext
8 FIN_COMPARTIMENTS
9
10 ARETES
11 {A,B}; {A,Env}; {B,Env}
12 FIN_ARETES

```



(a) Vue semi-éclatée simple



(b) Vue semi-éclatée avec coloration selon les volumes

FIGURE 3.42 – Modélisation d'un compartiment sans voisinage avec son contenant

Le quatrième cas traité concerne une compartimentation 3D similaire à celle 2D donnée en figure 3.30 page 65. Le compartiment centrale a toutes ses faces collées à d'autres compartiments, il n'a donc plus de surface d'échange avec l'environnement qui le contient. Dans ce cas, comme nous l'avons expliqué dans la section 3.3, le graphe d'échanges correspondant ne doit donc contenir aucune arête entre ce compartiment central et l'environnement. Ce qui est bien le cas, comme le montre le fichier d'extraction du graphe d'échanges (listing 3.5). En effet le compartiment centrale C0 est bien relié par une arête aux 6 autres compartiments C1 à C6, mais pas à l'environnement.

Listing 3.5 – Fichier d'extraction de graphe d'échanges correspondant

```

1 GRAPHE D'ECHANGES
2 TYPES_COMPARTIMENT
3 Cell, Ext
4 FIN_TYPES_COMPARTIMENT
5
6 COMPARTIMENTS
7 C1 : Cell ; C2 : Cell ; C3 : Cell ;
8 C4 : Cell ; C5 : Cell ; C6 : Cell ;

```

```
9 C7 : Cell ; Env : Ext
10 FIN_COMPARTIMENTS
11
12 ARETES
13 {C1,C2}; {C1,C3}; {C1,C4};
14 {C1,C5}; {C1,C6}; {C1,C7};
15 {C2,Env}; {C3,Env}; {C4,Env};
16 {C5,Env}; {C6,Env}; {C7,Env}
17 FIN_ARETES
```

3.6 Conclusion et perspectives

Le modeleur contient l'ensemble des opérations nécessaires pour construire les compartimentations cellulaires voulues, et permet ainsi d'extraire automatiquement tous les graphes d'échanges correspondants incluant l'ensemble des relations de voisinage et les contenus biochimiques initiaux.

La réutilisation du modeleur MOKA nous à permis de bénéficier d'une interface complète de visualisation des objets, appréciable quand on modélise des compartimentations complexes. Au contraire, nous n'avons pas intégré dans le modeleur bio-géométrique les nombreuses opérations de construction et modification des objets, afin de sécuriser la construction des compartiments et ainsi d'en garantir la cohérence. Ce qui dans l'absolu est bien, mais en pratique parfois contraignant.

Pour améliorer l'ergonomie du modeleur bio-géométrique, il conviendrait d'ajouter certaines opérations. En particulier les opérations de modification des compartiments comme des redimensionnements, des translations et des rotations. Toutes ces opérations devront préserver la cohérence du modèle bio-géométrique et donc intégrer une étape de vérification des contraintes géométriques de non chevauchement.

Notons que ces vérifications sont parfois complexes à réaliser en raison des erreurs d'arrondis sur les calculs géométriques. Ainsi deux compartiments proches peuvent être indûment détectés comme se chevauchant. C'est le cas notamment quand on construit des compartiments contiguës avant de les coudre. Les problèmes géométrique n'étant pas au cœur de cette thèse, nous avons simplement intégré une marge d'erreur à nos calculs.

Par ailleurs, nous avons volontairement limité la forme des compartiments à des parallélépipèdes rectangles pour simplifier les calculs géométriques, là encore car ils ne sont pas notre propos principal. Cependant cela donne des rendus cubiques peu réalistes et peu être trop contraignant pour modéliser certains assemblages complexes basés par exemple sur des hexagones. Il conviendrait donc d'enrichir les formes de départ des compartiments, par exemple en les généralisant à toute enveloppe convexe.

Chapitre 4

Langage de règles biochimiques orienté compartiments cellulaires : les règles génériques ou à variables de compartiments

Sommaire

4.1	Les règles génériques	84
4.1.1	Composantes d'une règle générique	86
4.1.2	Le modèle générique	93
4.2	Règles intermédiaires	95
4.2.1	Notions de substitution de termes	96
4.2.2	Les données d'illustration	100
4.2.3	La procédure de couplage règles génériques et graphe d'échanges	101
4.3	Traduction des règles intermédiaires	107
4.3.1	Traduction vers BIOCHAM	108
4.3.2	Traduction vers PATHWAY LOGIC	110
4.3.3	Exemple : traduction du modèle intermédiaire donné par <i>fichInterm3</i>	114
4.4	Implémentation	116
4.4.1	Le modèle générique	116
4.4.2	Implantation de l'instanciation et de la traduction	118

Dans ce chapitre, nous introduisons un langage dit de *règles génériques* qui nous permettra d'exprimer des règles de réactions biochimiques, en accord avec la nature et la répartition des compartiments où ont lieu les réactions considérées.

Comme annoncé précédemment, dans notre approche, la topologie du système biologique étudié et les règles biochimiques sont définies indépendamment : la topologie est fournie par un graphe d'échanges, en pratique extrait d'une 3-G-carte, garantissant le réalisme de la structure des compartiments considérée

tandis que les règles biochimiques sont exprimées abstraitement à partir de la seule nature des compartiments et de leur relation de voisinage. Ainsi, dans un premier temps, un ensemble de règles génériques pourra être instancié en différents ensembles de règles biochimiques, en fonction des graphes d'échanges considérés, qui jouent le rôle de contextes d'instanciation. Dans un second temps, les règles issues de l'instanciation d'un ensemble de règles génériques en fonction d'un graphe d'échanges pourront être interprétées, via un mécanisme de traduction, selon l'un ou l'autre des deux environnements considérés dans ce document : BIOCHAM et PATHWAY LOGIC. A partir donc d'un ensemble de règles génériques, sous réserve de nous donner un graphe d'échanges *pertinent* et un outil cible (BIOCHAM ou PATHWAY LOGIC), nous obtenons après instanciation et traduction un ensemble de règles directement exploitables.

Notre langage de règles génériques est donc « générique à deux niveaux » :

- le premier niveau est la généralité selon la structure topologique précise du système biologique considéré : il s'agit de substituer les variables de compartiment conformément à la structure topologique fournie par le graphe d'échanges associé au système biologique considéré ;
- le deuxième niveau concerne la généralité selon le langage cible utilisé pour exploiter l'ensemble des règles résultantes (par définition sans variables de compartiment) : l'outillage dédié du langage cible permettra à l'utilisateur d'analyser le système biologique étudié.

Remarque 4.1. *Dans cette section, lorsque nous parlons de généralité des règles, c'est à celle du premier niveau que nous nous référons.*

Les deux outillages considérés, BIOCHAM et PATHWAY LOGIC, étant fondés sur des paradigmes différents, les analyses issues de l'un ou l'autre, ne donneront pas nécessairement des résultats convergents. Néanmoins, nous pensons que notre approche offre ainsi un degré de flexibilité appréciable : elle permet à l'utilisateur d'expérimenter deux points de vue différents d'un même phénomène biologique dans une même structure topologique.

Le chapitre est organisé selon le plan suivant :

- La section 4.1 est consacrée à la présentation des *règles génériques*. Quelques éléments de syntaxe des règles, ainsi que le rôle des différents constituants des règles y sont détaillés.
- La section 4.2 décrit les mécanismes d'instanciation des règles génériques dans un graphe d'échanges en des règles dites *intermédiaires*. Par construction, ces règles intermédiaires reflètent la structure topologique décrite par le graphe d'échanges et sont exprimées dans un format proche des langages de règles des outils cibles BIOCHAM et PL. En effet, elles se présentent comme des règles biochimiques simples, sans variables de compartiments.
- la section 4.3 est consacrée à la présentation de la procédure de traduction vers les langages des outils cibles ;
- Enfin, la section 4.4 présente les éléments liés à l'implémentation.

4.1 Les règles génériques

Dans cette section, nous introduisons un langage à base de règles qui incorpore des réactions biochimiques relatives à la transformation, dégradation, synthèse des molécules et localise ces réactions dans certains compartiments.

La brique de base du langage est la règle dite générique encore appelée *règle à variables de compartiments*. Les règles génériques font intervenir des *molécules* et des *types de compartiment*, la généralité portant sur cette deuxième catégorie d'éléments. L'idée est d'exprimer à travers les règles génériques les comportements intrinsèques des types de compartiments biologiques, en faisant abstraction d'un quelconque système biologique. L'hypothèse sous-jacente est que les réactions biochimiques dépendent non seulement de la nature des compartiments concernés, mais aussi de la configuration explicite du système biologique sous étude selon ces compartiments. Plus précisément, deux intuitions essentielles militent en faveur d'un couplage entre règles biochimiques et structuration en compartiments. Il s'agit de :

- l'*identité moléculaire des compartiments biologiques* [?] qui suggère que, dans un système biologique donné, les compartiments de même type, de manière intrinsèque ou sous certaines conditions (notamment de voisinage avec d'autres compartiments), ont les mêmes comportements ;
- la *dépendance structuro-fonctionnelle* des compartiments biologiques qui impose :
 - qu'une règle de réaction faisant intervenir un compartiment de type T_1 donné, ne peut en aucun cas s'appliquer dans un système ne comportant pas de compartiment de type T_1 ;
 - qu'une règle de réaction faisant intervenir n compartiments C_1, \dots, C_n de types respectifs T_1, \dots, T_n n'est applicable dans un système comportant des compartiments de type T_1, \dots, T_n , que sous certaines conditions relatives aux configurations de voisinage des différents C_i .

Du fait de ces deux intuitions, une même règle générique appliquée à deux systèmes biologiques compartimentés différemment va donner lieu à deux ensembles différents de règles intermédiaires. Ces ensembles de règles sont induits par la topologie donnée par les graphes d'échanges.

Les avantages d'une telle approche sont les suivants :

- Les règles génériques permettent de s'affranchir de la prise en compte de la structure topologique des systèmes cellulaires auxquels elles seront rapportées. Cette structure est donnée par le graphe d'échanges tel que défini dans le chapitre 3.
- Le graphe d'échanges, sous réserve qu'il soit extrait d'une G-carte est cohérent et garantit ainsi que l'utilisation de la notion de compartiments au sein de règles biochimiques reflète une structure topologique réaliste des systèmes biologiques étudiés. Cet avantage correspond précisément à la motivation initiale du travail présenté dans ce manuscrit.
- L'ensemble des règles est concis. En effet, une règle générique est appelée à être instanciée autant de fois que nécessaire au regard du système biologique à l'origine de la construction du graphe d'échanges considéré. Par exemple, une règle générique de transport de molécules entre deux compartiments voisins typés de façon appropriée suffit pour capturer l'ensemble des cas particuliers de transports de molécules ayant lieu pour chaque voisinage de compartiments respectant le typage de la règle générique et apparaissant dans le graphe d'échanges. Le gain en taille dépendra clairement du graphe d'échanges considéré, tout particulièrement du fait que les motifs de voisinage privilégiés par les règles génériques s'y retrouvent souvent.

Les principales idées sous-jacentes à notre langage de règles génériques peuvent se résumer dans les points suivants :

- La structure sous-jacente d'une règle générique est celle usuelle d'une règle biochimique, avec essentiellement un membre gauche et un membre droit, chacun des membres comportant un certain nombre d'entités moléculaires.
- Chaque molécule ou structure moléculaire est localisée. En pratique, une variable de compartiment lui est attachée. Plus précisément, l'ensemble des molécules présentes dans un même compartiment est associé à une variable de compartiment.
- Une règle générique peut être instanciée sous la contrainte d'un graphe d'échanges. L'instanciation sera constituée de toutes les règles que l'on pourra construire en identifiant dans le graphe d'échanges

les configurations de compartiments conformes à celle attendue par la règle générique. Intuitivement, une règle est autorisée si les échanges biochimiques envisagés ont lieu dans le même compartiment ou dans des compartiments voisins.

- Une règle générique est éventuellement assortie de paramètres cinétiques dont les expressions peuvent impliquer des molécules et des variables de compartiment.
- Une règle générique peut contenir des préconditions portant sur les variables de compartiment présentes dans les membres gauche et/ou droit de la règle générique. Ces préconditions permettent de restreindre les configurations de voisinage légitimes pour construire les règles instanciées.

Nous souhaitons attirer l'attention du lecteur dès à présent sur un cas particulier : la plupart des langages de règles biochimiques ont la possibilité de considérer des réactions chimiques ne dépendant que des seules molécules en présence. Intuitivement, quel que soit le compartiment considéré, la règle s'applique, pour peu que le compartiment en question contienne les molécules du membre gauche. Ainsi la réaction peut avoir lieu quel que soit le compartiment de localisation des molécules. En raison de l'identité moléculaire des compartiments biologiques, l'existence de ce type de réactions suppose celle de molécules pouvant être contenues dans n'importe quel type de compartiment. Afin de ne pas être amené à écrire autant de règles qu'il y a de types de compartiment, nous allons distinguer dans la suite du document, selon leur degré de dépendance vis-à-vis des types de compartiment :

- les règles génériques à localisation *non sélective* qui sont totalement indépendantes des types de compartiment ;
- les règles génériques à localisation *sélective* dans lesquelles toutes les molécules sont nécessairement localisées dans des compartiments typés.

Nous prendrons soin d'avoir une syntaxe à la fois homogène et explicite pour repérer ces deux types de règles génériques.

4.1.1 Composantes d'une règle générique

Les règles génériques font essentiellement intervenir deux catégories d'éléments :

- La première catégorie est constituée des molécules ou complexes moléculaires en jeu dans les réactions biochimiques. En suivant la description fournie dans le chapitre 2, il s'agit de l'ensemble des constituants des langages de règles biochimiques. Nous en reprenons les principales facilités de manipulation. Les formes les plus simples de molécules sont dites *de base* et les autres sont formées à partir de deux opérations essentielles sur les molécules. Il s'agit de :
 - La *complexation* qui associe deux molécules pour en former une autre dite *complexe*. L'opération inverse de la complexation est la *décomplexation* qui dissocie les deux molécules d'un complexe ;
 - L'application d'une *modification de forme* à une molécule. On obtient une *forme modifiée* de la molécule concernée qui peut par exemple passer d'une forme inactive à une forme active. On peut citer comme exemple de modification de forme de molécule, la phosphorylation et son pendant la déphosphorylation, l'acétylisation, etc.
- Conformément aux arguments précédemment exposés dans ce chapitre, la seconde catégorie est celle des *variables de compartiment* qui permettent d'indiquer dans quel type de compartiment sont localisées les molécules en jeu dans la réaction.

Nous commençons par introduire quatre ensembles finis :

$$\text{CompartRG}, \text{MolsBase}, \text{Modifs}, \text{ParamsC}$$

qui seront systématiquement utilisés dans la suite pour dénoter respectivement :

- l'ensemble des identificateurs¹ de types de compartiments (quatre identificateurs sont associés à chaque type de compartiment pour prendre en compte les réactions localisées sur la face externe de la membrane, dans l'espace trans-membranaire, sur la face interne de la membrane et à l'intérieur du compartiment) ;
- L'ensemble des identificateurs de molécules de base (chaque molécule pouvant être assortie de ses sites de modification) manipulées ;
- l'ensemble des identificateurs de modifications de forme applicables aux molécules ;
- l'ensemble des identificateurs de paramètre cinétique.

L'ensemble *Mols* des molécules à considérer à partir de la donnée de *MolBase* et de *Modifs* est l'ensemble des expressions, défini comme le plus petit ensemble vérifiant :

- les éléments de *MolBase* sont des éléments de *Mols* ;
- si m_1 et m_2 sont deux éléments de *Mols*, alors $m_1 - m_2$ est un élément de *Mols* ;
- si m appartient à *Mols* et si mf appartient à *Modifs*, alors $(m : mf)$ est un élément de *Mols* ;
- si m appartient à *Mols* et si mf appartient à *Modifs*, alors $(m : mf < s_1, \dots, s_n >)$ est un élément de *Mols*

Dans la construction ci-dessus,

$$m_1 - m_2$$

représente la molécule obtenue par complexation de m_1 et de m_2 tandis que

$$(m : mf)$$

dénote la molécule m modifiée par la modification de forme mf et

$$(m : mf < s_1, \dots, s_n >)$$

la forme de m modifiée par la modification de forme de mf sur les sites de modification s_1 à s_n .

Nous signalons dès à présent que, pour prendre en compte la possibilité offerte par les outils cibles de définir des *schémas de règles biochimiques*, nous introduisons des *variables de molécules* comme des variables typées sur *Mols*. Il s'agit d'identificateurs introduits par « ? ».

Remarque 4.2. *Vocabulaire lié à l'utilisation de variables de molécules.*

1. *L'utilisation de variables de molécule dans une règle générique transforme celle-ci en un schéma de règle générique qui permet de générer des schémas de règles biochimiques. Dans la suite du document, nous utilisons plutôt l'expression « règle générique avec variables de molécule » au lieu de schémas de règles génériques.*
2. *Dans la suite de ce chapitre, nous utilisons « terme molécule » pour désigner soit un élément de *Mols*, soit une variable de molécule, soit un terme bien formé à partir de variables de molécule et d'éléments de *Mols*.*

En suivant les notations classiques de la logique du premier ordre typé, nous introduisons ci-dessous les quelques éléments syntaxiques relatifs à la manipulation des variables de compartiments :

Définition 4.1. *(Variable de compartiment).*

*Soit t un élément de *CompartRG*. Une variable de compartiment de type t sera notée $v : t$ où v est le nom de la variable.*

1. Les identificateurs sont représentés classiquement à l'aide de mots qui sont des suites de caractères alphanumériques commençant généralement par une lettre.

Exemple 4.1. (*Variable de compartiment*).

On donne $\text{CompartRG} = \{\text{cytoplasme}, \text{noyau}\}$.

x : cytoplasme et y : noyau sont deux exemples de variables de compartiment désignant respectivement une variable x de type cytoplasme et une variable y de type noyau.

Définition 4.2. (*Utilisation du mot « générique »*).

On considère l'ensemble CompartGE des types de compartiment. Toute expression comportant une variable de compartiment définie sur CompartGE sera qualifiée de générique.

Remarque 4.3. (*Compartiment générique*).

Pour plus de clarté, nous utilisons quelquefois dans la suite du document, « compartiment générique v », pour faire référence à la variable de compartiment v : t . Par exemple, on dira le contenu du compartiment générique v plutôt que le contenu de la variable de compartiment v .

Une règle à variable de compartiment prend la forme générale suivante :

$$\text{idrg} : [\text{PreC}] [\text{ParamC}] MG \Rightarrow MD [\text{CondVarMol}]$$

et permet de générer des règles de réactions biochimiques matérialisant pour un système biologique donné, le passage d'un état e_1 engendré à partir de l'état générique MG à un état e_2 engendré à partir de l'état générique MD , sous des conditions particulières de voisinage entre les valeurs prises par les variables de compartiment impliquées. Ces conditions de voisinage sont en partie données par la précondition de voisinage PreC .

Nous distinguons ainsi dans une règle générique deux parties :

- la partie *active* constituée de la description de la réaction biochimique, $MG \Rightarrow MD$, qui matérialise le changement d'état ;
- la partie *passive* comportant les autres composantes : la précondition de voisinage PreC , utilisée pour discriminer les règles générées, le paramètre cinétique ParamC qui permet de préciser le contexte d'application des règles générées et la condition sur les variables de molécules CondVarMol servant à contraindre les valeurs des variables de molécules apparaissant notamment dans la partie active.

Nous commençons par détailler la notion d'états génériques dont relèvent les parties MG et MD d'une règle générique. Les états génériques sont des modèles abstraits d'états de système biologique permettant de générer des états de système biologique. Ils consistent en des ensembles de termes molécules associés à différentes variables de compartiment. Les molécules présentes dans un même compartiment forment une solution.

Définition 4.3. (*Solution*).

Une solution S est un ensemble de termes molécules. Lorsque l'ensemble est vide, on parle de solution vide et on note $[\]$.

La solution non vide est de la forme $[\text{ensMol}]$ où ensMol est de la forme

$$m \text{ ou } m + \text{ensMol}$$

avec m un terme molécule.

Nous utilisons le symbole @ pour matérialiser l'association entre un compartiment et son contenu (solution), ce qui donne lieu à une *solution localisée*.

Définition 4.4. (*Solution localisée*).

Une solution localisée sur *CompartRG*, notée $S@v$ est la donnée d'une solution S et d'une variable de compartiment v sur *CompartRG*.

v est appelée variable de localisation de $S@v$.

Il est possible de donner une précision supplémentaire sur la localisation en faisant suivre la variable de compartiment de

$$extM, transM, intM, \text{ et } intC$$

mis entre parenthèses et correspondant respectivement à la face externe de la membrane, à la localisation trans-membranaire, à la face interne de la membrane ou à l'intérieur du compartiment. Dans ce cas, la solution localisée sera notée $S@v(loc)$, $loc \in \{extM, transM, intM, intC\}$.

Exemple 4.2. (*Solutions et solutions localisées*).

On considère :

- $Mols = \{m_1, m_2\}$;
- $Modifs = \{PHOS\}$;
- $CompartRG = \{t_1, t_2\}$.

Les expressions suivantes définies sur *Mols*, *Modifs* et *CompartRG*

$$[m_1 + m_1 - m_2]$$

$$[m_1 + m_1 - m_2]@v_1 : t_1$$

$$[m_1 + m_1 - m_2]@v_1 : t_1(intM)$$

correspondent respectivement à une solution composée des molécules m_1 et $m_1 - m_2$, la même solution localisée sur *CompartRG* dans un compartiment de type t_1 , la même solution localisée sur *CompartRG* sur la face interne d'un compartiment de type t_1 .

Les solutions localisées entrant dans la constitution d'un état générique doivent respecter certaines conditions sur les variables de compartiment et sur les ensembles de termes molécule :

- Une même variable de compartiment ne peut figurer qu'une seule fois dans l'expression d'un état générique. Cette option trouve sa raison d'être dans le fait que notre travail s'intéresse essentiellement aux compartiments cellulaires, ce qui nous fonde à privilégier une approche par les compartiments en décrivant le contenu qualitatif de chacun d'eux une seule fois, plutôt qu'une approche par les termes molécule.

Une telle option obéit à un souci de clarté et de concision dans l'écriture des états génériques mais permet également une meilleure lisibilité des états. Cependant, la motivation première de cette interdiction reste le souci d'alléger les traitements, en évitant de multiplier inutilement les variables de compartiment, toutes choses qui nécessiteraient des tests supplémentaires pour s'assurer qu'une même molécule n'est pas incluse plusieurs fois dans la même localisation.

- L'ensemble des solutions ne peut contenir à la fois une solution vide localisée et des solutions non vides localisées.

Ces deux conditions sont exprimées dans la définition 4.5 qui est celle d'état générique.

Définition 4.5. (*État générique*).

Un état générique sur CompartRG eg est soit un ensemble ayant comme seul élément la solution vide localisée sur CompartRG , soit un ensemble non vide de k solutions localisées sur CompartRG tel que toutes les solutions sont non vides et toutes les variables de compartiments sont disjointes deux à deux.

On appelle ensemble des variables de localisation de eg noté $\text{var}(eg)$, l'union des ensembles des variables de localisation des solutions localisées de eg :

$$\text{var}(eg) = \coprod_{S_i @ v_i : t_{v_i} \in eg} v_i : t_i$$

On appelle ensemble des types de définition de eg noté $\text{types}(eg)$, l'union des ensembles des types de compartiments de $\text{var}(eg)$:

$$\text{types}(eg) = \coprod_{v_i : t_{v_i} \in \text{var}(eg)} t_{v_i}$$

On note

$$S_1 @ v_1 : t_{v_1} \ \& \ \dots \ \& \ S_k @ v_k : t_{v_k}$$

l'état générique composé des solutions localisées $S_i @ v_i$ $i \in [1, k]$

Exemple 4.3. (*États génériques*).

En considérant les données de l'exemple 4.2,

$$[(M_1 - M_2)] @ v_1 : t_2 \text{ et}$$

$$[(M_1 : PHOS)] @ v_1 : t_1 \ \& \ [M_1 + M_2] @ v_2 : t_1$$

sont des états génériques composés respectivement d'une et de deux solution(s) localisée(s). Par contre

$$[(M_1 : PHOS)] @ v_1 : t_1 \ \& \ [M_1 + M_2] @ v_1 : t_1$$

$$[] @ v_1 : t_1 \ \& \ [M_1] @ v_2 : t_2$$

ne sont pas des états génériques parce que pour le premier cas, les variables de compartiment ne sont pas disjointes deux à deux et pour le second, la solution vide localisée n'est pas le seul élément de l'état générique.

Nous allons détailler les éléments de la partie passive des règles génériques en commençant par les paramètres cinétiques ParamC qui constituent la première catégorie de composantes optionnelles d'une règle générique. Il s'agit d'expressions de formes variées allant des réels à des expressions très complexes impliquant généralement les termes molécule apparaissant dans MG . L'expression de nos paramètres cinétiques s'inspire fortement de BIOCHAM qui est le seul des deux outils cibles à prendre en compte ces informations. On y rencontre les paramètres cinétiques conditionnels et les paramètres cinétiques non conditionnels.

Définition 4.6. (*Paramètres cinétiques simples pour une règle générique*).

Soit rg une règle générique. Un paramètre cinétique simple pour rg est défini inductivement par :

1. tout élément de \mathbf{R} est un paramètre cinétique ;

2. toute variable typée sur \mathbf{R} est un paramètre cinétique ;
3. si $S_i@v_i : t_{v_i}$ est une solution localisée de MG , alors $[[m_1 + \dots + m_k]@v_i : t_{v_i}]$ telle que $m_j \in S_i$ ($j \in [1, \dots, k]$) est un paramètre cinétique simple pour rg et désigne le produit des concentrations des termes molécule m_j contenus dans le compartiment générique $v : t_i$;
4. si k_1 et k_2 sont des paramètres cinétiques simples pour rg et n un entier, alors

$$k_1 * k_2, k_1 + k_2, k_1 - k_2, k_1/k_2$$

$$(k_1), \log(k_1), \exp(k_1), \cos(k_1), \sin(k_1), \text{frac}(k_1), MA(k_1)$$

$$MM(k_1, k_2), H(k_1, k_2, n)$$

sont des paramètres cinétiques simples pour rg ;

Définition 4.7. (Paramètres cinétiques conditionnels pour une règle générique).

Soit rg une règle générique. Un paramètre cinétique conditionnel est défini par la donnée :

1. d'une expression booléenne définie sur des paramètres cinétiques simples pour rg et utilisant les opérateurs $<, =, >$ et le connecteur logique *and* ;
2. de deux paramètres cinétiques simples pour rg , k_1 et k_2 .

les paramètres cinétiques conditionnels sont notés : *if condition then k_1 else k_2* où *condition* est l'expression booléenne et k_1, k_2 les deux paramètres cinétiques simples.

En considérant les données de l'exemple 4.2,

$$[[m_1]@v_1 : t_1]$$

$$|1.2| \text{ et } |k_1|$$

$$|if k_1 > 2 \text{ then } [m_1 + m_2]@v_1 : t_1 \text{ else } 1|$$

sont des exemples de paramètres cinétiques.

Nous avons signalé au niveau de la section 2.1 que dans certains cas, les concentrations en molécules des compartiments voisins de ceux impliqués dans la réaction influencent celle-ci : les paramètres cinétiques des règles générées doivent alors porter sur les compartiments voisins. Pour prendre cela en compte de façon concise dans les règles génériques, nous introduisons la fonction *Neighb* qui prend comme paramètre une variable de compartiment et désigne l'ensemble des compartiments voisins de celui passé en paramètre. Par exemple

$$Neighb(v_1 : t_1)$$

est l'ensemble des compartiments voisins du compartiment générique v_1 .

$$[[m_1]@Neighb(v_1 : t_1)]$$

représente les concentrations de la molécule m_1 dans les compartiments voisins de v_1 .

Les préconditions de voisinage constituent l'élément original de notre langage de règles. Ce sont des paires de variables de compartiment apparaissant dans la partie active de la règle générique. Chaque paire exprime une condition de voisinage entre ses deux membres. Les préconditions de voisinage sont utilisées dans la procédure de couplage entre règles génériques et graphe d'échanges pour la discrimination des

règles générées pour ne retenir pour traduction que celles pour lesquelles la condition sur chaque paire est respectée.

Étant donné qu'elles n'engagent que des variables de compartiment impliquées dans la partie active d'une règle générique, elles n'ont pas de raison d'être dans les règles n'impliquant qu'une seule variable de compartiment. Il s'agit de ce fait d'une composante optionnelle des règles génériques.

Définition 4.8. (*Précondition de voisinage*).

Une *précondition de voisinage* $PreC$ notée

$$PreC : \{\{v_1, v'_1\}, \dots, \{v_n, v'_n\}\}$$

est un ensemble de n paires $\{v_i, v'_i\}$ où v_i et v'_i ($i \in [1, n]$) sont des variables de compartiment distinctes deux à deux.

La dernière composante optionnelle des règles génériques, $CondVMol$ n'est utilisée que lorsque la partie active comporte des variables de molécule. Il s'agit d'une expression introduite par le mot-clé *if*, qui contraint les valeurs de chaque variable de molécule.

Définition 4.9. (*Conditions sur les variables de molécules*).

Soit rg une règle générique comportant les variables de compartiment $vmol_1 \dots vmol_k$. Une condition sur les variables de molécules de rg , $CondVMol$ est la donnée pour chaque $vmol_i$ $i \in [1, k]$, d'un sous-ensemble de $Mols$.

$CondVMol$ est notée

$$if (vmol_1 \text{ in } \{m_{vmol_{1_1}}, \dots, m_{vmol_{1_{n_1}}}\} \text{ and } \dots \text{ and } vmol_k \text{ in } \{m_{vmol_{k_1}}, \dots, m_{vmol_{k_{n_k}}}\})$$

où $m_{vmol_{i_j}}$ pour ($i \in \{1, \dots, k\}$ et ($j \in \{1, \dots, n_i\}$)) sont des éléments de $Mols$.

Toutes ses parties constitutives ayant été détaillées, nous donnons à présent la définition d'une règle générique.

Définition 4.10. (*Règle générique*).

Une règle générique rg sur $CompartRG$ est l'expression

$$idrg : [PreC] [ParamC] MG \Rightarrow MD [CondVMol]$$

où :

- MG et MD sont des états génériques sur $CompartRG$;
- $ParamC$ est un paramètre cinétique tel que

$$\forall v_i \in var(ParamC), v_i \in (var(MG) \cup var(MD)) ;$$

- $PreC$ est une précondition de voisinage telle que

$$\forall \{v_i, v'_i\} \in PreC, v_i, v'_i \in (var(MG) \cup var(MD)) ;$$

- $CondVMol$ est une condition portant sur les variables de molécules des états génériques.

- *idrg* est un identificateur de règle générique qui est par défaut *rg* ou *crg* selon que l'on est en présence d'une règle générique ou d'une règle générique avec variable de molécules aux valeurs contraintes par *CondVMol*.

On appelle types de définition de *rg* noté $types(rg)$, l'ensemble $types(MG) \cup types(MD)$ et variables de localisation de *rg* noté $var(rg)$, l'ensemble $var(MG) \cup var(MD)$

On considère les données de l'exemple 4.2; les règles du listing suivant sont des exemples de règles génériques.

```

rg1 : [m1]@v1 : t1 => []@v1 : t1.
rg2 : [PreC : {{v1 : t2, v2 : t2}}] [m2]@v1 : t2 => [m2]@v2 : t2.
rg3 : |[m2]@v1 : t2| [m2]@v1 : t2 & [m1 + m2]@v2 : t1 => [m1 - m2]@v2 : t1 & [m2]@v1 : t2.
crg1 : [?vmol]@t2 => [(?vmol : PHOS)]@t2 if (?vmol in {m1, m2})

```

rg1, *rg2*, *rg3* et *rg4* matérialisent respectivement :

- la dégradation de m_1 dans un compartiment de type t_1 ;
- le transport de m_2 d'un compartiment de type t_2 vers un autre de même type voisin ;
- la complexation de m_1 et m_2 dans un compartiment de type t_1 , catalysée par m_2 dans un compartiment de type t_2 , réaction dont la vitesse varie en fonction de la concentration de m_2 dans le compartiment de type t_2 ;
- la phosphorylation d'une molécule dans un compartiment de type t_2 , cette molécule pouvant être m_1 ou m_2 .

4.1.2 Le modèle générique

Les modèles génériques sont essentiellement composés de règles génériques. Avant d'en donner une définition mathématique, nous attirons l'attention du lecteur sur des types de compartiment particuliers que nous introduisons pour diverses raisons. En effet, l'obligation de localiser chaque molécule apparaissant dans une règle générique ainsi que le souci de concision dans l'écriture des règles génériques, nous amènent à introduire le type de compartiment particulier *AnyW* qui permettra d'écrire une seule fois les règles à localisation non sélective. Les variables de compartiment de ce type pourront être substituées par n'importe quel compartiment du graphe d'échanges, indépendamment du type de ce compartiment. En général, les réactions donnant lieu à des règles à localisation non sélective n'engagent qu'un unique compartiment d'où *AnyW* ne sera utilisé que pour des règles portant sur une seule variable de compartiment.

Par ailleurs, nous introduisons le type de compartiment t_{Env} comme type par défaut (l'utilisateur peut en effet typer l'environnement) pour l'environnement du système sous étude. Ce type de compartiment ne peut être utilisé que comme type de la variable de compartiment *Env* (qui est un identificateur réservé que nous introduisons également). En d'autres termes, nous n'avons pas des variables de compartiment de forme

$$v_1 : t_{Env}.$$

Par contre, nous pouvons avoir

$$Env : t_1,$$

t_1 étant le type défini par l'utilisateur pour l'environnement.

Définition 4.11. (Modèle générique).

Un modèle générique sur *CompartRG* est un 5-uplet

$$(CompartRG, MolsBase, Modifs, ParamsC, RGs)$$

tel que :

- *CompartRG* est l'ensemble de types de compartiments ;
- *MolsBase* l'ensemble des molécules de base apparaissant dans le modèle ;
- *Modifs* est l'ensemble de modifications de forme de molécules ;
- *ParamsC* est l'ensemble des identificateurs de paramètres cinétiques ;
- *RGs* est l'ensemble de règles génériques sur *CompartRG* tel que :
 - si $AnyW \in CompartRG$, $\exists rg \in RGs \mid AnyW \in types(rg)$;
 - si $AnyW \notin CompartRG$, $\forall tc \in CompartRG$, $\exists rg \in RGs \mid tc \in types(rg)$.

La condition sur *RGs* et *CompartRG* indique que si le type *AnyW* fait partie de *CompartRG*, c'est qu'il existe au moins une règle générique qui porte sur ce type. Dans ce cas, quels que soit les autres éléments de *CompartRG*, ils doivent être pris en compte même s'ils n'apparaissent pas dans une règle générique. Par contre, lorsque *AnyW* ne fait pas partie de *CompartRG* (aucune règle de réaction ne porte sur *AnyW*), tous les types de compartiments déclarés dans *CompartRG* doivent être utilisés par au moins une règle générique.

Exemple 4.4. (Modèle générique).

On considère les modèles génériques :

$MGen_1 = (CompartRG_1, MolsBase_1, Modifs_1, ParamsC_1, RGs_1)$ avec :

- $CompartRG_1 = \{AnyW, T_1, T_2\}$;
- $MolsBase_1 = \{M_1\}$;
- $Modifs_1 = \emptyset$;
- $ParamsC_1 = \emptyset$;
- $RGs_1 = \{rg_1 : [M_1]@AnyW \Rightarrow []@AnyW.\}$

$MGen_2 = (CompartRG_2, MolsBase_2, Modifs_2, ParamsC_2, RGs_2)$ avec :

- $CompartRG_2 = \{T_1, T_2\}$;
- $MolsBase_2 = \{M_1\}$;
- $Modifs_2 = \emptyset$;
- $ParamsC_2 = \emptyset$;
- $RGs_2 = \{rg_1 : [M_1]@T_1 \Rightarrow []@T_1.\}$

$MGen_1$ est correct du point de vue de la définition car l'existence de rg_1 portant sur *AnyW* fait que $MGen_1$ est équivalent à

$$MGen'_1 = (CompartRG'_1, MolsBase'_1, Modifs'_1, ParamsC'_1, RGs'_1)$$

avec :

- $CompartRG'_1 = \{T_1, T_2\}$;
- $MolsBase'_1 = \{M_1\}$;

- $Modifs'_1 = \emptyset$;
- $ParamsC'_1 = \emptyset$;
- $RGS'_1 = \{rg_1 : [M_1]@T_1 \Rightarrow []@T_1 \text{ ., } rg_2 : [M_1]@T_2 \Rightarrow []@T_2 \text{ .}\}$

Par contre $MGen_2$ n'est pas correct car le type de compartiment T_2 est déclaré sans être utilisé.

Les règles génériques données au titre de l'illustration de notre approche de couplage (listing 2.4) peuvent s'exprimer par les règles du listing 4.1 suivant.

Listing 4.1 – Modèle générique pour les règles du listing 2.4 - page 30 -

```

1 BEGIN_TYPEC C; E; N; V END_TYPEC
2 BEGIN_MOLSB M1; M2 END_MOLSB
3 BEGIN_MODIFS PHOS END_MODIFS
4 BEGIN_REGEN
5 rg_1 : [PreC : {{var_1, var_2}}] [M2]@var_1:V => [M2]@var_2:C.
6 rg_2 : [M1+M2]@C => [M1-M2]@C.
7 rg_3 : [PreC : {{var_1, var_2}}] [M1-M2]@var_1:C => [M1-M2]@var_2:N.
8 rg_4 : [M1+M2]@E => [(M2 : PHOS)+M1]@E.
9 rg_5 : [PreC : {{V, C}, {C, N}}] [M1]@V & [M2]@C => [M1-M2]@N.
10 rg_6 : [PreC : {{V, C}, {V, N}, {C, N}}] [M1]@V & [M2]@C => [M1-M2]@N.
11 END_REGEN

```

Ce listing introduit quelques éléments de syntaxe concrète des modèles génériques. La syntaxe intégrale sera donnée par l'annexe A.

4.2 Règles intermédiaires

Les *règles intermédiaires*, comme on peut le lire sur la figure 2.1, sont des règles de réactions biochimiques issues de la procédure de couplage règles génériques/graphes d'échanges. Ce couplage procède d'une instantiation des règles génériques en fonction des contraintes venant du graphe d'échanges, ces contraintes étant les types des compartiments du graphe et les relations de voisinage entre ces compartiments. En tant qu'instances définies sur les compartiments d'un graphe d'échanges, les règles intermédiaires sont des règles de réactions biochimiques impliquant des compartiments *concrets*. Notre objectif, nous le rappelons, étant d'avoir un modèle dans lequel toutes les règles de réaction sont compatibles avec les voisinages entre les compartiments impliqués, les instances de règles obtenues doivent être discriminées pour ne retenir que celles qui satisfont les conditions particulières de voisinage entre les compartiments impliqués, lesquelles conditions sont en partie données par la composante *PreC* des règles génériques.

Cette section a pour objet de détailler les deux étapes (calcul puis sélection des instances de règles génériques) de la procédure de couplage entre règles génériques et graphes d'échanges qui permet de passer d'un modèle générique à un *modèle intermédiaire*. Un exemple d'un tel modèle, obtenu à partir des règles génériques du listing 4.1 et du graphe d'échanges de la figure 4.1 est donné par le listing 4.2. On y constate en particulier le remplacement des variables de compartiment par des noms de compartiments issus du graphe d'échanges. Par ailleurs, on note l'absence de la composante *PreC* et la modification des identificateurs de règles, toutes choses qui confèrent aux règles intermédiaires la forme générale suivante :

$$idri : [ParamC] MG \Rightarrow MD [CondVMol].$$

La présentation de la procédure va se fonder sur des exemples donnés à titre d'illustration pour susciter les commentaires utiles. Mais auparavant, étant donné que l'instanciation procède d'une substitution de termes, nous faisons en 4.2.1 quelques rappels sur cette notion et y introduisons la notion de substitution dans un graphe d'échanges. Avant de détailler en 4.2.3 la procédure de couplage entre les règles génériques et les graphes d'échanges, nous faisons à travers le paragraphe 4.2.2, le point sur les données d'illustration choisies qui consistent en un modèle générique et quelques graphes d'échanges simples.

Listing 4.2 – Modèle intermédiaire pour les règles du listing 4.1 en fonction du graphe de la figure 4.1

```

1 BEGIN_MOLSB M1; M2 END_MOLSB
2 BEGIN_REGINTERM
3 ri_1.1 : [M2]@C1 => [M2]@C1.
4 ri_2.1 : [M1+M2]@C1 => [M1-M2]@C1.
5 ri_2.2 : [M1+M2]@C2 => [M1-M2]@C2.
6 ri_3.1 : [M1-M2]@C1 => [M1-M2]@N1.
7 ri_3.2 : [M1-M2]@C2 => [M1-M2]@N2.
8 ri_5.1 : [M1]@V1 & [M2]@C1 => [M1-M2]@N1.
9 END_REGINTERM
    
```

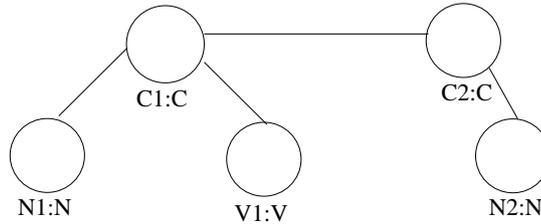


FIGURE 4.1 – Graphe d'échanges

4.2.1 Notions de substitution de termes

Dans cette section, nous rappelons quelques notions relatives à la substitution de termes, qui est une fonction σ d'un ensemble de variables typées, vers un ensemble de termes, respectant le typage. On représente une telle fonction par l'ensemble des couples noté

$$\{x_1 \setminus t_1, \dots, x_n \setminus t_n\} \text{ tels que } x_i \neq t_i$$

Avant de donner la définition mathématique des substitutions de termes, nous donnons celle d'*algèbre de termes du premier ordre sortée* qui décrit l'ensemble des termes à partir d'une *signature* et d'un ensemble de variables typées sur les *sortes* de la signature.

Définition 4.12. (*Algèbre de termes du premier ordre*).

On considère :

- Une signature Σ qui est la donnée
- d'un ensemble S_Σ de n types (sortes) de données : $S_\Sigma = \{s_1, \dots, s_n\}$;
- d'un ensemble Op_Σ de m opérations munies chacune d'un profil : $Op_\Sigma = \{op_1, \dots, op_m\}$, tel que $op_i = s_{i_1} \dots s_{i_{p+1}}$ (avec $1 \leq i \leq m$) est d'arité p ;

- Un ensemble X_Σ de variables typées sur S_Σ noté $X_\Sigma = \{x_1 : s_{x_1}, \dots, x_{nbv} : s_{x_{nbv}}\}$ où s_{x_i} est le type de la variable x_i

L'algèbre des termes du premier ordre, notée $\mathcal{A} = T(\Sigma, X)$ se définit par induction comme suit :

- Toute variable $x \in X_\Sigma$ est un terme.
- Toute opération $op_i \in Op_\Sigma$ d'arité 0 notée $op_i : \rightarrow s_{i1}$ est un terme.
- Pour toute opération $op_j \in Op_\Sigma$ d'arité $p > 0$, notée $op_j : s_{j1} \dots s_{jp} \rightarrow s_{jp+1}$, si $t_1 \dots t_p$ sont des termes tels que $\forall k \in [1, \dots, p]$, t_k est de même type que s_{jk} alors $op_j(t_1, \dots, t_p)$ est un terme.

Exemple 4.5. (Algèbre des termes de premier ordre).

Soient :

- une signature $\Sigma = (S, Op)$ avec :

$S = \{\text{entier}, \text{booléen}\}$ et $Op = \{0, \text{succ}, +, *, \text{true}, \text{false}, \text{estNul}\}$ tel que :

- $0 : \rightarrow \text{entier}$
- $\text{true} : \rightarrow \text{booléen}$
- $\text{false} : \rightarrow \text{booléen}$
- $\text{succ} : \text{entier} \rightarrow \text{entier}$
- $+$: $\text{entier entier} \rightarrow \text{entier}$
- $*$: $\text{entier entier} \rightarrow \text{entier}$
- $\text{estNul} : \text{entier} \rightarrow \text{booléen}$

- un ensemble de variables $X = \{x : \text{entier}, y : \text{entier}, z : \text{entier}\}$

Les expressions suivantes

$$\text{true}, 0, \text{succ}(x), +(\text{succ}(0), +(x, \text{succ}(y))), \text{estNul}(\text{succ}(0)), *(+(x, y), z)$$

sont des exemples de termes de l'algèbre des termes de premier ordre $\mathcal{A} = T(\Sigma, X)$.

Définition 4.13. (Substitution).

Soit $\mathcal{A} = T(\Sigma, X)$ une algèbre de termes du premier ordre sortée.

Une substitution σ définie sur \mathcal{A} est une fonction de X dans \mathcal{A} , telle que $\forall x_i : s_{x_i} \in X$, $\sigma(x_i)$ est de type s_{x_i} .

L'ensemble des $x : s_x \in X$ tels que $\sigma(x) \neq x$ est appelé domaine de σ et noté $\text{dom}(\sigma)$.

On appelle image de σ notée $\text{im}(\sigma)$ l'ensemble $\coprod_{x \in \text{dom}(\sigma)} \sigma(x)$.

Notons $BV(x)$ l'ensemble des variables apparaissant dans le terme x ; le codomaine de σ noté $\text{codom}(\sigma)$ est l'ensemble $\coprod_{y \in \text{im}(\sigma)} BV(y)$.

Une substitution σ est dite « close » lorsqu'elle n'introduit aucune variable ($\text{codom}(\sigma) = \emptyset$).

Notation 4.14. On note $\sigma_{(\mathcal{A})} = \coprod_{x \in \text{dom}(\sigma)} \{x \setminus \sigma(x)\}$ la substitution σ défini sur l'algèbre des termes de premier ordre sortée \mathcal{A}

En considérant les données de l'exemple 4.5

$$\sigma_{(\mathcal{A})} = \{x \setminus 0\} \text{ et } \sigma'_{(\mathcal{A})} = \{x \setminus \text{succ}(y)\}$$

	Domaine	Image	Codomaine
$\sigma_{(\mathcal{A})}$	$\{x\}$	$\{0\}$	\emptyset
$\sigma'_{(\mathcal{A})}$	$\{x\}$	$\text{succ}(y)$	$\{y\}$

 TABLE 4.1 – Domaine, image et codomaine des substitutions $\sigma_{(\mathcal{A})}$ et $\sigma'_{(\mathcal{A})}$.

sont des exemples de substitutions définies sur \mathcal{A} , dont les principales caractéristiques sont données par le tableau 4.1.

L'application d'une substitution σ à un terme t se fait en remplaçant simultanément dans t , toutes les variables x_i de σ par les $\sigma(x_i)$.

En considérant l'algèbre donnée dans l'exemple 4.5 et en appliquant au terme *(+(x, y), z) une substitution $\sigma = \{z \setminus \text{*(y, x)}\}$ on obtient le terme $t' = \text{*(+(x, y), *(x, y))}$

Après avoir abordé la substitution de termes dans un cadre général sorté, nous introduisons la notion de *substitution dans un graphe d'échanges* et nous définissons par la suite d'autres notions supplémentaires que nous utilisons dans notre procédure de couplage entre les règles génériques et les graphes d'échanges.

Définition 4.15. (*Signature issue d'un graphe d'échanges*).

Soit $GE = (V_{GE}, E_{GE}, l_{V_{GE}})$ un graphe d'échanges sur CompartGE .

On désigne par $\Sigma_{GE} = (\text{CompartGE}, OP_{GE})$ la signature issue de GE telle que

$$OP_{GE} = \coprod_{v:t_v \in V_{GE}} v \text{ :} \rightarrow t_v$$

Par exemple en considérant le graphe d'échanges G_1 de la figure 2.2(b), on a

$$\begin{aligned} \Sigma_{G_1} &= (\text{CompartGE}_{G_1}, OP_{G_1}) = \\ &(\{C, N, V\}, \{C1 \text{ :} \rightarrow C, C2 \text{ :} \rightarrow C, N1 \text{ :} \rightarrow N, N2 \text{ :} \rightarrow N, V1 \text{ :} \rightarrow V\}) \end{aligned}$$

Définition 4.16. (*Substitution dans un graphe d'échanges*).

Soient :

- GE un graphe d'échanges sur CompartGE ;
- X un ensemble de variables typées sur CompartGE .

Une substitution dans GE pour X , notée $\sigma_{(GE,X)}$ est une substitution close définie sur $T(\Sigma_{GE}, X)$ ($\sigma_{(GE,X)} : X \rightarrow OP_{GE}$).

Une substitution $\sigma_{(GE,X)}$ dans un graphe d'échanges GE pour un ensemble de variables X définit un sous-graphe qui est le sous-graphe de GE engendré par $\text{im}(\sigma_{(GE,X)})$.

Exemple 4.6. (*Substitution dans un graphe d'échanges*).

En considérant le graphe d'échanges G_1 (figure 2.2(b)) et l'ensemble

$$X = \{v_1 : C, v_2 : C, v_3 : N\}$$

de variables typées sur CompartGE_{G_1} , on a

$$\sigma_{(G_1, X)} = \{v_1 \setminus C1, v_2 \setminus C1\},$$

$$\sigma'_{(G_1, X)} = \{v_1 \setminus C1, v_2 \setminus C2, v_3 = N1\}$$

qui sont des exemples de substitutions dans le graphe d'échanges G_1 pour X .

Nous définissons dans ce qui suit les notions de *restriction d'une substitution à un ensemble de variables*, de *substitution connexe* et de *substitutions voisines* qui sont toutes relatives aux substitutions dans un graphe d'échanges et qui seront utilisées au niveau du filtrage des instances de règles génériques.

Définition 4.17. (*Restriction d'une substitution à un ensemble de variables*).

Soient

- GE un graphe d'échanges sur CompartGE ;
- $X = \{x_1, \dots, x_n\}$ un ensemble de variables typées sur CompartGE et $X' = \{x'_1, \dots, x'_m\} \subseteq X$;
- $\sigma_{(GE, X)}$ une substitution dans GE pour X .

La restriction de $\sigma_{(GE, X)}$ à X' est la substitution notée $\sigma_{(GE, X)|_{X'}}$ telle que

$$\forall i, 1 \leq i \leq m, \forall j, 1 \leq j \leq n, \text{ si } x'_i = x_j, \text{ alors } \sigma_{(GE, X)|_{X'}}(x'_i) = \sigma_{(GE, X)}(x_j)$$

.

Par exemple, en considérant les données de l'exemple 4.6 et un ensemble $X' = \{v_3\}$ de variables typées sur CompartGE_{G_1} , on a

$$\sigma_{(G_1, X)|_{X'}} = \emptyset$$

$$\sigma'_{(G_1, X)|_{X'}} = \{v_3 = N1\}$$

ce qui nous permet de dire qu'une substitution $\sigma_{(GE, X)|_{X'}}$ peut ou non définir un sous-graphe du sous-graphe engendré par $\sigma_{(GE, X)}$.

Définition 4.18. (*Substitution connexe*).

Soient

- GE un graphe d'échanges sur CompartGE ;
- X un ensemble de variables typées sur CompartGE ;
- $\sigma_{(GE, X)}$ une substitution dans GE pour X .

$\sigma_{(GE, X)}$ est dite connexe si le sous-graphe engendré par $\text{im}(\sigma_{(GE, X)})$ est connexe.

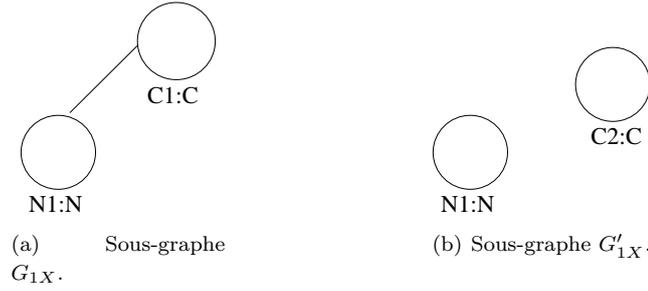
En considérant par exemple le graphe d'échanges G_1 (figure 2.2(b)) et l'ensemble $X = \{v_1 : C, v_2 : N\}$ de variables typées sur CompartGE_{G_1} , les substitutions

$$\sigma_{(G_1, X)} = \{v_1 \setminus C1, v_2 \setminus N1\} \text{ et } \sigma'_{(G_1, X)} = \{v_1 \setminus C2, v_2 \setminus N1\}$$

sont respectivement connexe et non connexe car donnant lieu respectivement aux sous-graphes G_{1X} (figure 4.2(a)) et G'_{1X} (figure 4.2(b)) de la figure 4.2 qui sont respectivement connexe et non connexe.

Définition 4.19. (*Substitutions voisines*).

Soient


 FIGURE 4.2 – Deux sous-graphes de G_1 engendrés respectivement par $im(\sigma_{(GE_1, X)})$ et $im(\sigma'_{(GE_1, X)})$.

- GE un graphe d'échanges sur $CompartGE$;
 - $X = \{x_1, \dots, x_n\}$ et $X' = \{x'_1, \dots, x'_m\}$ deux ensembles de variables typées sur $CompartGE$;
 - $\sigma_{(GE, X)}$ et $\sigma'_{(GE, X')}$ deux substitutions dans GE pour respectivement X et X' .
- On dit que $\sigma'_{(GE, X')}$ est voisine à $\sigma_{(GE, X)}$ si :
- $\forall x \in X \cap X', \sigma_{(GE, X)}(x) = \sigma'_{(GE, X')}(x)$
 - $\forall x' \in X' \setminus X, \exists v \in im(\sigma_{(GE, X)}), \{ \sigma'_{(GE, X')}(x'), v \} \in E_{GE}$.

on considère le graphe d'échanges G_1 (figure 2.2(b)) et les ensembles $X = \{v_1 : C, v_2 : N\}$ et $X' = \{v_3 : C, v_2 : N\}$ de variables typées sur $CompartGE_{G_1}$. La substitution

$$\sigma_{(G_1, X')} = \{v_3 \setminus C2, v_2 \setminus N1\}$$

est voisine à la substitution

$$\sigma_{(G_1, X)} = \{v_1 \setminus C1, v_2 \setminus N1\}$$

car $\sigma_{(G_1, X')}(v_2) = N1$ et $\{C1, C2\} \in E_{G_1}$.

4.2.2 Les données d'illustration

Nous fixons dans la présente section les éléments sur lesquels nous nous appuyons pour présenter la procédure de couplage entre règles génériques et graphes d'échanges. Nous tenons à préciser que ces éléments ont été choisis non parce qu'ils sont représentatifs de cas biologiques réels mais plutôt parce qu'ils permettront d'aborder toutes les particularités (subtilités) des mécanismes de substitutions et/ou de filtrage. Il s'agit des quatre graphes d'échanges donnés par la figure 4.3 et du modèle générique $MGen_1$ dont la description est donnée par le fichier *fichrg1*.

Listing 4.3 – Modèle générique pour illustration du couplage

```

1 BEGIN_TYPEC AnyW; Env; T1; T2 END_TYPEC .
2 BEGIN_MOLSB M1; M2; M3 END_MOLSB.
3 BEGIN_MODIF PHOS END_MODIF.
4 BEGIN_PARAM k1(0.1) END_PARAM.
5
6 BEGIN_REGEN
7 rg1 : [M1]@T1 => []@T1 .
8 rg2 : [M2]@T2 => [M2]@v1:T2 .
    
```

```

9  rg3 : [] @Env => [M1]@Env .
10 rg4 : [(M1:PHOS)]@T1 & [M2]@T2 => [M3]@T2 .
11 rg5 : [M2]@T1 => [M1]@T2 & [M2]@T1 .
12 rg6 : [PreC : (v1,v2)] [M1]@Env & [M2]@v1:T1 => [M2]v2:T1 .
13 rg7 : [M1]@AnyW => []@AnyW .
14 rg8 : | if (k1*[M1]@Neighb(v1) > 1) then 0.5 else 0.7| [M2+M2]@v1:t1 => [M1-M2]@v1 .
15 END_REGEN

```

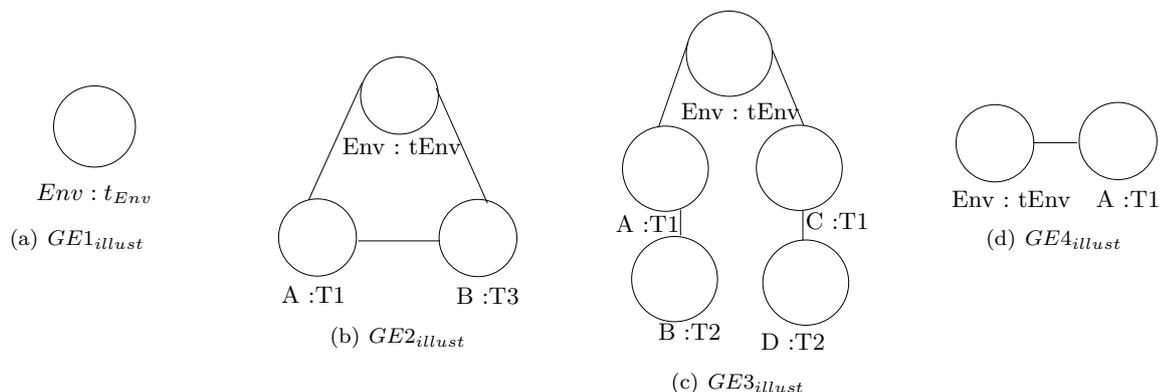


FIGURE 4.3 – Graphes d'échanges pour illustration de la procédure de couplage.

$MGen_1$ est défini sur trois molécules de base $M1$, $M2$ et $M3$ et sur deux types de compartiments déclarés $T1$ et $T2$ en plus de Env et $AnyW$. Les règles génériques $rg_{-(1)}, \dots, rg_{-(8)}$ qu'il contient, font intervenir respectivement 1, 2, 1, 2, 3, 1, 1 variable(s) de compartiments. Le modèle comporte également une déclaration d'une opération de modification de forme $PHOS$ et une déclaration d'un identificateur de paramètre cinétique $k1$ initialisé à 1.

4.2.3 La procédure de couplage règles génériques et graphe d'échanges

Nous présentons dans cette section les deux composantes de la procédure de couplage entre un ensemble de règles génériques et un graphe d'échanges. La phase d'instanciation consistant en l'application de substitutions, le paragraphe 4.2.3.1 va s'atteler à caractériser les substitutions permettant d'obtenir des instances de règles génériques intrinsèquement (respectant notamment le nombre de variables de compartiments) correctes. Nous qualifions ces substitutions de *substitutions éligibles*. La phase de sélection va permettre de retenir parmi les substitutions éligibles, celles dites *validées par les voisinages du graphe d'échanges* dont l'application donne les règles intermédiaires.

4.2.3.1 Calcul des substitutions éligibles

On considère un modèle générique $MGen = (CompartRG, MolsBase, Modifs, ParamsC, RGs)$ et un graphe d'échanges GE défini sur $CompartGE$. Pour effectuer les instanciations des règles génériques de RGs dans le graphe d'échanges GE , il convient dans un premier temps de *réajuster le cadre d'instanciation*. Ce traitement consiste à calculer et à considérer comme cadre d'instanciation, le sous-graphe de GE qui contient uniquement tous les compartiments dont les types se retrouvent dans $CompartRG$. La

principale raison d'une telle disposition est d'éviter que ne soient instanciées les règles comportant $AnyW$ en remplaçant $AnyW$ par des types de compartiment non déclarés dans $CompartRG$. En considérant par exemple le graphe d'échanges $GE2_{illustr}$ (figure 4.3(b)) et la règle générique $rg_{(7)}$ du listing 4.3, on aurait en l'absence de réajustement du cadre d'instanciation, une règle intermédiaire $[M1]@B \Rightarrow []@B$.

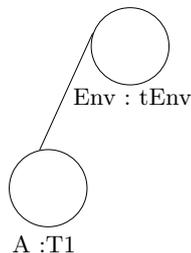
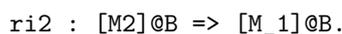


FIGURE 4.4 – Graphe d'échanges $GE2_{illustr}$ après réajustement.

Pour les données d'illustration, le seul ajustement à faire concerne le graphe $GE2_{illustr}$ qui se ramène au sous-graphe de la figure 4.4.

Une fois que le cadre d'instanciation est réajusté, il s'agit de déterminer pour chaque règle rg considérée, toutes les substitutions dans GE pour les variables de localisation de rg en veillant à respecter le nombre de variables de compartiments. C'est-à-dire que chaque substitution calculée doit comporter le même nombre de noms de compartiments que l'on a de variables de compartiments dans rg . Cela implique deux choses :

- Les substitutions doivent être définies sur toutes les variables de compartiments, plus exactement à chaque variable de compartiments apparaissant dans rg doit être associé un compartiment par la substitution.
- Deux variables de compartiments de même type doivent être substituées par des compartiments différents : en effet, les substituer par le même compartiment changerait la nature de la réaction modélisée par la règle générique. Par exemple, rg_2 est utilisée pour matérialiser le transport de la molécule $M2$ entre deux compartiments de type $T2$; en substituant les deux variables de compartiments par le même compartiment, on obtient une règle intermédiaire (par exemple dans $GE3$)



qui ne matérialise pas un transport de molécule.

Ces deux exigences caractérisent les substitutions éligibles que nous définissons mathématiquement par la définition 4.20 suivante.

Définition 4.20. (*Substitutions éligibles*).

Soient :

- GE un graphe d'échanges sur $CompartGE$ un ensemble de types de compartiment ;
- rg une règle générique telle que $types(rg) \subseteq CompartGE$.

Une substitution $\sigma_{(GE, var(rg))}$ est dite substitution éligible pour rg dans GE si :

1. $dom(\sigma) = var(rg)$ d'où σ est une application ;
2. $\forall v, v' \in var(rg), \sigma_{(GE, var(rg))}(v) \neq \sigma_{(GE, var(rg))}(v'),$ d'où $\sigma_{(GE, var(MD_{rg}))}$ est injective ;

Les instances de règles génériques intrinsèquement correctes sont les instances obtenues par application de substitutions éligibles. Le tableau suivant donne ces substitutions pour les données d'illustration.

rg	Substitutions éligibles			
	$GE1_{illustr}$	$GE2_{illustr}$ ajusté	$GE3_{illustr}$	$GE4_{illustr}$
rg1		{v1\A}	{v1\A} {v1\C}	
rg2			{v1\B, v2\D} {v1\D, v2\B}	
rg3	{v1\Env}	{v1\Env}	{v1\Env}	{v1\Env}
rg4			{v1\A, v2\B} {v1\A, v2\D} {v1\C, v2\B} {v1\C, v2\D}	
rg5			{v1\A, v2\B} {v1\A, v2\D} {v1\C, v2\B} {v1\C, v2\D}	
rg6			{v1\A, v2\C, v3\Env} {v1\C, v2\A, v3\Env}	
rg7	{v1\Env}	{v1\Env} {v1\A}	{v1\Env} {v1\A} {v1\B} {v1\C} {v1\D}	{v1\Env}
rg8		{v1\A}	{v1\A} {v1\C}	

TABLE 4.2 – Substitutions éligibles dans $GE1_{illustr}$ à $GE4_{illustr}$ pour les règles de $MGen_1$

Pour pratiquement toutes les règles génériques de $MGen_1$, les variables de compartiments sont sous la forme abrégée consistant en l'omission du nom de la variable. Aussi, pour le calcul des substitutions, des noms leur ont été associés ($v1$ pour la première variable anonyme rencontrée dans la règle, $v2$ pour la deuxième et ainsi de suite).

Sélection des substitutions éligibles : les substitutions validées par les voisinages

Toutes les substitutions éligibles obtenues ne respectent pas les voisinages entre compartiments. Par exemple, les deux substitutions calculées pour $rg_{(2)}$ impliquent deux compartiments B et D qui n'ont aucune relation de voisinage, et dont les molécules ne peuvent par conséquent pas interagir. L'objectif de cette étape est de discriminer les substitutions éligibles pour ne retenir que celles qui respectent les conditions de voisinage imposées par les règles génériques auxquelles elles doivent être appliquées. Ces conditions de voisinage doivent être évaluées sur la base du graphe d'échanges. Elles sont réparties selon deux catégories :

- Les conditions de voisinage explicites fournies au travers de la composante $PreC$ des règles génériques qui comme nous l'avons déjà vu exprime des nécessités de voisinage direct.
- Les conditions de voisinage implicites, définies par défaut dans le système. Ces dernières sont au nombre de deux pour une règle générique considérée :

- La première porte sur les variables de compartiment figurant dans le membre gauche de la règle générique ($var(MG)$). Elle stipule que deux compartiments génériques quelconques de cet ensemble doivent être accessibles l'un à partir de l'autre en utilisant, si nécessaire, comme seuls intermédiaires possibles, les autres compartiments génériques du même ensemble. Cette condition permet de garantir que tous les compartiments impliqués dans le déclenchement d'une réaction ont la possibilité de s'échanger des molécules indépendamment des autres éléments de la structure topologique.
- La deuxième porte sur les voisinages qui doivent exister entre les éléments de $var(MG)$ et ceux de $var(MD)$. Elle est vérifiée si chaque compartiment générique de $var(MD)$ est soit élément de $var(MG)$, soit directement voisin à au moins un compartiment générique de $var(MG)$. La raison d'être de cette condition est de garantir que les éléments de $var(MD)$ qui ne sont pas dans $var(MG)$ peuvent communiquer avec les éléments de $var(MG)$ pour en recevoir des molécules.

La prise en compte de toutes les conditions de voisinage par un mécanisme de sélection nous donne pour une règle générique rg et un graphe d'échanges GE , l'ensemble des substitutions dans GE pour $var(rg)$, validées par les voisinages de GE que nous définissons mathématiquement comme suit :

Définition 4.21. (*Substitutions validées par les voisinages d'un graphe d'échanges*).

Soient :

- $GE = (V, E, l_V)$ un graphe d'échanges sur $CompartGE$;
- $rg = [PreC] [ParamC] MG \Rightarrow MD CondVarMol$ une règle générique sur $CompartRG$;
- $\sigma_{(GE, var(rg))}$, une substitution éligible pour rg dans GE .

On dit que $\sigma_{(GE, var(rg))}$ est validée pour rg par les voisinages de GE si :

1. $\sigma_{(GE, var(rg)|var(MG))}$ est connexe ;
2. $\sigma_{(GE, var(rg)|var(MD))}$ est voisine à $\sigma_{(GE, var(rg)|var(MG))}$;
3. $\forall \{v, v'\} \in PreC, \{\sigma_{(GE, var(rg))}(v), \sigma_{(GE, var(rg))}(v')\} \in E$.

La sélection des substitutions éligibles du tableau 4.2 donne comme substitutions validées celles du tableau 4.3 qui ne soient pas hachurées.

En consultant le tableau 4.3, on constate que pour les règles génériques qui ne contiennent qu'une unique variable de compartiment ($rg1$, $rg3$ et $rg7$), toutes les substitutions éligibles sont de fait validées par le graphe d'échanges. En effet, de par le nombre de compartiments impliqués, aucune condition de voisinage n'a à être vérifiée, ce qui explique qu'on retrouve toutes ces substitutions dans le tableau 4.3.

En ce qui concerne les règles génériques n'impliquant que deux variables de compartiments, les substitutions éligibles qui ne sont pas validées (celles calculées pour $rg2$, et quelques unes de celles calculées pour $rg4$ et $rg5$) ne respectent pas les conditions de voisinage par défaut. En effet, en présence de deux variables de compartiment, la vérification des conditions de voisinage par défaut suffisent pour que les substitutions soient validées. Étant donné le nombre de compartiments impliqués, la vérification de la condition de connexité exigée sur les compartiments du membre gauche et celle de la deuxième condition par défaut, impliquent un voisinage direct entre les deux compartiments, qui se révèle être la seule condition de voisinage qui aurait pu faire l'objet d'une précondition de voisinage. Nous pouvons affirmer que les préconditions de voisinage ne sont pertinentes que pour les règles impliquant plus de deux compartiments.

Les substitutions éligibles calculées pour $rg6$ ne respectent pas les conditions de voisinage particulières données par la précondition de voisinage. En effet, les compartiments A et D appartenant à $GE3_{illustr}$ ne sont pas voisins. On remarquera que ces deux substitutions respectent les conditions de voisinage

rg	Substitutions éligibles et substitutions validées			
	$GE1_{illustr}$	$GE2_{illustr}$ ajusté	$GE3_{illustr}$	$GE4_{illustr}$
rg1		$\{v1 \setminus A\}$	$\{v1 \setminus A\}$ $\{v1 \setminus C\}$	
rg2			$\{v1 \setminus A\}$ $\{v1 \setminus B\}$ $\{v1 \setminus C\}$ $\{v1 \setminus D\}$	
rg3	$\{v1 \setminus Env\}$	$\{v1 \setminus Env\}$	$\{v1 \setminus Env\}$	$\{v1 \setminus Env\}$
rg4			$\{v1 \setminus A, v2 \setminus B\}$ $\{v1 \setminus A\}$ $\{v1 \setminus B\}$ $\{v1 \setminus C\}$ $\{v1 \setminus D\}$ $\{v1 \setminus C, v2 \setminus D\}$	
rg5			$\{v1 \setminus A, v2 \setminus B\}$ $\{v1 \setminus A\}$ $\{v1 \setminus B\}$ $\{v1 \setminus C\}$ $\{v1 \setminus D\}$ $\{v1 \setminus C, v2 \setminus D\}$	
rg6			$\{v1 \setminus A\}$ $\{v1 \setminus B\}$ $\{v1 \setminus C\}$ $\{v1 \setminus D\}$ $\{v1 \setminus A, v2 \setminus B\}$ $\{v1 \setminus C, v2 \setminus D\}$	
rg7	$\{v1 \setminus Env\}$	$\{v1 \setminus Env\}$ $\{v1 \setminus A\}$	$\{v1 \setminus Env\}$ $\{v1 \setminus A\}$ $\{v1 \setminus B\}$ $\{v1 \setminus C\}$ $\{v1 \setminus D\}$	$\{v1 \setminus Env\}$
rg8		$\{v1 \setminus A\}$	$\{v1 \setminus A\}$ $\{v1 \setminus C\}$	

TABLE 4.3 – Substitutions validées dans $GE1_{illustr}$ à $GE4_{illustr}$ pour les règles de $MGen_1$.

par défaut. De manière générale, la vérification des conditions de voisinage par défaut précède celle des conditions particulières.

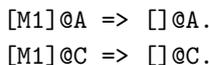
Instanciation des règles génériques : application des substitutions validées

L’instanciation des règles génériques consiste pour une règle rg en l’application à rg des substitutions validées pour rg dans le graphe d’échanges utilisé comme cadre d’instanciation. En tant qu’instances particulières de règles génériques, les règles intermédiaires ont globalement la même structure que les règles génériques dont elles sont issues. En particulier, on notera que si les membres gauche et droit conservent la même structure en ne différant que sur les variables de compartiments qui sont remplacées par des noms de compartiments, les structures des paramètres cinétiques peuvent changer complètement. C’est notamment le cas lorsque le paramètre cinétique de la règle générique est défini en utilisant *Neighb* : l’instanciation s’accompagne d’une évaluation de la fonction *Neighb* pour déterminer le bon paramètre cinétique.

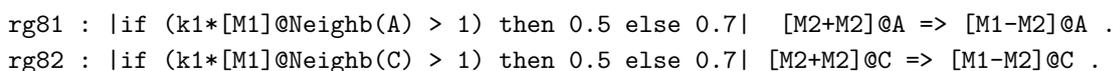
Nous obtenons pour les substitutions validées du tableau 4.3 les trois modèles intermédiaires donnés par les fichiers 4.4, 4.5 et 4.6 suivants (l’instanciation dans $GE1_{illustr}$ étant la même que celle dans $GE4_{illustr}$).

On notera en particulier que les informations de localisation n’apparaissent pas dans le modèle intermédiaire *fichInterm1* qui modélise un processus se déroulant en vase clos. Il est également à noter que

l'utilisation de *AnyW* entraîne quelques fois des situations dans lesquelles plusieurs règles intermédiaires expriment la même réaction : par exemple, en considérant $GE3_{illustr}$ et en appliquant toutes les substitutions validées pour $rg1$ et toutes celles validées pour $rg3$ respectivement à $rg1$ et $rg3$, on obtient deux fois les réactions



parce que $rg1$ est un cas particulier de $rg7$. Enfin, nous faisons remarquer que l'instanciation donne en principe :



pour $rg8$ instanciée respectivement dans $GE2_{illustr}$, $GE3_{illustr}$ et $GE3_{illustr}$. C'est l'évaluation dans les graphes $GE2_{illustr}$ et $GE3_{illustr}$ de $Neighb(A)$ et $Neighb(C)$ qui permettent d'obtenir respectivement les règles $rg81$ dans `fichInterm2` et `fichInterm3` et $rg81$ dans `fichInterm2`.

Listing 4.4 – Modèle intermédiaire obtenu avec $GE1_{illustr}$: fichier `fichInterm1`

```
1 BEGIN_TYPEC AnyW; Env; T1; T2 END_TYPEC .
2 BEGIN_MOLSB M1; M2; M3 END_MOLSB.
3 BEGIN_MODIF PHOS END_MODIF.
4 BEGIN_PARAM k1(0.1) END_PARAM.
5
6 BEGIN_REGINTERM
7 ri11 : [] =>[M1] .
8 ri11 : [M1] =>[] .
9 END_REGINTERM
```

Listing 4.5 – Modèle intermédiaire obtenu avec $GE2_{illustr}$: fichier `fichInterm2`

```
1 BEGIN_TYPEC AnyW; Env; T1; T2 END_TYPEC .
2 BEGIN_MOLSB M1; M2; M3 END_MOLSB.
3 BEGIN_MODIF PHOS END_MODIF.
4 BEGIN_PARAM k1(0.1) END_PARAM.
5
6 BEGIN_REGINTERM
7 ri11 : [M1]@A => []@A .
8 ri31 : []@Env =>[M1]@Env .
9 ri71 : [M1]@Env =>[]@Env .
10 ri81 : |if ([M1]@Env > 1) then 0.5 else 0.7| [M2+M2]@A => [M1-M2]@A .
11 END_REGINTERM
```

Listing 4.6 – Modèle intermédiaire obtenu avec *GE3_{illustr}* : fichier *fichInterm3*

```

1 BEGIN_TYPEC AnyW; Env; T1; T2 END_TYPEC .
2 BEGIN_MOLSB M1; M2; M3 END_MOLSB.
3 BEGIN_MODIF PHOS END_MODIF.
4 BEGIN_PARAM k1(0.1) END_PARAM.
5
6 BEGIN_REGINTERM
7 ri11 : [M1]@A => []@A .
8 ri12 : [M1]@C => []@C .
9
10 ri_31 : []@Env =>[M1]@Env .
11
12 ri41 : [(M1:PHOS)]@A & [M2]@B => [M3]@B .
13 ri42 : [(M1:PHOS)]@C & [M2]@D => [M3]@D .
14
15 ri51 : [M2]@A => [M1]@B & [M2]@A .
16 ri52 : [M2]@C => [M1]@D & [M2]@C .
17
18 ri71 : [M1]@Env =>[]@Env .
19 ri72 : [M1]@B => []@B .
20 ri73 : [M1]@D => []@D .
21
22 ri81 : | if (k1 * [M1]@Env + [M1]@B > 1) then 0.5 else 0.7| [M2+M2]@A => [M1-M2]@A .
23 ri82 : | if (k1 * [M1]@Env + [M1]@D > 1) then 0.5 else 0.7| [M2+M2]@C => [M1-M2]@C .
24 END_REGINTERM

```

4.3 Traduction des règles intermédiaires

Le schéma général de notre approche (cf. figure 2.1) indique que la procédure de traduction permet d’obtenir à partir d’un ensemble de règles intermédiaires et d’un outil cible choisi par l’utilisateur, un modèle de processus biologique, écrit dans le langage de règles de l’outil cible choisi. La présente section a pour objet de présenter cette phase de notre approche. Il s’agit pour une règle intermédiaire *ri* considérée, d’en transcrire chaque partie (états cellulaires, paramètres cinétiques ou conditions sur les variables de molécules) et d’agencer ces différentes transcriptions selon la syntaxe du langage de l’outil cible. Ces parties étant bâties sur les objets formels de base que sont les termes molécules et les localisations, leur traduction passe par celle de ces objets formels dont certains (molécules de base et localisations) en tant qu’identificateurs ne seront pas traduits et donc seront simplement repris tels quels dans les langages des outils cibles.

Nous organisons notre description autour des paragraphes suivants :

- le paragraphe 4.3.1 va concerner la traduction d’un modèle intermédiaire dans le langage de règles de BIOCHAM ; nous présenterons dans un premier temps les mécanismes de traduction des termes molécule avant de détailler ceux des règles intermédiaires à travers les transcriptions des états cellulaires, des paramètres cinétiques et des conditions sur les variables de molécules ;

- dans le paragraphe 4.3.2, nous détaillerons selon le même schéma que précédemment pour BIOCHAM, la transcription d'un modèle intermédiaire dans le langage de PATHWAY LOGIC ;
- le paragraphe 4.3.3 va donner la traduction en BIOCHAM et PATHWAY LOGIC du modèle intermédiaire `fichInterm3` et consister un en récapitulatif sous forme de tableau des deux paragraphes précédents.

4.3.1 Traduction vers BIOCHAM

La quasi-totalité des éléments du langage de règles génériques peut être traduit en BIOCHAM. Dans cette section, nous allons dans un premier temps présenter la traduction des molécules (complexes et formes modifiées). Ensuite, nous détaillerons le passage des états cellulaires intermédiaires à des états cellulaires exprimés en langage de règles BIOCHAM, avant de décrire la traduction des paramètres cinétiques. Enfin, nous discuterons de la fidélité des traductions surtout si des compromis ont été consentis.

4.3.1.1 Traduction des termes molécule

Les complexes et les formes modifiées de molécules que l'on rencontre dans le langage de règles génériques se traduisent assez naturellement en langage de règles BIOCHAM. En langage de règles génériques comme en langage de règles BIOCHAM, un complexe est une molécule issue d'une *liaison* entre deux molécules. Le complexe

CDC2-CYCLIN

sera rendu de la même manière en BIOCHAM :

CDC2-CYCLIN

Il en va autrement pour les opérations de modification de forme de molécules. En effet, toutes les informations manipulées en langage de règles génériques dans le cadre de ces opérations ne peuvent être *fidèlement* rendues en BIOCHAM qui ne considère que la phosphorylation comme modification de forme et qui ne précise que les sites de phosphorylation dans l'expression des formes modifiées de molécules. Deux cas sont à distinguer selon que les sites de modification sont précisés ou non : dans le premier cas, nous aurons comme schéma de traduction :

(Mol : Modif<s1, ... sn>)

(Mol : <s1, ... sn>)

qui donnent en BIOCHAM

Mol~{s1, ... sn}

et dans l'autre cas nous aurons

(Mol : Modif)

qui donne en BIOCHAM

Mol~{default}

fournissant l'information qu'aucun site de modification n'a été précisé. Il s'agit d'une convention que nous nous donnons pour pouvoir représenter ce genre de molécules modifiées.

La transformation d'une variable de molécule d'une règle intermédiaire en une variable de molécule BIOCHAM, consiste à remplacer le caractère ? par \$.

CDC2-?vMol

en langage de règles intermédiaires donnera en BIOCHAM

CDC2-\$vMol

4.3.1.2 Traduction des règles intermédiaires

Définition 4.22. (*Utilisation du mot « intermédiaire »*).

Nous qualifions d'« intermédiaire » toute instance d'une partie générique d'une règle générique. Ainsi on a les états intermédiaires qui sont des instances d'états génériques, les solutions intermédiaires sont les instances de solutions intermédiaires et ainsi de suite.

Dans les règles intermédiaires, un état cellulaire est un ensemble de k solutions localisées noté

$S_1@L_1, \dots, S_k@L_k$, une solution S_i , ($i \in [1, \dots, k]$) étant un ensemble de n_i termes molécule : m_{i1}, \dots, m_{in_i} contenues dans L_i .

En BIOCHAM par contre, un état cellulaire est une solution vue comme un ensemble d'objets formels BIOCHAM (molécule ou molécule assortie d'une localisation). La traduction d'un état cellulaire de règle intermédiaire en un état cellulaire de règle BIOCHAM va donc consister à transformer les k solutions intermédiaires en une solution BIOCHAM. Pour cela, chaque terme molécule de chaque solution intermédiaire sera transformée dans BIOCHAM, en molécule ou en schéma de molécule assortie d'une localisation. Ainsi on aura

$$[m_{11} + \dots + m_{1n_1}]@L_1, \& \dots, \& [m_{k1} + \dots + m_{kn_k}]@L_k$$

qui donne en BIOCHAM

$$m_{11}::@L_1 + \dots + m_{1n_1}::L_1 + \dots + m_{k1}::@L_k + \dots + m_{kn_k}::L_k$$

De manière générale, on a pour un état intermédiaire

$$ei = \prod_{i \in [1,k]} (S_i @ L_i)$$

une solution BIOCHAM

$$SolutionB = \prod_{i \in [1,k]} \left(\prod_{j \in [1,n_i], m_{ij} \in S_i @ L_i} (m_{ij} :: L_i) \right)$$

En ce qui concerne les paramètres cinétiques, la seule forme qui soit à traduire est celle exprimant des concentrations de molécules d'une solution :

$$| [m_1 + \dots + m_n] @ L |$$

qui sera transformé en

$$[m_1 :: L] * \dots * [m_n :: L]$$

correspondant à la loi d'action de masse avec comme paramètre 1, des molécules de la solution. Les autres formes sont simplement reprises en traitant cependant la particularité de `Neighb`.

La traduction des conditions sur les variables de molécules se fait assez facilement en remplaçant `if` par `where`.

La règle BIOCHAM issue d'une règle intermédiaire `ri` sera obtenue en agençant les différentes transcriptions des composantes de `ri` de la manière suivante

$$[ParamC \text{ for}] MG \Rightarrow MD [where \text{ CondVMol}].$$

les éléments entre crochets étant optionnels.

Au niveau modèle, les sections `MOLSB` et `PARAM` permettront respectivement de récupérer éventuellement les sites de phosphorylation des différentes molécules de base et les valeurs initiales des identificateurs de paramètres cinétiques.

4.3.2 Traduction vers PATHWAY LOGIC

La présentation de la traduction des règles intermédiaires vers PATHWAY LOGIC suivra à peu de choses près le même plan que celle de la traduction vers BIOCHAM, la principale différence étant liée au fait que PATHWAY LOGIC ne prend pas en compte les paramètres cinétiques.

4.3.2.1 Traduction des termes molécule

Avant toute chose, nous commençons par préciser que les variables de molécules sont reprises telles quelles dans PATHWAY LOGIC. En effet, elles s'expriment de la même manière dans les règles intermédiaires et dans les règles de PATHWAY LOGIC.

Tous les complexes et toutes les formes modifiées de molécules qu'on rencontre dans un modèle intermédiaire peuvent être transcrites dans PL. Les résultats de ces transcriptions sont des réarrangements syntaxiques des molécules de base et variables de molécules en utilisant les bons opérateurs. La traduction des complexes se ramène à un remplacement de l'opérateur «-» par « : ». Ainsi, le complexe

CDC2-CYCLIN donne en PATHWAY LOGIC (CDC2 : CYCLIN)

En ce qui concerne les modifications de formes, PATHWAY LOGIC n'autorise pas les formes ne précisant pas la nature de la modification. Aussi, nous aurons pour respectivement

(Mol : Modif<s1, ... sn>)

(Mol : Modif)

les schémas de traduction respectifs suivants :

[Mol-Modif(s1) ... Modif (sn)]
[Mol-Modif]

Pour transcrire la forme de modification n'indiquant que les sites de modification, nous introduisons une opération de modification de forme par défaut qui est *ModDefault* et ainsi, nous traduisons

(Mol : <s1, ... sn>)

en PATHWAY LOGIC par :

[Mol-ModDefault (s1) ... ModDefault (sn)]

4.3.2.2 Traduction des règles intermédiaires

Dans les règles intermédiaires de même qu'en PATHWAY LOGIC, un état cellulaire est défini selon la même compréhension : un ensemble de k localisations contenant chacune des termes molécules. La transcription de l'état exprimé en langage de règles intermédiaires vers l'état PATHWAY LOGIC consiste donc en de simples réarrangements syntaxiques des termes molécules et des localisations. (Avant de poursuivre, nous précisons que nous considérons la compartimentation de niveau 1, telle que définie dans la section 2.2.3). L'état intermédiaire

$$[m_{11} + \dots + m_{1n_1}] @L_1, \& \dots, \& [m_{k1} + \dots + m_{kn_k}] @L_k$$

donnera en PL

$$\{L_1 \mid l_1 \ m_{11} \ \dots \ m_{1n_1}\} \ \dots \ \{L_k \mid l_k \ m_{k1} \ \dots \ m_{kn_k}\}$$

De manière générale, on a pour un état intermédiaire

$$ei = \prod_{i \in [1,k]} (S_i @L_i)$$

où les S_i sont des solutions, un état PL

$$etatPL = \prod_{i \in [1,k]} (\{L_i \mid l_i \ S_i\})$$

où les S_i sont des **Soup** contenant les termes molécules qui sont produits, consommés ou transportés au cours de la réaction et les l_i des **Soup** désignant l'ensemble des autres molécules contenues dans L_i .

La traduction des conditions sur les variables de molécule donne lieu dans PATHWAY LOGIC à des *conditions de matching*, avec comme éléments de *matching* les différentes molécules des domaines de définition des variables. Rappelons que les conditions dans PATHWAY LOGIC sont introduites par *if*. Nous aurons pour la condition intermédiaire suivante

$$\text{if } (?vMol_1 \text{ in } \{M_1, M_2\} \text{ and } ?vMol_2 \text{ in } \{M_3, M_4\})$$

la condition PL

$$\text{if } ?vMol_1 \ S_1 : \text{Soup} := M_1 \ M_2 \ \wedge \ ?vMol_1 \ S_1 : \text{Soup} := M_3 \ M_4 \ .$$

En ce qui concerne les identificateurs de règles intermédiaires,

$$\text{ri_details} \text{ donne lieu à } \text{rl}[\text{details}] \text{ et } \text{cri_details} \text{ donne lieu à } \text{crl}[\text{details}]$$

Tout comme dans BIOCHAM, la règle PATHWAY LOGIC issue d'une règle intermédiaire **ri** sera obtenue en agençant les différentes transcriptions des composantes de **ri** de la manière suivante

$$\text{idrg} : \text{MG} \Rightarrow \text{MD} \ [\text{CondVMol}]$$

4.3.2.3 Obtention d'un modèle PL

L'obtention d'un modèle PL à partir d'un modèle intermédiaire passe par la constitution des quatre fichiers qui contiennent respectivement les modules donnant la définition du type algébrique de données (*theops.maude*), les molécules de base (*components.maude*), les règles de réaction (*rules.maude*) et les états initiaux (*qq.maude*).

Nous définissons un fichier *theops.maude* général qui sera utilisé pour tous les modèles que nous construirons à partir d'un ensemble de règles intermédiaires. Le principal module de ce fichier, *THEOPS* incorpore récursivement la définition des modules

PROTEIN
THING
SOUP
MODIFICATION
LOCATION
CELL
DISH

qui fournissent en particulier toutes les opérations non unaires pour définir toutes les formes de molécules possibles et leur localisation ainsi que les états initiaux. Le fichier *theops.mauve* est donné en annexe B.1.

A côté de ce fichier, nous avons ceux qui varient en fonction des processus étudiés. Il s'agit :

- du fichier *modOtherOps.mauve* qui comportera s'il y a lieu plusieurs modules :
 - dans le module MODIFS on a les opérations de modification de forme déclarés dans le modèle intermédiaire. Chaque élément *idModif* de la section MODIFS va donner lieu aux deux opérations suivantes :

```
op idModif :-> Modification
op idModif : site-> Modification
```

- dans le module LOCALISATION on a les informations de localisation issues de la section TYPEC du modèle intermédiaire. Chaque type de compartiment *idTypeC* ou chaque partie de type de compartiment *idPartC* donnent lieu respectivement à une opération de la forme

```
op idTypeC : -> locName.
op idPartC : -> locName.
```

- enfin le module TYPECELLULE va lister les opérations de déclaration des types de cellules utilisées.
- du fichier *modComponents.mauve* qui sera renseigné à partir de la section MOLSB. Le module PROTEINOPS donnera lieu pour chaque classe de molécules de base *idClasseMBase* à une déclaration de *sort* comme sous-sort de PROTEIN, ce qui correspond à l'insertion des lignes :

```
sort idClasseMBase
subsort idClasseMBase < PROTEIN.
```

et pour chaque molécule de base *idMBase idClasseBase (lstsites)* à une opération de la forme

```
op idMBase :-> idClasseBase
```

- du fichier *modRules.mauve* contenant le module RULES où sont déclarées en premier lieu les différentes variables qui seront utilisées. En particulier, on déclarera pour chaque donnée de localisation *idLoc*, la variable de type Soup qui correspond à son contenu non décrit dans les règles de réaction, à l'aide de l'instruction

```
var idContRestant_idLoc : Soup .
```

Ensuite, on aura les règles biochimiques traductions, dans PATHWAY LOGIC des règles intermédiaires.

– du fichier `modqq` qui donne l'état initial à considérer.

Une fois qu'on a tous ces fichiers dont les parties immuables sont données en annexes B, on peut définir comme suit le fichier `all.maude` qui sera chargé lors de l'ouverture du modèle.

```

1 load theops
2 load modOtherOps
3 load modComponents
4
5 fmod ALLOPS is
6   inc THEOPS .
7   inc MODIFS .
8   inc LOCALISATIONS .
9   inc PROTEINOPS .
10 endfm
11
12 load modRules
13 load model-checker
14 load modqq

```

4.3.3 Exemple : traduction du modèle intermédiaire donné par `fichInterm3`

Traducion en BIOCHAM

Listing 4.7 – Traduction en BIOCHAM du modèle donné `fichInterm3`

```

1 parameter(k1,0.1).
2
3 r_(1)_1 : M1::A => ..:A .
4 r_(1)_2 : M1::C => ..:C .
5
6 r_(3)_1 : ..:Env => M1::Env .
7
8 r_(4)_1 : M1~\{default\}::A + M2::B => M3::B .
9 r_(4)_2 : M1~\{default\}::C + M2::D => M3::D .
10
11 r_(5)_1 : M2::A => M1::B + M2::A .
12 r_(5)_2 : M2::C => M1::D + M2::C .
13
14 r_(7)_1 : M1::Env => ..:Env .
15 r_(7)_2 : M1::B => ..:B .
16 r_(7)_3 : M1::D => ..:D .
17
18 r_(8)_1 : if (k1 * ([M1::Env] + [M1::B]) > 1) then 0.5 else 0.7 for M2::A + M2::A => M1-M2::
    A .
19 r_(8)_2 : if (k1 * ([M1::Env] + [M1::D]) > 1) then 0.5 else 0.7 for M2::C + M2::C => M1-M2::
    C .

```

Traduction en PATHWAY LOGIC

Listing 4.8 – Traduction en PATHWAY LOGIC du modèle donné fichInterm3 : fichier modOtherOps

```

1  fichier  modOtherOps.maude
2
3  fmod MODIFS is inc THEOPS .
4    op PHOS : -> Modification .
5    op PHOS : Site -> Modification .
6  endfm
7
8  fmod LOCALISATIONS is inc THEOPS .
9    ops A B C D Env : -> LocName .
10 endfm

```

Listing 4.9 – Traduction en PATHWAY LOGIC du modèle donné fichInterm3 : fichier modOtherOps

```

1  Fichier  modComponents.maude
2
3  fmod PROTEINOPS is inc THEOPS .
4    op M1 : -> Protein .
5    ops M2 M3 : -> Protein .
6  endfm

```

Listing 4.10 – Traduction en PATHWAY LOGIC du modèle donné fichInterm3 : fichier modOtherOps

```

1  Fichier  modRules.maude
2
3  mod ALLBP is inc PROTEINOPS .
4
5  vars  a b c d env : Soup .
6
7  rl[1.1]: \{A | a M1\} => \{A | a\} .
8  rl[1.2]: \{C | c M1\} => \{C | c\} .
9
10 rl[3.1]: \{Env | env\} => \{Env | env M1\} .
11
12 rl[4.1]: \{A | a [M1-PHOS]\} \{B | b M2\} => \{A | a\} \{B | b M3\} .
13 rl[4.2]: \{C | c [M1-PHOS]\} \{D | d M2\} => \{C | c\} \{D | d M3\} .
14
15 rl[5.1]: \{A | a M2\} \{B | b\} => \{A | a M2\} \{B | b M1\} .
16 rl[5.2]: \{C | c M2\} \{D | d\} => \{C | c M2\} \{D | d M1\} .
17
18 rl[7.1]: \{Env | env M1\} => \{Env | env\} .
19 rl[7.2]: \{B | b M1\} => \{B | b\} .
20 rl[7.3]: \{D | d M1\} => \{D | d\} .
21
22 rl[8.1]: \{A | a M1 M2\} => \{A | a (M1:M2)\} .

```

```

23     rl [8_1]: \{C | c M1 M2\} => \{C | c (M1:M2)\} .
24
25     endm

```

4.4 Implémentation

Dans cette section, nous décrivons les principales classes et structures de données définies pour la représentation des modèles génériques et celles utilisées dans la mise en œuvre des procédures d’instanciation d’un modèle générique dans un graphe d’échanges et de traduction d’un modèle intermédiaire en langage d’outil cible. Nous donnerons également quelques algorithmes, notamment celui du couplage règles génériques/graphes d’échanges et l’algorithme de traduction. Le développement a été effectué en C/C++ dans l’Environnement de Développement Intégré (EDI) *Codeblocks*.

Remarque 4.4. *Dans la suite du document, nous utilisons le terme "fichier modèle générique" pour désigner le fichier texte d’extension « mg » écrit en langage de règle génériques.*

4.4.1 Le modèle générique

Nous avons défini en C++ une bibliothèque *libModeleGenerique* qui contient toutes les fonctions de définition (à partir d’un fichier modèle générique) et de manipulation d’un modèle générique (objet de la classe *CModeleGenerique*). Cette librairie statique utilise les fonctions d’une autre, la *libAnalyseSyntaxeRG* qui a été développée en C et qui regroupe uniquement les fonctions d’un analyseur syntaxique pour les fichiers modèle générique.

L’analyseur syntaxique

Les nœuds de l’arbre de syntaxe abstraite produit par l’analyseur syntaxique sont des instances de la structure de donnée *noeud* ci-dessous

```

typedef struct noeud
{
    typeNoeud natureN;
    detailNoeud valNoeud;
}noeud;

```

noeud contient un champ (*natureN*) qui donne la nature du nœud, c’est-à-dire l’information du type ou de la structure de données de l’élément que va contenir le champ *valNoeud* qui est une *union de données*. L’union de données *detailNoeud* est défini comme suit :

```

typedef union detailNoeud
{
    typeNoeud natureN;

```

```

n_modGen mGen;
n_typesC lesTypesC;
n_molsB lesMolsB;
n_molB uneMolB;
n_sitesM lesSitesM;
n_modifsF lesModifsF;
n_paramsC lesParamsC;

n_regleG * lesRG;
}detailNoeud;

```

Le nombre de champs de cette union correspond au nombre de types de nœuds différents que contient l'arbre de syntaxe abstraite. La librairie *libAnalyseSyntaxeRG* compte environ 1800 lignes de code (non généré).

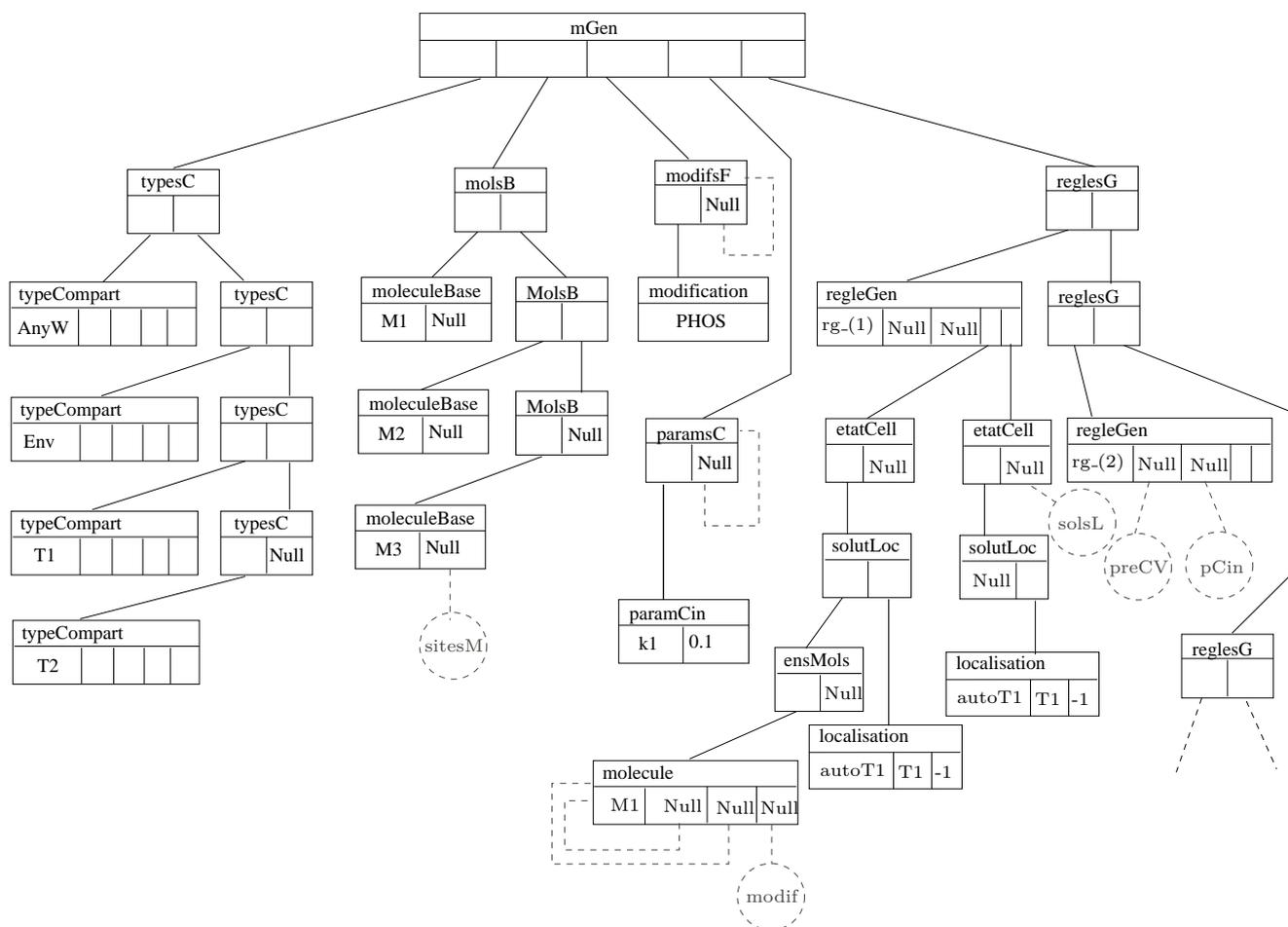


FIGURE 4.5 – arbre de syntaxe abstraite

La figure 4.5 représente une partie de l'implantation de l'arbre de syntaxe abstraite produit par l'analyse du fichier modèle générique *fichrg1*. On peut y lire sur les premières lignes de chaque forme rectangulaire

la nature du nœud. On a donc comme racine un nœud *MGen* qui a comme fils un nœud *typesC*, un nœud *molsB*, un nœud *modifsF*, un nœud *paramsC* et un nœud *reglesG* à partir desquels on peut parcourir respectivement les types de compartiments, les molécules de base avec éventuellement leurs sites de modification, les modifications de forme de molécules et les identificateurs de paramètres cinétiques avec leurs valeurs.

Les cercles gris en pointillés et les lignes qui y arrivent ne font pas partie de l'arbre. Ils sont mis comme indications pour renseigner sur la nature de certains nœuds fils des nœuds dont partent les traits : ainsi, on peut lire qu'un nœud *molécule* est une union d'un identificateur (qui est renseigné si on a affaire à une molécule de base), deux pointeurs sur des *molécule* (cas d'un complexe) et un pointeur sur un *modif* (si on a affaire à une forme modifiée de molécule). De même un nœud *moléculeBase* comprend un identificateur et un fils de nature *sitesM* qui dont le parcours donne l'ensemble des sites de modification déclarés pour la molécule.

La bibliothèque *libModeleGenerique*

L'arbre de syntaxe abstraite représente le modèle générique. Cependant nous avons défini une fonction qui transforme un arbre donné en un objet de la classe *CModeleGenerique*. Les raisons d'une telle transformation sont de deux ordres :

- nous voulions bénéficier des avantages de la programmation objet pour l'instanciation ;
- la transformation de l'arbre en plusieurs listes permet d'allouer à chaque objet l'espace effectivement nécessaire. La structure des nœuds de l'arbre est relativement grosse ;
- l'implantation de cette librairie a de nombreux points communs avec celle de la librairie *libModeleInterm* qui sera utilisée au niveau des procédures de traduction.

Nous donnons en annexe D.3, D.4 et D.5 les implantations (fichiers entêtes uniquement) de certaines des classes :

CModeleGenerique, CRegleGenerique CMolécule

4.4.2 Implantation de l'instanciation et de la traduction

Elle a donné lieu à la bibliothèque statique *libModeleInterm* qui fait appel aux bibliothèques statiques *libModeleGenerique* et *libGE*. Deux algorithmes ont été définis pour l'instanciation des règles génériques, la différence entre les deux se situant au niveau de la génération et du filtrage des substitutions valides :

- Le premier algorithme (algorithme 4.4.2) qui a été implanté est robuste mais n'est pas très efficace (en temps) : en effet, pour deux règles génériques différentes qui portent sur les mêmes types de compartiment aussi bien dans le membre gauche que dans le membre droit, l'algorithme génère deux fois les mêmes substitutions et effectue deux fois le même filtrage.
- Le second algorithme (algorithme 4.4.2.1) est une tentative d'amélioration du premier en évitant si possible la répétition de la génération des substitutions valides pour le membre gauche des règles.

Premier algorithme de couplage

Les étapes de cet algorithme sont les suivantes et elles s'appliquent à toutes les règles génériques :

1. On vérifie que la règle générique peut potentiellement être instanciée : si rg implique plus de variables de compartiment de type t que n'en compte le graphe d'échanges, elle ne peut être instanciée. On passe à la règle suivante.
2. On effectue la génération des substitutions pour les variables de compartiment du membre gauche de rg . Pour chaque substitution, on teste la connexité. A la sortie de cette étape, on a la liste des substitutions connexes pour les variables du membre gauche ($lstSubstMG$).
3. On effectue la génération des substitutions pour les variables de compartiment du membre droit de rg , ce qui donne lieu à la liste $lstSubstMD$.
4. Pour chaque élément $substMG$ de $lstSubstMG$, on considère tour à tour chaque élément $substMD$ de $lstSubstMD$ et on teste que $substMD$ est voisine à $substMG$. Si c'est le cas, on vérifie les préconditions de voisinage. Si celles-ci ne sont pas vérifiées, on passe au couple $(substMG, substMD)$ suivant.
5. Si le paramètre cinétique contient $Neighb(v_i)$ on le redéfinit en cherchant tous les voisins de v_i dans le graphe. Si v_i n'a pas de voisin et si le paramètre cinétique se réduisait à cette expression, rg ne peut être instanciée.
6. On crée une règle intermédiaire par application de $substMG \cup substMD$ à rg .

4.4.2.1 Deuxième algorithme de couplage

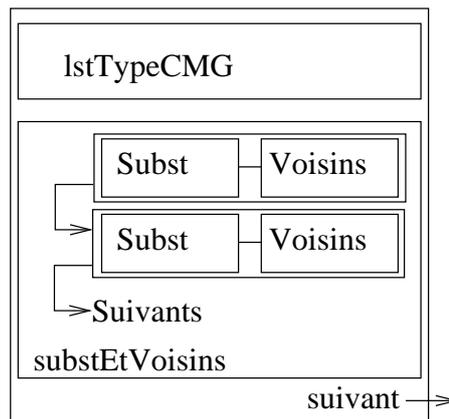


FIGURE 4.6 – Structure de la liste $lstSubstMG$

Cet algorithme est en cours d'implantation. Elle utilise une liste $lstSubstMG$ qui contient des éléments calculés pour l'instanciation d'une règle et pouvant être utiles pour l'instanciation des règles génériques non encore traitées. Nous donnons la structure de ces éléments dans la figure 4.6. Chaque élément contient deux composantes :

- la première ($lstTypeCMG$) est une liste (ordonnée) de types de compartiment ;
- la deuxième ($substEtVoisins$) est une liste d'éléments composés chacun :
 - d'une substitution $subst$ qui est en fait une liste de compartiments pris dans le graphe et ordonnée en fonction de $lstTypeCMG$: l'élément à l'indice i de $subst$ est de type l'élément à l'indice i de $lstTypeCMG$.
 - l'ensemble des compartiments voisins directs par le graphe des éléments de $subst$ qui ne sont pas dans $subst$: plus clairement, on ne considère pas les voisinages entre deux éléments de $subst$.

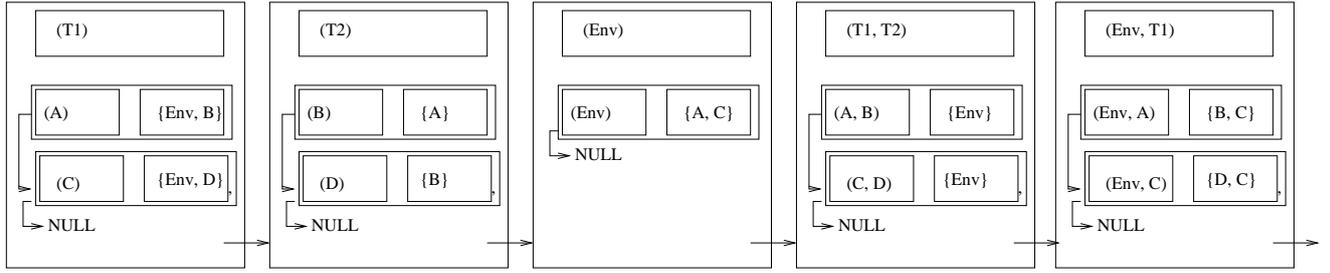


FIGURE 4.7 – Schématisation de la liste pour le graphe

Un exemple de liste (calculée pour le fichier de règles génériques *fichrg1* (4.3) sur la base du graphe d'échanges *GE3_{illustr}* (4.3(c)) est donné par la figure 4.7. Les quatre premiers éléments correspondent aux éléments calculés pour le membre gauche de respectivement rg_1 , rg_2 , rg_3 et rg_3 . Les règles rg_5 et rg_8 utiliseront le premier élément puisque leurs membres gauches portent sur exactement les mêmes types de compartiment que celui de rg_1 . Pour rg_7 portant sur *AnyW*, cette liste n'est pas utilisée.

Les étapes de l'algorithme sont les suivantes pour chaque règle générique :

1. On récupère la liste (ordonnée selon les types de compartiment) des variables de compartiments impliqués dans MG ($lstTypeCTemp$).
2. On vérifie que rg peut potentiellement être instanciée : si rg implique plus de variables de compartiment de type t que n'en compte le graphe d'échanges, elle ne peut être instanciée. On crée un nouvel élément qui a comme valeur ($lstTypeCTemp, NULL$) dans $lstSubstMG$ et on passe à la règle générique suivante.
3. On calcule $VMDEXclusif$ l'ensemble des variables de compartiments du membre droit qui ne sont pas dans le membre gauche ($Var(MD) \setminus Var(MG)$).
4. On recherche dans $lstSubstMG$ l'élément dont $lstTypeCMG$ correspond aux types de $lstTypeCTemp$. S'il existe, on va à l'étape 6.
5. On crée un nouvel élément dans $lstSubstMG$ ayant comme valeurs :
 - (a) pour $lstTypeCMG$ la liste des types de compartiments de $lstTypeCTemp$;
 - (b) pour $substEtVoisins$ la liste des objets tels que $subst$ est une substitution connexe pour les variables de compartiment de MG et $Voisins$ est la différence ensembliste entre l'union des ensembles des voisins des éléments de $subst$ et $subst$.
6. On génère la liste $lstSubstMD$ des substitutions de $VMDEXclusif$ à partir des éléments de $Voisins$.
7. Pour chaque substitution $subst$ de $substEtVoisins$, pour chaque élément de $lstSubstMD$, on teste que les préconditions de voisinage sont vérifiées. Si c'est le cas, on applique $subst$ à rg et $substMD$ à MD .

Le deuxième algorithme n'ayant pas été implanté, il est difficile de se prononcer sur son efficacité par rapport au premier.

Les différentes classes de la librairie *libModeleInterm* intègre des procédures de traduction en BIOCHAM ou en PATHWAY LOGIC. La traduction d'un modèle intermédiaire s'opère par la traduction de ses différentes parties dans le langage cible. Elles prennent en paramètre le langage cible et le fichier résultat.

Chapitre 5

Validation à partir de premiers exemples

Sommaire

5.1 Validation à partir de modèles BIOCHAM	122
5.1.1 Les processus biologiques modélisés	123
5.1.2 Compartimentation cellulaire et graphe d'échanges	124
5.1.3 Règles génériques	125
5.1.4 Application de l'approche de couplage/traduction	128
5.1.5 Bilan	135
5.2 Validation à partir d'un modèle PATHWAY LOGIC	136
5.2.1 Processus modélisé : premières étapes de l'activation de Ras	136
5.2.2 compartimentation cellulaire et graphe d'échanges	137
5.2.3 Les règles génériques	138
5.2.4 Application de l'approche de couplage/traduction	140
5.2.5 Commentaires	142
5.3 Perspectives	142

Avant toute chose, je fais remarquer avec regret qu'en raison d'un certain nombre de facteurs¹, nous n'avons pas pu définir dans le cadre de ces travaux un nouveau modèle de processus biologique.

Dans ce chapitre, nous présentons et appliquons une démarche de validation de notre approche, validation qui vise à établir que les « bonnes » entrées produisent les sorties attendues. Nous rappelons que notre objectif général est d'offrir au biologiste-modélisateur un moyen de définir des modèles de processus biologiques dans l'outil cible de son choix (BIOCHAM ou PATHWAY LOGIC), avec la garantie que toutes les règles de réaction contenues dans ces modèles respectent la compartimentation cellulaire du système étudié. Aussi, la validation de notre approche consistera à établir que :

1. Conditions de déroulement de la thèse notamment : « thèse en alternance » entre le Burkina Faso et la France. Le temps que j'ai passé en France (quatre mois chaque année) n'a pas permis de travailler avec une équipe de biologistes de ce pays à l'identification et à la compréhension d'un phénomène à modéliser. D'un autre côté, le problème traité par le sujet ne mobilise pas tellement les chercheurs au Burkina Faso : je n'ai pas pu identifier une équipe qui travaille à cette catégorie de problématique. Par ailleurs, étant sous contrat professionnel à temps plein, je n'ai pas pu me consacrer entièrement à la thèse.

- l’abstraction de la compartimentation cellulaire sous forme de graphe d’échanges rend fidèlement compte des compartiments et de leurs voisinages ;
- les règles intermédiaires issues du processus de couplage sont uniquement les instances de règles génériques respectant dans le graphe d’échanges les conditions de voisinage implicites et explicites ;
- la traduction de ces règles intermédiaires vers le langage cible, conserve scrupuleusement la sémantique (traduit exactement les mêmes réactions) avec éventuellement un nombre différent de règles de réaction.

Nous proposons de valider notre approche sur des modèles pris dans les deux outils cibles que nous découplerons, l’hypothèse étant que chaque modèle combine topologie et règles biochimiques. Cela nous permettra d’avoir les éléments en entrée de notre procédure de couplage et selon les résultats obtenus, nous pourrons nous prononcer sur la validité de notre approche. Ainsi dans un premier temps, nous considérons deux modèles BIOCHAM pour l’application de cette démarche de validation. Ensuite, nous nous intéresserons aux modèles PATHWAY LOGIC. Enfin, une petite discussion ouvrira des perspectives à plus ou moins court terme à ces travaux.

5.1 Validation à partir de modèles BIOCHAM

La distribution actuelle de BIOCHAM intègre des modèles de processus biologiques, regroupés en fonction des aspects du langage qui y sont mis en exergue ou en fonction des processus modélisés. Ainsi, on a les groupes *kinetics*, *celcycle*, *locations* etc. Nous proposons dans le tableau 5.1 (page 122) une brève description des deux modèles enregistrés dans la rubrique Locations : il s’agit du modèle de la *signalisation delta notch* adapté de [?] et du modèle du *réseau P53/Mdm2*² adapté de [?]. Le tableau donne pour chaque modèle le nombre de compartiments, celui de molécules (au sens BIOCHAM), de molécules de base et de molécules localisées, et le nombre de règles de réaction.

Nombre de :	p56Mdm2	signalisation Delta - Notch
compartiments	2 (n et c)	36 ($cij, i, j \in [1, 6]$)
molécules	7	72
molécules de base	3	2
molécules localisées	3	72
règles de réaction	20	144

A partir de ces éléments, nous retenons comme modèle à utiliser dans le cadre de la présente validation, le modèle de la *signalisation Delta - Notch*³ dans lequel selon le tableau 5.1, toutes les molécules sont localisées. En effet, pour être transcribable dans notre langage de règles, un modèle doit respecter une des deux conditions suivantes :

- aucune molécule n’est localisée ;
- toutes les molécules sont localisées.

Le principal intérêt de ce modèle qui contient des règles de réactions simples est l’importance de la compartimentation cellulaire, qui y est déterminante pour l’application des règles. Nous utilisons ce modèle pour

2. <http://contraintes.inria.fr/BIOCHAM/EXAMPLES/locations/p53Mdm2.bc>

3. <http://contraintes.inria.fr/BIOCHAM/EXAMPLES/locations/notch4n36c.bc>

valider l'extraction du graphe d'échanges, ainsi que l'utilisation de la fonction *Neighb* dans l'expression des paramètres cinétiques. En raison même de la simplicité des règles de ce modèle, nous allons utiliser un autre modèle, celui du *cycle cellulaire réduit de Tyson*⁴ qui contient des règles de réaction plus complexes, pour valider notre approche sur la préservation de la sémantique lors de la traduction. Comme ce modèle ne contient que des molécules non localisées, nous n'allons pas l'utiliser pour valider l'extraction du graphe d'échanges qui est trivial.

5.1.1 Les processus biologiques modélisés

5.1.1.1 Le cycle cellulaire

Selon [?, ?]

” le cycle cellulaire est l'ensemble des modifications qu'une cellule subit entre sa formation par division à partir d'une cellule mère et le moment où cette cellule a fini de se diviser en deux cellules filles. La durée du cycle cellulaire varie d'une espèce à l'autre et d'un type cellulaire à l'autre.”

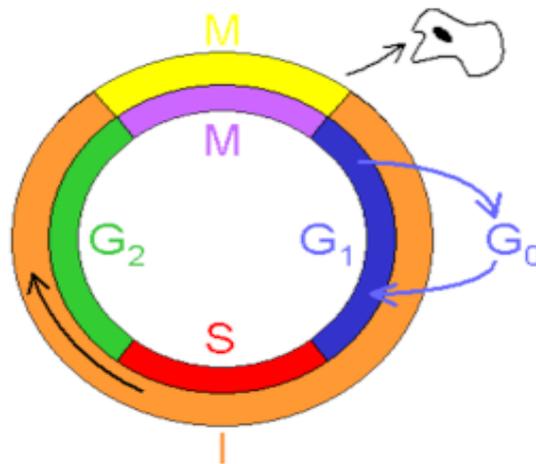


FIGURE 5.1 – Différentes phases du cycle cellulaire

Toutes nos autres sources [?, ?, ?] s'accordent sur le fait qu'il s'agit d'un mécanisme central qui régule la division cellulaire, clé du développement d'un organisme. La figure⁵ 5.1 montre les quatre phases que compte le cycle cellulaire : la phase G_1 qui contient une « sous-phase » G_0 pendant laquelle la cellule reste en repos. Ensuite, on a la phase S qui est celle de synthèse protéique commençant par la répllication de l'ADN. La phase G_2 précède la phase M qui est celle de division de la cellule.

Deux types de protéines présentent une activité importante durant ces quatre phases : il s'agit des *cyclines* et des *kinases cycline-dépendantes (cdk)* dont l'activité dépend de la liaison avec une cycline. En fonction des phases, les concentrations des différentes sortes de cyclines et de kinases cycline-dépendantes varient et l'atteinte de certains seuils déclenche le passage à la phase suivante. Le cycle cellulaire est caractérisé par des séries de complexations cycline-kinases cycline-dépendantes et de phosphorylations par des kinases ou des phosphatases.

4. http://contraintes.inria.fr/BIOCHAM/EXAMPLES/cell_cycle_Tyson_6Var/tyson91.bc

5. Source : Wikipédia

Le cycle cellulaire constitue un sujet bien documenté en raison principalement du rôle clé du processus dans l'organisme, qui a amené beaucoup de chercheurs à s'y intéresser. Ainsi, les modèles décrivant le cycle cellulaire sont relativement détaillés avec (pour les modèles à base de règles) un grand nombre de règles de réaction. Cependant, le modèle que nous proposons de traiter dans ce document est celui dit du *cycle cellulaire réduit de Tyson*, transcrit à partir des travaux de tyson [?] sur la modélisation des interactions cycline cdc2 au cours de la division cellulaire. Ce modèle comporte dix règles de réaction et deux molécules de base : la kinase cycline-dépendantes Cdc2 et la cycline Cyclin.

5.1.1.2 La signalisation Delta-Notch

La voie de signalisation Delta-Notch [?, ?] (couramment appelée *voie de signalisation Notch*) est une cascade de signalisations juxtacrines que l'on retrouve dans la quasi-totalité des organismes vivants : elle est conservée chez tous les métazoaires, du nématode *C.elegans* aux vertébrés supérieurs tels que l'homme. Elle contrôle la capacité de nombreux types de cellules à se différencier et implique essentiellement deux molécules : le *récepteur trans-membranaire Notch* dont le gène codant est le gène Notch et un ligand trans-membranaire *Delta* ou *Serrate*; en l'occurrence dans notre exemple, il s'agit de Delta.

L'activation du récepteur Notch est déclenchée par une liaison avec le ligand Delta fourni par une cellule juxtaposée à celle dans laquelle se déroule la signalisation. Cette liaison entraîne un premier clivage de Notch par l'enzyme Kuzbanian, libérant la partie extra-cellulaire et activant Notch. Un second clivage par la γ -sécrétase libère la partie intracellulaire dans le cytoplasme en conservant un court segment trans-membranaire. Le fragment intra cytoplasmique migre alors vers le noyau où il s'associe à d'autres protéines (*CSL*) pour former un complexe régulant la transcription de nombreux gènes cibles, notamment du gène Notch. Notons que la voie de signalisation Notch est un processus biologique très important. Des études [?, ?] ont montré que de nombreuses pathologies parfois létales sont associées à une mutation au niveau du gène Notch, ce qui donne lieu à d'autres travaux [?] qui explorent des stratégies bio-pharmacologiques pour modifier *in vivo* la signalisation Notch.

Les autres protéines intervenant dans cette voie de signalisation notamment au niveau des clivages et de la transcription ne sont pas mentionnées dans le modèle que nous proposons de traiter dans le présent document. Il s'agit d'un modèle comportant deux molécules de base et soixante douze règles de réaction.

5.1.2 Compartimentation cellulaire et graphe d'échanges

5.1.2.1 Le cycle cellulaire

Comme nous l'avons déjà dit, dans le cycle cellulaire réduit de Tyson, aucune information de localisation n'est donnée. Dans ce cas, nous considérons le processus comme se déroulant en vase clos, dans l'environnement.

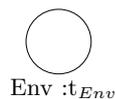


FIGURE 5.2 – Graphe d'échanges pour l'exemple du cycle cellulaire : $GE_{cellCycle}$

Le graphe d'échanges est ainsi réduit au seul compartiment représentant l'environnement Env de type t_{Env} . Nous le construisons donc directement, sans construire préalablement la compartimentation cellulaire dont il est issu, comme le montre la figure 5.2.

5.1.2.2 La signalisation Delta-Notch

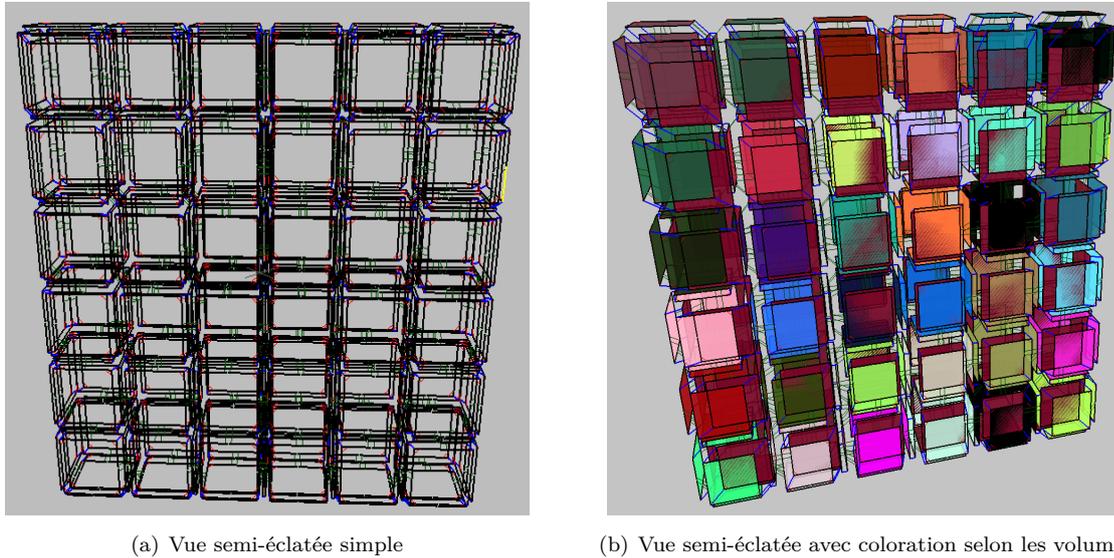


FIGURE 5.3 – Compartiment cellulaire pour l'exemple de la signalisation Delta-Notch

La signalisation Delta-Notch concerne un processus se déroulant dans une population de trente-six cellules disposées en matrice 6 X 6. Chaque cellule est désignée à l'aide de ses coordonnées en ligne et en colonne dans la matrice. Nous avons construit le modèle bio-géométrique correspondant, comme le montre la figure 5.3.

La figure 5.4 représente le graphe d'échanges extrait du modèle bio-géométrique précédent. Pour alléger la figure, seuls les noms des compartiments sont indiqués. En particulier, le type commun à tous les compartiments n'est pas affiché. Ce graphe d'échanges est bien conforme aux informations de voisinage entre les cellules données par [?] ainsi qu'aux arêtes données dans le fichier (annexe E) d'extraction du graphe d'échanges à partir du modèle bio-géométrique.

5.1.3 Règles génériques

5.1.3.1 Le modèle du cycle cellulaire

Le code complet des règles BIOCHAM est donné en annexe C (listing C.1). Les dix règles du modèle matérialisent des réactions de synthèse, de dégradation, de transformation et de complexations des deux molécules de base :

- rg1 est une règle de synthèse de la cycline (cyclin) dans l'environnement
- rg2 et rg8 sont des règles de dégradation respectivement de Cyclin et de la forme phosphorylée sur le site p1 de la même molécule

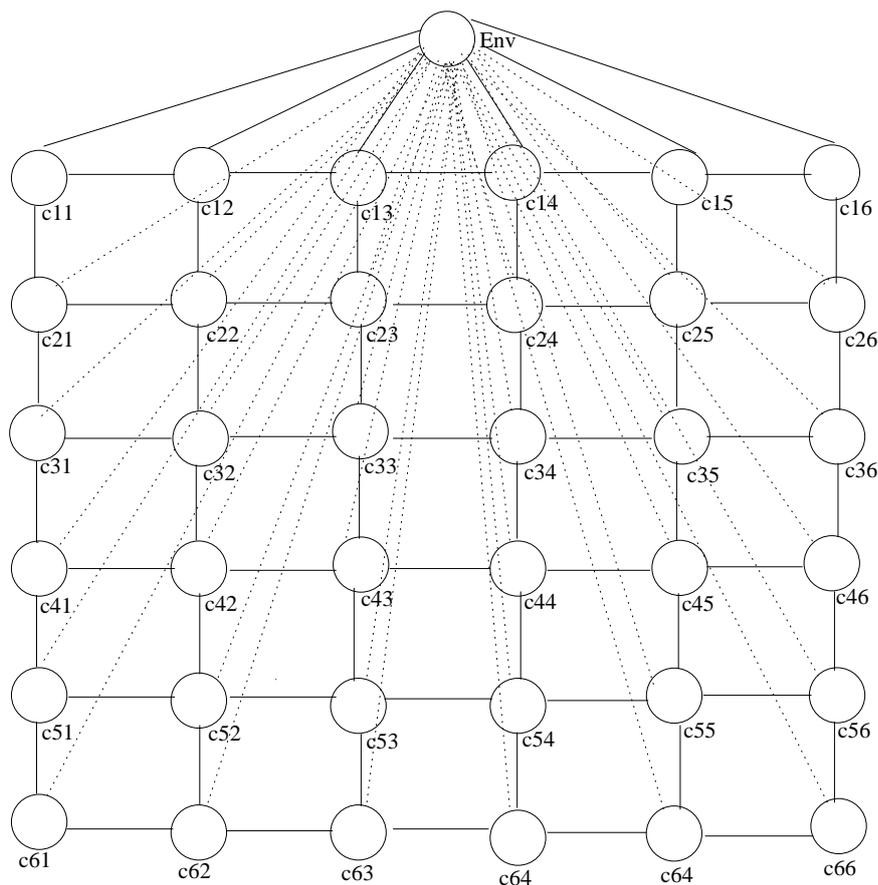


FIGURE 5.4 – Graphe d'échanges cohérent pour l'exemple du notch4n36c

```
((Cyclin : PHOS<p1>))
```

- rg3 matérialise une complexation de la cycline et de la forme phosphorylée en p1 de la kinase cycline-dépendante (cdc2). Le complexe résultant est ensuite phosphorylé en p2.
- rg4 et rg5 traduisent des réactions de déphosphorylation en p2 du complexe cyclin cdc2 doublement phosphorylé (en p1 et p2). Dans rg5, cette réaction est catalysée par la présence initiale du complexe résultat.
- rg6 est la réaction inverse de rg4. Elle consiste en une phosphorylation en p2 du complexe cyclin cdc2 phosphorylé en p1.
- rg7 traduit une décomplexation de cdc2 et de la forme phosphorylée en p1 de cyclin.
- rg9 est une réaction de phosphorylation de cdc2 en p1.
- rg10 est une réaction de déphosphorylation de cdc2 en p1.

Comme dans cet exemple, il n'y a pas de compartiment, les règles génériques qu'on obtient en considérant *Env* (de type *tEnv*) comme seul compartiment sont proches des règles BIOCHAM initiales. Le modèle générique (listing 5.1) compte dix règles génériques.

Listing 5.1 – Modèle générique du cycle cellulaire

```
1 BEGIN_TYPEC tEnv END_TYPEC
2 BEGIN_MOLSB MOLCLASS Protein {Cyclin(p1,p2); Cdc2(p1,p2)} END_MOLSB
3 BEGIN_MODIF PHOS END_MODIF
```

```

4 BEGIN_PARAM k1:0.015; k2:0; k3:200; k4p:0.018;
5 k4:180; k5:0; k6:1; k7:0.6; k8:100; k9:100 END_PARAM
6
7 BEGIN_REGEN
8 rg_1 : |k1| [] @Env=>[Cyclin]@Env.
9
10 rg_2: |(k2 * [Cyclin]@Env)| [Cyclin]@Env=>[]@Env.
11
12 rg_3: |(k3 * [Cyclin + (Cdc2:PHOS<p1>)]@Env)|
13 [Cyclin+(Cdc2:PHOS<p1>)]@Env=>[(Cdc2-Cyclin:PHOS<p1,p2>)]@Env.
14
15 rg_4: |(k4p * [(Cdc2-Cyclin:PHOS<p1,p2>)]@Env)|
16 [(Cdc2-Cyclin:PHOS<p1,p2>)]@Env=>[(Cdc2-Cyclin:PHOS<p1>)]@Env.
17
18 rg_5: |(k4*[(Cdc2-Cyclin:PHOS<p1>)+(Cdc2-Cyclin:PHOS<p1,p2>)+(Cdc2-Cyclin:PHOS<p1,p2>)]@Env)|
19 [(Cdc2-Cyclin:PHOS<p1,p2>) + (Cdc2-Cyclin:PHOS<p1>)]@Env
20 => [(Cdc2-Cyclin:PHOS<p1>) + (Cdc2-Cyclin:PHOS<p1>)]@Env.
21
22 rg_6: |(k5 * [(Cdc2-Cyclin:PHOS<p1>)]@Env)|
23 [(Cdc2-Cyclin:PHOS<p1>)]@Env=>[(Cdc2-Cyclin:PHOS<p1,p2>)]@Env.
24
25 rg_7: |(k6*[(Cdc2-Cyclin:PHOS<p1>)]@Env)|
26 [(Cdc2-Cyclin:PHOS<p1>)]@Env=>[(Cyclin:PHOS<p1>) + Cdc2]@Env.
27
28 rg_8: |(k7*[(Cyclin:PHOS<p1>)]@Env)|
29 [(Cyclin:PHOS<p1>)]@Env=>[]@Env.
30
31 rg_9: |(k8*[Cdc2]@Env)|
32 [Cdc2]@Env => [(Cdc2:PHOS<p1>)]@Env.
33
34 rg_10: |(k9*[(Cdc2:PHOS<p1>)]@Env)|
35 [(Cdc2:PHOS<p1>)]@Env=>[Cdc2]@Env.
36 END_REGEN

```

Le seul type de compartiment $tEnv$ se retrouve dans la section *TYPE_C* du modèle tandis que les deux molécules de base sont enregistrées dans la section *MOLSB*, chacune assortie de deux sites de modification qui se trouvent être les mêmes. Les dix (10) identificateurs de paramètres cinétiques

$$k_1, k_2, k_3, k_4, k_{4p}, k_5, k_6, k_7, k_8, k_9$$

avec leurs valeurs initiales font l'objet de la section *PARAM*. Dans la section *MODIFS*, est mentionnée comme seule opération de modification de forme de molécules *PHOS* mis pour phosphorylation. Cette information qui n'est pas explicitée dans le modèle initial est implicite, car *BIOCHAM* ne considère qu'une seule modification de forme de molécules, en l'occurrence la phosphorylation.

5.1.3.2 Modèle de la signalisation Delta-Notch inspiré de [?]

Les règles BIOCHAM initiales du modèle de la signalisation delta-notch sont données en annexe C, listing C.2

L'observation de ces soixante-douze (72) règles cibles, toutes réversibles, montre que tous les compartiments sont concernés par les mêmes règles qui sont au nombre de quatre et qui traduisent des réactions de synthèse et des réactions de dégradation des deux molécules D et N du modèle :

- la synthèse de N dans la cellule (rg1) qui est une règle conditionnelle ;
- la dégradation de N dans le cellule (rg2) ;
- la synthèse de D dans la cellule (rg3) qui est également une règle conditionnelle ;
- la dégradation de D dans le cellule (rg4) ;

Ce sont ces quatre réactions qui sont traduites à travers les règles du modèle générique dont le listing complet est donné (listing 5.2).

Listing 5.2 – Modèle générique de la signalisation Delta-Notch

```

1 BEGIN_TYPEC Cell END_TYPEC
2 BEGIN_MOLSB MOLCLASS Protein {D; N} END_MOLSB
3 BEGIN_PARAM ka : 1.0; kd : 1.0 END_PARAM
4 BEGIN_REGEN
5 rg_1 : | if (([D]@Neighb(Cell) < 0.2)) then 0.0 else ka| [] @Cell => [N]@Cell.
6 rg_2 : |MA(kd)| [N]@Cell => []@Cell.
7 rg_2 : | if (([N]@Cell < 0.5)) then 0.0 else ka| [] @Cell => [D]@Cell.
8 rg_4 : |MA(kd)| [D]@Cell => []@Cell.
9 END_REGEN

```

Le paramètre cinétique de rg1 montre

$$if([D]@Neighb(Cell) < 0.2) then 0 else ka$$

que ce sont les concentrations des molécules dans les cellules voisines (utilisation de *Neighb*) qui déterminent la survenue de la réaction. On note ici l'intérêt de cette fonction qui dans ce cas précis nous permet d'avoir un modèle concis. En effet, en l'absence de *Neighb*, nous aurions eu à définir les règles génériques en fonction du nombre de voisins des cellules. Ainsi, on aurait eu une règle pour les cellules à deux, trois, et quatre voisins.

5.1.4 Application de l'approche de couplage/traduction

Bien que les exemples de départ soient en BIOCHAM, nous allons traduire nos règles génériques à la fois en BIOCHAM et en PATHWAY LOGIC, afin de montrer la généralité de notre langage de règles génériques.

5.1.4.1 Cycle cellulaire

L'application de notre approche sur l'exemple du cycle cellulaire en BIOCHAM nous donne un modèle identique d'un point de vue sémantique au modèle initial (listing 5.3).

Listing 5.3 – Modèle réduit du cycle cellulaire de Tyson traduit vers BIOCHAM

```

1 parameter(k1,0.015).
2 parameter(k2,0).
3 parameter(k3,200).
4 parameter(k4p,0.018).
5 parameter(k4,180).
6 parameter(k5,0).
7 parameter(k6,1).
8 parameter(k7,0.6).
9 parameter(k8,100).
10 parameter(k9,100).
11
12 r_1 : k1 for _=>Cyclin.
13 r_2 : k2*[Cyclin] for Cyclin=>..
14 r_3 : k3*[Cyclin]*[Cdc2~{p1}] for Cyclin+Cdc2~{p1} => Cdc2-Cyclin~{p1,p2}.
15 r_4 : k4p*[Cdc2-Cyclin~{p1,p2}] for Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}.
16 r_5 : k4*[Cdc2-Cyclin~{p1}]*[Cdc2-Cyclin~{p1}]*[Cdc2-Cyclin~{p1,p2}] for Cdc2-Cyclin~{p1,
    p2}+Cdc2-Cyclin~{p1} => Cdc2-Cyclin~{p1}+Cdc2-Cyclin~{p1}.
17 r_6 : k5*[Cdc2-Cyclin~{p1}] for Cdc2-Cyclin~{p1} => Cdc2-Cyclin~{p1,p2}.
18 r_7 : k6*[Cdc2-Cyclin~{p1}] for Cdc2-Cyclin~{p1} => Cyclin~{p1}+Cdc2.
19 r_8 : k7*[Cyclin~{p1}] for Cyclin~{p1} =>..
20 r_9 : k8*[Cdc2] for Cdc2 => Cdc2~{p1}.
21 r10 : k9*[Cdc2~{p1}] for Cdc2~{p1} => Cdc2.
22
23 present(Cdc2,1).

```

On peut constater que chaque règle générique a donné lieu à une unique règle intermédiaire qui a été traduite dans le langage cible. Les seules différences notables sont au niveau de la règle issue de rg5. Dans le modèle initial, le paramètre cinétique de la règle dont est déduite rg.5 est :

$$k4 * ([Cdc2-Cyclin~{p1}])^2 * [Cdc2-Cyclin~{p1,p2}]$$

tandis que dans le modèle résultat il est :

$$k4 * [Cdc2-Cyclin~{p1}] * [Cdc2-Cyclin~{p1}] * [Cdc2-Cyclin~{p1,p2}]$$

Toujours par rapport à la même règle, nous pouvons constater que la forme abrégée des réactions catalysées de BIOCHAM est intéressante et qu'elle aurait pu s'intégrer avantageusement à notre langage de règles. En effet, la lecture de la règle initiale

$$k4 * ([Cdc2-Cyclin~{p1}])^2 * [Cdc2-Cyclin~{p1,p2}] \text{ for } \\ Cdc2-Cyclin~{p1,p2} = [Cdc2-Cyclin~{p1}] => Cdc2-Cyclin~{p1}.$$

est beaucoup plus facile à lire que celle de la règle obtenue par traduction

```
k4*[Cdc2-Cyclin~{p1}]*[Cdc2-Cyclin~{p1}]*[Cdc2-Cyclin~{p1,p2}] for
Cdc2-Cyclin~{p1,p2}+ Cdc2-Cyclin~{p1} => Cdc2-Cyclin~{p1}+Cdc2-Cyclin~{p1}.
```

Par ailleurs, on remarque qu'il n'est pas fait mention du compartiment *Env* qui était présent dans le modèle générique. Cela est voulu, car l'environnement ne fait pas partie de la compartimentation cellulaire et n'est pas déclaré explicitement par le biologiste modélisateur.

Remarque 5.1. *De façon générale, lorsque nous avons une compartimentation se résumant à l'environnement, nous ne le faisons pas apparaître dans les traductions. De même, lorsque la compartimentation comporte un seul compartiment de chaque type et que les localisations sont précisées dans les règles génériques, nous ne préfixons pas dans les règles traduites, ces localisations par les noms des compartiments.*

La section des molécules de base (*MOLBS*) n'est pas traduite. Ceci est normal parce que BIOCHAM considère toutes les molécules (de base, complexes ou modifiées) de la même manière : comme des objets formels. Cette flexibilité au niveau de BIOCHAM peut conduire à des situations inattendues : en se trompant sur une molécule dans une règle, le système considère le nom erroné comme une nouvelle molécule. Le passage par les règles génériques et le couplage permet d'éviter cela : en effet, seules les molécules déclarées peuvent apparaître dans les règles de réaction. De même, la section des modifications de forme de protéines (*MODIFS*) n'est pas non plus traduite pour la raison évidente que BIOCHAM comme nous l'avons déjà écrit n'explique pas les opérations de modification de forme.

On retrouve la transcription de la section des identificateurs de paramètres cinétiques (lignes 1 à 10). Les informations d'initialisation qui sont directement issues du graphe d'échanges sont données (ligne 23). On notera ici que contrairement au modèle initial qui précise les valeurs initiales pour chaque molécule, notre traduction se limite à donner les concentrations des molécules effectivement présentes (dans ce modèle, il s'agit de Cdc2).

Notons que l'ordre des sections importe peu : ainsi dans le modèle initial, l'état initial est placé avant l'ensemble des règles. On rencontre également des modèles (par exemple modèle de la signalisation delta notch) où c'est l'inverse. Pour terminer nous signalons l'absence (justifiée) de la macro, aspect que nous n'avons pas pris en compte dans notre langage de règles.

La traduction en PATHWAY LOGIC du modèle intermédiaire du cycle cellulaire a donné lieu à quatre fichiers (modOtherOps.maude, modComponents.maude, modRules.maude et qq.maude) présentés dans les listings 5.4 à 5.7.

Listing 5.4 – Modèle réduit du cycle cellulaire de Tyson : fichier modOtherOps.maude

```
1 fmod OTHEROPS is inc THEOPS .
2 op p1 : -> Site .
3 op p2 : -> Site .
4 op PHOS : -> Modification .
5 op PHOS : Site -> Modification .
6 endfm
```

Listing 5.5 – Modèle réduit du cycle cellulaire de Tyson : fichier modComponents.maude

```

1 fmod PROTEINOPS is inc THEOPS .
2 op Cdc2 : -> Protein .
3 op Cyclin : -> Protein .
4 endfm

```

Listing 5.6 – Modèle réduit du cycle cellulaire de Tyson : fichier modRules.maude

```

1 mod MODRULES is inc ALLOPS .
2 rl [1] : empty => Cyclin .
3 rl [2] : Cyclin => empty .
4 rl [3] : Cyclin [Cdc2-PHOS (p1)] => [[(Cdc2 : Cyclin) - PHOS (p1)] - PHOS (p2)] .
5 rl [4] : [[(Cdc2 : Cyclin) - PHOS (p1)] - PHOS (p2)] => [(Cdc2 : Cyclin) - PHOS (p1)] .
6 rl [5] : [[(Cdc2 : Cyclin) - PHOS (p1)] - PHOS (p2)] [(Cdc2 : Cyclin) - PHOS (p1)] => [(Cdc2
: Cyclin) - PHOS (p1)] [(Cdc2 : Cyclin) - PHOS (p1)] .
7 rl [6] : [(Cdc2 : Cyclin) - PHOS (p1)] => [[(Cdc2 : Cyclin) - PHOS (p1)] - PHOS (p2)] .
8 rl [7] : [(Cdc2 : Cyclin) - PHOS (p1)] => [Cyclin - PHOS(p1)] Cdc2 .
9 rl [8] : [Cyclin - PHOS(p1)] => empty .
10 rl [9] : Cdc2 => [Cdc2 - PHOS(p1)] .
11 rl [10] : [Cdc2 - PHOS(p1)] => Cdc2
12 endm

```

Listing 5.7 – Modèle réduit du cycle cellulaire de Tyson : fichier modQQ.maude

```

1 mod QQ is
2 inc ALLBP .
3 inc MODEL-CHECKER .
4 subsort Dish < State .
5
6 op etatinit : -> Dish .
7 eq etatinit = PD(Cdc2) .

```

Le fichier modOtherOps (listing 5.4) liste p1 et p2 comme sites de modification et déclare PHOS et PHOS associé à un site comme des modifications de forme de molécules. Ces deux informations sont issues d'une part de la section MOLSB et de la section MODIF du modèle générique. Les deux molécules sont déclarées dans le fichier modComponents.maude.

Considérant le fichier modRules.maude, on constate que les paramètres cinétiques n'apparaissent pas dans les règles. Par exemple rg3 est traduite par

```
rl[3] : Cyclin [Cdc2-PHOS (p1)] => [[(Cdc2 : Cyclin) - PHOS (p1)] - PHOS (p2)]
```

ignorant ainsi la vitesse de la transformation

```
|k3 * [cyclin + (Cdc2:PHOS<p1>)]|
```

Ceci est normal du moment que PATHWAY LOGIC ne prend pas en compte les paramètres cinétiques et de manière générale les aspects quantitatifs des modèles.

Nous attirons également l'attention sur la traduction des règles de synthèse et de dégradation telles que

```

r1[1] :empty => Cyclin
r1[9] : Cdc2 => empty
    
```

dans lesquelles la solution vide est donnée par *empty*. Il s'agit d'une convention que nous nous sommes fixée pour représenter cette catégorie de règles parce que la notion de solution vide n'est pas intégrée par PATHWAY LOGIC qui utilise plutôt des variables pour représenter le contenu non spécifié d'un compartiment. Mais étant dans un contexte de non localisation de nos règles, nous ne pouvons utiliser à ce niveau cette représentation.

5.1.4.2 Signalisation Delta-Notch

L'application de notre approche de couplage puis de traduction sur l'exemple de signalisation Delta-Notch nous donne un modèle BIOCHAM identique d'un point de vue sémantique au modèle initial (Compte tenu de la longueur de ce fichier, nous en avons extrait quelques règles que nous présentons listing 5.8.)

Listing 5.8 – extrait de la traduction vers BIOCHAM du modèle de la signalisation Delta-Notch

```

1 parameter(ka,1).
2 parameter(kd,1).
3
4 if [D::c12]+[D::c21] < 0.2 then 0 else ka for _::c11 => N::c11.
5 if [D::c13]+[D::c15]+[D::c24] < 0.2 then 0 else ka for _::c14 => N::c14.
6 if [D::c53]+[D::c55]+[D::c44]+[D::c64] < 0.2 then 0 else ka for _::c54 => N::c54.
7 MA(kd) for N::c11 => _::c11.
8 MA(kd) for N::c14 => _::c14.
9 MA(kd) for N::c54 => _::c54.
10 if [N::c41] > 0.5 then 0 else ka for _::c41 <=> D::c41.
11 MA(kd) for D::c35 => _::c35.
    
```

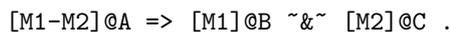
On peut constater à partir de cet extrait que les règles de réaction qui étaient toutes réversibles dans le modèle initial ont été traduites chacune en deux règles de réaction. Par exemple, la première règle (ligne 4)

```
(if [(D::c12) + (D::c21)<0,2 then 0 else ka, MA(kd)) for _ <=> N::c11.
```

donne dans le modèle traduit les deux règles suivantes :

```
if [D::c12] + [D::c21]<0,2 then 0 else ka for _::c11 => N::c11.  
MA(kd) for N::c11 => _::c11.
```

Cela est dû au fait que notre langage de règles n'autorise pas les réactions réversibles pour la principale raison que les conditions de voisinage entre compartiments que nous exigeons pour l'instanciation de nos règles ne sont pas forcément vérifiées dans les deux sens. Par exemple, considérons la règle générique suivante :



Elle exige dans le sens gauche droite, un voisinage direct entre le compartiment de type A et celui de type B et un autre entre le compartiment de type A et celui de type C (nous assimilerons le compartiment à son type dans la suite de cet exemple). Si cette règle était réversible, elle exigerait dans le sens droite gauche un voisinage direct entre B et C et un autre entre A et B ou entre A et C. On voit par là que les mêmes substitutions qui seraient générées dans un graphe d'échanges pour un sens ne marchent pas forcément pour l'autre sens. Ceci étant, nous n'intégrons naturellement pas les paramètres cinétiques qui valent pour les deux sens d'une règle réversible (comme ceux qu'on rencontre dans ce modèle). En effet, en considérant toujours la même règle (ligne 4 du listing),

```
if [D::c12] + [D::c21]<0,2 then 0 else ka et MA(kd)
```

sont respectivement utilisés dans le sens de la synthèse et dans celui de la dégradation.

On notera par ailleurs que la solution vide dans BIOCHAM n'est pas localisée tandis qu'elle l'est dans notre langage.

Cet exemple a permis de vérifier la prise en compte des relations de voisinage engageant des compartiments dont le contenu n'est pas affecté par la réaction. En effet, on peut voir dans l'exemple de la première règle que c12 et c21 apparaissent dans la règle sans que leur contenu soit affecté. Seul le contenu de c11 change.

Lors de la traduction en PATHWAY LOGIC, comme pour le modèle du cycle cellulaire, la signalisation Delta-Notch produit les quatre fichiers habituels (modOtherOps, modComponents, modRules et modQQ). Nous donnons ci-après (listing 5.9 à 5.11) les extraits (limités aux règles présentées dans la traduction vers BIOCHAM) des trois premiers fichiers cités.

Listing 5.9 – Modèle de la signalisation delta-notch : fichier modOtherOps.maude

```
1 fmod OTHEROPS is inc THEOPS .  
2 op c11 : -> LocName .  
3 op c12 : -> LocName .  
4 op c13 : -> LocName .  
5 op c14 : -> LocName .  
6 op c15 : -> LocName .  
7 op c21 : -> LocName .
```

```

8 op c24 : -> LocName .
9 op c44 : -> LocName .
10 op c53 : -> LocName .
11 op c54 : -> LocName .
12 op c55 : -> LocName .
13 op c64 : -> LocName .
14 endfm

```

Listing 5.10 – Modèle de la singalisation delta-notch : fichier modComponents.maude

```

1 fmod PROTEINOPS is inc THEOPS .
2 op D : -> Protein .
3 op N : -> Protein .
4 endfm

```

Listing 5.11 – Modèle de la singalisation delta-notch : fichier modRules.maude

```

1 mod MODRULES is inc ALLOPS .
2 vars cc11 cc14 cc13 cc14 cc15 cc21: -> Soup .
3 vars cc24 cc44 cc53 cc54 cc55 cc64: -> Soup .
4 rl : [c11 | cc11] => [c11 | cc11 N] .
5 rl : [c14 | cc14] => [c14 | cc14 N] .
6 rl : [c54 | cc54] => [c54 | cc54 N] .
7 rl : [c11 | cc11 N] => [c11 | cc11] .
8 rl : [c14 | cc14 N] => [c14 | cc14] .
9 rl : [c54 | cc54 N] => [c54 | cc54] .
10 rl : [c41 | cc41] => [c41 | cc41 D] .
11 rl : [c35 | cc35 D] => [c35 | cc35] .
12 endm

```

Tous les compartiments sont déclarés comme des localisations au niveau du fichier modOtherOps, tandis que les molécules sont déclarées dans le fichier modComponents.maude.

Considérant le fichier des règles modRules.maude, on s'aperçoit immédiatement de l'absence des paramètres cinétiques. Par exemple la dernière règle

```
(if [N::c35] > 0.5 then 0 else ka, MA(kd)) for _ <=> D::c35.
```

donne lieu aux deux règles suivantes :

```

r1 : [c35 | cc35] => [c35 | cc35 N] .
r1 : [c35 | cc35 N] => [c35 | cc35] .

```

On notera également dans le fichier modRules.maude la déclaration des variables de *Soup* (cc11, cc12 etc) qui représentent les contenus non détaillés des compartiments et qui sont utilisées au niveau des règles de réaction.

5.1.5 Bilan

Le tableau 5.1 propose un récapitulatif de certaines observations faites lors de l'application de notre démarche de validation sur ces deux modèles.

Indicateurs	Cycle cellulaire	Signalisation Notch
Intérêt du choix du modèle	Il s'agit de l'exemple « pédagogique » de BIOCHAM. Il nous a permis de traduire des règles à complexité moyenne, et de valider notre approche centrée sur les compartiments et leurs types, sur un modèle de processus se déroulant en vase clos sans précision d'une localisation.	La topologie dans ce modèle est importante. De plus l'information principale des règles est portée par les paramètres cinétiques. Nous avons ainsi pu vérifier que leur prise en compte est « soignée » dans notre approche.
Nombre de règles du modèle cible initial	10	72
Nombre de règles du modèle générique déduit	10	4
Nombre de règles du modèle cible après application de notre approche	10	144
Nombre de types de compartiment	0	1
Nombre de sommets de graphe d'échanges	1	37
Nombre d'arêtes du graphe d'échanges	0	98
Nombre de règles instanciées	10	4
Fidélité (Conservation de la sémantique sans ajout de nouvelles réactions) de la traduction	Oui	Oui sur les aspects pris en compte par PATHWAY LOGIC

TABLE 5.1 – Indicateurs sur les traductions du cycle cellulaire et de la signalisation de Notch

Notre démarche de validation a consisté dans cette première étape à découpler un modèle BIOCHAM existant, et à le reconstruire selon l'approche de couplage traduction que nous préconisons dans ce document. En l'appliquant avec succès sur le modèle du cycle cellulaire de BIOCHAM dans lequel les molécules ne sont pas localisées, nous montrons que l'approche qui s'intéresse en particulier aux comportements des types de compartiments, est relativement flexible.

L'application de la démarche de validation au modèle de la signalisation delta-notch avait pour objectif entre autre de tester la cohérence du graphe d'échanges extrait par rapport à la compartimentation cellulaire modélisée. En effet, la structure cellulaire considérée dans ce modèle est très *régulière* et les relations de voisinage entre compartiments qu'elle recèle peuvent être (difficilement) vérifiées à la main. De plus, la moitié des règles de réaction considérées dans ce modèle s'appuient essentiellement sur les voisinages entre les compartiments : pour un compartiment C considéré, les concentrations de la molécule D dans *tous* tous les compartiments voisins à C déterminent la possibilité d'une synthèse de N dans C . Le mot « tous » ici est important. En effet, en retrouvant à l'issue de notre processus de validation un modèle sémantiquement équivalent au modèle initial, nous montrons que le graphe d'échanges que nous avons extrait de notre représentation de la compartimentation étudiée est cohérent par rapport à cette compartimentation : il contient les mêmes compartiments et les mêmes relations de voisinage.

Cet exemple a également mis en lumière l'intérêt de la caractérisation des comportements des types de compartiments : si'il est admis que tous les compartiments d'un même type ont les mêmes réactions en présence des mêmes éléments biochimiques, une seule règle de réaction traduisant une de ces réactions suffit à rendre compte du comportement de tous les compartiments de ce type. Ainsi, nous avons eu besoin de seulement quatre règles génériques sur les types de compartiment pour exprimer les réactions d'une population de trente-six cellules. Notons que même s'il y en avait beaucoup plus, les quatre règles de réaction restent suffisantes.

5.2 Validation à partir d'un modèle PATHWAY LOGIC

PATHWAY LOGIC propose en téléchargement un certain nombre de modèles, parmi lesquels la base de connaissances *smallKB*. Ce modèle comporte des règles décrivant les premières réactions dans la cascade de signalisation RAS/MAPK. *SmallKB* est inclus par défaut dans le téléchargement du paquetage d'installation de PATHWAY LOGIC et est également utilisé pour les simulations en ligne de PATHWAY LOGIC. Nous reprenons ici cet exemple de référence.

5.2.1 Processus modélisé : premières étapes de l'activation de Ras

Selon [?] :

” La voie Ras/MAPK est une voie de signalisation intracellulaire qui joue un rôle important dans la régulation de la prolifération, de la survie, de la différenciation et de la migration cellulaire, ainsi que de l'angiogenèse. Elle est anormalement activée dans de nombreux cancers dont le cancer colorectal. Les mécanismes d'activation de cette voie sont principalement l'activation de récepteurs membranaires tels que l'EGFR, mais aussi la survenue de mutations somatiques, notamment au niveau des gènes codant pour la protéine Ras ou la protéine Raf. Une meilleure connaissance de cette voie de signalisation et des altérations oncogéniques en son sein ont permis l'identification de cibles thérapeutiques anticancéreuses potentiellement intéressantes, mais aussi de facteurs prédictifs de résistance à certaines thérapies anti-EGFR. ”

La cascade Ras/MAPK est une des voies de signalisation les plus étudiées en raison du rôle important qu'elle joue dans l'organisme. Elle est déclenchée par l'activation de la protéine Ras par des stimulus extra-cellulaires tels que la fixation du facteur de croissance épidermique *EGF* à son récepteur *EGFR*. Une fois activée, Ras déclenche une cascade de phosphorylations conduisant à la transcription de gènes impliqués dans la division cellulaire. Il est donc important pour éviter une prolifération de cellules, que la protéine Ras ne soit pas continuellement active, ce qui est le cas lors d'une mutation du gène Ras.

La figure 5.5⁶ montre la voie de signalisation des récepteurs aux *EGF* (*ErbB*).

La première étape de cette voie consiste en une liaison entre un récepteur membranaire (*Egfr* dans le modèle) et un facteur de croissance (*ErbB* dans le modèle) qui transmet le signal extra-cellulaire à l'intérieur de la cellule. Cette stimulation du récepteur l'amène à adopter sa forme active en se dimérisant. Dans cette conformation, le récepteur va déclencher dans le cytoplasme une cascade de phosphorylations de nombreuses protéines. Cela a pour effet l'activation de certaines voies de signalisation (parmi lesquelles

6. Source [?]

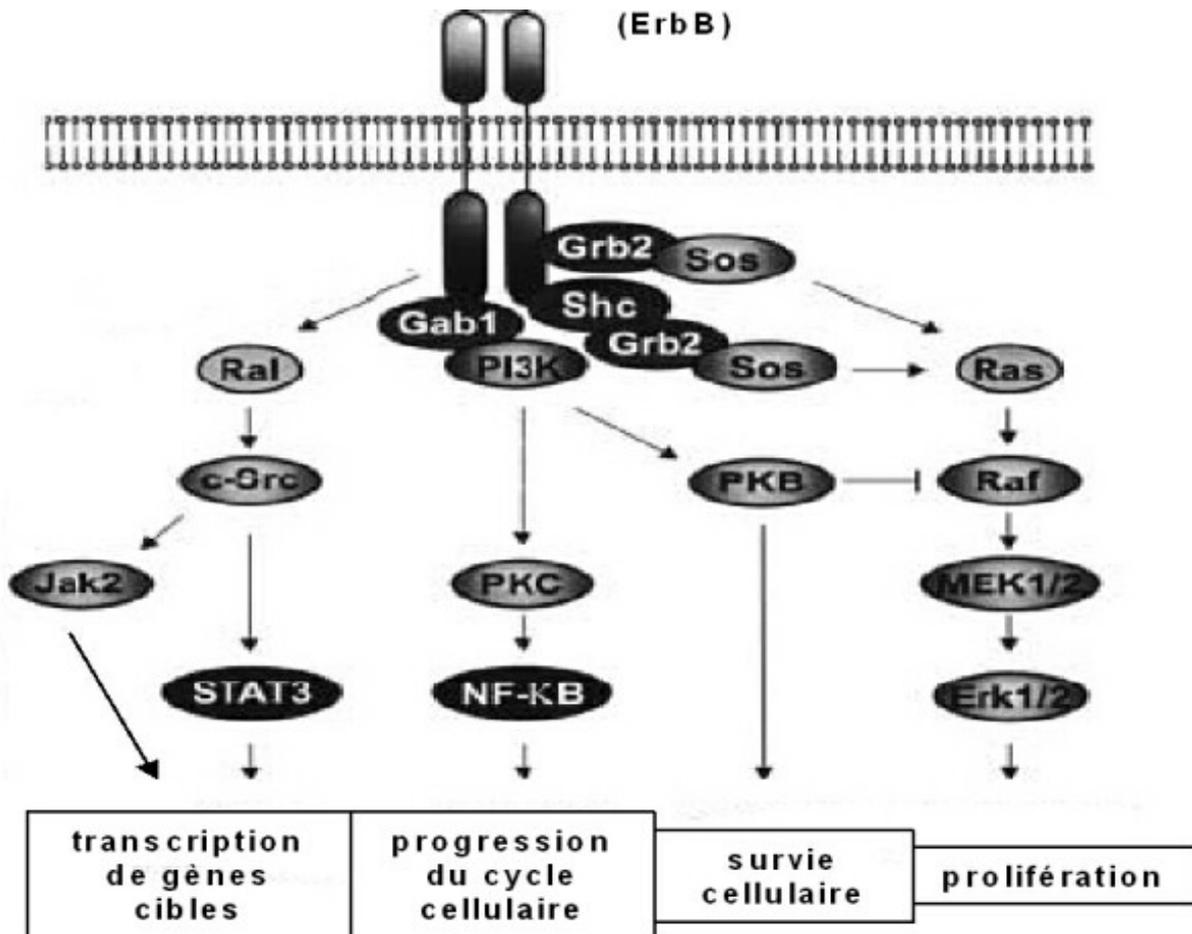


FIGURE 5.5 – Schéma de la voie Ras/MAPK

celles de Ras/MAPK) aboutissant à l'entrée en jeu de facteurs de transcription pouvant activer la transcription des gènes impliqués dans la progression du cycle cellulaire, la survie cellulaire et la prolifération cellulaire.

5.2.2 compartimentation cellulaire et graphe d'échanges

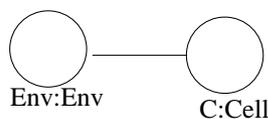


FIGURE 5.6 – Graphe d'échanges pour la voie RAS/MAPK

A partir du fichier des règles du modèle initial (listing C.4), on a l'information que la partie modélisée pour ce processus se déroule dans un unique compartiment qui est la cellule. Dans ce modèle les différentes localisations sont des parties de cellule : membrane cellulaire, intérieur de la cellule et également l'extérieur qui pour nous est l'environnement. Notre graphe d'échanges (figure 5.6) comporte donc deux compartiments : la cellule et l'environnement qui sont liés par une arête. Cependant étant donné que ce graphe

est le seul de tous ceux que nous avons vus jusque là à préciser les localisations, nous avons représenté la compartimentation sous-jacente pour en extraire l'en extraire. Cela a donné lieu au fichier 5.12 suivant.

Listing 5.12 – Fichier d'extraction du graphe d'échanges pour le modèle de l'activation de Ras

```

1 GRAPHE D'ECHANGES
2 TYPES_COMPARTIMENT
3 Cell, tEnv
4 FIN_TYPES_COMPARTIMENT
5
6 COMPARTIMENTS
7 C : Cell[CLo CLm CLi CLc] ; Env : TEnv
8 FIN_COMPARTIMENTS
9
10 ARETES
11 {C,Env}
12 FIN_ARETES

```

Ce fichier nous donne en ligne 7 les dénominations des différentes localisations de la cellule : CLo pour l'extérieur, CLm pour la membrane, CLi pour la face interne de la membrane et CLc pour l'intérieur de la cellule.

5.2.3 Les règles génériques

Nous donnons en annexe (listings C.3 et C.4) des versions dépouillées de leurs commentaires des fichiers `components.maude` et `rules.maude` du modèle. En raison de la longueur du fichier `theops.maude`, nous ne l'incluons pas dans le document.

Le modèle comporte dix-sept règles impliquant trente-sept molécules et traduisant des réactions diverses :

- rl[2] (ligne 12) concerne l'activation de EgfR par fixation avec un facteur de croissance épidermique (classe de molécule ErbB1L).
- rl[2] (ligne 21) concerne la dimérisation de EgfR.
- rl[4] (ligne 32) est la phosphorylation de Gab1 sur le site Y.
- rl[5] (ligne 42) traduit une relocalisation de Grb2 qui passe de l'intérieur de la cellule à la face interne de la membrane.
- rl[6] (ligne 52) traduit l'activation de Hras par association avec GTP.
- rl[7] (ligne 60) est une synthèse de IP3 par PIP2 sous l'action de Plc.
- rl[8] (ligne 70) traduit l'activation de Pi3k qui passe de l'intérieur de la cellule à la face interne de la membrane.
- rl[9] (ligne 78) est une transformation de PIP2 en PIP3 sous l'action de Pi3k.
- rl[10] (ligne 86) concerne l'activation de Plcg qui passe de l'intérieur de la cellule à la face interne de la membrane.
- rl[11] (ligne 96) est la phosphorylation sur la face interne de la membrane, de Shc sur le site Y.
- rl[12] (ligne 106) traduit le passage de Sos de la face interne de la membrane vers l'intérieur de la cellule.
- rl[13] (ligne 116) est la réaction inverse de rl[12].

- rl[15] (ligne 124) consiste en une phosphorylation de Cbl et en sa relocalisation sur la membrane interne de la cellule.
- rl[56] (ligne 134) concerne une activation de Pak1 qui passe ensuite du cytoplasme à la face interne de la membrane.
- rl[280] (ligne 144) consiste en l'activation de Raf1 par Hras dans le cytoplasme.
- rl[14] (ligne 152) traduit l'activation de Vav2 sur la face interne de la membrane.
- rl[clt1.ctest] (ligne 162) traduit une liaison entre les molécules Src, [Vav2-act] et [Cbl-Yphos].

Listing 5.13 – Modèle générique de l'activation de Ras

```

1 BEGIN_TYPEC Env, Cell END_TYPEC
2 BEGIN_MOLSB
3 MOLCLASS ErbB1L {Egf; Tgfa}
4 MOLCLASS Plc {Plcg; Plcg2}
5 MOLCLASS Protein {P1433x1; Cbl(p1, p2); DAG; EgfR; ErbB2; Gab1; Grb2; Hras; IP3; Pak1;
   Pi3k; Pik3ca; Pik3cb; Pik3cd; Pik3cg; Pik3c3; Pik3r1; Pik3r2; Pik3r3; PIP2; PIP3; PP2a; Raf1;
   Shc;
6     Sos1; Src; Ube213; Vav2}
7 END_MOLSB
8 BEGIN_MODIF act, ubiq, Yphos, reloc, GDP, GTP END_MODIF
9
10 BEGIN_REGEN
11 rg_1 : [ErbB1L]@Env & [EgfR]@Cell(TM)
12     => [(EgfR:act)-ErbB1L]@Cell(TM).
13
14 rg_2 : [(EgfR:act)-ErbB1L]@Cell(TM) & [(Cbl:Yphos) + (Ube213:ubiq)]@Cell(IM)
15     => [(EgfR:ubiq)-ErbB1L]@Cell(TM) & [(Cbl:Yphos) + Ube213]@Cell(IM).
16
17 rg_4 : [(EgfR:act)-ErbB1L]@Cell(TM) & [(Grb2:reloc)]@Cell(IM) & [Gab1]@Cell(IC)
18     => [(EgfR:act)-ErbB1L]@Cell(TM) & [(Grb2:reloc) + (Gab1:Yphos)]@Cell(IM).
19
20 rg_5 : [(EgfR:act)-ErbB1L]@Cell(TM) & [Grb2]@Cell(IC)
21     => [(EgfR:act)-ErbB1L]@Cell(TM) & [(Grb2:reloc)]@Cell(IM).
22
23 rg_6 : [PIP3]@Cell(TM) & [(Grb2:reloc) + (Sos1:reloc) + (Hras:GDP)]@Cell(IM)
24     => [PIP3]@Cell(TM) & [(Grb2:reloc) + (Sos1:reloc) + (Hras:GTP)]@Cell(IM).
25
26 rg_7 : [PIP2]@Cell(TM) & [(Plc : act)]@Cell(IM)
27     => [DAG]@Cell(TM) & [(Plc : act)]@Cell(IM) & [IP3]@Cell(IC).
28
29 rg_8 : [(Gab1 : Yphos)]@Cell(IM) & [Pi3k]@Cell(IC)
30     => [(Gab1 : Yphos)+(Pi3k:act)]@Cell(IM).
31
32 rg_9 : [PIP2]@Cell(TM) & [(Pi3k : act)]@Cell(IM)
33     => [PIP3]@Cell(TM) & [(Pi3k : act)]@Cell(IM).
34
35 rg_10 :[(EgfR:act)-ErbB1L + PIP3]@Cell(TM) & [Src]@Cell(IM) & [Plcg]@Cell(IC)
36     => [(EgfR:act)-ErbB1L + PIP3]@Cell(TM) & [Src + (Plcg : act)]@Cell(IM).

```

```

37
38 rg_11 :[( EgfR:act)–ErbB1L + PIP3]@Cell(TM) & [Src]@Cell(IM) & [Shc]@Cell(IC)
39       => [(EgfR:act)–ErbB1L + PIP3]@Cell(TM) & [Src + (Shc:Yphos)]@Cell(IM).
40
41 rg_12 :[( EgfR:act)–ErbB1L + PIP3]@Cell(TM) & [(Grb2:reloc)+(Sos1:reloc)]@Cell(IM)
42       => [(EgfR:act)–ErbB1L + PIP3]@Cell(TM) & [(Grb2:Yphos)]@Cell(IM) & [Sos1]@Cell(IC
43       ).
44
45 rg_13 : [(Grb2:reloc)]@Cell(IM) & [Sos1]@Cell(IC)
46       => [(Grb2:reloc)+(Sos1:reloc)]@Cell(IM).
47
48 rg_15 :[( EgfR:act)–ErbB1L + PIP3]@Cell(TM) & [Cbl]@Cell(IC)
49       => [(EgfR:act)–ErbB1L + PIP3]@Cell(TM) & [(Cbl:Yphos)]@Cell(IM).
50
51 rg_56 :[( EgfR:act)–ErbB1L + PIP3]@Cell(TM) & [Pak1]@Cell(IC)
52       => [(EgfR:act)–ErbB1L + PIP3]@Cell(TM) & [(Pak1:act)]@Cell(IM).
53
54 rg_280 :[( Hras:GTP) + (Pak1:act) + Src]@Cell(IM) & [Raf1+ P1433x1 + PP2a]@Cell(IC)
55       => [(Hras:GTP) + (Pak1:act) + Src + (Raf1:act) + P1433x1]@Cell(IM) & [PP2a]@Cell(IC
56       ).
57
58 rg_14 :[( EgfR:act)–ErbB1L]@Cell(TM) & [Src]@Cell(IM) & [Vav2]@Cell(IC)
59       => [(EgfR:act)–ErbB1L + PIP3]@Cell(TM) & [Src + (Vav2:act)]@Cell(IM).
60
61 rg_test :[Src+(Vav2:act)+(Cbl:Yphos)]@Cell(IM)
62       => [Src–(Vav2:act)–(Cbl:Yphos)]@Cell(IM).
63
64 END_REGEN

```

Le commentaire que suscite la vue de ce listing est que le modèle écrit en langage de règles est beaucoup plus compact que le modèle initial écrite en PATHWAY LOGIC. En effet, les deux fichiers (sans compter theops.maude et qq.maude) de ce dernier totalisent deux-cents huit (208) lignes contre soixante-deux (62) pour le modèle générique. Là encore, tout comme dans le modèle du cycle cellulaire déjà traité, une règle PATHWAY LOGIC a permis la déduction d’une unique règle générique. On note également l’absence (justifiée) de la section des identificateurs de paramètres cinétiques.

5.2.4 Application de l’approche de couplage/traduction

Nous obtenons après application de notre approche de couplage/traduction au modèle de l’activation de Ras, un modèle PATHWAY LOGIC sémantiquement équivalent au modèle initial. Du point de vue de la syntaxe également, l’équivalence est quasi-parfaite. Les différentes sections du modèle générique ont été concernées par la traduction. Il est à noter en particulier la transformation des localisations précises des molécules en utilisant les informations de localisation issues du graphe d’échanges :

- EM donne CLo;
- TM donne CLm;

- IM donne CLi ;
- IC donne CLc ;

Notons pour terminer que nous avons dû ajouter un « P » pour protéine à la molécule 1433x1 pour la compilation de nos règles génériques parce que dans ce langage les identificateurs sont sensés commencer par un caractère alphabétique.

Le listing 5.14 est le résultat de la traduction vers BIOCHAM du modèle de l'activation de Ras. Seule la section des règles génériques a été traduite en BIOCHAM. Plusieurs adaptations ont été nécessaires :

- Les règles de réactions dans lesquelles apparaissent des classes de molécules (par exemple ErbB1L, Plc) ont donné lieu à des schémas de règles de réaction. En effet, ces classes de molécules représentent plusieurs molécules et lorsqu'elles apparaissent dans une règle, elles sont considérées comme des variables des molécules de la classe. Aussi ayant connaissances de ces dernières grâce à la section MOLSB, on peut écrire le schéma de règle correspondant. Notons que l'ajout d'une molécule à la classe nécessite que tous les schémas de règles fondés sur cette classe soient mis à jour.
- La prise en compte des opérations de modification de forme. PATHWAY LOGIC en use beaucoup et comme nous l'avons déjà dit, BIOCHAM ne tient compte que de la phosphorylation. L'adaptation ici a été de considérer la modification comme un site pour pouvoir écrire la règle en BIOCHAM.

Listing 5.14 – Traduction vers BIOCHAM du modèle d'activation de Ras

```

1  ?ErbB1L::Env + EgfR::CLm => EgfR~{act}-?ErbB1L::CLm if ?ErbB1L in {Egf, Tgfa}.
2
3  EgfR~{act}-?ErbB1L::CLm + Cbl~{Yphos}::CLi+Ube213~{ubiq}::CLi .
4  => EgfR~{ubiq}-?ErbB1L::CLm + Cbl~{Yphos}::CLi+Ube213::CLi if ?ErbB1L in {Egf, Tgfa}.
5
6  EgfR~{act}-?ErbB1L::CLm + Grb2~{reloc}::CLi+Gab1::CLc
7  => EgfR~{act}-?ErbB1L::CLm + Grb2~{reloc}::CLi + Gab1~{Yphos}::CLi if ?ErbB1L in {Egf,
   Tgfa}.
8
9  EgfR~{act}-?ErbB1L::CLm + Grb2::CLc
10 =>if ?ErbB1L in {Egf, Tgfa}+Grb2~{reloc}::CLi if ?ErbB1L in {Egf, Tgfa}.
11
12 PIP3::CLm+Grb2~{reloc}::CLi+Sos1~{reloc}::CLi+Hras~{GDP}::CLi
13 =>PIP3::CLm+Grb2~{reloc}::CLi+Sos1~{reloc}::CLi+Hras~{GTP}::CLi.
14
15 PIP2::CLm + ?Plc~{act}::CLi
16 =>DAG::CLm+?Plc~{act}::CLi+IP3::CLc if ?Plc in {Plcg, Plcg2}.
17
18 Gab1~{Yphos}::CLi+Pi3k::CLc
19 => Gab1~{Yphos}::CLi + Pi3k~{act}::CLi.
20
21 PIP2::CLm+Pi3k~{act}::CLi => PIP3::CLm+Pi3k~{act}::CLi.
22
23 EgfR~{act}-?ErbB1L::CLm+Src::CLi+Shc::CLc
24 =>EgfR~{act}-?ErbB1L::CLm+Src::CLi+Plcg~{act}::CLi if ?ErbB1L in {Egf, Tgfa}.
25
26 EgfR~{act}-?ErbB1L::CLm+PIP3::CLm+Grb2~{reloc}::CLi+Sos~{reloc}::CLi

```

```

27 =>EgfR~{act}-?ErbB1L::CLm+PIP3::CLm+Grb2~{Yphos}::CLi+Sos1::CLc if if ?ErbB1L in {Egf,
    Tgfa}.
28
29 Grb2~{reloc}::CLi+Sos1::CLc => Grb2~{reloc}::CLi+Sos1~{reloc}::CLi.
30
31 EgfR~{act}-?ErbB1L::CLm+PIP3::CLm+Cbl::CLc
32 =>EgfR~{act}-?ErbB1L::CLm+PIP3::CLm+Cbl~{Yphos}::CLi if ?ErbB1L in {Egf, Tgfa}.
33
34 EgfR~{act}-?ErbB1L::CLm+PIP3::CLm+Pak1::CLc
35 ==>EgfR~{act}-?ErbB1L::CLm+PIP3::CLm+Pak1~{act}::CLi if ?ErbB1L in {Egf, Tgfa}.
36
37 Hras~{GTP}::CLi+Pak1~{act}::CLi+Src::CLi+Raf1::CLc+P1433x1::CLc+PP2a::CLc
38 =>Hras~{GTP}::CLi+Pak1~{act}::CLi+Src::CLi+Raf1~{act}::CLi+P1433x1::CLi+PP2a::CLc.
39
40 EgfR~{act}-?ErbB1L::CLm+Src::CLi+Vav2::CLc
41 =>EgfR~{act}-?ErbB1L::CLm+PIP3::CLm+Src::CLi+Vav2~{act}::CLi if ?ErbB1L in {Egf, Tgfa}.
42
43 Src::CLi+Vav2~{act}::CLi+Cbl~{Yphos}::CLi => Src-Vav2~{act}-Cbl~{Yphos}::CLi.

```

5.2.5 Commentaires

Les résultats obtenus par l'application de notre approche au modèle de l'activation de Ras, viennent confirmer la pertinence de considérer les systèmes cellulaires comme combinant topologie et réaction biochimiques.

5.3 Perspectives

Nous estimons en toute objectivité que la phase de validation de notre approche aurait gagné à modéliser un cas nouveau à partir de données prise dans la littérature, et à confronter les résultats des simulations et analyses à ceux répertoriés. Dans notre quête d'un processus cellulaire nouveau à modéliser, nous nous sommes rendus compte que la littérature ne regorge pas tant de processus engageant plusieurs cellules. Par ailleurs, les processus modélisés ne concernant qu'une seule cellule, toutes les localisations sont pratiquement voisines. Probablement en raison des limitations des outils actuels.

Pour démontrer pleinement le potentiel de la chaîne de modélisation que nous proposons, il conviendrait donc de traiter un nouveau cas d'étude comprenant une compartimentation plus riche. Cependant un tel travail doit ce faire en concertation avec des biologiste. Or comme nous l'avons rappelé en introduction de ce chapitre, cela n'a pas été possible dans le cadre de cette thèse.

A défaut de ce cas nouveau, nous souhaitons à très court terme créer un modèle extrapolé mais réaliste avec un nombre conséquent de compartiments et de règles génériques. Cela nous permettra de tester le comportement de notre outil sur des données plus volumineuses.

Nous avons dit dans le présent chapitre que notre approche s'appliquait dans un système où soit aucune molécule n'est localisée, soit toutes les molécules sont localisées. cela peut être restrictif eu égard

au nombre de modèles qui combinent des règles « mixtes ». Nous souhaitons nous pencher sur la question pour proposer un moyen crédible de considérer ces modèles dans notre approche.

Conclusion

L'objectif initial des présents travaux de recherche était de définir la première version d'un environnement générique dédié aux biologistes-modélisateurs pour la définition de modèles (à base de règles) de processus biologiques qui tiennent compte de la topologie des structures cellulaires étudiées. Le contexte de la modélisation des phénomènes biologiques étant entre autres caractérisé par l'existence de nombreux outils de modélisation offrant d'intéressantes capacités de simulation et d'analyse, nous exploitons cette donne en garantissant *en amont* que les modèles étudiés ne comportent que des règles de réaction *autorisées au vu de la structure cellulaire sous étude* tout en donnant des moyens souples et génériques pour définir les processus.

Pour résoudre cette problématique, nous avons adopté une approche qui consiste à construire les modèles en partant de deux structures : les compartimentations cellulaires et les règles génériques, structures que nous rapprochons sur la base de données de *types de compartiment*.

Résultats obtenus

Notre contribution peut concrètement se décliner selon les trois axes que sont 1) La modélisation des compartimentations cellulaires, 2) La définition d'un langage de règles génériques sur les types de compartiment et 3) Le couplage entre un modèle générique et une compartimentation cellulaire.

Selon le premier axe, nous nous sommes appuyés sur le noyau de MOKA, modeleur manipulant les cartes généralisées de dimension 3, pour implanter un modeleur dit *bio-géométrique* qui permet une représentation réaliste des structures cellulaires compartimentées. Dans l'implantation de ce modeleur, la structure des brins de MOKA a naturellement été conservée ainsi que le plongement géométrique des sommets nécessaire pour la visualisation des modèles. Nous avons défini un plongement spécifique, le plongement compartiment porté par les volumes topologiques qui donne l'identification des compartiments représentés ainsi que leur contenu en termes de molécules.

Le modeleur prend en compte deux types de relations de voisinage entre compartiments : l'inclusion d'un compartiment dans un autre et l'accolement de deux compartiments. Il permet à ce jour la création et la suppression de compartiments, l'attachement de molécules aux compartiments créés et l'extraction d'un graphe d'échanges cohérent par rapport à la compartimentation cellulaire modélisée.

De manière concrète, deux bibliothèques statiques ont été développées : `lib-gmapbiokernel` et `lib-controller-gmapbio`, ainsi qu'une interface graphique basique permettant l'utilisation de leurs fonctions. Une bibliothèque additionnelle `libGE` contient toutes les fonctions liées aux graphes d'échanges.

En ce qui concerne le deuxième axe, nous avons proposé un langage de règles *générique* permettant aux utilisateurs d'exprimer des réactions biochimiques caractérisant les comportements intrinsèques des types de compartiments en fonction des molécules qu'ils contiennent et des types des compartiments voisins.

Notre langage est suffisamment expressif pour capturer les réactions manipulables par l'outil BIOCHAM et la formalisation PATHWAY LOGIC de Maude. Il intègre de nombreux aspects de la modélisation à base de règles des processus cellulaires, analysables dans l'une ou l'autre de ces deux approches. Du point de vue de l'implantation, nous avons développé les outils de vérification de modèle et de manipulation des règles. Ils font l'objet des bibliothèques `libCompilerG` et `libModelGen`.

Quant au troisième axe, nous avons défini une procédure d'instanciation des règles génériques dans un graphe d'échanges respectant une topologie donnée, et sans relation de voisinage implicite. Les fonctions développées à cet effet sont regroupées dans la `libSubstitution`. La synthèse est réalisée dans des fonctions de traduction vers les modèles cibles BIOCHAM et PATHWAY LOGIC, en fonction du point de vue d'analyse que souhaite suivre l'utilisateur.

Le modéleur spécialisé réalisé nous permet de traiter des cas aussi complexes que ceux rencontrés dans la voie de signalisation Delta-Notch. Nous présentons une telle modélisation, avec les garanties apportées par notre construction.

Quelques perspectives

Notre bio-modéleur constitue une base solide pour un environnement de modélisation et d'analyse qui peut être étendu selon plusieurs directions.

À court terme et du point de vue outil, on peut dans un premier temps étoffer les primitives et les techniques de manipulation des compartiments, en particulier en ce qui concerne les multi-adjacences.

Nous avons montré que notre approche était réaliste en traitant des exemples non triviaux. Au delà de BIOCHAM et les formalisations PATHWAY LOGIC, il est maintenant envisageable de chercher à traduire nos modèles vers d'autres outils d'analyse à compartiments quasi-statiques et, par là, à étudier des cas de plus en plus complexes.

À moyen et long terme, des perspectives plus ambitieuses s'offrent à nous. Notre prototype de modéleur bio-géométrique considère comme composante unitaire l'objet compartiment ; il serait intéressant d'individualiser les membranes comme objets du langage et contenant à part entière de molécules afin de permettre une spécification des réactions membranaires (et donc leur étude). Une première approche pourrait alors prendre la forme d'une définition adéquate d'un modèle de plongement *face* ou *demi-face* pertinent. Cela permettrait d'attacher d'autres informations aux membranes telles que le sens de perméabilité des molécules, surface d'échanges offerte par la membrane, etc.

Dans le même ordre d'idées, les propriétés géométriques telles que le volume peuvent se révéler intéressantes à intégrer. De telles informations peuvent, à première vue, être prises en compte dans l'expression des paramètres cinétiques.

Remarquons qu'étendre l'expressivité de notre langage contraindra l'ensemble des outils cibles dans la mesure où nous pourrions éventuellement distinguer des situations trop fines pour certains d'entre eux, ou des propriétés hors de leur champ d'expertise.

On pourrait encore offrir de nouveaux niveaux de détail, en permettant une description plus précise des molécules elles-mêmes : sites de liaison, etc., certaines caractérisations étant susceptibles d'influer sur les résultats de simulation et d'analyse.

Enfin, les systèmes cellulaires sont des objets dynamiques. Prendre en compte cette dimension importante autoriserait la modélisation de divisions et fusions de compartiments, voire la migration de cellules, etc. qui sont pour l'instant hors d'atteinte de nos outils cibles.

Annexe A

Éléments de syntaxe concrète

A.1 Le modèle générique

Listing A.1 – Syntaxe de déclaration des différentes parties d'un modèle générique

```
1 MGen          ::= declTypesC declMolsBase declModifs declParam defRGs
2
3 declTypesC    ::= BEGIN_TYPEC lstDeclTypeC END_TYPEC
4 lstDeclTypeC ::= idTypeC | idTypeC;lstDeclTypeC
5
6 declMolsBase  ::= BEGIN_MOLSB lstDeclMolBase END_MOLSB
7 lstDeclMolBase ::= declMolBase | declMolBase; lstDeclMolBase
8 declMolBase   ::= idMolBase | idMolbase (lstSites)
9
10 declModifs    ::= BEGIN_MODIF lstModifs END_MODIF
11 lstModifs     ::= idModif | idModif, lstModifs
12
13 declParamC    ::= BEGIN_PARAM lstDeclParam END_PARAM
14 lstDeclParam  ::= declParam | declParam; lstDeclParam
15 declParam     ::= idParam(reel)
16
17 defRGs       ::= BEGIN_REGEN lstRGs END_REGEN
18 lstRGs       ::= rg | rg lstRGs
```

Le listing A.1 montre que chaque partie d'un modèle générique est introduite par un mot-clé : *BEGIN_TYPEC*, *BEGIN_MOLSB* etc. Il donne en lignes 3 à 4 les éléments de syntaxe relatifs à la déclaration des types de compartiments qui seront utilisés dans le modèle. La déclaration des molécules de base (ligne 6 à 8) suit à peu près la même logique que celle des types de compartiment. On y lit en particulier que celles-ci peuvent être déclarées avec la précision de leurs sites de modification (ligne 10). Par exemple

BEGIN_MOLSB cdc2; cyclin(p1,p2) END_MOLSB

déclare les molécules *cdc2* sans site de modification et *cyclin* avec deux sites de modification *p1* et *p2*.

La déclaration des opérations de modification de forme des molécules est donnée par les lignes 10 et 11. On peut constater que ces opérations consistent juste en des identificateurs. Par exemple

```
BEGIN_MODIF PHOS, ACTI, INHI END_MODIF
```

déclare trois opérations de modifications de forme de molécules. Les identificateurs de paramètres cinétiques sont déclarés (lignes 13 à 15) chacun avec une valeur initiale.

A.2 Syntaxe des règles génériques

Listing A.2 – Syntaxe des règles génériques

```

1  rg          ::= idrg : precondition pCinetiq etatCel => etatCel CondVarMol.
2
3  precondition ::= [PreC : lstCondV]
4  lstCondV    ::= CondV | condV, lstCondV
5  condV       ::= {varCompart, varCompart}
6
7  pCinetiq    ::= %paramC% | %if condition then paramC else paramC%
8  paramC      ::= reel | idParam | solLocalisee | paramC Op paramC
9              | fonction(paramC) | MM(paramC, paramC)
10             | H(paramC, paramC, entier) | (paramC)
11 Op          ::= * | / | + | -
12 fonction    ::= log | exp | cos | sin | frac | MA
13 condition   ::= paramC OpCompare paramC
14 OpCompare   ::= < | = | >
15
16 etatCel     ::= solLoc1 | etatNonVide
17 solLoc1     ::= []@varCompart
18 etatNonVide ::= solLoc2 | solLoc2 & etatNonVide
19 solLoc2     ::= [ensMol]@varCompart
20 ensMol      ::= molecule | molecule + ensMol
21 molecule    ::= idMolBase | (molecule : modif) | molecule-molecule
22 modif       ::= idModif | <lstSites> | idModif <lstSites>
23 lstSites    ::= idSite | idSite, lstSites
24 varCompart  ::= idVarC : idTypeCpartTC | idVarC | idTypeC partTC
25 partTC     ::= (idMemb) | (idMExt) | (idMInt) | (idInt) | -
26
27 CondVarMol ::= if (idVarMol in {ensMol}) suiteCondVM
28 suiteCondVM ::= and CondVarMol | -

```

La ligne 1 du listing A.2 donne la structure générale de la règle générique : on retrouve en effet les éléments de syntaxe pour les six différentes parties d'une règle générique que sont l'identificateur de la règle générique, la précondition de voisinage et le paramètre cinétique, les deux états cellulaires tenant lieu de membre gauche et de membre droit et la condition sur les variables de molécules.

Les identificateurs de paramètres cinétiques (*idParam*) que l'on retrouve dans les règles génériques doivent avoir été déclarés dans la section adéquate. De même tous les types de compartiment doivent avoir été déclarés. Même lorsque les sites de modification ont été précisés dans la déclaration d'une molécule de base, conformément à la ligne 21 du listing A.2, celle-ci apparaît sans mention des sites de modification dans les règles génériques. Par exemple le complexe formé de *cdc2* et de *cyclin* est

$$cdc2 - cyclin \text{ et non } cdc2 - cyclin(p1, p2)$$

Les lignes 24 et 25 correspondent à la syntaxe d'expression des variables de compartiment. La forme standard est celle comportant un identificateur de variable et un type de compartiment, les deux parties étant séparées par « : ». Par exemple

$$v_1 : cell$$

$$v_1 : cell(c/m)$$

désignent respectivement la variable de compartiment *v_1* de type *cell* et la variable *v_2* de type *cell* avec une précision de la localisation des molécules dans *cell*. À côté de cette forme standard, on a deux autres formes qui consistent en des abréviations se traduisant par l'omission dans l'expression de la variable, de l'une ou l'autre des deux parties : l'identificateur ou le type de compartiment. Les abréviations sont utilisables sous certaines conditions :

- l'abréviation qui consiste à omettre le type de compartiment ne peut apparaître que lorsqu'un type de compartiment lui avait déjà été associé dans la même règle générique. Ainsi, en dehors de la *variable de compartiment Env*, cette forme d'abréviation ne peut, en raison de la première condition de définition des états génériques, se retrouver que dans le membre droit d'une règle générique. Par exemple la règle générique

$$[]@v_1 \Rightarrow [A]@v_1 : Type_1$$

est une utilisation incorrecte de cette abréviation.

- l'abréviation qui consiste à omettre le nom de la variable peut apparaître partout dans la règle générique mais son utilisation nécessite d'être sûr de ce que l'on souhaite exprimer. Aussi, il est conseillé de ne l'utiliser que lorsque la règle générique ne contient qu'une seule variable du type concerné. Par exemple, la règle générique

$$[A]@v_1 : Type_1 \Rightarrow [A]@Type_1$$

sera interprétée comme faisant intervenir deux variables de compartiment, tandis que la règle générique

$$[]@Type_1 \Rightarrow [A]@Type_1$$

sera interprétée comme ne faisant intervenir qu'une unique variable de compartiment.

Annexe B

Parties immuables des fichiers d'un modèle PL

B.1 theops.maude

```
1 fmod PROTEIN is pr NAT .
2 sorts AminoAcid Protein .
3   subsort AminoAcid < Protein .
4 endfm
5 *****
6 fmod THING is pr PROTEIN .
7
8 sort Thing Family Composite DNA Complex Chemical Signature Stimulus .
9 subsorts Protein Family Composite DNA Complex Chemical Signature
10   Stimulus < Thing .
11
12   op (.:_) : Thing Thing -> Complex [assoc comm] .
13
14 **** removing duplicates in complexes for new complexes
15   op reduce : Complex -> Complex .
16   eq reduce((T1:Thing : T2:Thing)) = (T1:Thing : T2:Thing) .
17
18 endfm
19 *****
20 fmod SOUP is pr THING .
21
22   sort Soup .
23   subsort Thing < Soup .
24   op empty : -> Soup .
25   op -- : Soup Soup -> Soup [assoc comm id: empty] .
26
```

```

27   op _has_ : Soup Thing -> Bool .
28   eq (T1:Thing S:Soup) has T2:Thing =
29       if T1:Thing == T2:Thing then true else S:Soup has T2:Thing fi .
30   eq (S:Soup has T:Thing) = false [owise] .
31
32   ****!!! cltfix
33   op <_> : Soup -> Complex . **** for new complexes
34   op reduceS : Soup -> Soup .
35   eq reduceS(T:Thing T:Thing S:Soup) = reduceS(T:Thing S:Soup) .
36   eq reduceS(S:Soup) = S:Soup [owise] .
37
38   eq reduce(< s:Soup >) = < reduceS(s:Soup) > .
39   ****!!! end cltfix
40
41   *** A quick way to add more than one of the same thing to qt
42   op # : Nat Thing -> Soup .
43   eq #((s N:Nat),T:Thing) = T:Thing #(N:Nat,T:Thing) .
44   eq #(0,T:Thing) = empty .
45
46   endfm
47   *****
48   fmod MODIFICATION is pr SOUP .
49
50   sorts Site Modification ModSet .
51   subsort Modification < ModSet .
52
53   op none : -> ModSet .
54   op __ : ModSet ModSet -> ModSet [assoc comm id: none] .
55   op __ : AminoAcid Nat -> Site .
56
57   op _contains_ : ModSet Modification -> Bool .
58   var M M' : Modification .
59   var MS : ModSet .
60
61   eq none contains M' = false .
62   eq (M MS) contains M' = if M == M' then true
63       else MS contains M' fi .
64
65   op [_-] : Protein ModSet -> Protein
66       [right id: none] . *** format (nt d d d d d) .
67   op [_-] : Family ModSet -> Family
68       [right id: none] . *** format (nt d d d d d) .
69   op [_-] : Composite ModSet -> Composite
70       [right id: none] . *** format (nt d d d d d) .
71   op [_-] : DNA ModSet -> DNA
72       [right id: none] . *** format (nt d d d d d) .

```

```

73   op [_-] : Chemical ModSet -> Chemical
74           [right id: none ] . *** format (nt d d d d d) ] .
75
76   endfm
77   *****
78   fmod LOCATION is inc MODIFICATION .
79
80   sort Location LocName .
81   subsort Location < Soup .
82   op {_|_} : LocName Soup -> Location [format (n d d t d d)] .
83
84   endfm
85   *****
86   fmod CELL is inc LOCATION .
87
88   sort Cell CellType .
89   subsort Cell < Soup .
90
91   op [_|_] : CellType Soup -> Cell .
92   op Cell : -> CellType .
93
94   endfm
95   *****
96   fmod DISH is inc CELL .
97
98   sort Dish .
99   op PD : Soup -> Dish .
100
101  endfm
102
103  *****
104  fmod THEOPS is
105    inc DISH .
106  endfm
107  *****

```

B.2 modOtherOps.maude

```

1  fmod OTHEROPS is inc THEOPS .
2  **** Operations de modification de forme
3
4  **** Localisations
5
6  **** type de cellules
7

```

```
8 **** etc
9 endfm
```

B.3 modCompnents.maude

```
1 fmod PROTEINOPS is inc THEOPS .
2 **** Classes de molecules
3
4 **** Molecules
5 endfm
```

B.4 modRules.maude

```
1 mod MODRULES is inc ALLOPS .
2 **** variables de Soup
3
4 **** regles biochimiques
5 endm
```

Annexe C

Modèles pour la validation de l'approche

C.1 Modèles initiaux BIOCHAM

Listing C.1 – Modèle réduit du cycle cellulaire de Tyson (version initiale)

```
1  %A model of the cell cycle based on the interactions between cdc2 and cyclin.
2
3  present(Cdc2,1).
4  present(Cdc2~{p1},0).
5  present(Cyclin,0).
6  present(Cdc2-Cyclin~{p1,p2},0).
7  present(Cdc2-Cyclin~{p1},0).
8  present(Cyclin~{p1},0).
9
10 %macro(CT,[Cdc2-Cyclin~{p1,p2}]+[Cdc2-Cyclin~{p1}]+[Cdc2~{p1}]+[Cdc2]).
11
12 k1          for _=>Cyclin.
13 k2*[Cyclin] for Cyclin=>..
14
15 k3*[Cyclin]*[Cdc2~{p1}] for Cyclin+Cdc2~{p1} => Cdc2-Cyclin~{p1,p2}.
16
17 k4p*[Cdc2-Cyclin~{p1,p2}] for Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}.
18
19 k4*([Cdc2-Cyclin~{p1}])^2*[Cdc2-Cyclin~{p1,p2}]
20 for Cdc2-Cyclin~{p1,p2} =[Cdc2-Cyclin~{p1}]=> Cdc2-Cyclin~{p1}.
21
22 k5*[Cdc2-Cyclin~{p1}] for Cdc2-Cyclin~{p1} => Cdc2-Cyclin~{p1,p2}.
23
24 k6*[Cdc2-Cyclin~{p1}] for Cdc2-Cyclin~{p1} => Cyclin~{p1}+Cdc2.
25 k7*[Cyclin~{p1}] for Cyclin~{p1} =>..
```

```

26
27 k8*[Cdc2]          for Cdc2 => Cdc2~{p1}.
28 k9*[Cdc2~{p1}]    for Cdc2~{p1} => Cdc2.
29
30 macro(YT,[Cyclin]+[Cyclin~{p1}]+[Cdc2-Cyclin~{p1,p2}]+[Cdc2-Cyclin~{p1}]).
31
32 parameter(k1,0.015).
33 parameter(k2,0).
34 parameter(k3,200).
35 parameter(k4p,0.018).
36 parameter(k4,180).
37 parameter(k5,0).
38 parameter(k6,1).
39 parameter(k7,0.6).
40 parameter(k8,100).
41 parameter(k9,100).

```

Listing C.2 – Modèle de la signalisation Delta-Notch (version initiale)

```

1  % http://www.loria.fr/publications/2002/A02-R-026/A02-R-026.ps
2
3  parameter(ka,1).
4  parameter(kd,1).
5
6  %%% Notch protein production
7
8  % two neighbors
9
10 ( if [D::c12]+[D::c21] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c11.
11 ( if [D::c15]+[D::c26] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c16.
12 ( if [D::c62]+[D::c51] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c61.
13 ( if [D::c65]+[D::c56] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c66.
14
15 % 3 neighbors
16
17 ( if [D::c11]+[D::c13]+[D::c22] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c12.
18 ( if [D::c12]+[D::c14]+[D::c23] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c13.
19 ( if [D::c13]+[D::c15]+[D::c24] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c14.
20 ( if [D::c14]+[D::c16]+[D::c25] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c15.
21
22 ( if [D::c61]+[D::c63]+[D::c52] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c62.
23 ( if [D::c62]+[D::c64]+[D::c53] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c63.
24 ( if [D::c63]+[D::c65]+[D::c54] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c64.
25 ( if [D::c64]+[D::c66]+[D::c55] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c65.
26
27 ( if [D::c11]+[D::c31]+[D::c22] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c21.
28 ( if [D::c21]+[D::c41]+[D::c32] < 0.2 then 0 else ka,MA(kd) for _ <=> N::c31.

```

```

29 (if [D::c31]+[D::c51]+[D::c42] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c41.
30 (if [D::c41]+[D::c61]+[D::c52] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c51.
31
32 (if [D::c16]+[D::c36]+[D::c25] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c26.
33 (if [D::c26]+[D::c46]+[D::c35] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c36.
34 (if [D::c36]+[D::c56]+[D::c45] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c46.
35 (if [D::c46]+[D::c66]+[D::c55] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c56.
36
37 % 4 neighbors
38
39 (if [D::c21]+[D::c23]+[D::c12]+[D::c32] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c22.
40 (if [D::c22]+[D::c24]+[D::c13]+[D::c33] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c23.
41 (if [D::c23]+[D::c25]+[D::c14]+[D::c34] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c24.
42 (if [D::c24]+[D::c26]+[D::c15]+[D::c35] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c25.
43
44 (if [D::c31]+[D::c33]+[D::c22]+[D::c42] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c32.
45 (if [D::c32]+[D::c34]+[D::c23]+[D::c43] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c33.
46 (if [D::c33]+[D::c35]+[D::c24]+[D::c44] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c34.
47 (if [D::c34]+[D::c36]+[D::c25]+[D::c45] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c35.
48
49 (if [D::c41]+[D::c43]+[D::c32]+[D::c52] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c42.
50 (if [D::c42]+[D::c44]+[D::c33]+[D::c53] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c43.
51 (if [D::c43]+[D::c45]+[D::c34]+[D::c54] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c44.
52 (if [D::c44]+[D::c46]+[D::c35]+[D::c55] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c45.
53
54
55 (if [D::c51]+[D::c53]+[D::c42]+[D::c62] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c52.
56 (if [D::c52]+[D::c54]+[D::c43]+[D::c63] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c53.
57 (if [D::c53]+[D::c55]+[D::c44]+[D::c64] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c54.
58 (if [D::c54]+[D::c56]+[D::c45]+[D::c65] < 0.2 then 0 else ka,MA(kd)) for _ <=> N::c55.
59
60
61 %%% Delta protein production
62
63 (if [N::c11] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c11.
64 (if [N::c12] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c12.
65 (if [N::c13] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c13.
66 (if [N::c14] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c14.
67 (if [N::c15] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c15.
68 (if [N::c16] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c16.
69 (if [N::c21] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c21.
70 (if [N::c22] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c22.
71 (if [N::c23] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c23.
72 (if [N::c24] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c24.
73 (if [N::c25] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c25.
74 (if [N::c26] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c26.

```

```

75 (if [N::c31] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c31.
76 (if [N::c32] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c32.
77 (if [N::c33] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c33.
78 (if [N::c34] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c34.
79 (if [N::c35] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c35.
80 (if [N::c36] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c36.
81 (if [N::c41] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c41.
82 (if [N::c42] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c42.
83 (if [N::c43] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c43.
84 (if [N::c44] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c44.
85 (if [N::c45] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c45.
86 (if [N::c46] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c46.
87 (if [N::c51] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c51.
88 (if [N::c52] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c52.
89 (if [N::c53] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c53.
90 (if [N::c54] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c54.
91 (if [N::c55] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c55.
92 (if [N::c56] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c56.
93 (if [N::c61] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c61.
94 (if [N::c62] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c62.
95 (if [N::c63] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c63.
96 (if [N::c64] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c64.
97 (if [N::c65] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c65.
98 (if [N::c66] > 0.5 then 0 else ka,MA(kd)) for _ <=> D::c66.
99
100
101 %%% Initial state
102
103 % stochastic distribution :
104 % +/- 30% about the value 1 in the first article
105 % normal distribution with unity mean and a variance of 0.05 in the second article
106 % here we choose the second option
107
108 present(D::c11,1) . present(N::c11,0.91) .
109 present(D::c12,1.02) . present(N::c12,1) .
110 present(D::c13,1) . present(N::c13,0.95) .
111 present(D::c14,1) . present(N::c14,1) .
112 present(D::c15,1.06) . present(N::c15,1) .
113 present(D::c16,1.06) . present(N::c16,1) .
114 present(D::c21,0.97) . present(N::c21,1) .
115 present(D::c22,1.04) . present(N::c22,1.01) .
116 present(D::c23,1.02) . present(N::c23,0.95) .
117 present(D::c24,1) . present(N::c24,1.03) .
118 present(D::c25,1) . present(N::c25,1) .
119 present(D::c26,1.06) . present(N::c26,1.09) .
120 present(D::c31,1.02) . present(N::c31,1) .

```

```

121 present(D::c32,0.99) . present(N::c32,1.01) .
122 present(D::c33,1) . present(N::c33,0.98) .
123 present(D::c34,1) . present(N::c34,0.96) .
124 present(D::c35,1.12) . present(N::c35,1.06) .
125 present(D::c36,1.12) . present(N::c36,1.06) .
126 present(D::c41,0.94) . present(N::c41,0.99) .
127 present(D::c42,1.03) . present(N::c42,1.01) .
128 present(D::c43,1) . present(N::c43,1.02) .
129 present(D::c44,0.91) . present(N::c44,0.99) .
130 present(D::c45,1.1) . present(N::c45,0.99) .
131 present(D::c46,1.1) . present(N::c46,0.99) .
132 present(D::c51,0.94) . present(N::c51,0.99) .
133 present(D::c52,1.03) . present(N::c52,1.01) .
134 present(D::c53,1) . present(N::c53,1.02) .
135 present(D::c54,0.91) . present(N::c54,0.99) .
136 present(D::c55,1.11) . present(N::c55,0.99) .
137 present(D::c56,1.0) . present(N::c56,0.99) .
138 present(D::c61,0.94) . present(N::c61,0.99) .
139 present(D::c62,1.03) . present(N::c62,1.01) .
140 present(D::c63,1) . present(N::c63,1.02) .
141 present(D::c64,0.91) . present(N::c64,0.93) .
142 present(D::c65,0.87) . present(N::c65,0.99) .
143 present(D::c66,1.1) . present(N::c66,0.99) .

```

C.2 Modèle initial PATHWAY LOGIC

Listing C.3 – Modèle de l'activation de Ras : fichier components.maude

```

1 fmod PROTEINOPS is inc DISH .
2 **** inc DOMAIN .
3
4 sort ErbB1L .
5 subsort ErbB1L < Protein .
6 op 1433x1 : -> Protein .
7 op Cbl : -> Protein .
8 op DAG : -> Chemical .
9 op Egf : -> ErbB1L .
10 op EgfR : -> Protein .
11 op ErbB2 : -> Protein .
12 op Gab1 : -> Protein .
13 op Grb2 : -> Protein .
14 op Hras : -> Protein .
15 op IP3 : -> Chemical .
16 op Pak1 : -> Protein .
17 op Pi3k : -> Composite .

```

```

18 op Pik3ca : -> Protein .
19 op Pik3cb : -> Protein .
20 op Pik3cd : -> Protein .
21 op Pik3cg : -> Protein .
22 op Pik3c3 : -> Protein .
23 op Pik3r1 : -> Protein .
24 op Pik3r2 : -> Protein .
25 op Pik3r3 : -> Protein .
26 op PIP2 : -> Chemical .
27 op PIP3 : -> Chemical .
28 sort Plc .
29 subsort Plc < Protein .
30 op Plcg : -> Plc .
31   op Plcg1 : -> Protein .
32   op Plcg2 : -> Protein .
33 op PP2a : -> Composite .
34 op Raf1 : -> Protein .
35 op Shc : -> Protein .
36 op Sos1 : -> Protein .
37 op Src : -> Protein .
38 op Tgfa : -> ErbB1L .
39 op Ube2l3 : -> Protein .
40 op Vav2 : -> Protein .
41 endfm

```

Listing C.4 – Modèle de l'activation de Ras : fichier rules.mauve

```

1 mod ALLBP is inc PROTEINOPS .
2
3 var cell : CellType .
4 vars clo clm cli cle nuo num nui nuc : Soup .
5 vars moo mom moi moc mio mim mii mic : Soup .
6 vars ero erm eri erc pxo pxm pxi pxc : Soup .
7 vars gao gam gai gac lyo lym lyi lyc : Soup .
8 vars eeo eem eei eec leo lem lei lec : Soup .
9 vars cpo cpm cpi cpc ct ptc sig : Soup .
10 var ms : ModSet .
11
12 rl [1.EgfR.act]:
13   ?ErbB1L:ErbB1L
14   [CellType:CellType | ct
15   {CLm | clm EgfR           } ]
16   =>
17   [CellType:CellType | ct
18   {CLm | clm ([EgfR - act] : ?ErbB1L:ErbB1L)} ] .
19   -----
20

```

```

21 rl [2. EgfR.ubiq]:
22   {CLm | clm ([EgfR - act] : ?ErbB1L:ErbB1L) }
23   {CLi | cli [Cbl - Yphos] [Ube2l3 - ubiq] }
24   =>
25   {CLm | clm ([EgfR - ubiq] : ?ErbB1L:ErbB1L) }
26   {CLi | cli [Cbl - Yphos] Ube2l3          } .
27   -----
28   *** 10531381(D) Cbl-Ring binds Ube2l3
29     *** Together they ubiq EgfR after Egf stim
30
31
32 rl [4. Gab1.Yphosed]:
33   {CLm | clm ([EgfR - act] : ?ErbB1L:ErbB1L) }
34   {CLi | cli [Grb2 - reloc]                  }
35   {CLc | clc Gab1                            }
36   =>
37   {CLm | clm ([EgfR - act] : ?ErbB1L:ErbB1L) }
38   {CLi | cli [Grb2 - reloc] [Gab1 - Yphos] }
39   {CLc | clc                            } .
40   -----
41
42 rl [5. Grb2.reloc]:
43   {CLm | clm ([EgfR - act] : ?ErbB1L:ErbB1L) }
44   {CLi | cli                               }
45   {CLc | clc Grb2                          }
46   =>
47   {CLm | clm ([EgfR - act] : ?ErbB1L:ErbB1L) }
48   {CLi | cli [Grb2 - reloc]                }
49   {CLc | clc                               } .
50   -----
51
52 rl [6. Hras.act .1]:
53   {CLm | clm PIP3                            }
54   {CLi | cli [Grb2 - reloc] [Sos1 - reloc] [Hras - GDP] }
55   =>
56   {CLm | clm PIP3                            }
57   {CLi | cli [Grb2 - reloc] [Sos1 - reloc] [Hras - GTP] } .
58   -----
59
60 rl [7. IP3.from.PIP2.by.Plc]:
61   {CLm | clm PIP2                            }
62   {CLi | cli [?Plc:Plc - act]                }
63   {CLc | clc                                }
64   =>
65   {CLm | clm DAG                            }

```

```

66  {CLi | cli [?Plc:Plc - act] }
67  {CLc | clc IP3          } .
68  -----
69
70  rl [8. Pi3k.act]:
71  {CLi | cli [Gab1 - Yphos]      }
72  {CLc | clc Pi3k                }
73  =>
74  {CLi | cli [Gab1 - Yphos] [Pi3k - act] }
75  {CLc | clc                      } .
76  -----
77
78  rl [9. PIP3.from.PIP2.by.Pi3k]:
79  {CLm | clm PIP2          }
80  {CLi | cli [Pi3k - act] }
81  =>
82  {CLm | clm PIP3          }
83  {CLi | cli [Pi3k - act] } .
84  -----
85
86  rl [10. Plcg.act]:
87  {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) PIP3 }
88  {CLi | cli Src          }
89  {CLc | clc Plcg        }
90  =>
91  {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) PIP3 }
92  {CLi | cli Src [Plcg - act] }
93  {CLc | clc          } .
94  -----
95
96  rl [11. Shc.Yphosed]:
97  {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) }
98  {CLi | cli Src          }
99  {CLc | clc Shc          }
100 =>
101 {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) }
102 {CLi | cli Src [Shc - Yphos] }
103 {CLc | clc          } .
104 -----
105
106 rl [12. Sos1. reinit ]:
107 {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) }
108 {CLi | cli [Grb2 - reloc] [Sos1 - reloc] }
109 {CLc | clc          }
110 =>
111 {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) }

```

```

112 {CLi | cli [Grb2 - Yphos]          }
113 {CLc | clc Sos1                    } .
114 -----
115
116 rl [13.Sos1.reloc]:
117 {CLi | cli [Grb2 - reloc]          }
118 {CLc | clc Sos1                    }
119 =>
120 {CLi | cli [Grb2 - reloc] [Sos1 - reloc] }
121 {CLc | clc                          } .
122 -----
123
124 rl [15.Cbl.reloc.Yphos]:
125 {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) }
126 {CLi | cli                          }
127 {CLc | clc Cbl                      }
128 =>
129 {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) }
130 {CLi | cli [Cbl - Yphos]           }
131 {CLc | clc                          } .
132 -----
133
134 rl [E56.Pak1.irt.Egf]:
135 {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) }
136 {CLi | cli                          }
137 {CLc | clc Pak1                    }
138 =>
139 {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) }
140 {CLi | cli [Pak1 - act]            }
141 {CLc | clc                          } .
142 -----
143
144 rl [280.Raf1.by.Hras]:
145 {CLi | cli [Hras - GTP] [Pak1 - act] Src }
146 {CLc | clc Raf1 1433x1 PP2a          }
147 =>
148 {CLi | cli [Hras - GTP] [Pak1 - act] Src [Raf1 - act] 1433x1 }
149 {CLc | clc PP2a                      } .
150 -----
151
152 rl [14.Vav2.act]:
153 {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) }
154 {CLi | cli Src                      }
155 {CLc | clc Vav2                    }
156 =>
157 {CLm | clm ([Egfr - act] : ?ErbB1L:ErbB1L) }

```

```
158 {CLi | cli Src [Vav2 - act] }
159 {CLc | clc          } .
160 -----
161
162 rl [clt1 . ctest]:
163 {CLi | cli Src [Vav2 - act] [Cbl - Yphos] }
164 =>
165 {CLi | cli ( Src : ( [Vav2 - act] : [Cbl - Yphos] ) ) } .
166 -----
167 endm
```

Annexe D

Quelques fichiers entêtes de classes

D.1 Classe CCBio

```
1 #ifndef C_BIO_HH
2 #define C_BIO_HH
3
4 #include "inline-macro.hh"
5 #include <iostream>
6 #include <cstring>
7 #include "dart.hh"
8 using namespace std;
9 using namespace GMap3d;
10
11 struct ListeBrins
12 {
13     CDart * Unbrin;
14     ListeBrins * suiv;
15 };
16
17 ListeBrins * ajouterEltListe(ListeBrins * L, CDart * ADart);
18 ListeBrins * supprimerEltListe(ListeBrins * L, CDart * ADart);
19
20 class CCBio
21 {
22 private:
23     string TypeCompart;
24     string NomCompart;
25     ListeBrins * LesInterieurs;
26     CDart * bPorteur;
27
28 public:
29     CCBio();
```

```

30     CCBio (string TypeC, string NomC);
31     ~CCBio();
32
33     string getLibTypeC() const;
34     CTypeCompartiment * getTypeC();
35     string getNomC() const;
36     ListeBrins * getInterieurs ();
37     CDart * getBrinPorteur();
38     void setTypeC(string NewTypeC);
39     void setNomC(string NewNomC);
40     void setInterieurs (ListeBrins * lb)
41     void setBrinPorteur(CDart * bp);
42     void setTypeNomC(string NewTypeC, string NewNomC);
43     void insererInt (GMap3d::CDart * TC);
44     bool est_egal (CCBio C);
45     void deleteInt (CDart * b);
46     bool contient (CCBio C);
47     bool est_contenu_dans (CCBio C);
48     bool est_colle_a (CCBio C);
49     bool est_voisin (CCBio C);
50 };
51     #include INCLUDE_INLINE("cbio.icc")
52 #endif // C_BIO_HH

```

D.2 Classe CExchangeGraph

```

1  #ifndef EXCHANGEGRAPH_HH
2  #define EXCHANGEGRAPH_HH
3
4  #include "typescompart.hh"
5  #include "compartiment.hh"
6  #include "arete.hh"
7  #include "substitution.hh"
8  #include <stdio.h>
9  #include <string>
10 #include <iostream>
11 extern "C"{
12 #include "analyseGE.h"
13 #include "analyseGE.tab.h"
14 }
15 using namespace std;
16
17 class CExchangeGraph
18 {
19     public:

```

```
20     CExchangeGraph();
21     CExchangeGraph(CCompartment * lc, CArete * la);
22     CExchangeGraph(string nge);
23     CExchangeGraph(noeudGE * nge);
24     CExchangeGraph(CExchangeGraph * G);
25     CExchangeGraph(CExchangeGraph &G);
26     ~CExchangeGraph();
27
28     CCompartment * getListVertex();
29     CArete * getListArete();
30     string getNomGE();
31     CTypeCompartment * getTypesC();
32     void setListCompart(CCompartment * ACompart);
33     void setListArete(CArete * a);
34     void setNomGE(string nge);
35     void setTypesC(CTypeCompartment * tc);
36
37     void insertVertex(string tc, string nc);
38     void insertVertex(CCompartment * ACompart);
39     void supprimVertex(string nc);
40     void supprimVertex(CCompartment * ACompart);
41     void insertEdge(string nc1, string nc2);
42     void insertEdge(CCompartment * c1, CCompartment * c2);
43     void insertEdge(CArete * a);
44     void supprimEdge(CCompartment * ACompart);
45     void supprimEdge(string nc1, string nc2);
46     void supprimEdge(CCompartment * c1, CCompartment * c2);
47     void supprimEdge(CArete * a);
48     void insertTypeC(string libtc, string em, string tm, string im, string ic);
49     void insertTypeC(CTypeCompartment * tc);
50     void insertTypeC(CTypeCompartment & tc);
51
52     CExchangeGraph * restrictPartieListCompart(CCompartment * lc);
53     CCompartment * findCompartment(string nc);
54     CCompartment * findCompartment(CCompartment * ACompart);
55     CArete * findEdge(CCompartment * c1, CCompartment * c2);
56     CArete * findEdge(string nc1, string nc2);
57     bool sontVoisins(string nc1, string nc2);
58     CCompartment * lesVoisinsDe(CCompartment * ACompart);
59     CCompartment * lesVoisinsDe(string nc);
60     bool contientVertex(string nc);
61     bool contientType(string tc);
62     bool contientListCompart(CCompartment * ACompart);
63     void differenceVertex(CExchangeGraph * G2);
64     void fusionnerVertex(CExchangeGraph * G2);
65     bool sousGraphesIssusSontVoisins(CCompartment * L1, CCompartment * L2);
```

```

66     bool sousGraphesVoisins(CExchangeGraph * G1, CExchangeGraph * G2);
67     int nbVertex();
68     bool estConnexe();
69     CSubstitution * genererSubstUneVar(CCompartmentVariable * cv);
70     CSubstitution * genererSubstListVariables(CCompartmentVariable * lesVars, CSubstitution
        * res);
71     bool tousTypesRepresentes(CCompartmentVariable * vars);
72     CSubstitution * genererSubstListVariables2(CCompartmentVariable * lesVars,
        CSubstitution * res);
73     CTypeCompartment * findTypeCompartment(string ltc);
74     void enregistrerGE();
75     void writeGraphe();
76
77     private:
78         string nomGE;
79         CTypeCompartment * firstTC;
80         CCompartment * firstVertex;
81         CArete * firstEdge;
82 };
83 #endif // EXCHANGEGRAPH_HH

```

D.3 Classe CModeleGenerique

Listing D.1 – entête de la classe CModeleGenerique

```

1  #ifndef MODELE_GEN_HH
2  #define MODELE_GEN_HH
3  #include <string>
4  #include "typescompart.hh"
5  #include "classe-molecule-base.hh"
6  #include "modif-forme.hh"
7  #include "parametre-cinetiq.hh"
8  #include "rg.hh"
9
10 extern "C"
11 {
12     #include "analyse.h"
13     #include "analyse.tab.h"
14 }
15
16 using namespace std;
17 class CModeleGenerique
18 {
19     public:
20     CModeleGenerique();

```

```
21 CModeleGenerique(noeud * tn);
22 ~CModeleGenerique();
23
24 CRegleGenerique * getFirtsRG();
25
26 void setFirstRG(CRegleGenerique * rg);
27
28 CRegleGenerique * recupererLaRG(noeud * tn);
29 CEtatCellule * recupererMembre(noeud * tn);
30 CSolution * recupererSolutionLocalisee(noeud * tn);
31 CMolecule * recupererMolecule(noeud * tn);
32 CModification * recupererModification(noeud * tn);
33
34 void addRules(CRegleGenerique * rg);
35
36 void insertTypeC(string libtc , string em, string tm, string im, string ic);
37 void insertTypeC(CTypeCompartiment * tc);
38 void insertTypeC(CTypeCompartiment & tc);
39 CTypeCompartiment * findTypeCompartiment(string ltc);
40 CTypeCompartiment * getTypesC();
41 void setTypesC(CTypeCompartiment * tc);
42 void recupererLesTypesC(noeud * tn);
43
44 CClassMols * getClassMolsB();
45 void setClassMolsB(CClassMols * cm);
46
47 void insertClassMolsB(CClassMols * cm);
48 void recupererMoleculeBase(noeud * tn);
49 CMoleculeBase * recupererMoleculesClasse(noeud * tn);
50
51 CModifForme * getModifForme();
52 void setModifForme(CModifForme * mf);
53
54 void insertModifForme(string mf);
55 void insertModifForme(CModifForme * mf);
56 void recupererModifsForme(noeud * tn);
57
58 CParametresCinetiq * getFirstPC();
59 void setFirstPC(CParametresCinetiq * pcin);
60 void insertParamCin(CParametresCinetiq * pcin);
61 void insertParamCin(string idp, float valp);
62 void insertParamCin(string idp);
63 void recupererParametresCinetiques(noeud * tn);
64
65 void recupererLesReglesGen(noeud * tn);
66
```

```
67
68     private:
69         CTypeCompartiment * firstTC;
70         CClassMols * firstClassMolsB;
71         CModifForme * firstModification;
72         CParametresCinetiq * firstPC;
73         CRegleGenerique * firstRG;
74
75
76 };
77
78 #endif // MODELE_GEN_HH
```

D.4 Classe CRegleGenerique

Listing D.2 – entête de la classe CRegleGenerique

```
1 #ifndef RG_HH
2 #define RG_HH
3
4 #include <stdio.h>
5 #include <string>
6 #include <stdlib.h>
7 #include "precond-voisinage.hh"
8 #include "kinetics-rg.hh"
9 #include "param-cin-conditionnel.hh"
10 #include "etat-cellule.hh"
11 #include "condition-variable-molecule.hh"
12
13 #include <string>
14 #include <iostream>
15 #include <fstream>
16
17
18 typedef enum{pcinAbsent, pcinSimple, pcinCond} optionPCin;
19 class CRegleGenerique
20 {
21     public:
22         CRegleGenerique();
23         CRegleGenerique(CEtatCellule * E1, CEtatCellule * E2);
24         CRegleGenerique(CRegleGenerique * rg);
25         CRegleGenerique(CRegleGenerique & rg);
26         ~CRegleGenerique();
27
28         string getNomRG();
```

```
29     void setNomRG(string nrg);
30     void recupererIdrg(noeud * tn);
31
32     CPrecondVoisinage * getPrecondV();
33     void setPrecondV(CPrecondVoisinage * prec);
34     void insertPrecondV(CPrecondVoisinage * prec);
35     void insertPrecondV(string v1, string v2);
36     void recupererPrecVoisinage(noeud * tn);
37
38     optionPCin getExistenceEtTypePC();
39     CKineticsRG * getSimpleKineticsRG();
40     CParamCinCond * getConditionnalKineticsRG();
41     void setExistenceEtTypePC(optionPCin optPC);
42     void setSimpleKineticsRG(CKineticsRG * pc);
43     void setConditionnalKineticsRG(CParamCinCond * pc);
44     void recupererParametrsCinetiques(noeud * tn);
45
46     CEtatCellule * getMG();
47     CEtatCellule * getMD();
48     CRegleGenerique * getRGSuivant();
49
50
51     void setMG(CEtatCellule * E1);
52     void setMD(CEtatCellule * E2);
53     void recupererEtatsCellule(noeud * tn);
54
55     void setCondVMol(CConditionVarMol * cvm);
56     CConditionVarMol * getCondVMol();
57     void addCondVMol(CConditionVarMol * cvm);
58     void recupererCondVMol(noeud * tn);
59     void writeCondVMol();
60     void setRGSuivante(CRegleGenerique * rgS);
61
62         void writeRG(ostream &f);
63
64     private:
65         string nomRG;
66         CPrecondVoisinage * firstPrecV;
67
68         optionPCin optPC;
69         CKineticsRG * simpleKin;
70         CParamCinCond * condKin;
71
72         CEtatCellule * MG;
73         CEtatCellule * MD;
74         CConditionVarMol * firstCondVM;
```

```
75     CRegleGenerique * rgSuiv;  
76 };  
77 #endif //RG_HH
```

D.5 Classe CMolecule

Listing D.3 – entête de la classe CMolecule

```
1  #ifndef MOLECULE_HH  
2  #define MOLECULE_HH  
3  #include <string>  
4  #include <iostream>  
5  #include <fstream>  
6  #include "modification.hh"  
7  
8  extern "C"  
9  {  
10     #include "analyse.h"  
11     #include "analyse.tab.h"  
12 }  
13  
14 using namespace std;  
15  
16 typedef enum {molvide, MBase, MBaseModif, Complexe, ComplexeModif, vMol1, vMol1Modif,  
17             vMol2, vMol2Modif} typeMol;  
18  
19 class CMolecule  
20 {  
21 public:  
22     CMolecule();  
23     CMolecule(string nomM);  
24     CMolecule(CMolecule * M);  
25     CMolecule(CMolecule & M);  
26     ~CMolecule();  
27     typeMol getTypeM();  
28     string getNomM();  
29     string getVarMol1();  
30     string getVarMol2();  
31     CMolecule * getComposante1Complexe();  
32     CMolecule * getComposante2Complexe();  
33     CModification * getModification();  
34  
35     CMolecule * getMoleculeSuivante();  
36
```

```
37 void setTypeM(typeMol tm);
38 void setNomM(string nomM);
39 void setVarM1(string nomVM);
40 void setVarM2(string nomVM);
41
42 void createComplex(CMolecule * M1, CMolecule * M2);
43 void setMolSuiivante(CMolecule * M);
44 void addModif(string nmod);
45 void addModification(CModification * modM);
46
47 void recupererMolecule(noeud * tn);
48
49 void writeMolecule(typeModele tmod, ostream &fichier);
50
51 void writeMoleculeBiocham(ostream &fichier);
52 void writeMoleculePL(ostream &fichier);
53 void writeMoleculeGenerique(ostream &fichier);
54 void writeMoleculeGenerique();
55 void writeMoleculeIntermediaire(ostream &fichier);
56
57 private:
58 typeMol typeM;
59 string nomMol;
60 string varMol1;
61 string varMol2;
62 CMolecule * C1;
63 CMolecule * C2;
64 string modif;
65 CModification * modifMol;
66
67 CMolecule * molSuiv;
68 };
69
70 #endif // MOLECULE_HH
```


Annexe E

Fichier d'extraction du graphe d'échanges : informations topologiques

Listing E.1 – Fichier d'extraction du graphe d'échanges pour le modèle de la signalisation Delta-Notch

```
1 GRAPHE D'ECHANGES
2 TYPES_COMPARTIMENT
3 Cellule, Ext
4 FIN_TYPES_COMPARTIMENT
5
6 COMPARTIMENTS
7 c11 : Cellule ; c12 : Cellule ; c13 : Cellule ;
8 c14 : Cellule ; c15 : Cellule ; c16 : Cellule ;
9 c21 : Cellule ; c22 : Cellule ; c23 : Cellule ;
10 c24 : Cellule ; c25 : Cellule ; c26 : Cellule ;
11 c31 : Cellule ; c32 : Cellule ; c33 : Cellule ;
12 c34 : Cellule ; c35 : Cellule ; c36 : Cellule ;
13 c41 : Cellule ; c42 : Cellule ; c43 : Cellule ;
14 c44 : Cellule ; c45 : Cellule ; c46 : Cellule ;
15 c51 : Cellule ; c52 : Cellule ; c53 : Cellule ;
16 c54 : Cellule ; c55 : Cellule ; c56 : Cellule ;
17 c61 : Cellule ; c62 : Cellule ; c63 : Cellule ;
18 c64 : Cellule ; c65 : Cellule ; c66 : Cellule ;
19 Env : Ext
20 FIN_COMPARTIMENTS
21
22 ARETES
23 {c11,c12}; {c11,c21}; {c11,Env};
24 {c12,c13}; {c12,c22}; {c12,Env};
25 {c13,c14}; {c13,c23}; {c13,Env};
```

```
26 {c14,c15}; {c14,c24}; {c14,Env};
27 {c15,c16}; {c15,c25}; {c15,Env};
28 {c16,c26}; {c16,Env}; {c21,c22};
29 {c21,c31}; {c21,Env}; {c22,c23};
30 {c22,c32}; {c22,Env}; {c23,c24};
31 {c23,c33}; {c23,Env}; {c24,c25};
32 {c24,c34}; {c24,Env}; {c25,c26};
33 {c25,c35}; {c25,Env}; {c26,c36};
34 {c26,Env}; {c31,c32}; {c31,c41};
35 {c31,Env}; {c32,c33}; {c32,c42};
36 {c32,Env}; {c33,c34}; {c33,Env};
37 {c34,c35}; {c34,c44}; {c34,Env};
38 {c35,c36}; {c35,c45}; {c35,Env};
39 {c36,c46}; {c36,Env}; {c41,c42};
40 {c41,c51}; {c41,Env}; {c42,c43};
41 {c42,c52}; {c42,Env}; {c43,c44};
42 {c43,c53}; {c43,Env}; {c44,c45};
43 {c44,c54}; {c44,Env}; {c45,c46};
44 {c45,c55}; {c45,Env}; {c46,c56};
45 {c46,Env}; {c51,c52}; {c51,c61};
46 {c51,Env}; {c52,c53}; {c52,c62};
47 {c52,Env}; {c53,c54}; {c53,c63};
48 {c53,Env}; {c54,c55}; {c54,c64};
49 {c54,Env}; {c55,c56}; {c55,c65};
50 {c55,Env}; {c56,c66}; {c56,Env};
51 {c61,c62}; {c61,Env}; {c62,c63};
52 {c62,Env}; {c63,c64}; {c63,Env};
53 {c64,c65}; {c64,Env}; {c65,c66};
54 {c65,Env}; {c66,Env}
55 FIN_ARETES
```

Table des figures

1.1 Synthèse (\rightarrow) et dégradation (\leftarrow) de molécules	6
1.2 Transport de molécules	6
1.3 Représentation d'une molécule (M_1) avec ses domaines fonctionnels.	6
1.4 Complexation entre les molécules M_1 et M_2 par leurs sites de liaison respectifs B et D	7
2.1 Schématisation de l'approche	27
2.2 Une compartimentation cellulaire et le graphe d'échanges correspondant	28
3.1 Deux plongements pour une même structure topologique	38
3.2 Ambiguïté du modèle en fil de fer	40
3.3 Représentation par arbre CSG	40
3.4 Représentation CSG d'un objet non usinable	41
3.5 Un objet représenté par les bords	42
3.6 Décomposition d'un objet en cellules	42
3.7 Graphe d'incidence de l'objet figure 3.6	43
3.8 Ambiguïté du graphe d'incidence	44
3.9 Non prise en compte de l'inclusion par les graphes d'incidence	44
3.10 Graphe d'adjacence pour les cellules de dimension 1 de l'objet figure 3.6 (page 42)	44
3.11 Graphe d'Adjacence des régions	45
3.12 Ambiguïté des graphes d'Adjacence des régions	45
3.13 Arbre d'inclusion	46
3.14 Objet 3D à représenter par une 3-G-Carte	47
3.15 Décomposition de l'objet 3D (figure 3.14) en brins	47

3.16	Etablissement des liaisons α_0 à α_3 entre les brins de la figure 3.15(d)	48
3.17	Représentation des voisinages de brins	48
3.18	Modélisation 2-G-Carte de l'objet de la figure 3.6 (page 42)	49
3.19	Les différentes cellules topologiques dans une 2-G-Carte	50
3.20	Collage de i - <i>cellules</i> le long de $(i - 1)$ - <i>Cellules</i>	51
3.21	Exemples de représentations qui ne sont pas des n-G-Cartes	52
3.22	Graphe G_1 : exemple de graphe partiellement étiqueté	54
3.23	Deux sous graphes	55
3.24	Graphe non orienté correspondant au graphe G_2 (figure 3.23(b))	56
3.25	Compartmentation cellulaire et le graphe d'échanges l'abstrayant	57
3.26	Représentation d'une compartimentation cellulaire par un modèle bio-géométrique	60
3.27	Les informations du plongement compartiment sur deux exemples	62
3.28	Vérifications à effectuer pour le positionnement des compartiments à créer	63
3.29	Extraction du graphe d'échange	64
3.30	Compartiment inclus mais non voisin du contenant	65
3.31	Architecture 3 tiers de MOKA	66
3.32	Portion du diagramme des classes de MOKA.	67
3.33	Deux vues compactes de compartiments et identification des faces dans la représentation des compartiments	68
3.34	Représentation de la double-enveloppe sur deux faces de cube	69
3.35	Diagramme de classes de lib-gmapbiokernel.	70
3.36	Compartmentation et implémentation du plongement biologique correspondant	71
3.37	Cas de collage de faces de dimensions différentes	73
3.38	Diagramme de classes de libGE.	75
3.39	Modélisation d'une compartimentation cellulaire et extraction de son graphe d'échange	78
3.40	Modélisation d'une compartiment cellulaire aux compartiments de taille différents	79
3.41	Modélisation d'un collage avec inclusion de la face de collage	80
3.42	Modélisation d'un compartiment sans voisinage avec son contenant	81
4.1	Graphe d'échanges	96

4.2	Deux sous-graphes de G_1 engendrés respectivement par $im(\sigma_{(GE_1, X)})$ et $im(\sigma'_{(GE_1, X)})$. . .	100
4.3	Graphes d'échanges pour illustration de la procédure de couplage.	101
4.4	Grphe d'échanges $GE2_{illustr}$ après réajustement.	102
4.5	arbre de syntaxe abstraite	117
4.6	Structure de la liste $lstSubstMG$	119
4.7	Schématisation de la liste pour le graphe	120
5.1	Différentes phases du cycle cellulaire	123
5.2	Grphe d'échanges pour l'exemple du cycle cellulaire : $GE_{cellCycle}$	124
5.3	Compartiment cellulaire pour l'exemple de la signalisatation Delta-Notch	125
5.4	Grphe d'échanges cohérent pour l'exemple du notch4n36c	126
5.5	Schéma de la voie Ras/MAPK	137
5.6	Grphe d'échanges pour la voie RAS/MAPK	137

Liste des tableaux

2.1	Tableau récapitulatif des éléments à inclure dans le nouveau langage de règles	35
4.1	Domaine, image et codomaine des substitutions $\sigma_{(\mathcal{A})}$ et $\sigma'_{(\mathcal{A})}$	98
4.2	Substitutions éligibles dans $GE1_{illustr}$ à $GE4_{illustr}$ pour les règles de $MGen_1$	103
4.3	Substitutions validées dans $GE1_{illustr}$ à $GE4_{illustr}$ pour les règles de $MGen_1$	105
5.1	Indicateurs sur les traductions du cycle cellulaire et de la signalisation de Notch	135

Bibliographie

- [ABS⁺07] Alessandro Abate, Yu Bai, Nathalie Sznajder, Carolyn Talcott, and Ashish Tiwari. Quantitative and probabilistic modeling in pathway logic. In *In the IEEE proceedings of the 7th International Symposium on Bioinformatics and Bioengineering*, 2007.
- [AGM08] Marc Aiguier, Pascale Le Gall, and Mbarka Mabrouki. Emergent properties in reactive systems. In *15th Asia-Pacific Software Engineering Conference (APSEC 2008)*, pages 273–280. IEEE, 2008.
- [AMAP02] Eric J. Allenspach, Ivan Maillard, Jon C. Aster, and Warren S. Pear. Notch signaling in cancer. *Cancer Biol Ther*, 1(5) :466–476, September 2002.
- [And10] Bruno André. Organisation interne et physiologie de la cellule. <http://www.ulb.ac.be/sciences/physcell/student.htm>, 2010. Cours de master année académique 2010-2011.
- [Bau75] Bruce G. Baumgart. A polyhedron representation for computer vision. In *AFIPS Nat*, volume 44, pages 589–596, <http://www.informatik.uni-trier.de/ley/db/conf/afips/ncc75.html>, 1975.
- [Ber] S. Berthoumieux. Estimation de paramètres : comment utiliser des données biologiques incomplètes? <http://www.inria.fr/centre/grenoble/actualites/estimation-de-parametres-et-donnees-biologiques-incompletees>.
- [BL] Christel Brou and Frédérique Logeat. Endocytose et voie de signalisation notch. <http://id.erudit.org/iderudit/013770ar>. M/S : médecine sciences, vol. 22, 8-9, 2006, p. 685-688.
- [BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- [Car04] Luca Cardelli. Brane calculi. In *Computational Methods in Systems Biology, International Conference, CMSB 2004*, volume 3082 of *Lecture Notes in Computer Science*, pages 257–278. Springer, 2004.
- [CCRF⁺07] Laurence Calzone, Nathalie Chabrier-Rivier, François Fages, L. Fosse, and Sylvain Soliman. Langages formels dans la machine abstraite biochimique biocham. *Technique et Science Informatiques*, 26(1-2) :47–72, 2007.
- [CCRFS05] Laurence Calzone, Nathalie Chabrier-Rivier, François Fages, and Sylvain Soliman. Apprentissage de règles de réactions biochimiques à partir de propriétés en logique temporelle. In Alain Guénoche et Christophe Geourjon Guy Perrière, editor, *Actes de JOBIM'05*, pages 183–192, Lyon, July 2005.

-
- [CCRFS06] Laurence Calzone, Nathalie Chabrier-Rivier, François Fages, and Sylvain Soliman. Machine learning biochemical networks from temporal logic properties. 4220 :68–94, 2006.
- [CDE⁺07] Manuel Clavel, Francisco Duran, Steven Eker, Patrick Lincoln, Narciso Marti-Oliet, José Meseguer, and Carolyn Talcott. *Maude Manual (Version 2.3)*. January 2007.
- [CF03] Nathalie Chabrier and François Fages. Symbolic model checking of biochemical networks. In *Computational Methods in Systems Biology, First International Workshop (CMSB)*, volume 2602 of *Lecture Notes in Computer Science*, pages 149–162. Springer, 2003.
- [CFS06] Laurence Calzone, François Fages, and Sylvain Soliman. Biocham : an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14) :1805–1807, 2006.
- [CL97] P. COLANTONI and B. LAGET. Color image segmentation using region adjacency graphs. In *IPA97, 15-17 July 1997, Conference Publication No. 443 0 IEE*, 1997.
- [cnr] Le cycle cellulaire. <http://www.cnr.fr/cw/dossiers/doscel/imgAr/anim/cycleCell.html>.
- [CNT05] A. Ciliberto, B. Novak, and J. J. Tyson. Steady states and oscillations in the p53/Mdm2 network. *Cell Cycle*, 4(3) :488–493, March 2005.
- [CRCD⁺04a] Nathalie Chabrier-Rivier, Marc Chiaverini, Vincent Danos, François Fages, and Vincent Schächter. Modeling and querying biomolecular interaction networks. *Theor. Comput. Sci.*, 325(1) :25–44, 2004.
- [CRCD⁺04b] Nathalie Chabrier-Rivier, Marc Chiaverini, Vincent Danos, François Fages, and Vincent Schächter. Modeling and querying biomolecular interaction networks. *Theor. Comput. Sci.*, 325 :25–44, September 2004.
- [CRFS04] Nathalie Chabrier-Rivier, François Fages, and Sylvain Soliman. The biochemical abstract machine biocham. In Vincent Danos and Vincent Schächter, editors, *CMSB*, volume 3082 of *Lecture Notes in Computer Science*, pages 172–191. Springer, 2004.
- [Cri58] Francis H. C. Crick. On protein synthesis. In Academic Press, editor, *Proceedings of the 12th Symposia of the Society for Experimental Biology : The biological replication of macromolecules*, page 138–163, 1958.
- [Dav03] Laetitia Davidovic. *Caractérisation de la protéine PARL, le prototype d’une nouvelle sous-famille de sérine-protéases assurant la protéolyse intramembranaire régulée*. PhD thesis, Université Laval, 08 2003. <http://archimede.bibl.ulaval.ca/archimede/fichiers/21080/ch01.html>.
- [DFF⁺07] Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, Jean Krivine, Plectix Biosystems, École Normale Supérieure, and École Polytechnique. Rule-based modelling of cellular signalling. In *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR’07), Lecture Notes in Computer Science*, pages 17–41, 2007.
- [DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–320. North Holland, 1990.
- [DL86] J.F. Hughes D.H. Laidlaw, W.B. Trumbore. Constructive solid geometry for polyhedral objects. In *SIGGRAPH Comput. Graph.*, volume 20, pages 161–170, <http://www.informatik.uni-trier.de/ley/db/conf/siggraph/siggraph1986.html>. 1986.
- [Duf89] J.-F. Dufourd. Algebraic map-based topological kernel for polyhedron modelers : algebraic specification and logic prototyping. In *Proc. of Eurographics*, 1989.
-

- [EKL+02a] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and K. Sonmez. Pathway Logic : Symbolic analysis of biological signaling. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 400–412, January 2002.
- [EKL+02b] Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, and Carolyn Talcott. Pathway Logic : Executable models of biological networks. In *Proceedings of Fourth International Workshop on Rewriting Logic and Its Applications (WRLA)*. 2002.
- [FBGH05] James R. Faeder, Michael L. Blinov, Byron Goldstein, and William S. Hlavacek. Rule-based modeling of biochemical networks. *Complexity*, 10(4) :22–41, 2005.
- [FJRS10] François Fages, Dragana Jovanovska, Aurélien Rizk, and Sylvain Soliman. *BIOCHAM 3.2 Reference Manual*, October 2010.
- [FS08a] François Fages and Sylvain Soliman. Formal cell biology in biocham. In Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro, editors, *SFM*, volume 5016 of *Lecture Notes in Computer Science*, pages 54–80. Springer, 2008.
- [FS08b] François Fages and Sylvain Soliman. Abstract interpretation and types for systems biology. *Theor. Comput. Sci.*, 403 :52–70, August 2008.
- [FSCR04] F. Fages, S. Soliman, and N. Chabrier-Rivier. Modeling and quering interaction networks in the biochemical abstract machine biocham. *Journal of Biological Physics and Chemistry*, 4(2) :64–73, Oct 2004.
- [FT99] E. Favier and A. Tremeau. Une mesure de contraste couleur inter-régions. In *Dix-septième colloque GRETSI*, Vannes, September 1999.
- [GT01] Ronojoy Ghosh and Claire Tomlin. Lateral inhibition through delta-notch signaling : A piecewise affine hybrid model. In Maria Di Benedetto and Alberto Sangiovanni-Vincentelli, editors, *Hybrid Systems : Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 232–246. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-45351-2.21.
- [HFH+08] Michael Hucka, Andrew M. Finney, Stefan Hoops, Sarah M. Keating, and Nicolas Le Novère. Systems biology markup language (sbml) level 2 : Structures and facilities for model definition. <http://sbml.org/documents/>, sbml.org, 2008.
- [HFS+03] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, , the rest of the SBML Forum :, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (sbml) : a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4) :524–531, 2003.
- [IXX] Institut des systèmes complexes IXXI. Les systèmes complexes : un sujet intrinsèquement interdisciplinaires. http://www.ixxi.fr/OLD/Les_SC.php.
- [JT08] Salim Jouili and Salvatore Tabbone. Applications des graphes en traitement d’images. In Y. Boudabbous and N. Zaguia, editors, *International Conference on Relations, Orders and Graphs : Interaction with Computer Science - ROGICS’08*, pages 434–442, Mahdia, Tunisie, 2008. University of Ottawa, Canada and University of Sfax, Tunisia.
- [KCD00] Mohammed Khachan, Patrick Chenin, and Hafsa Deddi. Polyhedral representation and adjacency graph in n-dimensional digital images. *Comput. Vis. Image Underst.*, 79 :428–441, September 2000.

-
- [KDMH99] Boris N. Kholodenko, Oleg V. Demin, Gisela Moehren, and Jan B. Hoek. Quantification of short term signaling by the epidermal growth factor receptor. *Journal of Biological Chemistry*, 274(42) :30169–30181, 1999.
- [Kép06] François Képès. *Morphogenèse, l'origine des formes*, chapter Morphodynamique de la voie sécrétoire., pages 129–152. Belin, Paris, 2006.
- [LDR⁺10] Chen Li, Marco Donizelli, Nicolas Rodriguez, Harish Dharuri, Lukas Endler, Vijayalakshmi Chelliah, Lu Li, Enuo He, Arnaud Henry, Melanie I. Stefan, Jacky L. Snoep, Michael Hucka, Nicolas Le Novère, and Camille Laibe. BioModels Database : An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, 4 :92, Jun 2010.
- [Lie94] P. Lienhardt. N-dimensional generalised combinatorial maps and cellular quasimanifolds. *International Journal of Computational Geometry and Applications*, 1994.
- [LLP10] Astrid Lièvre and Pierre Laurent-Puig. La voie de signalisation ras/mapk ras/mapk signaling pathway. *Cancero digest*, 2(1) :38–42, 2010. article de mise au point.
- [LM04] Marie-Claude Lebart and Jean Mariani. La régulation du cycle cellulaire. Biologie et multimédia. <http://www.snv.jussieu.fr/bmedia/cyclecellBM/>, juin 2004.
- [Mil99] Robin Milner. *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press, 1999.
- [NOM03] B.J. Nickoloff, B.A. Osborne, and L. Miele. Notch signaling as a therapeutic target in cancer : a new approach to the development of cell fate modifying agents. *Oncogene*, 22(42) :6598–608, 2003.
- [OSV⁺05] R J Orton, O E Sturm, Vladislav Vyshemirsky, Muffy Calder, D R Gilbert, and Walter Kolch. Computational modelling of the receptor-tyrosine-kinase-activated mapk pathway. *The Biochemical journal*, 392(Pt 2) :249–261, 2005.
- [Pou09] Mathieu Poudret. *Transformations de graphes pour les opérations topologiques de la modélisation géométrique. Application à l'étude de la dynamique de l'appareil de Golgi*. PhD thesis, Université de Poitiers, 2009.
- [PP10] Michael Pedersen and Gordon Plotkin. A Language for Biochemical Systems : Design and Formal Specification. In Corrado Priami, Rainer Breitling, David Gilbert, Monika Heiner, and Adelinde M. Uhrmacher, editors, *Transactions on Computational Systems Biology XII*, volume 5945, chapter 3, pages 77–145. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [RPS⁺04a] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Shapiro. Bioambients : an abstraction for biological compartments. *Theor. Comput. Sci.*, 325 :141–167, September 2004.
- [RPS⁺04b] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Y. Shapiro. Bioambients : an abstraction for biological compartments. *Theor. Comput. Sci.*, 325(1) :141–167, 2004.
- [Sei] D. Seigneurin. Les membranes cellulaires. Cours 2007-2008, chapitre 2, umvf.biomedicale.univ-paris5.fr/.../Seigneurin_Daniel.P02.pdf.
- [SIM] M. SIMON. Les propriétés des membranes cellulaires. Cours pharmacie 2009-2011, <http://www.cours-pharmacie.com/biologie-cellulaire/les-membranes-cellulaires.html>.
- [Sim06] Arnauld Simon. *Mutations de Notch dans les leucémies aiguës lymphoblastiques T de l'adulte*. PhD thesis, Université René Descartes (Paris 5), 2006.
-

- [Sri81] Sargur N. Srihari. Representation of three-dimensional digital images. *ACM Comput. Surv.*, 13 :399–424, December 1981.
- [Tal06a] Carolyn Talcott. Formal executable models of cell signaling primitives. In Tiziana Margaria, Anna Philippou, and Bernhard Steffen, editors, *2nd International Symposium On Leveraging Applications of Formal Methods, Verification and Validation ISOLA06*, pages 303–307, 2006.
- [Tal06b] Carolyn Talcott. Symbolic modeling of signal transduction in pathway logic. In L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, editors, *2006 Winter Simulation Conference*, December 2006.
- [Tal08] Carolyn Talcott. Pathway logic. In *Formal Methods for Computational Systems Biology*, volume 5016 of *LNCS*, pages 21–53. Springer, 2008. 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems.
- [TC00] Alain Trémeau and Philippe Colantoni. Regions adjacency graph applied to color image segmentation. In *IEEE Transactions on Image Processing*, 2000.
- [TD05] Carolyn Talcott and David L. Dill. The pathway logic assistant. In *Proceedings of the Workshop Computational Methods in Systems Biology (CMSB, G. Plotkin, Ed., 2005, pp. 228–239)*, March 2005.
- [TEK⁺04] Carolyn Talcott, Steven Eker, Merrill Knapp, Patrick Lincoln, and Keith Laderoute. Pathway logic modeling of protein functional domains in signal transduction. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 568–580, January 2004.
- [Tut84] W. T. Tutte. *Graph theory*. Encyclopedia of Mathematics and its Applications, volume 21, Addison-Wesley Publishing Company, Menlo Park, CA, USA, 1984.
- [Tys91] J J Tyson. Modeling the cell division cycle : cdc2 and cyclin interactions. *Proceedings of the National Academy of Sciences*, 88(16) :7328–7332, 1991.
- [VD03] F. Vidil and G. Damiand. *Moka, Documentation Utilisateur*. www.sic.sp2mi.univ-poitiers.fr/moka/, 2003.
- [Wei88] K. Weiler. The radial-edge structure : A topological representation for non-manifold geometric boundary modeling. In *Geometric Modelling for CAD Applications*, 1988.
- [Wika] Wikipédia. Encyclopédie libre. <http://fr.wikipedia.org>.
- [Wikb] Wikipédia. Encyclopédie libre. page : Biologie des systèmes. http://fr.wikipedia.org/wiki/Biologie_des_systèmes.
- [Wikc] Wikipédia. Encyclopédie libre. page : Système complexe. http://fr.wikipedia.org/wiki/Système_complexe.