Université d'Évry Val d'Essonne
Programme d'Épigénomique

Thèse

présentée en première version en vue d'obtenir le grade de docteur, spécialité « Informatique »

par

Mbarka Mabrouki

Étude de la préservation des propriétés temporelles des réseaux de régulation génétique au travers du plongement : vers une caractérisation des systèmes complexes par l'émergence de propriétés

Thèse soutenue le 13 décembre 2010 devant le jury composé de :

Yamine Aït-Ameur	Professeur ENSMA Poitiers	(Rapporteur)
Gilles Bernot	Professeur Université de Nice	(Rapporteur)
Pascale Le Gall	Professeure Université d'Évry	(Directrice de thèse)
Marc Aiguier	Professeur ECP	(Examinateur)
François Képès	Directeur CNRS, Université d'Évry	(Examinateur)
Daniel Krob	Directeur CNRS, École Polytechnique	(Examinateur)

Au petit Ayoub . . .
Au martyr Mohamed Bouazizi . . .

Je voudrais tout d'abord exprimer mes plus profonds remerciements à mes encadrants de thèse Pascale Le Gall et Marc Aiguier de m'avoir confié ce sujet de thèse et de m'avoir accompagnée tout au long de la durée de thèse. Ils se sont montrés disponibles afin de pouvoir débattre, parfois longuement, sur mon travail. Au cours de ces discussions, ils n'ont pas cessé de me conseiller et de m'encourager tout en me laissant libre des orientations que je voulais donner à ce travail. La qualité de leur encadrement m'a permis de corriger et perfectionner tout ce que j'ai pu effectuer et notamment ce manuscrit. Cette thèse ne serait certainement pas aussi aboutie s'il n'avaient pas été présents. C'est véritablement une chance d'avoir travaillé avec eux.

Un grand merci à mes rapporteurs, Yamine Aït-Ameur et Gilles Bernot, qui ont pris le temps de lire le manuscrit et m'ont fait l'honneur de l'évaluer. Merci aussi à François Képès et Daniel Krob qui ont accepté de participer à mon jury et de s'être intéressés à mon travail. Je sais combien leur temps à tous est compté et je leur en suis d'autant plus reconnaissante.

Je souhaite aussi remercier l'ensemble des équipes de recherche et d'enseignement qui m'ont accueillie et toutes les personnes que j'ai pu y côtoyer.

Je remercie mon entourage, à commencer par ma famille qui m'a soutenue et m'a encouragée à effectuer les études que je souhaitais et donc finalement d'entreprendre cette thèse. Je remercie mes amis, en France et en Tunisie, qui ont suivi mon travail et qui ont été à mes côtés dans les bons moments comme dans les mauvais.

Évry, le 6 avril 2011.

Table des matières

Liste des figures

Introduction générale

1 Problématique

Le travail de cette thèse porte sur la caractérisation des systèmes complexes à base de composants au travers de la présence de propriétés émergentes, qui soit entrent en conflit avec les propriétés attachées aux composants les constituant, soit sont directement issues de la coopération ou de l'interaction des composants.

La problématique de la caractérisation de systèmes complexes par les propriétés émergentes fait écho à un souci récurrent du génie logiciel : est-il possible de concevoir un système en le structurant de telle sorte que les propriétés attachées aux composants ou modules le constituant puissent être héritées systématiquement par le système tout entier ? Cette propriété est souvent perçue comme un préalable à la réutilisation des composants.

Une des pistes qui ont été explorées pour répondre à cette problématique a consisté à cerner une famille de systèmes, les *systèmes modulaires*, qui ont pour particularité de préserver par construction toutes les propriétés importées des sous-systèmes.

Les premières études dans ce sens ont surtout été menées dans le cadre des langages de programmation et des spécifications algébriques. Lorsqu'il s'agit de décrire de grands systèmes, les spécifications algébriques sont généralement construites de manière incrémentale par l'utilisation de primitives de structuration comme l'enrichissement [132, 29]. Des résultats de modularité ont été établis dans le cadre de spécifications dites axiomatiques dont les axiomes sont des équations conditionnées par d'autres équations (i.e. des clauses de Horn élémentaires réduites au prédicat égalité). Néanmoins, pour obtenir ces résultats de préservation, il a fallu imposer des contraintes fortes sur la classe des modèles (appelés classiquement algèbres) vérifiant ces axiomes. Ces résultats de modularité dans le cadre des spécifications algébriques conditionnelles positives seront présentés en détail dans le chapitre 2 de cette thèse. Néanmoins, nous pouvons dès à présent souligner que ces résultats concernent la préservation des modèles au travers de la structuration.

Très vite, il a été constaté que les spécifications algébriques de base n'étaient pas toujours bien adaptées à la description des systèmes informatiques. Elles ont alors inspiré différents formalismes axiomatiques dérivés, et prenant en compte différents concepts tels que les fonctions partielles, les fonctions d'ordre supérieur, les aspects dynamiques, le temps, le traitement d'exception, les structures de données bornées, etc. Ces formalismes pouvaient alors être munis des mêmes primitives de structuration pour lesquelles on étudiait les mêmes résultats de préservation des

modèles. Pour pouvoir généraliser les concepts indépendamment des systèmes logiques sous-jacents, Goguen et Burstall ont proposé le cadre général des institutions [65, 63] pour s'abstraire de différents formalismes logiques. Une institution comprend la donnée d'un ensemble de signatures, qui décrivent l'ensemble des interfaces possibles, un ensemble de formules pour chaque signature et un ensemble de modèles pour chaque signature, représentant l'ensemble des implantations partageant la même interface, celle modélisée par la signature, ainsi qu'une relation de satisfaction qui indique pour chaque signature, quels sont les modèles satisfaisant une formule donnée. Les institutions introduisent aussi un moyen de mettre en relation les signatures et donc de structurer les spécifications. Par exemple, une signature dédiée à la description des entiers naturels sera intuitivement incluse dans une signature dédiée à décrire les entiers relatifs, puisque cette dernière contiendra tous les éléments de l'interface des entiers naturels ainsi que, par exemple, la notion d'opposé. De même, il peut être utile par exemple de renommer certains symboles de la signature. Ce type de transformation se fait à travers les morphismes de signatures, qui ont leurs équivalents au niveau des modèles et au niveau des formules. Les travaux autour des institutions sont ainsi formulés en utilisant les notions issues de la théorie des catégories.

Le cadre offert par les institutions a permis de mener de nombreuses études sur la structuration de spécifications notamment par le biais des morphismes de signatures, mais aussi en exploitant la propriété particulière des institutions appelée condition de satisfaction qui assure la préservation de la relation de satisfaction par changement de signature, c'est-à-dire qu'un changement de signature selon un morphisme de signatures ne doit pas affecter la satisfaction des formules par les modèles. Les premiers travaux sur la modularité dans le cadre des spécifications algébriques ont été étendus à d'autres formalismes axiomatiques indépendamment de toute logique, c'est-à-dire définis au-dessus des institutions et ce, pour tirer profit de la structuration par morphismes de signatures et de la préservation qu'assure la condition de satisfaction [26, 44, 100].

En général, le mécanisme de base pour construire les spécifications est de les définir à l'aide d'une signature et d'un ensemble d'axiomes exprimés sur cette signature : la spécification définit la classe (éventuellement, une sous-classe particulière) des algèbres satisfaisant les axiomes de la spécification. Lorsque les spécifications sont structurées à l'aide d'une inclusion de signatures, alors la condition de satisfaction permet de transporter de concert, via les morphismes de signatures, les formules, les modèles et la validité des premières par les seconds. La plupart des travaux portant sur la notion de spécifications modulaires dans le cadre général des institutions, s'attachent à assurer la préservation des modèles au travers de la structuration, assurant de fait la préservation des formules validées. Dans ce document, nous allons proposer un cadre général pour la définition de spécifications structurées qui n'assure pas par défaut la préservation des propriétés des sous-systèmes au niveau du système global. Ce cadre sera défini au-dessus des institutions, qui ont été jugées appropriées (ou à défaut, ont constitué une source d'inspiration) pour définir des concepts en toute généralité, indépendamment des formalismes

logiques sous-jacents : spécifications structurées [100, 29], spécifications hétérogènes [23, 96], test à partir de spécifications [85, 15], systèmes de preuve [92] ou de prototypage [6], spécifications multi-vues [10].

2 Systèmes complexes : présence de propriétés émergentes

Par opposition aux systèmes modulaires, nous qualifions de *système complexe* tout système violant au moins une propriété importée d'un de ses sous-systèmes ou pour lequel une nouvelle propriété émerge de l'interconnexion des sous-systèmes le constituant.

Dans le domaine de l'informatique, un exemple typique de systèmes perçus comme complexes sont les systèmes à base de services qu'ils rendent et des interactions entre ces derniers. Le domaine des services en télécommunications a fait émerger la notion d'interactions de services signalant des conflits entre les différents services en jeu. Un service téléphonique est défini comme une fonctionnalité supplémentaire au comportement initial du système téléphonique de base, fournie à un abonné ou servant à l'administration du réseau. Des intégrations multiples de tels services peuvent engendrer l'apparition ou la disparition, souhaitée ou non, de propriétés du réseau initial produisant ce que l'on appelle des interactions de services [33, 55]. À titre d'exemple considérons un système téléphonique qui fournit les services suivants :

- *Interdiction d'appels entrants*, qui permet à un abonné, utilisateur d'un poste A, d'interdire toute tentative de communication provenant d'une certaine liste de postes que lui-même a spécifiés. Supposons que le poste B fait partie de cette liste.
- *Communication à trois*, qui permet à un abonné, utilisateur d'un poste C, d'être l'initiateur d'une communication à trois.

Supposons maintenant que l'utilisateur du poste C appelle le poste B et se trouve en communication avec lui. Supposons maintenant que l'utilisateur du poste C initie, durant cette communication, une communication à trois avec le poste A. Si l'utilisateur du poste A décroche, il se retrouve en communication à trois et donc en particulier en communication avec B ce qui contredit la définition du service d'interdiction d'appels entrants.

Pour résoudre le problème des interactions de services, des solutions ont été proposées. Elles sont souvent fondées sur une définition d'une architecture dédiée du système, cloisonnant autant que possible les interactions et résolvant les interactions par des mécanismes de prioritarisation. Ainsi, les systèmes de télécommunication à base de services intègrent les services en résolvant les interactions de services en se basant essentiellement sur des décisions d'expert. Ainsi, par construction, les systèmes à base de services ne préservent donc pas l'ensemble des propriétés attachées aux services les constituant.

Au-delà de la conception des systèmes logiciels, les systèmes complexes sont omniprésents et font l'objet d'études dans plusieurs disciplines : la biologie, la physique, les sciences humaines les sciences cog-

nitives. Sans pour autant cerner complètement la notion même de système complexe, ces disciplines mettent en général en avant les faits suivants : (1) un grand nombre d'entités interagissant entre elles d'une façon non triviale et simultanée, (2) l'émergence au niveau global des nouvelles propriétés non observables au niveau des entités constitutives et (3) une dynamique de fonctionnement global non prévisible à partir de l'observation et de l'analyse des interactions élémentaires entre les différentes entités [116, 41, 46].

Dans le contexte de la biologie, les réseaux de régulation génétique sont bel et bien un exemple de tels systèmes complexes si l'on considère qu'un sous-réseau construit avec quelques gènes constitue un sous-système. On montrera dans la dernière partie de cette thèse que conformément à l'idée commune à propos de la régulation des gènes, le comportement d'un réseau peut être remis en cause lorsqu'il est plongé dans un réseau plus grand où de nouvelles interactions sur le réseau initial peuvent apparaître et perturber ainsi le comportement du sous-réseau initial.

3 Un cas d'étude des systèmes complexes : les réseaux de régulation génétique

Les gènes constituent une collectivité structurée dont l'organisation et le fonctionnement dépendent en partie de l'expression des gènes le constituant : l'expression d'un gène est très souvent sous le contrôle d'autres gènes, via la synthèse de protéines régulatrices [86]. L'état et les propriétés d'une cellule varient selon que les protéines régulatrices sont en mesure de favoriser ou de bloquer, i.e. d'activer ou d'inhiber, l'activité de protéines cibles. Ces interactions entre les gènes sont capturées par la notion de *réseau de régulation génétique*. Ainsi, les réseaux de régulation génétique pilotent le fonctionnement et le développement des organismes vivants.

Afin de comprendre et de prédire les comportements régulés par un réseau de régulation, il existe deux principales pistes d'exploration. La première consiste à constituer les interactions constitutives du réseau pour affiner par organisme vivant la connaissance de la carte des interactions entre gènes. La seconde consiste à étudier les comportements des réseaux. En raison de nombreuses boucles de rétroaction qui interdisent tout raisonnement intuitif, et des connaissances partielles à propos des paramètres cinétiques gouvernant la dynamique des réseaux, seules les dynamiques de petits réseaux sont étudiées, en général en faisant appel à des méthodes et outils de modélisation et de simulation. Les deux points évoqués ci-dessus, c'est-à-dire (1) le manque d'information sur les graphes d'interactions de réseaux de grande taille et (2) la mise en évidence des graphes d'interactions de réseaux de petite taille d'intérêt, sont deux facteurs qui réduisent les perspectives d'étudier des réseaux de régulation pris dans leur ensemble aux dépens de l'étude de leurs éléments constitutifs pris isolément ou en petits groupes.

Ces deux activités sont complémentaires, car la première construit la partie descriptive, ou statique, des réseaux tandis que la seconde s'attache à analyser leur comportement. Dans l'objectif de renforcer la complémen-

tarité des deux démarches, une question s'impose alors : dans quelle mesure les comportements identifiés pour des petits réseaux peuvent-ils être transposés pour les réseaux de régulation génétique englobant le (petit) réseau de régulation sous étude ? Si l'on part de l'idée que les petits réseaux sélectionnés pour être étudiés sous l'angle de la dynamique le sont parce que les biologistes leur attribuent un rôle fonctionnel, c'est-à-dire en charge de pilotage d'une fonction particulière de la cellule, il s'agit alors d'établir quelles sont les règles ou conditions restrictives qui permettent d'exploiter les connaissances établies à propos de la dynamique des sous-réseaux au niveau du réseau complet de régulation génétique. En d'autres termes, sous réserve que ces conditions soient identifiées, tout réseau étudié sous l'angle de la dynamique sera susceptible de représenter un module fonctionnel : le réseau complet pourrait alors être décomposé en un ensemble de modules fonctionnels interagissant entre eux afin de fournir des fonctionnalités plus complexes. Cette vue simplifiée représente un objectif à atteindre. Cependant, cet objectif reste encore hors de portée dans toute sa généralité, essentiellement pour les deux raisons suivantes : (1) ce qui caractérise les sous-réseaux susceptibles de représenter un module ne fait pas encore l'unanimité : identification par des biologistes, un motif récurrent détecté par des méthodes de physiques statistiques, ... et (2) l'exploitation des mécanismes de structuration en modules n'est pas mûre notamment en vue de jeter les bases d'une ingénierie génétique où les modules seraient rangés dans une boîte à outils et combinés à volonté pour produire de nouvelles fonctions.

Dans ce document, nous nous sommes intéressés à la modélisation proposée par R. Thomas [124] sous la forme de modèles discrets multivalués et analysée au travers de propriétés exprimées en logique temporelle [22, 21]. On montrera dans la dernière partie de cette thèse que conformément à l'idée commune à propos de la régulation des gènes, le comportement d'un réseau peut être remis en cause lorsqu'il est plongé dans un réseau plus grand où de nouvelles interactions sur le réseau initial peuvent apparaître et perturber ainsi le comportement du sous-réseau initial. Notre objectif est alors d'identifier quelques configurations pour lesquelles un sous-réseau dont on a cerné quelques propriétés dynamiques jugées caractéristiques peut être retrouvé fidèlement au niveau d'un réseau englobant.

4 Une présentation intuitive de systèmes complexes

Dans cette thèse, nous introduirons un cadre de spécifications au sein duquel il sera possible de caractériser formellement ce qu'est un système complexe. Notre cadre est général puisqu'il est défini au-dessus des institutions. Nous considérons ici qu'un système est complexe dès lors qu'il est composé de plusieurs sous-systèmes interconnectés entre eux et que ses propriétés ne sont pas déductibles de celles des sous-systèmes le composant. Dans le cas contraire, c'est-à-dire, dans le cas où les propriétés du système sont déductibles de celles des composants, le système est qualifié de modulaire. Pour spécifier les interactions des sous-systèmes, nous

introduisons la notion de *connecteur* : un connecteur permet d'assembler plusieurs systèmes pour obtenir un système plus large.

Dans le domaine de l'architecture logicielle, le concept de connecteur a été bien étudié. La définition d'une architecture d'un système comme un ensemble de composants interconnectés par des connecteurs devient largement acceptée dans la communauté [13]. À ce niveau, l'architecture est perçue globalement comme une collection de composants logiciels, une collection de connecteurs (pour décrire les interactions entre composants) et des configurations, c'est-à-dire des assemblages de composants et de connecteurs. Pour définir, spécifier et manipuler ces éléments, de nombreux langages et des outils ont été définis. Il s'agit principalement des langages de description d'architecture (Architecture Description Languages - ADLs) telles que ACME/ADML [53], Wright [12] et Community [51, 50].

Notre caractérisation de système complexe permet de définir dans un cadre unificateur la notion de systèmes complexes et de systèmes modulaires et donc de proposer un cadre suffisamment lâche pour que les propriétés des sous-systèmes soient ou ne soient pas transposables au niveau du système global.

Le potentiel d'un système d'être complexe ou modulaire est fortement lié au connecteur qui lie ses composants. Considérons par exemple un système composé de n sous-systèmes $S_1, \ldots, S_n$ qui sont connectés à travers un connecteur c. Notons par $\mathcal{P}(S_i)$, où $i \in \{1, \ldots, n\}$, et $\mathcal{P}(c(S_1, \ldots, S_n))$ l'ensemble des propriétés attachées respectivement des sous-systèmes S_i et du système global $c(S_1, \ldots, S_n)$. Intuitivement, nous définirons les systèmes complexes à partir des idées suivantes :

- $c(S_1, \ldots, S_n)$ est considéré comme complexe si,
  - au moins une propriété d'un composant S_i n'est pas transporté au niveau du système global $c(S_1, \ldots, S_n)$ ou
  - au moins une propriété du système global $c(S_1, \ldots, S_n)$ ne peut être déduite des propriétés des sous-systèmes S_i.
- il est considéré comme modulaire s'il n'est pas complexe.

Ainsi, les propriétés émergentes seront de deux types :

- Les propriétés qui sont des propriétés de certains composants quand ils sont pris isolément, mais qui ne sont pas des propriétés du système global et vice versa.
- Les propriétés du système global qui ne peuvent pas être inférées à partir des propriétés des composants.

Notre contribution consiste donc à présenter une caractérisation formelle générique unificatrice de la notion de systèmes complexes et de systèmes modulaires centrée sur la notion de composant/propriété, et permettant l'utilisation de technique formelle pour la vérification. L'enjeu devient donc de faciliter la classification des propriétés en caractérisant des conditions suffisantes, voire nécessaire pour assurer une classification

aisée des propriétés héritées des sous-systèmes. Dans ce cadre, nous définirons un connecteur générique comme moyen de composition d'un système global à partir de sous-systèmes et nous identifierons des conditions nécessaires et/ou suffisantes pour que les propriétés des sous-systèmes soient héritées par le système global [8, 9].

Nous allons illustrer ce cadre générique sur deux exemples concrets de nature différente pour appuyer nos propos. Le premier fait écho aux problématiques rencontrées dans le monde du génie logiciel. Nous avons étudié un formalisme dédié à la spécification des systèmes réactifs où une spécification est un système de transitions étiquetées. Les propriétés d'un système réactif sont exprimées en logique modale équationnelle et sont interprétées dans des systèmes de transitions dont les états sont des algèbres. Nous avons fourni des conditions suffisantes de préservation de propriétés lorsque deux spécifications sont composées à l'aide du produit synchronisé [7].

Une dernière partie de ce manuscrit est consacrée à la problématique de l'émergence des propriétés au sein des réseaux de régulation génétique. Nous avons étudié le plongement des réseaux de régulation génétique dans le cadre de la modélisation discrète introduite par R. Thomas et nous avons fourni de nouveaux résultats en mettant à jour des conditions suffisantes de préservation de propriétés exprimées sous forme de logique temporelle, susceptible de caractériser des modules fonctionnels au sein des réseaux de régulation [89].

5 Organisation du document

La thèse se compose de trois parties organisées comme suit :

La première partie est consacrée à la présentation des résultats fondamentaux de la modularité des spécifications algébriques structurées. Nous présentons tout d'abord un état de l'art des spécifications algébriques et de leurs primitives de structuration. Ensuite, nous présentons une contribution à la notion de modularité et de complexité au travers de la structuration, par enrichissement ou par union, induite par les primitives de structuration correspondantes, et à l'étude des conditions sous lesquelles la modularité d'une spécification algébrique structurée, par enrichissement ou par union, est assurée.

La seconde partie présente une généralisation des notions de structuration, modularité et complexité à des spécifications abstraites. Ces spécifications étant définies indépendamment de toute logique, nous présentons tout d'abord le cadre général des institutions afin de fixer les notions et les notations des différentes composantes d'une logique dont nous aurons besoin pour définir le cadre des spécifications abstraites. Nous définissons ensuite notre cadre générique de spécifications. Pour structurer les spécifications, nous définirons au travers des notions catégorielles de diagrammes et de colimites, un connecteur architectural générique permettant d'interconnecter des spécifications abstraites entre elles pour en construire d'autres plus larges. Les notions de la modularité et de la

complexité que nous avons définies dans le cadre des spécifications algébriques sont ensuite généralisées dans le cadre de ce connecteur générique. Dans ce cadre, nous fournirons des conditions nécessaires et/ou suffisantes pour qu'un connecteur soit modulaire. Nous illustrerons toutes les notions définies dans cette partie au travers deux exemples : les spécifications axiomatiques et les spécifications des systèmes réactifs.

La troisième partie est consacrée aux problématiques de l'émergence des propriétés au sein des réseaux de régulation génétique. Nous présentons tout d'abord les réseaux de régulation génétique dans la modélisation discrète introduite par R. Thomas [125, 124]. Ensuite, nous définissons le plongement d'un réseau de régulation dans un réseau de régulation plus large. Enfin, nous présenterons des conditions suffisantes sur le plongement pour assurer la préservation d'une certaine classe de propriétés, exprimées sous forme de logique temporelle, et susceptibles de caractériser des fonctions biologiques au sein des réseaux de régulation génétique.

L'annexe A est un bref récapitulatif des notions de la théorie des catégories nécessaires à la lecture de cette thèse.

L'annexe B inclut les preuves de quelques théorèmes du chapitre 6 de la partie III.

Première partie

Structuration et modularité dans le cadre des spécifications algébriques

Spécifications algébriques

1

Sommaire

Les spécifications algébriques ont émergé dans les années 70, avec les travaux de Liskov et Zilles [87], de Guttag et Horning [66, 67], et du groupe ADJ composé de Goguen, Thatcher, Wagner et Wright [62, 64]. Ce type de spécifications a montré son efficacité pour spécifier/modéliser le comportement des programmes informatiques dits standards, c'est-à-dire des programmes définis par des structures de données souvent complexes mais des structures de contrôle simples[1]. L'idée principale des spécifications algébriques consiste à décrire les programmes standards au travers des structures de données qu'ils manipulent. Toute structure de données peut être vue comme une structure algébrique, c'est-à-dire un ensemble de valeurs muni d'applications et d'éléments distingués. Par exemple, les listes d'entiers peuvent être vues comme l'ensemble de tous les mots sur $\mathbb{N}$ sur lequel on peut insérer ou retirer un entier au travers des opérations d'ajout et de suppression et de distinguer parmi tous ces mots le mot vide pour dénoter la liste vide. Ainsi, les structures de données partagent avec les structures algébriques mathématiques usuelles telles que les groupes, les anneaux ou les corps leur forme. Cependant, les structures de données diffèrent des structures mathématiques usuelles en ce que l'ensemble des valeurs sous-jacent est souvent construit inductivement. Par exemple, l'ensemble des listes d'entiers peut être défini inductivement par la liste vide et l'opération d'ajout d'un entier en tête de liste. C'est pour cette dernière raison qu'il est aisé de programmer des fonctions sur ces structures de données.

[1] Ceci les différencie des systèmes informatiques tels que les systèmes distribués et réactifs qui se définissent par des structures de données simples mais des structures de contrôles compliquées (parallélisme, entrelacement, ...). Pour de tels systèmes, ce sont surtout les logiques modales (surtout temporelles) et les systèmes de transitions de toute sorte les mieux adaptés pour à la fois raisonner dessus et modéliser de tels systèmes.

Cette forme commune permet naturellement d'utiliser le formalisme des algèbres universelles et sa logique sous-jacente, *la logique équationnelle*, pour spécifier les structures de données et donc les programmes les manipulant. Ainsi, les spécifications algébriques sont des systèmes axiomatiques (i.e. des ensembles des propriétés élémentaires) définis dans la logique équationnelle permettant de décrire le comportement de chacune des opérations du type de données considéré.

1 La logique équationnelle

Comme toute logique, la logique équationnelle est définie par un volet syntaxique et un volet sémantique[2]. Pour une présentation plus détaillée de la logique équationnelle, incluant les spécifications algébriques, le lecteur pourra se référer à [88] ou à [118].

1.1 Syntaxe

1.1.1 Signatures

En pratique, le volet syntaxique repose sur la construction inductive de formules à partir de symboles de fonctions, de connecteurs logiques et de quantificateurs. Ce que l'on remarque alors dans la définition d'un formalisme de spécification, est qu'elle contient des parties fixes telles que l'ensemble des connecteurs logiques qu'elle accepte. Elle contient aussi des parties utilisateurs que l'on introduit pour résoudre un problème donné. Par exemple, si l'on veut raisonner sur les entiers naturels, on va se placer dans un langage où l'on dispose des fonctions *zero*, *succ*, $+$, $\times$, etc, alors que si l'on veut traiter des propriétés des files d'attente, on va se munir d'une constante représentant une file vide, d'opérations d'ajout, de retrait, etc. La partie utilisateur évoquée ci-dessus, est appelée classiquement une *signature*. Elle déclare les divers éléments mis en jeu dans la spécification.

Définition 1.1 (Signature) *Une* ***signature*** Σ *est un couple* (S, F) *où :*

- S *est un ensemble fini dont les éléments sont appelés* ***sortes*** *ou* ***types***,
- $F = (F_{w,s})_{w \in S^*, s \in S}$ *est une famille d'ensembles indexés par* $S^* \times S$ *de symboles d'****opérations***[3]. *Chaque* $f \in F_{w,s}$ *est une opération d'arité* w, *de sorte* s *et de profil* (w, s). *On écrit* $f : s_1 \times \cdots \times s_n \to s$ *où* $w = s_1 \dots s_n$. *Si* f *est d'arité* 0 *alors* w *est le mot vide* ε *et on dit que* $f \in F_{\varepsilon,s}$ *est une* ***constante*** *de sorte* s *qu'on note alors* $f :\to s$.

Pour des raisons de commodité d'usage, il est aussi possible d'utiliser une notation plus classique pour certaines opérations que la notation purement fonctionnelle. Dans cette notation, dite infixée, les éléments de l'arité figurent de part et d'autre des éléments de notation de l'opération. On prendra alors soin de placer dans le nom de l'opération (au niveau du profil) un certains nombres de caractères _, le $n$ième caractère _ symbolisant la place du $n$ième élément de l'arité.

Exemple 1.1 *Une signature* $\Sigma_{nat} = (S_{nat}, F_{nat})$ *de l'arithmétique usuelle peut être :*

$$S_{nat} = \{nat, bool\}$$

[2] Il y a aussi un volet symbolique défini par un ensemble de règles d'inférence qui traduit correctement et le plus complètement possible la sémantique. Nous ne présentons pas ici le volet calcul, car nous donnerons une caractérisation uniquement sémantique de la complexité. En outre, dans la partie II, nous généraliserons les résultats présentés ici dans un cadre abstrait des formalismes logiques, les institutions, qui généralisent la partie sémantique de ces formalismes, et n'incluent donc pas le volet calcul.

[3] S^* désigne l'ensemble des suites finies construites sur S.

$$
\begin{aligned}
F_{nat} = \{ & \mathit{zero} :\rightarrow \mathit{nat}\,, \\
& \mathit{succ} : \mathit{nat} \rightarrow \mathit{nat}\,, \\
& \_ + \_ : \mathit{nat} \times \mathit{nat} \rightarrow \mathit{nat}\,, \\
& \_ - \_ : \mathit{nat} \times \mathit{nat} \rightarrow \mathit{nat}\,, \\
& \_ * \_ : \mathit{nat} \times \mathit{nat} \rightarrow \mathit{nat}\,, \\
& \mathit{vrai} :\rightarrow \mathit{bool}\,, \\
& \mathit{faux} :\rightarrow \mathit{bool}\,, \\
& \_ < \_ : \mathit{nat} \times \mathit{nat} \rightarrow \mathit{bool}\,\}
\end{aligned}
$$

Notons qu'aucune signification particulière n'est, pour l'instant, associée à ces symboles. Bien entendu, on pense à interpréter par exemple zero comme le zéro des entiers ou succ comme le successeur d'un entier, mais il serait tout à fait possible, comme nous le verrons en section 1.2, d'interpréter ces symboles d'une autre manière.

Exemple 1.2 *Une signature associée aux listes d'entiers naturels peut être* $\Sigma_{liste} = (S_{liste}, F_{liste})$ *:*

$$
\begin{aligned}
S_{liste} = \; & S_{nat} \cup \{\mathit{liste}\} \\
F_{liste} = \; & F_{nat} \cup \{\, \mathit{nil} :\rightarrow \mathit{liste}\,, \\
& \qquad \mathit{vide} : \mathit{liste} \rightarrow \mathit{bool}\,, \\
& \qquad \mathit{cons} : \mathit{nat} \times \mathit{liste} \rightarrow \mathit{liste}\,, \\
& \qquad \mathit{appartient} : \mathit{nat} \times \mathit{liste} \rightarrow \mathit{bool}\,, \\
& \qquad \mathit{tête} : \mathit{liste} \rightarrow \mathit{nat}\,, \\
& \qquad \mathit{queue} : \mathit{liste} \rightarrow \mathit{liste}\,\}
\end{aligned}
$$

Pour refléter la structuration dans les programmes qui est souvent définie par l'ajout de nouveaux types et de nouvelles opérations tels que l'héritage dans la programmation orientée objet, ou la déclaration de nouvelles structures qui spécialisent celles déjà existantes dans la programmation modulaire, nous introduisons la notion de morphisme de signatures.

Définition 1.2 (Morphisme de signatures) *Soient* $\Sigma = (S, F)$ *et* $\Sigma' = (S', F')$ *deux signatures. Un* ***morphisme de signatures*** $\sigma : \Sigma \rightarrow \Sigma'$ *est défini par :*

- *une application* $\sigma_S : S \rightarrow S'$ *;*
- *une application* $\sigma_F : F \rightarrow F'$ *telle que,* $\forall f :\rightarrow s \in F, \sigma_F(f) :\rightarrow \sigma_S(s) \in F'$ *et* $\forall f : s_1 \times \cdots \times s_n \rightarrow s \in F$, $\sigma_F(f) : \sigma_S(s_1) \times \cdots \times \sigma_S(s_n) \rightarrow \sigma_S(s) \in F'$.

Notation 1.1 *Soient* $\Sigma = (S, F)$ *et* $\Sigma' = (S', F')$ *deux signatures telles que* $S \subseteq S'$ *et* $F \subseteq F'$. *On note* $\Sigma \hookrightarrow \Sigma'$ *le morphisme d'inclusion définie par les applications* $S \hookrightarrow S'$ *associant à tout élément de S l'élément lui même et* $F \hookrightarrow F'$ *associant à tout élément de F l'élément lui même.*

Exemple 1.3 *Comme* $S_{nat} \subseteq S_{liste}$ *et* $F_{nat} \subseteq F_{liste}$, *on peut alors considérer le morphisme d'inclusion* $\Sigma_{nat} \hookrightarrow \Sigma_{liste}$.

Bien entendu, la définition de morphisme de signatures est plus générale que le simple ajout de types et d'opérations défini par un morphisme d'inclusion.

Les morphismes de signatures étant définis à partir de deux applica-

tions, il est aisé de définir leur composition. Ainsi, les signatures munies des morphismes forment une catégorie[4] notée SIG_{EL}.

1.1.2 Formules

Une fois que l'on a fixé l'ensemble des symboles associés à un type de données, il est possible de former des expressions, *les formules*, visant à décrire les propriétés de ce type de données. Pour définir les formules, nous devons tout d'abord introduire les premiers éléments syntaxiques que sont *les termes*. Les termes vont dénoter des valeurs symboliques. Pour permettre de manipuler des valeurs génériques, nous introduisons au préalable la notion de *variable*.

Définition 1.3 (S-ensemble) *Étant donné un ensemble S de sortes, un **S-ensemble** est un ensemble A muni d'une partition indexée par S :*

$$A = \amalg_{s \in S} A_s$$

où $\amalg$ désigne l'union disjointe.

Définition 1.4 (Ensemble de variables) *Soit $\Sigma = (S, F)$ une signature. On appelle **ensemble de variables** sur Σ un S-ensemble V tel que pour tout $s \in S$, $V_s \cap (S \cup F) = \emptyset$.*

Définition 1.5 (Termes) *Soit $\Sigma = (S, F)$ une signature. Soit V un ensemble de variables sur Σ. Le S-ensemble des **termes**, noté $T_\Sigma(V)$, est l'ensemble défini de la façon suivante :*

- *pour tout $x \in V_s$, $x \in T_\Sigma(V)_s$;*
- *pour tout $f \in F_{\varepsilon,s}$, $f \in T_\Sigma(V)_s$;*
- *pour tout $f : s_1 \times \cdots \times s_n \to s \in F$, pour tout $(t_1, \ldots, t_n) \in T_\Sigma(V)_{s_1} \times \cdots \times T_\Sigma(V)_{s_n}$, $f(t_1, \ldots, t_n) \in T_\Sigma(V)_s$.*

*Un terme sans variables est appelé un **terme clos**, et on note par T_Σ le S-ensemble des termes clos $T_\Sigma(\emptyset)$.*

Exemple 1.4 (Liste) *Sur la signature donnée dans l'exemple 1.2, et sur l'ensemble des variables $V_{nat} = \{x, y\}$ et $V_{liste} = \{l, ll\}$, on peut construire les termes suivants :*

$$nil, cons(x, l), cons(y, cons(x, l)), cons(tête(l), queue(l))$$

Définition 1.6 (Équation) *Soit $\Sigma = (S, F)$ une signature. Soit V un ensemble de variables sur Σ. Une **équation** sur Σ est un élément de $T_\Sigma(V)_s \times T_\Sigma(V)_s$, pour $s \in S$. Pour tout $t, t' \in T_\Sigma(V)_s$, on notera $t = t'$ l'équation (t, t').*

Les formules sont construites inductivement à partir des équations, des connecteurs booléens usuels, et des quantificateurs universel et existentiel.

Définition 1.7 (Formules) *Soit $\Sigma = (S, F)$ une signature. Soit V un ensemble de variables sur Σ. L'ensemble des **formules** sur Σ, noté $For_{EL}(\Sigma)$, est l'ensemble défini de la façon suivante :*

- *pour tout $s \in S$, pour tout $t, t' \in T_\Sigma(V)_s$, $t = t' \in For_{EL}(\Sigma)$;*
- *pour tout $\varphi \in For_{EL}(\Sigma)$, $\neg\varphi \in For_{EL}(\Sigma)$;*
- *pour tout φ et ψ de $For_{EL}(\Sigma)$, $\varphi \wedge \psi$, $\varphi \vee \psi$ et $\varphi \Rightarrow \psi$ appartiennent à $For_{EL}(\Sigma)$;*
- *pour tout $x \in V$, pour tout $\varphi \in For_{EL}(\Sigma)$, $\forall x\varphi$ et $\exists x\varphi$ appartiennent à $For_{EL}(\Sigma)$.*

[4] Voir la définition A.1 de l'annexe A pour la notion de catégorie.

Parfois, pour alléger les notations, on omet d'indiquer le symbole $\forall$ dans les formules, on dit alors que les variables sont implicitement quantifiées universellement.

Exemple 1.5 (Liste) *On peut exprimer quelques propriétés caractérisant les opérations sur la signature des listes d'entiers naturels donnée dans l'exemple 1.2 par les formules suivantes :*

$$tête(cons(x,l)) = x$$
$$queue(cons(x,l)) = l$$
$$cons(tête(l), queue(l)) = l$$

Pour ses bonnes propriétés algébriques telles que l'existence d'un modèle initial ou encore d'un morphisme d'adjonction au travers des morphismes de signatures (voir la section 3.1.3 du chapitre 2), mais aussi ses résultats de décidabilité, nous introduisons ci-après une restriction des formules, *les formules conditionnelles positives*. Ces formules permettent d'exprimer le fait qu'une propriété, sous forme d'une équation, est conséquence d'un ensemble de conditions, exprimé sous la forme d'une conjonction d'équations. Cette restriction nous sera utile pour la suite pour donner des conditions suffisantes à une bonne modularité des systèmes structurés par les primitives de structuration présentées en section 2 du chapitre 2.

Définition 1.8 (Formules conditionnelles positives) *Soit Σ une signature. Une **formule conditionnelle positive** sur Σ est une formule sur Σ de la forme :*

$$t_1 = t'_1 \wedge \cdots \wedge t_{n-1} = t'_{n-1} \Rightarrow t_n = t'_n$$

où $t_i = t'_i$ est une équation sur Σ pour tout i, $1 \leq i \leq n$.

Tout morphisme de signature $\sigma : \Sigma \to \Sigma'$ se prolonge aux termes puis aux formules en changeant les symboles de Σ apparaissant dans les termes et les formules par les symboles correspondants dans Σ'.

Définition 1.9 (Prolongement des morphismes de signatures aux ensembles des formules) *Soient $\Sigma = (S, F)$ et $\Sigma' = (S', F')$ deux signatures. Soient V et V' deux ensembles de variables respectivement sur Σ et Σ'. Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signature. On définit $\sigma_V : V \to V'$ une application injective*[5] *telle que pour tout $s \in S$, pour tout $x \in V_s$, $\sigma_V(x) \in V'_{\sigma_S(s)}$.*

*Le **prolongement de σ aux termes**, noté $\ddot{\sigma} : T_\Sigma(V) \to T_{\Sigma'}(V')$, est défini pour tout terme t inductivement sur la forme de t de la façon suivante :*

- *si t est une variable $x \in V$, alors $\ddot{\sigma}(x) = \sigma_V(x)$;*
- *si t est une constante $f \in F_{\varepsilon,s}$, alors $\ddot{\sigma}(f) = \sigma_F(f)$;*
- *si t est de la forme $f(t_1, \ldots, t_n)$, alors $\ddot{\sigma}(t) = \sigma_F(f)(\ddot{\sigma}(t_1), \ldots, \ddot{\sigma}(t_n))$*

***Le prolongement de $\ddot{\sigma}$ aux formules**, noté $\overline{\sigma} : For_{EL}(\Sigma) \to For_{EL}(\Sigma')$, est défini pour toute formule φ inductivement sur la forme de φ de la façon suivante :*

- *si φ est de la forme $t = t'$, avec $t, t' \in T_\Sigma(V)_s$ et $s \in S$, alors $\overline{\sigma}(\varphi)$ vaut $\ddot{\sigma}(t) = \ddot{\sigma}(t')$;*
- *si φ est de la forme $\neg\psi$, alors $\overline{\sigma}(\varphi)$ vaut $\neg\overline{\sigma}(\psi)$;*

[5] La propriété d'être injective pour σ_V est importante pour obtenir une propriété fondamentale à une bonne structuration qui est *la condition de satisfaction* (voir la section 3.2 du chapitre 2).

- *si φ est de la forme $\psi \ominus \varrho$ avec $\ominus \in \{\wedge, \vee, \Rightarrow\}$, alors $\overline{\sigma}(\varphi)$ vaut $\overline{\sigma}(\psi) \ominus \overline{\sigma}(\varrho)$;*
- *si φ est de la forme $Qx\psi$ avec $Q \in \{\forall, \exists\}$, alors $\overline{\sigma}(\varphi)$ vaut $Q\sigma_V(x)\overline{\sigma}(\psi)$.*

1.2 Sémantique

Les signatures, termes et formules ne sont que des assemblages de symboles. Pour pouvoir raisonner sur les spécifications algébriques et donc le comportement des systèmes spécifiés, il est nécessaire d'associer une signification mathématique à tous ces symboles et ces assemblages de symboles.

1.2.1 Algèbre

Comme nous l'avons vu précédemment, un type de données est une structure algébrique, c'est-à-dire un ensemble de valeurs représentant ces données muni d'éléments distingués ainsi que d'opérations agissant sur ces valeurs. On parlera d'algèbre associée à une signature. Parfois, ces structures sont aussi appelées *modèles*.

Définition 1.10 (Algèbre) *Soit $\Sigma = (S, F)$ une signature. Une **algèbre** $\mathcal{A}$ de Σ est un S-ensemble A, appelé **support de l'algèbre**, muni pour chaque symbole d'opération $f : s_1 \times \cdots \times s_n \rightarrow s \in F$ d'une application $f_{\mathcal{A}} : A_{s_1} \times \cdots \times A_{s_n} \rightarrow A_s$, l'interprétation de f. Si $f :\rightarrow s \in F$, alors $f_{\mathcal{A}} \in A_s$, et est appelée une constante de A.*

Une algèbre peut être alors vue comme l'abstraction mathématique des réalisations possibles (i.e. des programmes sous-jacents). En effet, supposons le programme suivant calculant le PGCD (Plus Grand Commun Diviseur) de deux entiers positifs décrit dans un pseudo-langage où les mots clés sont donnés en français et sont utilisés conformément à l'usage commun :

```
Fonction pgcd( n, m : entier) : entier
    res : entier
    Si (n < m) Alors
        res ← pgcd(m,n) ;
    Sinon
        Si (n mod m = 0) Alors
            res ← m ;
        Sinon
            res ← pgcd(m,n mod m) ;
        Fin Si
    Fin Si
    Retourner res ;
Fin
Fonction mod( n, m : entier) : entier
    r : entier
    r ← n - m ;
    Si (m < r) Alors
        r ← mod(n-m,m) ;
    Fin Si
    Retourner r ;
Fin
```

Algorithme 1 – *PGCD de deux entiers positifs*

Pour obtenir une signature associée à la spécification du PGCD de deux entiers positifs, on peut étendre la signature Σ_{nat} de l'exemple 1.1 par l'ajout de deux symboles : un symbole pour le calcul du PGCD et un pour le calcul du reste d'une division euclidienne. Ainsi on obtient la signature $\Sigma_{pgcd} = (S_{pgcd}, F_{pgcd})$, où :

- $S_{pgcd} = S_{nat}$;
- $F_{pgcd} = F_{nat} \cup \{pgcd : nat \times nat \to nat, mod : nat \times nat \to nat\}$.

Ainsi, à l'algorithme ci-dessus, on peut trivialement associer l'algèbre $\mathcal{A}$ de Σ_{pgcd} définie par le S-ensemble A, avec $A_{nat} = \mathbb{N}$ et $A_{bool} = \{0, 1\}$, muni des applications suivantes :

$zero_{\mathcal{A}} = 0$, $+_{\mathcal{A}} = +_{\mathbb{N}}$, $-_{\mathcal{A}} = -_{\mathbb{N}}$, $*_{\mathcal{A}} = *_{\mathbb{N}}$, $<_{\mathcal{A}}=<_{\mathbb{N}}$, $vrai_{\mathcal{A}} = 1$, $faux_{\mathcal{A}} = 0$, où $+_{\mathbb{N}}, -_{\mathbb{N}}, *_{\mathbb{N}}$ et $<_{\mathbb{N}}$ sont les fonctions naturelles sur les entiers,

$succ_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ définie pour tout $n \in \mathbb{N}$ par $Succ_{\mathcal{A}}(n) = n +_{\mathbb{N}} 1$,

$mod_{\mathcal{A}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ définie pour tout $n, m \in \mathbb{N}$ par

$$mod_{\mathcal{A}}(n,m) = \begin{cases} m -_{\mathbb{N}} n \text{ si } (n -_{\mathbb{N}} m <_{\mathbb{N}} m) = 1 \\ mod_{\mathcal{A}}(n -_{\mathbb{N}} m, m) \text{ sinon} \end{cases}$$

et $pgcd_{\mathcal{A}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ définie pour tout $n, m \in \mathbb{N}$ par

$$pgcd_{\mathcal{A}}(n,m) = \begin{cases} m & \text{si } mod_{\mathcal{A}}(n,m) = 0 \\ pgcd_{\mathcal{A}}(m,n) & \text{si } (n <_{\mathbb{N}} m) = 1 \\ pgcd_{\mathcal{A}}(m, mod_{\mathcal{A}}(n,m)) & \text{sinon} \end{cases}$$

Il existe des algèbres particulières dans la classe des algèbres associées à une signature : l'*algèbre syntaxique*, l'*algèbre des termes clos* et l'*algèbre tri-*

viale. Nous présentons ces dernières, car elles nous seront utiles en section 3.1 du chapitre 2.

Définition 1.11 (Algèbre syntaxique, Algèbre des termes clos) *Soit $\Sigma = (S, F)$ une signature et V un ensemble de variables sur Σ.*

*L'**algèbre syntaxique** de Σ sur V, notée $\mathcal{T}_\Sigma(V)$, est le S-ensemble $T_\Sigma(V)$, muni pour chaque opération $f : s_1 \times \cdots \times s_n \to s \in F$ de l'application $f_{\mathcal{T}_\Sigma(V)} : T_\Sigma(V)_{s_1} \times \cdots \times T_\Sigma(V)_{s_n} \to T_\Sigma(V)_s$ qui à tout élément $(t_1, \ldots, t_n)$ associe $f(t_1, \ldots, t_n)$.*

*Si pour tout $s \in S, (T_\Sigma)_s \neq \emptyset$, l'**algèbre des termes clos** $\mathcal{T}_\Sigma$ est définie comme l'algèbre syntaxique quand $V = \emptyset$.*

Définition 1.12 (Algèbre libre) *Soit Σ une signature, et soit $\mathcal{A}$ une algèbre de Σ. L'algèbre particulière $\mathcal{T}_\Sigma(A)$ est appelée **algèbre libre** de Σ engendrée par A. À partir de cette algèbre libre, on peut définir de façon canonique l'évaluation $ev_\mathcal{A} : \mathcal{T}_\Sigma(A) \to \mathcal{A}$ qui est le prolongement de l'identité de A, et qui est définie par :*

- *pour tout $s \in S$, pour tout $a \in A_s$, $ev_\mathcal{A}(a) = a$;*
- *pour tout $f : s_1 \times \cdots \times s_n \to s \in F$, pour tout $(t_1, \ldots, t_n) \in T_\Sigma(A)_{s_1} \times \cdots \times T_\Sigma(A)_{s_n}$, $ev_\mathcal{A}(f(t_1, \ldots, t_n)) = f_\mathcal{A}(ev_\mathcal{A}(t_1), \ldots, ev_\mathcal{A}(t_n))$*

Définition 1.13 (Algèbre triviale) *Soit $\Sigma = (S, F)$ une signature. L'**algèbre triviale** de Σ, notée $\mathcal{T}riv$, est le S-ensemble Triv tel que pour tout s, $Triv_s = \{s\}$, et muni pour chaque opération $f : s_1 \times \cdots \times s_n \to s \in F$ d'une application $f_{\mathcal{T}riv} : Triv_{s_1} \times \cdots \times Triv_{s_n} \to Triv_s$ qui à l'unique élément $(s_1, \ldots, s_n)$ associe s.*

Nous verrons dans la suite que, par cette démarche axiomatique, nous pouvons associer à une spécification non pas une réalisation, mais plusieurs possibles. Nous devons alors disposer d'un moyen de comparer ces réalisations entre elles pour choisir les plus pertinentes par exemple celle répondant exactement à la spécification (définie comme l'algèbre initiale, voir la section 3.1 du chapitre 2). Le moyen de comparer les algèbres entre elles se définit, comme pour les signatures, au travers de la notion d'*homomorphismes*.

Définition 1.14 (Homomorphisme) *Soit $\Sigma = (S, F)$ une signature. Soient $\mathcal{A}$ et $\mathcal{A}'$ deux algèbres de Σ. Un **Σ-homomorphisme d'algèbres** $h : \mathcal{A} \to \mathcal{A}'$ est une famille d'applications $h = (h^s : A_s \to A'_s)_{s \in S}$ telle que pour toute opération $f : s_1 \times \cdots \times s_n \to s \in F$, pour tout $(a_1, \ldots, a_n) \in A_{s_1} \times \cdots \times A_{s_n}$, on a :*

$$h^s(f_\mathcal{A}(a_1, \ldots, a_n)) = f_{\mathcal{A}'}(h^{s_1}(a_1), \ldots, h^{s_n}(a_n))$$

Fait 1.1 *Soit Σ une signature, et soient $\mathcal{A}$ et $\mathcal{A}'$ deux algèbres de Σ et soit $h : \mathcal{A} \to \mathcal{A}'$ un Σ-homomorphisme d'algèbres.*

L'application $\overline{h} : \mathcal{T}_\Sigma(A) \to \mathcal{T}_\Sigma(A')$ qui est le prolongement de h sur $\mathcal{T}_\Sigma(A)$ définie de la façon suivante :

- *pour tout $s \in S$, pour tout $a \in A_s$, $\overline{h}(a) = h(a)$;*
- *pour tout $f : s_1 \times \cdots \times s_n \to s \in F$, pour tout $(t_1, \ldots, t_n) \in T_\Sigma(A)_{s_1} \times \cdots \times T_\Sigma(A)_{s_n}$, $\overline{h}(f(t_1, \ldots, t_n)) = f(\overline{h}(t_1), \ldots, \overline{h}(t_n))$.*

est un homomorphisme d'algèbres associées à Σ.

Les homomorphismes étant définis à partir d'applications, leur composition est définie à partir de la composition des applications sous-jacentes.

Les algèbres associées à une signature Σ munies des homomorphismes d'algèbres ainsi définis, forment alors une catégorie notée $Alg(\Sigma)$.

De la même manière qu'un morphisme de signature $\sigma : \Sigma \to \Sigma'$ définissait une application $\overline{\sigma} : For_{EL}(\Sigma) \to For_{EL}(\Sigma')$, ce même morphisme de signature définit un foncteur[6] $U_\sigma : Alg(\Sigma') \to Alg(\Sigma)$ appelé foncteur d'oubli.

Définition 1.15 (Foncteur d'oubli U_σ) *Soient Σ et Σ' deux signatures. Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signatures. On définit le* ***foncteur d'oubli*** *$U_\sigma : Alg(\Sigma') \to Alg(\Sigma)$ comme étant le foncteur :*

- *qui associe à chaque algèbre $\mathcal{A}' \in |Alg(\Sigma')|$ l'algèbre[7] $U_\sigma(\mathcal{A}') \in |Alg(\Sigma)|$ telle que l'ensemble sous-jacent $U_\sigma(A')$ est tel que pour tout $s \in S, U_\sigma(A')_s = A'_{\sigma_S(s)}$, et muni pour chaque symbole d'opération $f : s_1 \times \cdots \times s_n \to s \in F$ de l'application $f_{U_\sigma(\mathcal{A}')} = \sigma_F(f)_{\mathcal{A}'}$;*
- *qui associe à chaque Σ'-homomorphisme d'algèbres $h' : \mathcal{A}'_1 \to \mathcal{A}'_2$ le Σ-homomorphisme d'algèbres $U_\sigma(h') : U_\sigma(\mathcal{A}'_1) \to U_\sigma(\mathcal{A}'_2)$ défini pour tout $s \in S$ par $U_\sigma(h')^s = h'^{\sigma_S(s)}$.*

Soit $\mathcal{A}' \in |Alg(\Sigma')|$, on note aussi $\mathcal{A}'_{|\sigma}$ la Σ-algèbre $U_\sigma(\mathcal{A}')$. Si $\Sigma \hookrightarrow \Sigma'$, on note $U_{|\Sigma}$ le foncteur d'oubli et $\mathcal{A}'_{|\Sigma}$ la Σ-algèbre $U_{|\Sigma}(\mathcal{A}')$.

1.2.2 Algèbres remarquables

Les algèbres présentées dans les définitions 1.11 et 1.13 ont des propriétés remarquables dans la catégorie $Alg(\Sigma)$ que nous énonçons ci-dessous et qui seront utiles à notre propos en section 3.1 du chapitre 2.

Théorème 1.1 *Soit Σ une signature. L'algèbre des termes clos $\mathcal{T}_\Sigma$ est initiale[8] dans $Alg(\Sigma)$.*

Preuve.

Soit V un ensemble quelconque de variables sur Σ. Pour toute Σ-algèbre $\mathcal{A}$ et toute application $\nu : V \to A$ préservant les types, il existe un unique Σ-homomorphisme $\overline{\nu} : \mathcal{T}_\Sigma(V) \to \mathcal{A}$ tel que pour tout $v \in V$, $\overline{\nu}(v) = \nu(v)$. $\overline{\nu}$ est défini par induction sur les termes par :

$$\overline{\nu}(v) = \nu(v)$$

$$\overline{\nu}(f(t_1, \ldots, t_n)) = f_{\mathcal{A}}(\overline{\nu}(t_1), \ldots, \overline{\nu}(t_n))$$

Ceci s'applique en particulier à $\mathcal{T}_\Sigma$ quand $V = \emptyset$. Donc pour toute Σ-algèbre $\mathcal{A}$ il existe un unique Σ-homomorphisme $\_{\mathcal{A}} : \mathcal{T}_\Sigma \to \mathcal{A}$, qui est défini par :

$$f(t_1, \ldots, t_n)_{\mathcal{A}} = f_{\mathcal{A}}(t_{1_{\mathcal{A}}}, \ldots, t_{n_{\mathcal{A}}})$$

Ainsi, $\mathcal{T}_\Sigma$ est initiale dans $Alg(\Sigma)$. □

À partir de l'homomorphisme $\_{\mathcal{A}}$, on peut définir une sous-catégorie d'algèbres intéressante, les *algèbres finiment engendrées*, qui correspondent à des quotients de l'algèbre initiale. Modulo ce quotient, elles possèdent aussi certaines des propriétés de l'algèbre initiale $\mathcal{T}_\Sigma$ telle que la possibilité de raisonner par induction sur les termes.

[6] Voir la définition A.7 de l'annexe A pour la notion de foncteur.

[7] La notion $|Alg(\Sigma')|$ désigne la classe des objets de la catégorie $Alg(\Sigma')$, voir la définition A.1 de l'annexe A.

[8] Voir la définition A.6 de l'annexe A pour la notion d'objet initial ou final dans une catégorie.

Définition 1.16 (Algèbre finiment engendrée) *Soit $\Sigma = (S, F)$ une signature et V un ensemble de variables sur Σ. Soit $\mathcal{A} \in |Alg(\Sigma)|$ une algèbre de Σ. On dit que $\mathcal{A}$ est **finiment engendrée** si et seulement si $\_^{\mathcal{A}}$ est surjective, c'est-à-dire que pour tout $s \in S$, pour tout $a \in A_s$, il existe au moins un terme clos $t \in (T_\Sigma)_s$ tel que $t^{\mathcal{A}} = a$. On note $Gen(\Sigma)$ la sous-catégorie pleine[9] de $Alg(\Sigma)$ dont les objets sont les algèbres de Σ finiment engendrées.*

Théorème 1.2 *Soit Σ une signature. L'algèbre triviale $\mathcal{T}riv$ est finale dans $Alg(\Sigma)$.*

Preuve. En effet, l'algèbre triviale qui associe à chaque terme sa sorte ne laisse aucun choix pour interpréter les opérations. C'est une algèbre finale au sens où pour toute algèbre $\mathcal{A} \in |Alg(\Sigma)|$, il existe un unique Σ-homomorphisme de $\mathcal{A}$ dans $\mathcal{T}riv$. □

1.2.3 Relation de satisfaction

Dans la section précédente, nous avons donné un sens mathématique aux signatures, ici nous allons donner un sens mathématique aux termes et aux formules.

Définition 1.17 (Interprétation des termes) *Soient $\Sigma = (S, F)$ une signature, V un ensemble de variables sur Σ et $\mathcal{A}$ une algèbre de Σ.*

*Une **interprétation des variables** dans $\mathcal{A}$ est une application $\nu : V \to A$ telle que pour tout $s \in S$, pour tout $x \in V_s$, $\nu(x) \in A_s$.*

*L'**interprétation des termes** $\overline{\nu} : T_\Sigma(V) \to A$ se définit naturellement par un prolongement de ν aux termes de la manière suivante :*

- *si t est une variable $x \in V$, alors $\overline{\nu}(t) = \nu(x)$;*
- *si t est une constante $f \in F_{\varepsilon,s}$, alors $\overline{\nu}(t) = f_{\mathcal{A}}$;*
- *si t est de la forme $f(t_1, \ldots, t_n)$, alors $\overline{\nu}(t) = f_{\mathcal{A}}(\overline{\nu}(t_1), \ldots, \overline{\nu}(t_n))$.*

Dans la suite, on notera aussi ν le prolongement $\overline{\nu}$ de l'interprétation des variables aux termes.

À partir d'une interprétation des variables, on définit simplement la satisfaction des formules par induction sur la structure des formules.

Définition 1.18 (Relation de satisfaction) *Soient $\Sigma = (S, F)$ une signature, V un ensemble de variables sur Σ, et $\mathcal{A}$ une algèbre de Σ. Soient $\nu : V \to A$ une interprétation des variables et soit $\varphi \in For_{EL}(\Sigma)$.*

*La **satisfaction** de φ par l'algèbre $\mathcal{A}$ pour l'interprétation ν, notée $\mathcal{A} \models^{\nu}_{\Sigma} \varphi$, est définie inductivement sur la structure de φ de la manière suivante :*

- *si φ est de la forme $t = t'$, alors $\mathcal{A} \models^{\nu}_{\Sigma} \varphi$ ssi $\nu(t) = \nu(t')$;*
- *si φ est de la forme $\neg\psi$, alors $\mathcal{A} \models^{\nu}_{\Sigma} \varphi$ ssi $\mathcal{A} \not\models^{\nu}_{\Sigma} \psi$;*
- *si φ est de la forme $\psi \wedge \varrho$, alors $\mathcal{A} \models^{\nu}_{\Sigma} \varphi$ ssi $\mathcal{A} \models^{\nu}_{\Sigma} \psi$ et $\mathcal{A} \models^{\nu}_{\Sigma} \varrho$;*
- *si φ est de la forme $\psi \vee \varrho$, alors $\mathcal{A} \models^{\nu}_{\Sigma} \varphi$ ssi $\mathcal{A} \models^{\nu}_{\Sigma} \psi$ ou $\mathcal{A} \models^{\nu}_{\Sigma} \varrho$;*
- *si φ est de la forme $\psi \Rightarrow \varrho$, alors $\mathcal{A} \models^{\nu}_{\Sigma} \varphi$ ssi, si $\mathcal{A} \models^{\nu}_{\Sigma} \psi$ alors $\mathcal{A} \models^{\nu}_{\Sigma} \varrho$;*
- *si φ est de la forme $\forall x\psi$, alors $\mathcal{A} \models^{\nu}_{\Sigma} \varphi$ ssi pour toute interprétation $\nu' : V \to A$ qui vérifie $\nu(y) = \nu'(y)$ pour tout $y \in V \setminus \{x\}$, $\mathcal{A} \models^{\nu'}_{\Sigma} \psi$;*
- *si φ est de la forme $\exists x\psi$, alors $\mathcal{A} \models^{\nu}_{\Sigma} \varphi$ ssi il existe une interprétation qui vérifie $\nu(y) = \nu'(y)$ pour tout $y \in V \setminus \{x\}$, telle que $\mathcal{A} \models^{\nu'}_{\Sigma} \psi$.*

[9] Voir la définition A.4 de l'annexe A pour la notion de sous-catégorie pleine.

$\mathcal{A}$ valide φ, noté $\mathcal{A} \models_\Sigma \varphi$, est défini par : pour tout $v : V \to A$, $\mathcal{A} \models^v_\Sigma \varphi$.

*La relation $\models_\Sigma$ sur $|Alg(\Sigma)| \times For_{EL}(\Sigma)$ ainsi définie est appelée **relation de satisfaction**, et on note $\models_{EL}$ la famille $(\models_\Sigma)_{\Sigma \in |SIG_{EL}|}$ des relations de satisfaction.*

Soient $\mathbb{A}$ un ensemble de Σ-algèbres et $\varphi \in For_{EL}(\Sigma)$. Si pour tout $\mathcal{A} \in \mathbb{A}$, $\mathcal{A} \models_\Sigma \varphi$, alors on note $\mathbb{A} \models_\Sigma \varphi$ cette relation de satisfaction.

Pour une signature donnée, il est possible de distinguer les algèbres de cette signature au travers des formules qu'ils valident. Deux algèbres de même signature qui valident exactement les mêmes formules sont alors dites *élémentairement équivalentes*.

Définition 1.19 (Équivalence élémentaire d'algèbres) *Soit Σ une signature. Soient $\mathcal{A}$ et $\mathcal{A}'$ deux Σ-algèbres. On dit que $\mathcal{A}$ et $\mathcal{A}'$ sont **élémentairement équivalentes**, noté $\mathcal{A} \approx \mathcal{A}'$ si et seulement si elles valident exactement les mêmes formules de $For_{EL}(\Sigma)$:*

$$\forall \varphi \in For_{EL}(\Sigma), \mathcal{A} \models_\Sigma \varphi \Leftrightarrow \mathcal{A}' \models_\Sigma \varphi$$

2 Spécifications algébriques

Comme il est usuel de le faire en mathématiques, les structures de données étant des structures algébriques, nous spécifierons leur comportement par un système axiomatique dans la logique équationnelle. On appellera de tels systèmes axiomatiques, *des spécifications algébriques*.

Définition 1.20 (Spécification algébrique) *Une **spécification algébrique** Sp est un couple (Σ, Ax) où Σ est une signature et $Ax \subseteq For_{EL}(\Sigma)$ est un ensemble de formules appelées **axiomes**. Une spécification algébrique est dite **conditionnelle positive** si tous ses axiomes sont des formules conditionnelles positives.*

Exemple 1.6 *On peut donner comme exemple de spécification pour le PGCD de deux entiers le couple $(\Sigma_{pgcd}, Ax_{pgcd})$, où :*

$$\begin{aligned} Ax_{pgcd} = \{ & mod(n, pgcd(n,m)) = zero\,, \\ & mod(m, pgcd(n,m)) = zero\,, \\ & mod(n,p) = zero \wedge mod(m,p) = zero \Rightarrow \\ & p = pgcd(n,m) \vee p < pgcd(n,m) = vrai \} \end{aligned}$$

Exemple 1.7 *À la signature des listes donnée à l'exemple 1.2, on peut associer l'ensemble des axiomes caractérisant les opérations sur cette signature :*

$$\begin{aligned} Ax_{liste} = \{ & vide(nil) = vrai\,, \\ & vide(cons(x,l)) = faux\,, \\ & tête(cons(x,l)) = x\,, \\ & queue(cons(x,l)) = l\,, \\ & cons(x,l) = ll \Rightarrow \neg(ll = nil)\,, \\ & cons(x,l) = cons(y,ll) \Rightarrow x = y \wedge l = ll\,, \\ & appartient(x,nil) = faux\,, \\ & \neg(x = y) \Rightarrow appartient(x, cons(y,l)) = appartient(x,l)\,, \\ & appartient(x, cons(x,l)) = vrai \} \end{aligned}$$

Ainsi, la paire $(\Sigma_{liste}, Ax_{liste})$ définit une spécification des listes.

Les axiomes décrivent donc le comportement attendu du système. Les algèbres représentant des réalisations possibles, une spécification va donc servir au travers de la relation de satisfaction à sélectionner celles répondant aux attentes.

Définition 1.21 (Algèbres d'une spécification) *Soit $Sp = (\Sigma, Ax)$ une spécification. Une **algèbre de la spécification** Sp est une Σ-algèbre qui valide toutes les formules de Ax. On note $Alg(Sp)$ la sous-catégorie pleine de $Alg(\Sigma)$ dont les objets sont des algèbres de Sp appelées aussi des Sp-algèbres. On appelle Sp-*homomorphisme *un Σ-homomorphisme entre deux Sp-algèbres.*

On peut constater ainsi que selon le niveau d'abstraction, on peut avoir un grand nombre de réalisations possibles. D'une façon usuelle, on dit que la sémantique est lâche [26, 20].

Exemple 1.8 *Considérons la signature $\Sigma = (S, F)$ suivante :*

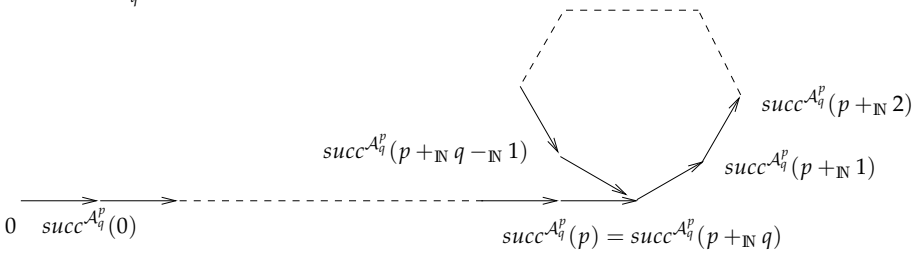
$$S = \{nat, bool\}$$
$$F = \{\, zero :\rightarrow nat\,,\ succ : nat \rightarrow nat\,,\ \_ + \_ : nat \times nat \rightarrow nat\,,$$

Définissons la spécification $Sp = (\Sigma, Ax)$, où :

$$Ax = \{\, n + zero = n\,,\ n + succ(m) = succ(n + m)\,\}$$

Sur la spécification Sp, nous pouvons associer toutes les algèbres de $Alg(Sp)$ finiment engendrées suivantes :

- *l'algèbre $\mathcal{A}$ définie par l'ensemble $A_{nat} = \mathbb{N}$ muni des applications : $zero_{\mathcal{A}} = 0$, $+_{\mathcal{A}} = +_{\mathbb{N}}$ et $succ_{\mathcal{A}} : \mathbb{N} \rightarrow \mathbb{N}$ définie par : $\forall n \in \mathbb{N}$, $Succ_{\mathcal{A}}(n) = n +_{\mathbb{N}} 1$.*
- *les algèbres $\mathcal{A}_q^p$ avec $p, q \in \mathbb{N}$ définies par les ensembles $(A_q^p)_{nat}$ donné par les intervalles $[0, p + q[$, munis des applications :*
  $0_{\mathcal{A}_q^p} = 0$, $succ_{\mathcal{A}_q^p}$ est définie par :



et $+_{\mathcal{A}_q^p}$ est définie par :

$$\forall a, b \in (A_q^p)_{nat},\ a +_{\mathcal{A}_q^p} b = \left[\begin{array}{ll} a +_{\mathbb{N}} b & si\ a +_{\mathbb{N}} b <_{\mathbb{N}} p +_{\mathbb{N}} q \\ p +_{\mathbb{N}} mod_{\mathbb{N}}((a +_{\mathbb{N}} b -_{\mathbb{N}} p), q) & sinon \end{array}\right]$$

avec $mod_{\mathbb{N}} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ définie par :

$$\forall n, m \in \mathbb{N}, mod_{\mathbb{N}}(n, m) = \left\{\begin{array}{l} m -_{\mathbb{N}} n\ si\ (n -_{\mathbb{N}} m <_{\mathbb{N}} m) = 1 \\ mod_{\mathbb{N}}(n -_{\mathbb{N}} m, m)\ sinon \end{array}\right.$$

Ainsi, nous constatons que pour un exemple basique de signature et d'axiomes, nous pouvons déjà définir une grande classe de modèles.

Les axiomes décrits dans les spécifications définissent les comportements élémentaires des systèmes à étudier. À partir de ces axiomes,

comme il est usuel dans les approches axiomatiques, d'autres comportements peuvent émerger. On parle de *conséquences sémantiques*. C'est au travers de ces dernières que l'on étudie les spécifications et que nous pouvons les comparer entre elles. Pour ce qui nous concerne, c'est à partir de ces dernières que nous définirons notre notion de complexité des systèmes (voir la section 3.2 du chapitre 2 et le chapitre 4 de ce document).

Définition 1.22 (Conséquences sémantiques d'une spécification) *Soit $Sp = (\Sigma, Ax)$ une spécification. L'**ensemble des conséquences sémantiques** de Sp, noté $Sp^\bullet$, est l'ensemble des formules validées par toutes les Sp-algèbres :*

$$Sp^\bullet = \{\varphi \mid |Alg(Sp)| \models_\Sigma \varphi\}$$

Exemple 1.9 *En considérant une nouvelle spécification de l'arithmétique $Sp'_{nat} = (\Sigma'_{nat}, Ax'_{nat})$ où $\Sigma'_{nat} = (S'_{nat}, F'_{nat})$ est la signature suivante :*

$$\begin{aligned} S'_{nat} = & \{nat\} \\ F'_{nat} = & \{ \_ + \_ : nat \times nat \to nat , \\ & \quad \_ \times \_ : nat \times nat \to nat , \\ & \quad \_^2 : nat \to nat \} \end{aligned}$$

et l'ensemble des axiomes Ax'_{nat} est le suivant :

$$\begin{aligned} Ax'_{nat} = & \{ x + y = y + x , \\ & \quad x \times y = y \times x , \\ & \quad x + (y + z) = (x + y) + z , \\ & \quad x \times (y \times z) = (x \times y) \times z , \\ & \quad x \times (y + z) = (x \times y) + (x \times z) , \\ & \quad 2 \times x = x + x , \\ & \quad x^2 = x \times x \} \end{aligned}$$

Alors, on peut montrer à partir de la spécification ci-dessus que l'identité remarquable suivante :

$$(x + y)^2 = x^2 + 2 \times (x \times y) + y^2$$

est une conséquence sémantique de Sp.

Preuve. *Soit $\mathcal{A} \in Alg(Sp'_{nat})$. Montrons que*

$$Alg(Sp'_{nat}) \models_{\Sigma'_{nat}} (x + y)^2 = x^2 + 2(x \times y) + y^2$$

Soit ν une interprétation des variables qui à x associe n et à y associe m, n et m étant deux valeurs de type nat. Montrons alors que

$$\mathcal{A} \models^{\nu}_{\Sigma'_{nat}} (x + y)^2 = x^2 + 2(x \times y) + y^2$$

$\mathcal{A} \in Alg(Sp'_{nat})$, donc les égalités suivantes sont vérifiées :

$$\begin{aligned} (n +^{\mathcal{A}} m)^2 &= (n +^{\mathcal{A}} m) \times^{\mathcal{A}} (n +^{\mathcal{A}} m) \\ &= ((n +^{\mathcal{A}} m) \times^{\mathcal{A}} n) +^{\mathcal{A}} ((n +^{\mathcal{A}} m) \times^{\mathcal{A}} m) \\ &= (n \times^{\mathcal{A}} (n +^{\mathcal{A}} m)) +^{\mathcal{A}} (m \times^{\mathcal{A}} (n +^{\mathcal{A}} m)) \\ &= (n \times^{\mathcal{A}} n +^{\mathcal{A}} n \times^{\mathcal{A}} m) +^{\mathcal{A}} (m \times^{\mathcal{A}} n +^{\mathcal{A}} m \times^{\mathcal{A}} m) \\ &= n \times^{\mathcal{A}} n +^{\mathcal{A}} (n \times^{\mathcal{A}} m +^{\mathcal{A}} m \times^{\mathcal{A}} n) +^{\mathcal{A}} m \times^{\mathcal{A}} m \\ &= n^2 +^{\mathcal{A}} (n \times^{\mathcal{A}} m +^{\mathcal{A}} m \times^{\mathcal{A}} n) +^{\mathcal{A}} m^2 \\ &= n^2 +^{\mathcal{A}} (n \times^{\mathcal{A}} m +^{\mathcal{A}} n \times^{\mathcal{A}} m) +^{\mathcal{A}} m^2 \\ &= n^2 +^{\mathcal{A}} 2(n \times^{\mathcal{A}} m) +^{\mathcal{A}} m^2 \end{aligned}$$

Ainsi, $\mathcal{A} \models^{\nu}_{\Sigma'_{nat}} (x + y)^2 = x^2 + 2(x \times y) + y^2$.

□

3 Conclusion

Dans ce chapitre nous avons présenté le formalisme de spécifications algébriques qui est un formalisme approprié pour spécifier un grand nombre de structures de données usuelles utilisées en informatique. Plusieurs langages fondés sur les spécifications algébriques tels que CLEAR [32], LARCH [69] et la famille OBJ [52, 60] ont été proposés. Puisque la famille des spécifications algébriques est grande, le projet CoFI (Common Framework Initiative for algebraic specification and development) a été proposé [112] pour présenter un cadre unifié pour l'ensemble des spécifications algébriques. Ce projet a amené à développer le langage CASL (Common Algebraic Specification Language) a été développé [27, 17].

Pour aider à l'écriture des spécifications, ces langages de spécification sont munis de primitives de structuration qui permettent de construire de façon incrémentale des spécifications à partir de spécifications de base, tels que les entiers et les listes, et qui sont définies dans des librairies de spécifications.

Historiquement, les premières spécifications algébriques ont été utilisées comme logique sous-jacente la logique des prédicats du premier ordre typé. Néanmoins, les premiers travaux portant sur l'étude de la modularité au travers des spécifications axiomatiques ont constaté qu'elle était trop expressive. Dans le prochain chapitre, nous nous focaliserons sur la logique équationnelle, et particulièrement la logique conditionnelle positive, qui a donné des résultats fondamentaux concernant la modularité des spécifications algébriques.

Structuration des spécifications algébriques

2

Sommaire

Pour pouvoir être utilisées dans la conception de programmes de taille réelle, les spécifications algébriques ont été munies de primitives de structuration. Comme nous le verrons dans ce chapitre, c'est sur ces primitives de structuration que nous illustrerons le concept de modularité et de complexité d'une spécification que nous généraliserons dans la suite de ce document.

Ce chapitre se découpe de la manière suivante : en section 2, nous présentons les primitives de structuration des spécifications algébriques de Wirsing [132]. En section 3, nous proposons une structuration des spécifications algébriques qui est largement inspirée de ce jeu de primitives. Nous rappelons en section 3.1 la notion de modularité classique, une modularité orientée modèles, que nous avons adaptée selon notre définition de la structuration des spécifications algébriques et nous donnons les résultats de modularité qui s'y attachent. Enfin, en section 3.2 nous introduisons notre définition de la modularité, qui est une définition orientée propriétés et nous donnons les résultats fondamentaux liés à cette modularité.

1 Besoin de spécifications structurées

Durant le processus du développement d'un logiciel, l'écriture de spécifications se fait naturellement par la réutilisation de spécifications déjà existantes. En effet, supposons par exemple que l'on souhaite spécifier les listes triées d'entiers et que l'on dispose des spécifications $Sp_{bool} = (\Sigma_{bool}, Ax_{bool})$ et $Sp_{nat} = (\Sigma_{nat}, Ax_{nat})$ des booléens et des entiers qui sont définies de la manière suivante :

$Sp_{bool} =$

$S_{bool} = \{bool\}$

$F_{bool} = \{\, vrai :\rightarrow bool\,,$
$\quad faux :\rightarrow bool\,\}$

$Ax_{bool} = \{\, \forall b, b = vrai \vee b = faux\,,$
$\quad \neg(vrai = faux)\,\}$

$Sp_{nat} =$

$S_{nat} = \{nat, bool\}$

$F_{nat} = \{\, zero :\rightarrow nat\,,$
$\quad succ : nat \rightarrow nat\,,$
$\quad \_ + \_ : nat \times nat \rightarrow nat\,,$
$\quad \_ \leq \_ : nat \times nat \rightarrow bool\,\}$

$Ax_{nat} = \{\, n + zero = n\,,$
$\quad n + succ(m) = succ(n + m)\,,$
$\quad n + m = m + n\,,$
$\quad (n + m) + p = n + (m + p)\,,$
$\quad zero \leq zero = vrai\,,$
$\quad succ(n) \leq zero = faux\,,$
$\quad zero \leq succ(n) = vrai\,,$
$\quad n \leq m = succ(n) \leq succ(m)\,\}$

On peut tout d'abord commencer par unir les spécifications d'entiers et des booléens en réutilisant Sp_{bool} et Sp_{nat} et en répercutant l'opération d'union sur chacune des composantes, c'est-à-dire :

$Sp_{natbool} = Sp_{bool} \oplus Sp_{nat}$, où :
- $Sp_{natbool} = ((S_{natbool}, F_{natbool}), Ax_{natbool})$,
- $S_{natbool} = S_{nat} \cup S_{bool}$,
- $F_{natbool} = F_{nat} \cup F_{bool}$,
- $Ax_{natbool} = Ax_{nat} \cup Ax_{bool}$.

Ensuite, on définit la spécification des listes d'entiers à partir de $Sp_{natbool}$ en enrichissant $Sp_{natbool}$:

$Sp_{liste} = Sp_{natbool} \oplus (S_{liste}, F_{liste}, Ax_{liste})$, où :

$S_{liste} = \{liste\}$

$F_{liste} = \{\, nil :\rightarrow liste\,,$
$\quad vide : liste \rightarrow bool\,,$
$\quad cons : nat \times liste \rightarrow liste\,,$
$\quad appartient : nat \times liste \rightarrow bool\,,$
$\quad tête : liste \rightarrow nat\,,$
$\quad queue : liste \rightarrow liste\,\}$

$$
\begin{aligned}
Ax_{liste} = \{\ & vide(nil) = vrai\,,\\
& vide(cons(x,l)) = faux\,,\\
& tête(cons(x,l)) = x\,,\\
& queue(cons(x,l)) = l\,,\\
& cons(x,l) = ll \Rightarrow \neg(ll = nil)\,,\\
& cons(x,l) = cons(y,ll) \Rightarrow x = y \wedge l = ll\,,\\
& appartient(x,nil) = faux\,,\\
& \neg(x = y) \Rightarrow appartient(x,cons(y,l)) = appartient(x,l)\,,\\
& appartient(x,cons(x,l)) = vrai\ \}
\end{aligned}
$$

Remarquons que $(S_{liste}, F_{liste}, Ax_{liste})$ dénote « un module » d'enrichissement et diffère d'une spécification bien formée : par exemple l'opération $appartient : liste \rightarrow bool$ est de sorte $bool$, mais $bool$ n'appartient pas à S_{liste} mais plutôt à $S_{natbool}$.

Maintenant, si l'on veut tenir compte du besoin d'une opération de tri sur les listes d'entiers, on peut enrichir la spécification Sp_{liste} pour obtenir une spécification des listes triées :

$Sp_{tliste} = Sp_{liste} \oplus (\emptyset, F_{tliste}, Ax_{tliste})$, où :

$$
\begin{aligned}
F_{tliste} = \{\ & trier : liste \rightarrow liste\,,\\
& trié\text{: } liste \rightarrow bool\ \}
\end{aligned}
$$

$$
\begin{aligned}
Ax_{tliste} = \{\ & trié(nil) = vrai\,,\\
& \forall x, (trié(cons(x,l)) = vrai \Rightarrow (\forall y, (appartient(y,l) = vrai \Rightarrow\\
& x \leq y = vrai) \wedge tri(l) = vrai))\,,\\
& \forall x, ((\forall y, (appartient(y,l) = vrai \Rightarrow\\
& x \leq y = vrai) \wedge tri(l) = vrai) \Rightarrow trié(cons(x,l)) = vrai)\,,\\
& trié(trier(l)) = vrai\,,\\
& appartient(x,l) = appartient(x,trier(l))\ \}
\end{aligned}
$$

Ainsi, pour spécifier les listes triées d'entiers, on n'a pas spécifié les entiers et les booléens de nouveau, mais plutôt réutilisé des spécifications déjà existantes : l'opérateur $\oplus$ dénote la relation «est réutilisable pour» entre une spécification à écrire et les spécifications réutilisables.

La réutilisation de spécification a fait l'objet de nombreuses recherches, essentiellement autour de la problématique de l'aide à l'écriture de spécifications de manière incrémentale, par exemple par enrichissement ou par union. Ce processus d'écriture des spécifications a abouti aux *spécifications structurées*. Cette structuration est caractérisée par des morceaux de spécifications nommés, reliés entre eux par des *primitives de structurations*.

2 Structuration par jeu de primitives

De nombreux travaux ont porté sur l'étude des spécifications algébriques structurées [100, 26, 71, 132, 29] dans le but de pouvoir spécifier de manière incrémentale les systèmes de grande taille à partir de spécifications déjà existantes. Certains de ces travaux [71, 132, 29] ont abouti à la définition précise de quelques jeux de primitives de structuration qui aident à l'écriture de spécifications par la réutilisation des spécifications plus élémentaires. Ces jeux de primitives comportent essentiellement les primitives suivantes :

- l'*enrichissement* qui consiste en la possibilité d'ajouter de nouveaux types, opérations et/ou axiomes à une spécification existante. L'enrichissement peut être illustré par exemple par la spécification des listes triées d'entiers par l'ajout des opérations *trier* et *tri* et de l'ensemble d'axiomes Ax_{tliste} au dessus de la spécification des listes d'entiers Sp_{liste},
- le *renommage* qui consiste à renommer les types et les opérations de la signature d'une spécification et qui est caractérisé par un morphisme de signatures bijectif,
- le *masquage* de types et d'opérations, appelé aussi *restriction* ou *encapsulation*, qui consiste à rendre impossible leur invocation hors de la spécification en question. Il peut être illustré par la spécification des listes d'entiers par la réutilisation des listes triées d'entiers. Le masquage est caractérisé par un morphisme d'inclusion de la signature de la spécification résultante du masquage dans celle de la spécification sur laquelle on applique le masquage,
- l'*union* qui consiste à combiner deux spécifications et qui peut être illustrée par exemple par la spécification des entiers et des booléens par la réutilisation de leurs spécifications respectives.

Dans la suite, nous reprenons le jeu de primitives de Wirsing [132, 72] qui nous a inspirés pour introduire notre définition de la structuration des spécifications algébriques (cf. définition 2.8).

Les primitives de Wirsing sont définies dans un cadre où les morphismes de signatures sont restreints aux morphismes d'inclusion. Les spécifications structurées sont obtenues à partir de spécifications dites *plates* et à partir de l'application de trois primitives basiques : l'*union* de deux spécifications, le *renommage* d'une spécification et la *restriction* d'une spécification à une partie de sa signature.

Chaque spécification plate est caractérisée par une *syntaxe* donnée par une signature et par un ensemble d'axiomes sur cette signature et par une *sémantique* donnant la sous-catégorie pleine d'algèbres validant les axiomes.

Définition 2.1 (Spécification plate) *Une spécification plate Sp est donnée par la syntaxe et la sémantique suivantes :*

- ***La syntaxe*** *est donnée par une spécification algébrique (Σ, Ax) (cf. définition 1.20).*
- ***La sémantique*** *est donnée par :*
  - $Sig(Sp) = \Sigma$,
  - $Mod(Sp) = Alg((\Sigma, Ax))$.

Notons que les signatures font partie de la sémantique, ce qui est cohérent dans le sens où la définition des modèles leur est intrinsèquement liée, plus explicitement une classe d'algèbres peut être représentée par le couple (signature,classe d'algèbres).

Pour construire des spécifications structurées à partir des spécifications plates comme briques de base, les primitives de structuration sont utilisées. Tout comme les spécifications plates, on associe à chaque primitive une *syntaxe* et une *sémantique*. La sémantique fournit la signature

Σ de la spécification résultante de l'application de la primitive en question et fournit aussi une sous-catégorie pleine de $Alg(\Sigma)$ dont les objets représentent les *réalisations* de la spécification résultante.

Dans la suite, on va introduite la syntaxe et la sémantique de ces primitives qui utilisent les trois types *Spec*, *Signature* et *renommage*. Les constantes de type *Spec*, *signature* et *renommage* sont respectivement les spécifications plates, les signatures de SIG_{EL} et les morphismes de signatures bijectifs.

Définition 2.2 (Syntaxe des primitives de structuration basiques) *La syntaxe des primitives de structuration est définie de la façon suivante :*

(1) union : $\_ + \_ : Spec \times Spec \rightarrow Spec$

(2) restriction : $\_|\_ : Spec \times Signature \rightarrow Spec$

(3) renommage : $renommer\_par\_ : Spec \times Renommage \rightarrow Spec$

Définition 2.3 (Spécification structurée) *Une* ***spécification structurée*** *est un terme de type Spec dans la syntaxe donnée par la définition 2.2.*

On interprète les termes (les spécifications structurées). À chaque terme Sp, on associe une signature notée $Sig(Sp)$ et une catégorie de modèles notée $Mod(Sp)$. La sémantique est définie comme suit :

Définition 2.4 (Sémantique de l'union) *La sémantique de l'union est donnée par :*

- $Sig(Sp_1 + Sp_2) = Sig(Sp_1) \cup Sig(Sp_2)$
- $Mod(Sp_1 + Sp_2)$, *la sous-catégorie pleine de* $Alg(Sig(Sp_1 + Sp_2))$, *telle que* $|Mod(Sp_1 + Sp_2)| = \{\mathcal{A} \in |Alg(Sig(Sp_1 + Sp_2))| \mid \mathcal{A}_{|Sig(Sp_1)} \in |Mod(Sp_1)| \text{ et } \mathcal{A}_{|Sig(Sp_2)} \in |Mod(Sp_2)|\}$

Définition 2.5 (Sémantique de la restriction) *Soit* Σ *une signature telle que* $\Sigma \subseteq Sig(Sp)$. *La sémantique de la restriction est donnée par :*

- $Sig(Sp_{|\Sigma}) = \Sigma$
- $Mod(Sp_{|\Sigma})$, *la sous-catégorie pleine de* $Alg(\Sigma)$, *telle que* $|Mod(Sp_{|\Sigma})| = \{\mathcal{A}_{|\Sigma} \in |Mod(\Sigma)| \mid \mathcal{A} \in |Mod(Sp)|\}$

Définition 2.6 (Sémantique de renommage) *Soit* $\sigma : Sig(Sp) \rightarrow \Sigma$ *un morphisme de signature bijectif. La sémantique du renommage est donnée par :*

- $Sig(renommer\ Sp\ par\ \sigma) = \Sigma$
- $Mod(renommer\ Sp\ par\ \sigma)$, *la sous-catégorie pleine de* $Alg(\Sigma)$, *telle que* $|Mod(renommer\ Sp\ par\ \sigma)| = \{\mathcal{A} \in |Alg(\Sigma)| \mid \mathcal{A}_{|\sigma} \in |Mod(Sp)|\}$

Pour une spécification structurée Sp donnée, on appelle les objets de $Mod(Sp)$ des réalisations de Sp ou encore des Sp-algèbres. On appelle *Sp-homomorphisme* un Σ-homomorphisme entre deux Sp-algèbres.

Pour exprimer l'enrichissement, Wirsing propose tout simplement d'utiliser la primitive de l'union comme l'indique la définition suivante :

Définition 2.7 (Primitive d'enrichissement) *Soit* Sp *une spécification structurée et soit* $\Delta = (S, F, \Delta Ax)$ *tels que :*

$$(S', F') = Sig(Sp) \text{ et } \Delta Ax \subseteq For_{EL}((S' \cup S, F' \cup F))$$

La primitive d'enrichissement d'une spécification Sp *par* $\Delta = (S, F, \Delta Ax)$ *est*

définie de la façon suivante :

$$enrich_{\Delta}(Sp) = Sp + ((S_{Sig(Sp)} \cup S, F_{Sig(Sp)} \cup F), \Delta Ax)$$

L'ensemble de ces primitives ne constitue pas la liste exhaustive des primitives de structuration des spécifications algébriques. Il en existe d'autres ou des variantes dans la littérature [71, 29]. Par exemple dans [29], Borzyszkowski propose, avant d'unir deux spécifications, de les ramener à une même signature. Les divers choix sont motivés par des raisons techniques ou intuitives. Toutefois, les différents jeux couvrent à peu près le même champ de structuration et on peut généralement retrouver une primitive d'un jeu en combinant d'autres primitives comme nous l'avons vu par exemple pour le cas de l'enrichissement (cf. définition 2.7).

3 Modularité des spécifications algébriques

Bien que les primitives de structuration permettent simplement et naturellement d'augmenter (par enrichissement ou par union) ou de réduire (par restriction) la taille des spécifications, elles ne permettent pas de garantir gratuitement la réutilisation des réalisations relatives aux spécifications existantes. Pour cet objectif, des études ont été menées et des résultats ont été établis, plus précisément dans le cadre des spécifications structurées caractérisées par l'enrichissement d'une spécification plate (Σ, Ax) par une spécification conditionnelle positive plate (Σ', Ax'), où Σ est incluse dans Σ' et $Ax \subseteq Ax'$. Ce résultat classique reflète l'idée suivante : toute réalisation de la spécification « enrichie » fournit de façon canonique une réalisation de la spécification « pauvre » et toute réalisation de la spécification pauvre peut s'étendre en une réalisation de la spécification enrichie. Dans la littérature ce résultat est connu sous le nom de la modularité des spécifications structurées. Dans la suite, on appelle cette modularité une *modularité orientée modèles* puisqu'elle garantit la préservation des modèles (réalisations) au travers de la structuration des spécifications algébriques.

Rappelons que les spécifications structurées que nous avons vues en section 2 sont construites inductivement à partir de primitives de structuration. De ce fait, leur forme ne permet pas de réutiliser le résultat de la modularité orientée modèles des spécifications algébriques plates. Pour retrouver ce résultat classique, nous sommes amenés à ramener les spécifications structurées à des spécifications plates. Pour ce faire, nous allons exploiter un résultat établi dans [132] qui consiste à ramener toute spécification structurée à une forme quasi-plate, appelée *forme normale*.

Théorème 2.1 *Pour toute spécification structurée Sp, il existe une spécification, notée $nf(Sp)$, de la forme $(\Sigma', Ax')_{|Sig(Sp)}$, appelée **forme normale** de Sp, où Σ' est une signature et Ax' un ensemble de Σ'-formules, tel que $Mod(Sp) = Mod(nf(Sp))$.*

Ce résultat est donné conjointement avec un système d'équations entre les spécifications qui permet d'obtenir cette forme normale. Ce système d'équations est donné en détail dans [132].

L'intérêt de ce théorème est qu'il permet d'associer à toute spécification structurée, par enrichissement ou par union, une spécification plate, et

donc de retrouver le résultat classique de la modularité orientée modèles des spécifications structurées par enrichissement.

Définition 2.8 (Enrichissement, union) *Soient Sp, Sp_1 et Sp_2 des spécifications structurées et soit $Sp' = (\Sigma', Ax')$ une spécification plate.*

- *Sp' est un **enrichissement** de Sp si et seulement s'il existe une spécification Sp'' et $\Delta = (S, F, \Delta Ax)$ tels que $Sp'' = enrich_\Delta(Sp)$ et $nf(Sp'') = Sp'_{|Sig(Sp'')}$.*
- *Sp' est une **union** de Sp_1 et Sp_2 si et seulement s'il existe une spécification Sp'' telle que $Sp'' = Sp_1 + Sp_2$ et $nf(Sp'') = Sp'_{|Sig(Sp'')}$.*

Remarque 2.1 *Soient Sp, Sp_1, Sp_2 des spécifications structurées et soit $Sp' = (\Sigma', Ax')$ une spécification plate.*

- *Si Sp' est un enrichissement de Sp, alors il existe un morphisme d'inclusion $Sig(Sp) \hookrightarrow \Sigma'$, et*

$$\forall \mathcal{A}' \in |Mod(Sp')|, \mathcal{A}'_{|\Sigma} \in |Mod(Sp)|$$

- *Si Sp' est l'union de Sp_1 et Sp_2, alors il existe deux morphismes d'inclusion $Sig(Sp_1) \hookrightarrow \Sigma'$ et $Sig(Sp_2) \hookrightarrow \Sigma'$, et*

$$\forall \mathcal{A}' \in |Mod(Sp')|, \mathcal{A}'_{|Sig(Sp_1)} \in |Mod(Sp_1)| \text{ et } \mathcal{A}'_{|Sig(Sp_2)} \in |Mod(Sp_2)|$$

*Les spécifications Sp, Sp_1, Sp_2 sont appelées **des spécifications enrichies**.*

3.1 Modularité orientée modèles

Dans cette section, nous allons étudier la modularité des spécifications structurées par enrichissement. Étant données deux spécifications structurées Sp et Sp' telles que Sp' est un enrichissement de Sp, la modularité orientée modèles impose les deux points suivants :

(1) Toute réalisation de Sp' fournit une réalisation de Sp.

(2) Toute réalisation de Sp peut s'étendre à une réalisation de Sp'.

Le premier point est assuré par définition même de l'enrichissement. En effet d'après la remarque 2.1, on peut déduire que

$$|Mod(Sp')|_{|\Sigma} \subseteq |Mod(Sp)|, \text{ avec } |Mod(Sp')|_{|\Sigma} = \{\mathcal{A}'_{|\Sigma} \in Mod(Sp) | \mathcal{A}' \in Mod(Sp')\}$$

Dans le reste de la section, nous allons étudier des conditions suffisantes fondées sur la notion d'adjonction catégorielle permettant d'obtenir le deuxième point. Ces conditions seront obtenues dans le cadre restreint des spécifications conditionnelles positives.

Les conditions que nous allons présenter dans cette section reposent sur un résultat classique dit de *minimalité* [64], qui pour toute spécification conditionnelle positive montre qu'il existe un quotient minimal de toute algèbre permettant de valider l'ensemble des axiomes de la spécification. Rappelons ce résultat.

3.1.1 Congruence

Rappelons tout d'abord que pour tout ensemble E donné, une relation d'équivalence sur E est une relation binaire $\equiv \in E \times E$ qui est réflexive, symétrique et transitive. La relation $\equiv$ partitionne E en un ensemble disjoint de classes d'équivalence $\lfloor e \rfloor = \{c \in E \mid e \equiv c\}$. Le quotient de E par la relation $\equiv$ donne l'ensemble de ses classes : $E_{/\equiv} = \{\lfloor e \rfloor \mid e \in E\}$. La notion de *congruence* étend la notion de relation d'équivalence aux algèbres en préservant l'application des opérations.

Définition 2.9 (Congruence) *Soient $\Sigma = (S, F)$ une signature et $\mathcal{A}$ une algèbre sur Σ. Soit $\equiv$ une relation d'équivalence sur A, c'est à dire une famille indexée par S de relations d'équivalence sur chaque A_s. Alors $\equiv$ est **une congruence** sur $\mathcal{A}$ si et seulement si pour tout $f : s_1 \times \cdots \times s_n \to s \in F$, pour tout $(a_1, \ldots, a_n) \in A_{s_1} \times \cdots \times A_{s_n}$, pour tout $(a'_1, \ldots, a'_n) \in A_{s_1} \times \cdots \times A_{s_n}$, on a :*

$$\forall i, 1 \leq i \leq n, a_i \equiv_{s_i} a'_i \Rightarrow f_{\mathcal{A}}(a_1, \ldots, a_n) \equiv_s f_{\mathcal{A}}(a'_1, \ldots, a'_n)$$

Exemple 2.1 *Soient $\Sigma = (S, F)$ une signature et $\mathcal{A} \in |Alg(\Sigma)|$ une algèbre de Σ. La relation binaire $\equiv_{Triv} = (\equiv_{Triv_s})_{s \in S}$ définie par :*

$$a \equiv_{Triv_s} a' \Leftrightarrow a \in A_s \wedge a' \in A_s$$

est une congruence sur $\mathcal{A}$.

Exemple 2.2 *Soient $\Sigma = (S, F)$ une signature. Soient $\mathcal{A}$ et $\mathcal{B}$ deux algèbres de Σ et soit $h : \mathcal{A} \to \mathcal{B}$ un Σ-homomorphisme. On peut prouver, en se référant aux définitions 1.14 et 2.9, que la relation binaire $\equiv_h$ sur A, définie par $a \equiv_h a'$ si et seulement si $h(a) = h(a')$, est une congruence sur $\mathcal{A}$.*

Étant données une algèbre $\mathcal{A}$ et une congruence $\equiv$ sur $\mathcal{A}$, il est possible de munir A quotienté par $\equiv$ d'une structure d'algèbre, appelé *algèbre quotient*, comme l'indique le théorème suivant.

Théorème 2.2 (Algèbre quotient) *Soient $\Sigma = (S, F)$ une signature et $\mathcal{A}$ une algèbre de Σ. Soit $\equiv$ une congruence sur $\mathcal{A}$. L'ensemble $A_{/\equiv}$ est canoniquement muni d'une structure d'algèbre de Σ définie de la façon suivante :*

- *$A_{/\equiv} = ((A_{/\equiv})_s)_{s \in S}$ où $(A_{/\equiv})_s = \{\lfloor a \rfloor | a \in A_s\}$ pour tout $s \in S$, et $\lfloor a \rfloor$ la classe de congruence définie par $\lfloor a \rfloor = \{a' \in A_s | a \equiv_s a'\}$.*
- *pour tout $f : s_1 \times \cdots \times s_n \to s \in F$, l'interprétation de f est l'application $f_{\mathcal{A}_{/\equiv}} : (A_{/\equiv})_{s_1} \times \cdots \times (A_{/\equiv})_{s_n} \to (A_{/\equiv})_s$ définie pour tout n-uplet $(\lfloor a_1 \rfloor, \ldots, \lfloor a_n \rfloor)$ par $f_{\mathcal{A}_{/\equiv}}(\lfloor a_1 \rfloor, \ldots, \lfloor a_n \rfloor) = \lfloor f_{\mathcal{A}}(a_1, \ldots, a_n) \rfloor$.*

et $h = (h^s : A_s \to (A_{/\equiv})_s)_{s \in S}$, telle que pour tout $s \in S$, et pour tout $a \in A_s$, $h^s(a) = \lfloor a \rfloor$, est un Σ-homomorphisme, appelé homomorphisme quotient de $\mathcal{A}$ sur $\mathcal{A}_{/\equiv}$.

Preuve. Soit $f : s_1 \times \cdots \times s_n \to s \in F$ et soit $(a_1, \ldots, a_n) \in A_{s_1} \times \cdots \times A_{s_n}$.

$$\begin{aligned} h^s(f_{\mathcal{A}}(a_1, \ldots, a_n)) &= \lfloor f_{\mathcal{A}}(a_1, \ldots, a_n) \rfloor \\ &= f_{\mathcal{A}_{/\equiv}}(\lfloor a_1 \rfloor, \ldots, \lfloor a_n \rfloor) \\ &= f_{\mathcal{A}_{/\equiv}}(h^{s_1}(a_1), \ldots, h^{s_n}(a_n)) \end{aligned}$$

d'où h est un Σ-homomorphisme. □

3.1.2 Minimalité

Comme nous l'avons expliqué au début de la section 3.1, le résultat suivant est au centre des conditions pour avoir la modularité orientée modèles des spécifications algébriques. En effet, ce résultat nous permettra de définir un foncteur adjoint à gauche au foncteur d'oubli sur lequel seront définies des conditions pour assurer le point (2) de la section 3.1.

Théorème 2.3 (Minimalité) *Soit $Sp = (\Sigma, Ax)$ une spécification conditionnelle positive. Soient $\mathcal{A}$ une algèbre sur Σ et $\mathcal{R}$ une relation binaire sur A, c'est à dire une famille indexée par S de relations binaires sur chaque A_s. Il existe une plus petite Sp-algèbre $\mathcal{B} \in |Alg(Sp)|$ (au sens du préordre induit par les homomorphismes) telle que :*

1. *Il existe un homomorphisme $h_{\mathcal{B}} : \mathcal{A} \to \mathcal{B}$;*
2. *$(\mathcal{B}, h_{\mathcal{B}})$ est compatible avec $\mathcal{R}$, c'est-à-dire*

$$\forall a, a' \in A, a\mathcal{R}a' \Rightarrow h_{\mathcal{B}}(a) = h_{\mathcal{B}}(a')$$

En plus, pour tout $\mathcal{C} \in |Alg(Sp)|$ satisfaisant les conditions (1) et (2), il existe un unique homomorphisme $\mu_{\mathcal{C}} : \mathcal{B} \to \mathcal{C}$ tel que $\mu_{\mathcal{C}} \circ h_{\mathcal{B}} = h_{\mathcal{C}}$.

Preuve. Soit $F = \{(\mathcal{C}, h_{\mathcal{C}}) \mid \mathcal{C} \in |Alg(Sp)| \text{ et } h_{\mathcal{C}} : \mathcal{A} \to \mathcal{C} \ tq \ \forall s \in S, \forall a, a' \in A_s, a\mathcal{R}_s a' \Rightarrow h_{\mathcal{C}}(a) = h_{\mathcal{C}}(a')\}$.

F n'est pas vide puisque $(\mathcal{T}riv, h_{\mathcal{T}riv}) \in F$, où $h_{\mathcal{T}riv}$ est l'unique homomorphisme de $\mathcal{A}$ dans $\mathcal{T}riv$ (d'après le théorème 1.2) défini pour tout $s \in S$, pour tout $a \in A_s$ par $h^s_{\mathcal{T}riv}(a) = s$.

Considérons la relation binaire $\equiv$ sur A définie par

$$\forall s \in S, \forall a, a' \in A_s, a \equiv_s a' \Leftrightarrow \forall (\mathcal{C}, h_{\mathcal{C}}) \in F, h^s_{\mathcal{C}}(a) = h^s_{\mathcal{C}}(a')$$

La relation $\equiv$ est une congruence sur $\mathcal{A}$. En effet, elle est l'intersection de toutes les congruences induites par chacun des morphismes $h_{\mathcal{C}}$, pour $(\mathcal{C}, h_{\mathcal{C}}) \in F$.

Appelons $\mathcal{B}$ le quotient de $\mathcal{A}$ par la congruence $\equiv$. D'après le Théorème 2.2, $h_{\mathcal{B}} : \mathcal{A} \to \mathcal{B}$ est un homomorphisme. De plus, pour tout $(\mathcal{C}, h_{\mathcal{C}}) \in F$, on peut considérer l'homomorphisme $\mu_{\mathcal{C}} : \mathcal{B} \to \mathcal{C}$ défini par $\mu_{\mathcal{C}} \circ h_{\mathcal{B}} = h_{\mathcal{C}}$. Cet homomorphisme est de plus unique par la surjectivité de $h_{\mathcal{B}}$. Par conséquent, si $(\mathcal{B}, h_{\mathcal{B}}) \in F$ alors il est nécessairement le plus petit élément de F. $(\mathcal{B}, h_{\mathcal{B}})$ vérifie bien les conditions (1) et (2) du théorème, donc pour montrer que $(\mathcal{B}, h_{\mathcal{B}}) \in F$ il suffit de montrer que $\mathcal{B} \in |Alg(Sp)|$.

Soit $t_1 = t'_1 \wedge \cdots \wedge t_n = t'_n \Rightarrow t = t'$ un axiome de Ax et soit $\nu_{\mathcal{B}} : V \to \mathcal{B}$ une interprétation telle que pour tout i, $1 \leq i \leq n$,

$$\nu_{\mathcal{B}}(t_i) = \nu_{\mathcal{B}}(t'_i)$$

Montrer que $\mathcal{B} \in |Alg(Sp)|$ revient à montrer que $\nu_{\mathcal{B}}(t) = \nu_{\mathcal{B}}(t')$.

Pour tout $(\mathcal{C}, h_{\mathcal{C}}) \in F$, considérons $\nu_{\mathcal{C}} : V \to \mathcal{C}$ l'interprétation définie par,

$$\nu_{\mathcal{C}} = \mu_{\mathcal{C}} \circ \nu_{\mathcal{B}}$$

Par définition, on a pour tout i, $1 \leq i \leq n$,

$$\nu_{\mathcal{C}}(t_i) = \nu_{\mathcal{C}}(t'_i)$$

et donc

$$\nu_{\mathcal{C}}(t) = \nu_{\mathcal{C}}(t')$$

c'est-à-dire

$$\mu_{\mathcal{C}}(\nu_{\mathcal{B}}(t)) = \mu_{\mathcal{C}}(\nu_{\mathcal{B}}(t'))$$

Soient, $a, a' \in A$ tels que,

$$\nu_{\mathcal{B}}(t) = h_{\mathcal{B}}(a) \text{ et } \nu_{\mathcal{B}}(t') = h_{\mathcal{B}}(a')$$

On a donc,

$$\mu_{\mathcal{C}}(h_{\mathcal{B}}(a)) = \mu_{\mathcal{C}}(h_{\mathcal{B}}(a'))$$

d'où nous déduisons,

$$h_{\mathcal{C}}(a) = h_{\mathcal{C}}(a')$$

Comme ceci est vrai pour tout $(\mathcal{C}, h_{\mathcal{C}}) \in F$, on peut conclure

$$h_{\mathcal{B}}(a) = h_{\mathcal{B}}(a')$$

c'est-à-dire,

$$\nu_{\mathcal{B}}(t) = \nu_{\mathcal{B}}(t')$$

□

Corollaire 2.1 *Soit $Sp = (\Sigma, Ax)$ une spécification conditionnelle positive. $Alg(Sp)$ et $Gen(Sp)$ admettent un objet initial, noté $\mathcal{T}_{Sp}$.*

Preuve. On choisit $\mathcal{A} = \mathcal{T}_\Sigma$ et $\mathcal{R}$ la relation vide dans le théorème 2.3. $\mathcal{T}_{Sp}$ est la plus petite Sp-algèbre qui valide Sp. Ce qui signifie que pour tout $\mathcal{C} \in |Alg(Sp)|$, il existe un unique homomorphisme $\mu_{\mathcal{C}} : \mathcal{T}_{Sp} \to \mathcal{C}$ tel que $h_{\mathcal{A}} = \mu_{\mathcal{A}} \circ h_{\mathcal{T}_{Sp}}$, avec $h_{\mathcal{T}_{Sp}} : \mathcal{T}_\Sigma \to \mathcal{T}_{Sp}$. Il reste à prouver que cet homomorphisme est unique. Ceci résulte du fait que $\mathcal{T}_\Sigma$ est initiale dans $Alg(\Sigma)$ et que $h_{\mathcal{A}} = \mu_{\mathcal{A}} \circ h_{\mathcal{T}_{Sp}}$.

Nous avons prouvé que $\mathcal{T}_{Sp}$ est initiale dans $Alg(Sp)$, il reste à prouver que $\mathcal{T}_{Sp}$ est initiale dans $Gen(Sp)$, c'est-à-dire que $\mathcal{T}_{Sp}$ est finiment engendrée. Ceci est immédiat puisque $\mathcal{T}_{Sp}$ est un quotient de $\mathcal{T}_\Sigma$. □

3.1.3 Adjonction

Étant données deux spécifications structurées Sp et Sp' telles que Sp' est un enrichissement de Sp au sens de la définition 2.8 et Sp' est de plus conditionnelle positive. On va alors montrer que nous pouvons construire un foncteur $F : Mod(Sp) \to Mod(Sp')$ adjoint à gauche au foncteur d'oubli $U_{|\Sigma}$. Cette adjonction nous assurera l'existence pour toute réalisation $\mathcal{A} \in |Mod(Sp)|$ d'un homomorphisme $I_{\mathcal{A}} : \mathcal{A} \to F(\mathcal{A})_{|\Sigma}$, appelé *morphisme d'adjonction*.

Quand ce morphisme d'adjonction est un isomorphisme[1], il nous assure alors la modularité orientée modèles. C'est sur cet isomorphisme que

[1] Voir la définition A.11 de l'annexe A pour la notion d'isomorphisme.

la modularité des spécifications algébriques a été définie à la fin des années 70 par Guttag et Horning [67]. En fait, ils ont décomposé cette propriété en deux sous propriétés, *la consistance hiérarchique* et *la suffisante complétude* selon que ce morphisme d'adjonction $I_{\mathcal{A}}$ soit un monomorphisme et un épimorphisme, respectivement.

Rappelons maintenant la définition de ce foncteur F, appelé *foncteur de synthèse*.

Théorème 2.4 (Foncteur de synthèse F) *Soit Sp une spécification structurée et soit $Sp' = (\Sigma', Ax')$ une spécification conditionnelle positive enrichissement de Sp au sens de la définition 2.8 et soit $\mathcal{A} \in |Mod(Sp)|$. Soit $\mathcal{R}$ la relation binaire sur $\mathcal{T}_{\Sigma'}(A)$ définie par :*

$$\forall a'_1, a'_2 \in T_{\Sigma'}(A), (a'_1 \mathcal{R} a'_2) \Leftrightarrow ((a'_1 \in (T_\Sigma(A)) \wedge (a'_2 \in T_\Sigma(A)) \wedge (ev_{\mathcal{A}}(a'_1) = ev_{\mathcal{A}}(a'_2)))$$

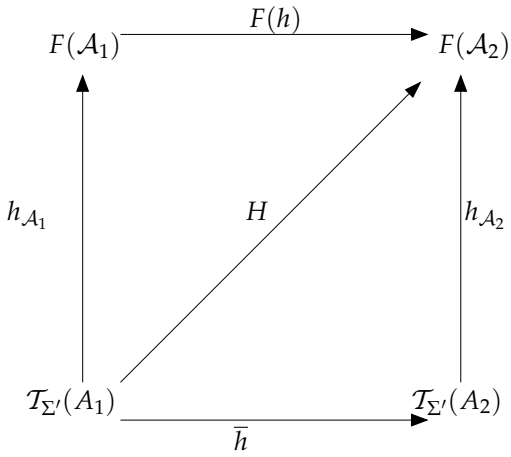
(cf. définition 1.12 pour la définition de $ev_{\mathcal{A}}$)

On définit $F : Mod(Sp) \rightarrow Mod(Sp')$ comme étant le foncteur :
- *qui associe à chaque Sp-algèbre $\mathcal{A} \in |Mod(Sp)|$ la plus petite Sp'-algèbre $F(\mathcal{A})$ tel que :*
  - *Il existe un homomorphisme $h_{\mathcal{A}} : \mathcal{T}_{\Sigma'}(A) \rightarrow F(\mathcal{A})$;*
  - *$(F(\mathcal{A}), h_{\mathcal{A}})$ est compatible avec $\mathcal{R}$, c'est-à-dire*

$$\forall a, a' \in T_{\Sigma'}(A), a \mathcal{R} a' \Rightarrow h_{\mathcal{A}}(a) = h_{\mathcal{A}}(a')$$

  (cette Sp'-algèbre existe et est unique d'après le théorème 2.3)
- *qui associe à chaque Sp-homomorphisme $h : \mathcal{A}_1 \rightarrow \mathcal{A}_2$ le Sp'-homomorphisme $F(h) : F(\mathcal{A}_1) \rightarrow F(\mathcal{A}_2)$ tel que $H = F(h) \circ h_{\mathcal{A}_1}$, où :*
  - *$H = h_{\mathcal{A}_2} \circ \overline{h}$,*
  - *$h_{\mathcal{A}_1}$ est l'homomorphisme quotient de $\mathcal{T}_{\Sigma'}(A_1)$ sur $F(\mathcal{A}_1)$,*
  - *$h_{\mathcal{A}_2}$ est l'homomorphisme quotient de $\mathcal{T}_{\Sigma'}(A_2)$ sur $F(\mathcal{A}_2)$,*
  - *$\overline{h} : \mathcal{T}_{\Sigma'}(A_1) \rightarrow \mathcal{T}_{\Sigma'}(A_2))$ est l'extension canonique de h.*

  $F(h)$ est l'unique Sp'-homomorphisme tel que le diagramme suivant commute.



F ainsi défini est un foncteur appelé ***foncteur de synthèse****.*

Preuve. Soit $\mathcal{A} \in |Mod(Sp)|$ une Sp-algèbre. $F(\mathcal{A})$ existe d'après le théorème 2.3.

Montrons que pour tout $h : \mathcal{A}_1 \rightarrow \mathcal{A}_2$, $F(h)$ existe et vérifie $H = F(h) \circ h_{\mathcal{A}_1}$. Pour tout $h : \mathcal{A}_1 \rightarrow \mathcal{A}_2$, $\overline{h}$ existe, et par suite $H = h_{\mathcal{A}_2} \circ \overline{h}$ existe. D'après le théorème 2.3, il existe un unique homomorphisme de h_F tel que $H = h_F \circ h_{\mathcal{A}_1}$. On prend $F(h) = h_F$.

Montrons maintenant que pour tout Sp-morphisme $h_1 : \mathcal{A}_2 \rightarrow \mathcal{A}_3$ et $h_2 : \mathcal{A}_2 \rightarrow \mathcal{A}_3$, $F(h' \circ h) = F(h') \circ F(h)$. Ceci résulte du fait que $\overline{h' \circ h} = \overline{h'} \circ \overline{h}$ et du fait que $h_F : F(\mathcal{A}_1) \rightarrow F(\mathcal{A}_3)$ est unique, et qui est par définition égal à $F(h' \circ h)$. □

Théorème 2.5 *Soit Sp une spécification structurée et soit $Sp' = (\Sigma', Ax')$ une spécification conditionnelle positive enrichissement de Sp.*

Le foncteur de synthèse $F : Mod(Sp) \rightarrow Mod(Sp')$ est adjoint à gauche au foncteur d'oubli $U_{|\Sigma} : Mod(Sp') \rightarrow Mod(Sp)$.

Preuve. Soit $\mathcal{A} \in |Mod(Sp)|$. Soit $\alpha : \mathcal{A} \rightarrow \mathcal{T}_{\Sigma'}(A)$ l'inclusion de $\mathcal{A}$ dans $\mathcal{T}_{\Sigma'}(A)$. Soit $I_{\mathcal{A}} : \mathcal{A} \rightarrow F(\mathcal{A})_{|\Sigma}$ la co-restriction de $h_{\mathcal{A}} \circ \alpha$ sur $F(\mathcal{A})_{|\Sigma}$, avec $h_{\mathcal{A}} : \mathcal{T}_{\Sigma'}(A) \rightarrow F(\mathcal{A})$ l'homomorphisme quotient de $\mathcal{T}_{\Sigma'}(A)$ sur $F(\mathcal{A})$. Pour montrer que F est le foncteur adjoint à gauche à $U_{|\Sigma}$, il suffit de montrer que $F(\mathcal{A})$ est libre en $\mathcal{A}$ (voir l'annexe A). Ceci revient à montrer que :

$$\forall \mathcal{A}' \in |Mod(Sp')|, \forall h : \mathcal{A} \rightarrow \mathcal{A}'_{|\Sigma}, \exists! h', h = U_{|\Sigma}(h') \circ I_{\mathcal{A}}$$

Il existe un unique Σ'-homomorphisme $h_{\mathcal{A}'} : \mathcal{T}_{\Sigma'}(A) \rightarrow \mathcal{A}'$, prolongement de h de $\mathcal{T}_{\Sigma'}(A)$ dans $\mathcal{A}'$. D'après le théorème 2.3, il existe un unique homomorphisme $h' : F(\mathcal{A}) \rightarrow \mathcal{A}'$ tel que

$$h_{\mathcal{A}'} = h' \circ h_{\mathcal{A}}$$

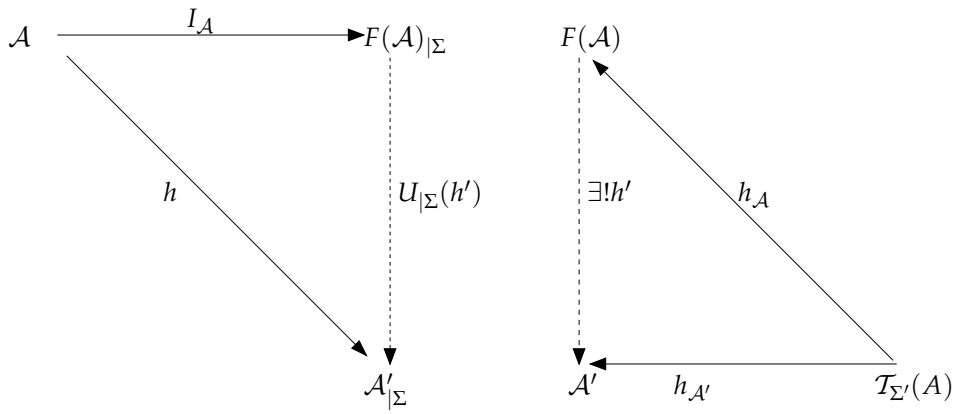
Ainsi $h_{\mathcal{A}'} \circ \alpha = h' \circ h_{\mathcal{A}} \circ \alpha$. Puisque $h_{\mathcal{A}'}$ est le prolongement de h et $h_{\mathcal{A}} \circ \alpha$ est le prolongement de $I_{\mathcal{A}}$, on obtient

$$h = U_{|\Sigma}(h') \circ I_{\mathcal{A}}$$

En plus, pour tout homomorphisme h'' tel que $h = U_{|\Sigma}(h'') \circ I_{\mathcal{A}}$, autre que h', est tel que $h'' \circ h_{\mathcal{A}}$ est un prolongement de h de $\mathcal{T}_{\Sigma'}(A)$ dans $\mathcal{A}'$. Or $h_{\mathcal{A}'}$ est l'unique prolongement de h de $\mathcal{T}_{\Sigma'}(A)$ dans $\mathcal{A}'$, donc

$$h'' \circ h_{\mathcal{A}} = h_{\mathcal{A}'}$$

Finalement, l'unicité de h' (c'est à dire $h' = h''$) résulte du théorème 2.3. Le schéma suivant est une illustration de la preuve.



□

Définition 2.10 (Consistance hiérarchique, suffisante complétude) *Soit Sp une spécification structurée et soit $Sp' = (\Sigma', Ax')$ une spécification conditionnelle positive enrichissement de Sp.*

- *La spécification Sp' est dite **hiérarchiquement consistante** au dessus de Sp si et seulement si pour tout $\mathcal{A} \in |Mod(Sp)|$ l'homomorphisme d'adjonction $I_{\mathcal{A}} : \mathcal{A} \to F(\mathcal{A})_{|\Sigma}$ est un monomorphisme.*
- *La spécification Sp' est dite **suffisamment complète** au dessus de Sp si et seulement si pour tout $\mathcal{A} \in |Mod(Sp)|$ l'homomorphisme d'adjonction $I_{\mathcal{A}} : \mathcal{A} \to F(\mathcal{A})_{|\Sigma}$ est un épimorphisme.*

3.2 Modularité orientée propriétés et complexité

Comme nous l'avons déjà vu, la structuration des spécifications algébriques par enrichissement ou par union vise à réutiliser des spécifications déjà existantes. Néanmoins, il est tout à fait possible que l'enrichissement et l'union tels que nous les avons définis (cf. définition 2.8) induisent à l'émergence d'un comportement non attendu des spécifications enrichies, de l'enrichissement ou de l'union. Cette émergence de comportement peut être caractérisée par la présence de propriétés, appelés *propriétés de non-conformité*, qui sont :

- soient des propriétés attendues d'une spécification enrichie (resp. des specifications enrichies), et qui ne sont pas des propriétés de l'enrichissement (resp. de l'union),
- soient des propriétés attendues de l'enrichissement (resp. de l'union), construites sur la signature de la spécification enrichie (resp. les signatures des spécifications enrichies), mais qui ne sont pas des propriétés de l'enrichissement (resp. de l'union).

Formellement, les propriétés attendues d'une spécification sont dénotées par l'ensemble des conséquences sémantiques de cette spécification.

Définition 2.11 (Conséquences sémantiques d'une spécification) *Soit Sp une spécification structurée. L'**ensemble des conséquences sémantiques** de Sp, noté $Sp^\bullet$, est l'ensemble des formules validées par toutes les Sp-algèbres :*

$$Sp^\bullet = \{\varphi \mid |Mod(Sp)| \models_\Sigma \varphi\}$$

En l'absence de propriétés de non-conformité, un enrichissement est dit *modulaire par propriétés*. Il est dit *complexe* dans le cas contraire.

Définition 2.12 (Enrichissement modulaire, complexe) *Soit Sp une spécification structurée et soit Sp' une spécifications enrichissement de Sp. Sp' est **modulaire par propriétés** au dessus de Sp si et seulement si,*

$$\forall \varphi \in For_{EL}(\Sigma), (\varphi \in Sp^\bullet \Leftrightarrow \varphi \in Sp'^\bullet)$$

*sinon elle est dite **complexe**.*

De la même façon, on définit une union modulaire par propriétés et une union complexe.

Définition 2.13 (Union modulaire, complexe) *Soient Sp_1, Sp_2 et Sp' trois spécifications structurées telles que Sp' est une union de Sp_1 et Sp_2. Sp' est **modulaire par propriétés** au dessus de Sp_1 et Sp_2 si et seulement si,*

$$\forall i \in \{1,2\}, \forall \varphi \in For_{EL}(\Sigma_i), (\varphi \in Sp_i^{\bullet} \Leftrightarrow \varphi \in Sp'^{\bullet})$$

*sinon elle est dite **complexe**.*

La condition de la définition 2.12 (resp. 2.13) est appelée *condition de modularité orientée propriétés*. S'il existe une formule $\varphi \in For_{EL}(\Sigma)$ (resp. $\varphi \in For_{EL}(\Sigma_i)$) qui ne vérifie pas la condition de modularité, alors φ est appelée une *propriété de non-conformité* pour l'enrichissement (resp. l'union).

Le sens "$\Rightarrow$" est toujours vérifié. En effet, on a le résultat suivant bien connu, appelé *condition de satisfaction* :

Théorème 2.6 *Soient Σ et Σ' deux signatures de SIG_{EL}. Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signatures. Pour toute formule $\varphi \in For_{EL}(\Sigma)$ et pour toute algèbre $\mathcal{A}' \in |Alg(\Sigma')|$,*

$$\mathcal{A}' \models_{\Sigma'} \overline{\sigma}(\varphi) \Leftrightarrow U_\sigma(\mathcal{A}') \models_{\Sigma} \varphi$$

Preuve. Soient V et V' deux ensembles de variables respectivement sur Σ et Σ' et soit $\sigma_V : V \to V'$ une application injective telle que pour tout $s \in S$, pour tout $x \in V_s$, $\sigma_V(x) \in V'_{\sigma_S(s)}$.

($\Rightarrow$) Soit $\nu : V \to U_\sigma(A')$ une interprétation quelconque et soit l'interprétation $\nu' : V' \to A'$ telle que pour tout $x \in V$, $\nu(x) = \nu'(\sigma_V(x))$ (une telle application existe puisque σ_V est injective). Par induction structurelle sur les termes de $T_\Sigma(V)$, on peut montrer que pour tout $t \in T_\Sigma(V)$,

$$\overline{\nu}(t) = \overline{\nu'}(\ddot{\sigma}(t))$$

En effet,

– si t est une variable $x \in V$, alors on a l'égalité par définition de ν'.

– si t une constante $f \in F_\varepsilon$, alors

$$\begin{array}{rcl} \overline{\nu}(f) & = & f_{U_\sigma(\mathcal{A}')} \\ & = & \sigma_F(f)_{\mathcal{A}'} \\ & = & \overline{\nu'}(\sigma_F(f)) \end{array}$$

– si t est de la forme $f(t_1, \ldots, t_n)$, alors

$$\begin{array}{rcl} \overline{\nu}(f(t_1, \ldots, t_n)) & = & f_{U_\sigma(\mathcal{A}')}(\overline{\nu}(t_1), \ldots, \overline{\nu}(t_n)) \\ & = & \sigma_F(f)_{\mathcal{A}'}(\overline{\nu}(t_1), \ldots, \overline{\nu}(t_n)) \\ & = & \overline{\nu'}(\sigma_F(f)(t_1, \ldots, t_n)) \end{array}$$

Par induction structurelle sur les formules de $For_{EL}(\Sigma)$, on peut montrer que pour tout $\varphi \in For_{EL}(\Sigma)$,

$$\mathcal{A}' \models_{\Sigma'} \overline{\sigma}(\varphi) \Rightarrow U_\sigma(\mathcal{A}') \models_{\Sigma} \varphi$$

En effet, si φ est de la forme $t = t'$, avec $t, t' \in T_\Sigma(V)$, alors

$$\mathcal{A}' \models_{\Sigma'} \overline{\sigma}(t = t') \Rightarrow \overline{\nu'}(\ddot{\sigma}(t)) = \overline{\nu'}(\ddot{\sigma}(t'))$$

et par conséquent,

$$\overline{\nu}(t) = \overline{\nu}(t')$$

Ceci pour tout $\nu : V \to U_\sigma(A')$. D'où,

$$U_\sigma(\mathcal{A}') \models_\Sigma t = t'$$

Pour les autres formes des formules, il est facile de déduire le résultat.

($\Leftarrow$) Soit $\nu' : V' \to A'$ une interprétation quelconque et soit une interprétation $\nu : V \to U_\sigma(A')$ telle que pour tout $x \in V$, $\nu(x) = \nu'(\sigma_V(x))$. Le reste de la preuve est similaire au sens ($\Rightarrow$) de la preuve.

□

De ce théorème, et du fait que $|Mod(Sp')|_{|\Sigma} \subseteq |Mod(Sp)|$, on obtient le résultat suivant :

Corollaire 2.2 *Soient Sp et Sp' deux spécifications structurées telles que Sp' est un enrichissement de Sp. Alors on a, $Sp^\bullet \subseteq Sp'^\bullet$*

Preuve. Par définition même de l'enrichissement (cf. remarque 2.1), on a :

$$|Mod(Sp')|_{|\Sigma} \subseteq |Mod(Sp)|$$

Donc pour toute formule $\varphi \in For_{EL}(\Sigma)$:

$$|Mod(Sp)| \models_\Sigma \varphi \Rightarrow |Mod(Sp')|_{|\Sigma} \models_\Sigma \varphi$$

Par la condition de satisfaction,

$$|Mod(Sp')|_{|\Sigma} \models_\Sigma \varphi \Leftrightarrow |Mod(Sp')| \models_{\Sigma'} \varphi$$

donc

$$|Mod(Sp)| \models_\Sigma \varphi \Rightarrow |Mod(Sp')| \models_{\Sigma'} \varphi$$

d'où $Sp^\bullet \subseteq Sp'^\bullet$ □

Bien entendu, et par analogie, on obtient le résultat suivant pour l'union de deux spécifications.

Corollaire 2.3 *Soient Sp_1, Sp_2 et Sp' trois spécifications structurées telles que Sp' est une union de Sp_1 et Sp_2. Alors on a, $\forall i \in \{1,2\}, Sp_i^\bullet \subseteq Sp'^\bullet$*

Preuve. La preuve se fait par analogie avec celle du corollaire 2.2. □

Une condition simple sur la classe des algèbres pour empêcher l'émergence de propriétés de non-conformité est alors la suivante :

$$\forall \mathcal{A} \in |Mod(Sp)|\ \exists \mathcal{A}' \in |Mod(Sp')|, \mathcal{A} \approx \mathcal{A}'_{|\Sigma}$$

En effet, quand cette condition est vérifiée, on a pour tout $\varphi \in For_{EL}(\Sigma)$,

$$|Mod(Sp')|_{|\Sigma} \models_\Sigma \varphi \Rightarrow |Mod(Sp)| \models_\Sigma \varphi$$

et par la condition de satisfaction on a,

$$|Mod(Sp')|_{|\Sigma} \models_\Sigma \varphi \Leftrightarrow |Mod(Sp')| \models_{\Sigma'} \varphi$$

ce qui entraîne,

$$|Mod(Sp')| \models_{\Sigma'} \varphi \Rightarrow |Mod(Sp)| \models_{\Sigma} \varphi$$

d'où $Sp'^{\bullet} \subseteq Sp^{\bullet}$

Cependant, on peut montrer que deux algèbres isomorphes sont élémentairement équivalentes. Ainsi, si pour toute algèbre $\mathcal{A} \in |Mod(Sp)|$, il existe une algèbre $\mathcal{A}' \in |Mod(Sp')|$ telle que $\mathcal{A}$ et $\mathcal{A}'_{|\Sigma}$ soient isomorphes, alors ceci peut être vu comme le pendant sémantique de la condition de la modularité orientée propriétés. Or, cette condition est garantie sous l'hypothèse que Sp' enrichissement de Sp soit conditionnelle positive et sous l'hypothèse de la complétude suffisante et de et de la consistance hiérarchique de Sp' au dessus de Sp.

Théorème 2.7 *Soit Sp une spécification structurée et soit $Sp' = (\Sigma', Ax')$ une spécification conditionnelle positive enrichissement de Sp. Si Sp' est suffisamment complète et hiérarchiquement consistante au dessus de Sp, alors Sp' est modulaire par propriétés au dessus de Sp.*

De ce premier résultat de modularité de l'enrichissement émane un second résultat induit par l'union de deux spécifications conditionnelles positives.

Théorème 2.8 *Soient Sp_1 et Sp_2 deux spécifications structurées et soit $Sp' = (\Sigma', Ax')$ une spécification conditionnelle positive union de Sp_1 et Sp_2. Si Sp' est suffisamment complète et hiérarchiquement consistante au dessus de Sp_1 et Sp_2, alors Sp' est modulaire par propriétés au dessus de Sp_1 et Sp_2.*

Preuve. La spécification Sp' est une union de Sp_1 et Sp_2. Donc par définition même de l'union de deux spécifications algébriques, Sp' a les caractéristiques d'un enrichissement de Sp_1 et de Sp_2 (cf. remarque 2.1). Si Sp' est suffisamment complète et hiérarchiquement consistante au dessus de Sp_1 et Sp_2, alors d'après le théorème 2.7,

$$\forall \varphi \in For_{EL}(\Sigma_1), (\varphi \in Sp_1^{\bullet} \Leftrightarrow \varphi \in Sp'^{\bullet})$$

et

$$\forall \varphi \in For_{EL}(\Sigma_2), (\varphi \in Sp_2^{\bullet} \Leftrightarrow \varphi \in Sp'^{\bullet})$$

d'où

$$\forall i \in \{1,2\}, \forall \varphi \in For_{EL}(\Sigma_i), (\varphi \in Sp_i^{\bullet} \Leftrightarrow \varphi \in Sp'^{\bullet})$$

Par conséquent, Sp' est une union modulaire par propriétés au dessus de Sp_1 et Sp_2.

□

4 Conclusion

Les méthodes algébriques standard décrites dans le chapitre 1 ont montré leur efficacité à capturer les différents aspects essentiels des structures de données utilisées en informatique. Nous avons aussi abordé la question de

la structuration des spécifications algébriques au travers les deux primitives d'enrichissement et de l'union. À l'issue de quoi, nous avons défini formellement la notion de modularité des spécifications algébriques, puis nous avons fourni certaines hypothèses qui nous ont permis d'avoir un résultat de modularité. Ainsi spécifier en terme de types abstraits est d'une grande utilité surtout dans le domaine de réutilisation des logiciels.

Cependant, en utilisant les types abstraits classiques, la sémantique des opérations exclut des hypothèses liées à l'environnement de l'exécution. Par exemple, elle ne prend pas en compte les aspects de concurrence des environnements coopératifs. Ainsi, il s'avère difficile de spécifier tous les systèmes qu'on puisse rencontrer par des spécifications algébriques : nous détaillerons dans la partie suivante de ce document un formalisme pour spécifier des systèmes réactifs. Ce formalisme est différent des spécifications algébriques dans le sens où le comportement d'un système ne se définit pas au travers de propriétés logiques attendues, mais plutôt par la donnée d'un système de transitions étiquetées.

Dans la suite de la thèse, et pour faire face à l'hétérogénéité des formalismes de spécification, nous définissons dans le cadre des institutions, des spécifications génériques qui peuvent être instanciées par d'autres spécifications classiques. Nous verrons que les notions de structuration, de modularité et de complexité que nous avons abordées ont une portée très générale, et peuvent être reformulées dans le cadre général des institutions. L'intérêt des institutions est de fournir des résultats généraux indépendants d'une logique sous-jacente choisie.

Deuxième partie

Cadre général de spécifications abstraites

Les institutions

3

Sommaire

D'une manière générale, et comme nous l'avons vu pour la logique sous-jacente aux spécifications algébriques, un formalisme logique est défini par la donnée d'une *syntaxe* et d'une *sémantique* :

- la syntaxe est caractérisée par la donnée d'une classe de signatures, de morphismes de signatures, qui représentent les relations entre signatures, et d'un ensemble de formules associé à chaque signature.
- la sémantique est caractérisée par la donnée, pour chaque signature, d'une classe de modèles et d'une relation de satisfaction des formules par les modèles, qui lui sont associés.

En effet, même si la définition de ces éléments diffère d'une logique à une autre, de nombreux points communs existent traduisant l'essence même d'un formalisme logique. En pratique la partie syntaxique d'un formalisme logique, tels que la logique propositionnelle, la logique du premier ordre, la logique équationnelle, la logique des clauses de Horn, la logique modale, les logiques temporelles et bien d'autres, repose sur la construction inductive de formules à partir de symboles de deux types, des symboles fixes et des symboles variables que l'on introduit dans le but de décrire le système à spécifier. Par exemple, et comme nous l'avons vu pour la logique équationnelle, les symboles variables sont les symboles de fonctions, alors que les symboles fixes sont les connecteurs logiques et les quantificateurs. La partie des symboles variables qui caractérise les

éléments de base du système à spécifier forme donc sa signature. De plus, et comme nous l'avons vu pour la logique équationnelle, un système spécifié au-dessus d'une logique est amené à être réutilisé. Cette réutilisation se caractérise en premier lieu par une modification de sa signature, par exemple par enrichissement, puis par une répercussion de cette modification au niveau syntaxique, c'est-à-dire sur les formules, puis au niveau sémantique, c'est-à-dire sur les modèles.

Pour appréhender la notion de signatures ainsi qu'un moyen de les comparer au niveau syntaxique et au niveau sémantique tout en capturant ces éléments indépendamment de la forme des formules ou des modèles, c'est-à-dire indépendamment de la logique sous-jacente, Goguen et Burstall ont proposé le cadre général des institutions [65, 63]. Le but était essentiellement d'abstraire la notion des formalismes logiques dans l'intérêt de développer des concepts sur des spécifications indépendamment des systèmes logiques sous-jacents.

En particulier le cadre offert par les institutions a permis de mener de nombreuses études sur la structuration de spécifications notamment grâce aux morphismes de signatures, mais aussi grâce à une propriété particulière des institutions appelée *condition de satisfaction* [63, 118] et des études sur le raffinement [113] et sur le test [84, 45].

Compte tenu des bénéfices importants apportés par les institutions, il est déjà usuel de représenter la logique sous-jacente à la spécification d'un système donné par une institution. Ceci permet donc de profiter des résultats généraux établis dans le cadre des institutions.

Dans cette thèse, et pour pouvoir généraliser les concepts liés à la problématique de la structuration et de la modularité des spécifications, vus au chapitre 2, indépendamment des formalismes logiques sous-jacents, nous opterons pour la même démarche, et ce pour deux raisons :

- le cadre offert par les institutions nous permettra de définir la notion de *spécifications abstraites* indépendamment du formalisme logique sous-jacent.
- ce choix nous permettra de tirer profit des travaux déjà menés pour la structuration des spécifications établis dans le cadre des institutions.

Dans ce chapitre, nous présentons des définitions élémentaires liées aux institutions et nous donnons quelques exemples qui nous seront utiles dans la suite.

1 Institutions

Une institution abstrait la notion de formalisme logique en donnant :

- une catégorie de signatures qui caractérisent les éléments de base nécessaires à la description d'un système à spécifier. Les morphismes de signatures qui représentent les relations entre signatures traduisent généralement un lien entre une signature « pauvre » et une signature « riche » en associant, à chaque élément de la signature pauvre, un élément de la signature riche. Intuitivement, les morphismes de signatures sont des applications qui permettent la modification de signatures par enrichissement, renommage, etc.
- un moyen d'associer, à chaque signature, un ensemble de formules bien formées sur cette signature. Chaque morphisme de signatures est transporté au niveau des formules en une fonction qui transforme les formules en renommant les symboles de la signature utilisés dans les formules en accord avec les morphismes de signatures.
- un moyen d'associer, à chaque signature, une catégorie de modèles. Les morphismes de signatures sont transportés aux modèles de façon à pouvoir restreindre un modèle associé à une signature « riche » en un modèle d'une signature plus « pauvre », le long du dit morphisme.
- un moyen d'associer à chaque signature, une relation entre les modèles et les formules associés à cette signature, appelée *relation de satisfaction*, permettant de dire si un modèle valide ou non une formule.

Une condition sur ces éléments impose la préservation de la relation de satisfaction par morphisme de signatures, c'est-à-dire qu'un changement de signature par morphisme de signatures ne doit pas modifier la satisfaction des formules par les modèles.

Nous donnons ci-dessous la définition des institutions introduite dans [65].

Définition 3.1 (Institution) *Une **institution** est un quadruplet $(SIG, For, Mod, \models)$ où :*

- *SIG est une catégorie dont les objets sont appelés des signatures ;*
- *$For : SIG \to SET$ est un foncteur[1] qui, à toute signature $\Sigma \in |SIG|$ associe l'ensemble $For(\Sigma)$ des formules sur Σ. On appelle Σ-formules les éléments de $For(\Sigma)$;*
- *$Mod : SIG \to CAT^{op}$ est un foncteur[2] qui, à chaque signature $\Sigma \in |SIG|$ associe la catégorie $Mod(\Sigma)$ des modèles de Σ. On appelle Σ-modèles les objets de $Mod(\Sigma)$;*
- *$\models = (\models_\Sigma)_{\Sigma \in |SIG|}$ est la famille de relations $\models_\Sigma$ sur $Mod(\Sigma) \times For(\Sigma)$ appelées relations de satisfaction.*

De plus, pour tout morphisme de signatures $\sigma : \Sigma \to \Sigma'$, pour tout Σ'-modèle $\mathcal{M}' \in |Mod(\Sigma')|$, pour toute Σ-formule $\varphi \in For(\Sigma)$, on a la propriété suivante appelée condition de satisfaction :

[1] SET est la catégorie dont les objets sont les ensembles et dont les morphismes sont les fonctions totales.

[2] CAT est la catégorie dont les objets sont des catégories et dont les morphismes sont les foncteurs. CAT^{op} est la catégorie duale de CAT, voir la définition A.5.

$$\mathcal{M}' \models_{\Sigma'} For(\sigma)(\varphi) \Leftrightarrow Mod(\sigma)(\mathcal{M}') \models_{\Sigma} \varphi$$

*Le Σ-modèle $Mod(\sigma)(\mathcal{M}')$ est appelé **oubli** de $\mathcal{M}'$ au travers du morphisme σ.*

L'intérêt de se restreindre aux institutions réside dans le fait que la condition de satisfaction permet d'obtenir comme conséquences immédiates des résultats intéressants dans le cadre de la structuration des spécifications (cf. chapitre 4).

Comme nous le verrons au travers des exemples de la section 3, il existe des logiques au sens de la définition 3.1, appelées *des logiques abstraites*, mais qui ne vérifient pas la condition de satisfaction. Sans détailler, la perte de la condition de satisfaction est essentiellement due à un élagage dans les modèles.

Définition 3.2 (Logique abstraite) *Une **logique abstraite** est un quadruplet $(SIG, For, Mod, \models)$ au sens de la définition 3.1 qui ne vérifie pas la condition de satisfaction.*

Pour le reste de ce chapitre, nous nous plaçons dans une institution notée $\mathcal{I} = (SIG, For, Mod, \models)$ fixée. Nous allons alors présenter dans le reste de cette section des notions classiques définies au dessus des institutions. Ces dernières nous seront utiles pour la suite.

Un modèle d'une signature donnée peut être caractérisé par l'ensemble des formules qu'il valide. Cet ensemble forme ce qu'on appelle la *théorie du modèle*.

Définition 3.3 (Théorie d'un modèle) *Soient Σ une signature de SIG et $\mathcal{M}$ un modèle de Σ. **La théorie de** $\mathcal{M}$, notée $Th(\mathcal{M})$, est l'ensemble de formules suivantes :*

$$Th(\mathcal{M}) = \{\varphi \in For(\Sigma) \mid \mathcal{M} \models_{\Sigma} \varphi\}$$

De même, pour une signature Σ et pour une formule φ sur Σ données, il est possible de considérer l'ensemble des modèles qui valident φ. On dit alors qu'un modèle appartenant à cet ensemble est *un modèle de φ*. Plus généralement, étant donné un ensemble de Σ-formules, on définit un modèle associé à cet ensemble de la façon suivante :

Définition 3.4 (Modèle d'un ensemble de formules) *Soient Σ une signature de SIG et $\Phi \subseteq For(\Sigma)$ un ensemble de Σ-formules. Un **modèle de** Φ est un Σ-modèle $\mathcal{M}$ qui valide toutes les formules de Φ, c'est-à-dire tel que :*

$$\forall \varphi \in \Phi, \mathcal{M} \models_{\Sigma} \varphi$$

On note $Mod(\Phi)$ la sous-catégorie pleine de $Mod(\Sigma)$ dont les objets sont des modèles de Φ.

Étant donné un ensemble de formules Φ, un modèle de Φ est donc un modèle qui valide en particulier les formules de Φ mais en général, valide aussi d'autres formules. Parmi celles-ci, il en existe qui sont également validées par tous les modèles de Φ, on dit que ce sont des conséquences sémantiques de Φ.

Définition 3.5 (Conséquences sémantiques d'un ensemble de formules) *Soient Σ une signature de SIG et $\Phi \subseteq For(\Sigma)$ un ensemble de Σ-formules. L'ensemble des* ***conséquences sémantiques de*** *Φ, noté $\Phi^\bullet$, est l'ensemble des formules validées par tous les modèles de Φ :*

$$\Phi^\bullet = \{\varphi \mid \forall \mathcal{M} \in Mod(\Phi), \mathcal{M} \models_\Sigma \varphi\}$$

Si $\varphi \in \Phi^\bullet$, on note $\Phi \models_\Sigma \varphi$ la relation de conséquence sémantique. Si $\Phi^\bullet = \Phi$ on dit que Φ est ***une théorie*** *au dessus de Σ.*

Pour une signature donnée, il existe une multitude de modèles entre lesquels il n'est possible d'établir de comparaison qu'au travers des formules qu'ils valident. Deux modèles d'une même signature qui valident exactement les mêmes formules sont dits *élémentairement équivalents*.

Définition 3.6 (Équivalence élémentaire de modèles) *Soit Σ une signature. Soient $\mathcal{M}$ et $\mathcal{M}'$ deux Σ-modèles. On dit que $\mathcal{M}$ et $\mathcal{M}'$ sont* ***élémentairement équivalents*** *si et seulement s'ils valident exactement les mêmes formules de $For(\Sigma)$:*

$$\forall \varphi \in For(\Sigma), \mathcal{M} \models_\Sigma \varphi \Leftrightarrow \mathcal{M}' \models_\Sigma \varphi$$

La notion d'équivalence élémentaire de modèles nous permet alors de définir la notion d'institution *close par isomorphismes*.

Définition 3.7 (Close par isomorphismes) *$\mathcal{I}$ est* ***close par isomorphismes*** *si et seulement si, pour tout $\Sigma \in |SIG|$, les Σ-modèles isomorphes sont élémentairement équivalents.*

2 Exemples d'institutions

Dans cette section, nous présentons quelques exemples d'institutions :

- Nous abordons la logique équationnelle vue en section 1 du chapitre 1 comme une première illustration des institutions.
- Nous présentons ensuite une institution pour deux logiques temporelles connues sous le nom de CTL (Computational Tree Logic) [47, 37] et CTL* [49, 48], suivi d'une variante de la logique CTL* qui nous sera utile par la suite pour modéliser les propriétés des réseaux de régulation génétique dans la partie III.
- Enfin, nous présentons une institution pour capturer une logique dédiée aux systèmes réactifs afin d'illustrer notre cadre général de spécifications abstraites (cf. section 1 du chapitre 4 où cet exemple sera largement repris).

2.1 Institution équationnelle

Dans cette section, nous allons introduire l'institution pour la logique équationnelle. Dans le chapitre précédent, nous avons bien avancé dans ce sens. En effet, nous avons donné :

- une catégorie SIG_{EL} dont les objets sont les signatures données par la définition 1.1 et dont les morphismes de signatures sont donnés par la définition 1.2.

- un moyen d'associer, à chaque signature Σ, la catégorie d'algèbres $Alg(\Sigma)$ (cf. section 1.2.1 du chapitre 1). En plus, pour tout morphisme de signatures $\sigma : \Sigma \to \Sigma'$, on considère $U_\sigma : Alg(\Sigma') \to Alg(\Sigma)$, qu'on note ici $Alg(\sigma)$ (cf. définition 1.15).
- un moyen d'associer, à chaque signature Σ, un ensemble de formules $For_{EL}(\Sigma)$ (cf. définition 1.7). En plus, tout morphisme de signatures $\sigma : \Sigma \to \Sigma'$ est étendu aux formules par l'application $\overline{\sigma} : For_{EL}(\Sigma) \to For_{EL}(\Sigma')$, qu'on note ici $For_{EL}(\sigma)$ (cf. définition 1.9).
- pour chaque signature Σ, une relation de satisfaction $\models_\Sigma$ sur $Alg(\Sigma) \times For_{EL}(\Sigma)$ (cf. définition 1.18).

Dans le chapitre 1, For_{EL} (resp. Alg) a été introduit comme une application qui à une signature Σ associe un ensemble de formules $For_{EL}(\Sigma)$ (resp. une catégorie d'algèbres $Alg(\Sigma)$). En plus, à tout morphisme de signatures $\sigma : \Sigma \to \Sigma'$ de SIG_{EL}, on considère l'application $For_{EL}(\sigma) : For_{EL}(\Sigma) \to For_{EL}(\Sigma')$ et l'application $Alg(\sigma) : Alg(\Sigma') \to Alg(\Sigma)$. $For_{EL} : SIG_{EL} \to SET$ et $Alg : SIG_{EL} \to CAT$ définissent bien des foncteurs.

Définition 3.8 (Foncteur For_{EL}) *$For_{EL} : SIG_{EL} \to SET$ est le foncteur qui associe :*

- *à chaque signature $\Sigma \in |SIG_{EL}|$ l'ensemble $For_{EL}(\Sigma)$ des Σ-formules ;*
- *à chaque morphisme de signatures $\sigma : \Sigma \to \Sigma'$ le prolongement de σ sur les formules $\overline{\sigma} : For_{EL}(\Sigma) \to For_{EL}(\Sigma')$ introduit à la définition 1.9.*

For_{EL} ainsi défini est bien un foncteur puisqu'il satisfait les conditions de la préservation de l'identité et la préservation de la composition.

Preuve. Soient $\Sigma, \Sigma', \Sigma''$ des signatures dans SIG_{EL}, $id_\Sigma : \Sigma \to \Sigma$ le morphisme identité sur Σ et $\sigma : \Sigma \to \Sigma'$ et $\sigma' : \Sigma' \to \Sigma''$ deux morphismes de signatures. Soient V, V' et V'' trois ensembles de variables respectivement sur Σ, Σ' et Σ'' et soient $\sigma_V : V \to V'$ et $\sigma'_{V'} : V' \to V''$ deux applications injecttives telles que pour tout $s \in S$, pour tout $x \in V_s$, $\sigma_V(x) \in V'_{\sigma_S(s)}$ et pour tout $s' \in S'$, pour tout $x' \in V'_{s'}$, $\sigma_{V'}(x') \in V'_{\sigma'_{S'}(s')}$.

Alors on a :

- $For_{EL}(id_\Sigma) = id_{For_{EL}(\Sigma)}$.
- Par induction structurelle sur les formules de $For_{EL}(\Sigma)$ et sur les termes, on peut montrer que pour tout $\varphi \in For_{EL}(\Sigma)$, $\overline{\sigma' \circ \sigma}(\varphi) = \overline{\sigma'} \circ \overline{\sigma}(\varphi)$. En effet, si φ est de la forme $t_1 = t_2$, avec $t_1, t_2 \in T_\Sigma(V)$, alors si t_1 et t_2 sont deux variables respectives x_1 et x_2 de V, alors :

$$\begin{array}{rcl} \overline{\sigma' \circ \sigma}(\varphi) & = & (\sigma' \ddot{\circ} \sigma(t_1) = \sigma' \ddot{\circ} \sigma(t_2)) \\ & = & (\sigma'_{V'} \circ \sigma_V(x_1) = \sigma'_{V'} \circ \sigma_V(x_2)) \\ & = & (\ddot{\sigma'} \circ \ddot{\sigma}(t_1) = \ddot{\sigma'} \circ \ddot{\sigma}(t_2)) \\ & = & \overline{\sigma'} \circ \overline{\sigma}(t_1 = t_2) \\ & = & \overline{\sigma'} \circ \overline{\sigma}(\varphi) \end{array}$$

Pour les autres formes des formules, il est facile de déduire le résultat. Ainsi,

$$\begin{array}{rcl} For_{EL}(\sigma' \circ \sigma)(\varphi) & = & \overline{\sigma' \circ \sigma}(\varphi) \\ & = & \overline{\sigma'} \circ \overline{\sigma}(\varphi) \\ & = & For_{EL}(\sigma') \circ For_{EL}(\sigma)(\varphi) \end{array}$$

□

Définition 3.9 (Foncteur Alg) *$Alg : SIG_{EL} \to CAT^{op}$ est le foncteur qui associe :*

– *à chaque signature* $\Sigma \in |SIG_{EL}|$ *la catégorie* $Alg(\Sigma)$ *des* Σ*-algèbres ;*
– *à chaque morphisme de signatures* $\sigma : \Sigma \to \Sigma'$ *le foncteur d'oubli* $U_\sigma : Alg(\Sigma') \to Alg(\Sigma)$ *introduit à la définition 1.15.*

De même, Alg ainsi défini est bien un foncteur puisqu'on peut vérifier que les conditions de la préservation de l'identité et la préservation de la composition sont satisfaites.

D'après le théorème 2.6 la condition de satisfaction des institutions est bien vérifiée par la logique abstraite $(SIG_{EL}, Alg, For_{EL}, \models_{EL})$, donc ce quadruplet est bien une institution.

Tout comme la logique équationnelle, plusieurs autres logiques utilisées comme formalismes de spécifications et souvent vues comme des extensions simples de la logique équationnelle ont été étudiées dans le cadre des institutions. On peut citer notamment la logique du premier ordre multi-sorte avec égalité, la logique du premier ordre partielle multi-sorte avec égalité, la logique du premier ordre partielle multi-sorte avec égalité et sous-sortes, la logique des clauses de Horn, la logique du second ordre, et les algèbres partielles [16, 18].

2.2 Institution CTL

La logique CTL, pour Computation Tree Logic, est une logique temporelle qui a été introduite par Clarke, Emerson et Sistla [38, 47] précisément pour exprimer des propriétés servant à vérifier les exécutions des systèmes à processus, comme les systèmes parallèles, les systèmes concurrents, les systèmes interactifs ou les systèmes réactifs, modélisées par des systèmes de transitions finis. On trouve notamment parmi ces systèmes des protocoles de communications et des contrôleurs de processus industriels potentiellement critiques.

Le type des systèmes de transitions auxquels cette logique s'applique est celui des systèmes de transitions non étiquetées, où chaque état est représenté comme une valuation d'un ensemble fini de variables propositionnelles. En pratique, ces variables propositionnelles abstraient des propriétés élémentaires du système. Ces variables propositionnelles sont appelées *propositions atomiques* et leur ensemble est traditionnellement noté AP (Atomic Propositions).

Dans les systèmes de transitions que considèrent Clarke, Emerson et Sistla, un état est toujours l'origine d'au moins une transition. Le type de système de transitions qu'ils proposent est celui des *structures de Kripke totales* où la relation de transition est une relation totale[3]. Formellement cette structure de Kripke est définie de la façon suivant :

Définition 3.10 (Structure de Kripke) *Soit AP un ensemble fini de propositions atomiques. Une* ***structure de Kripke*** *sur AP est un triplet (S, T, L) où :*
– *S est un ensemble fini d'états.*
– *$T \subseteq S \times S$ est une relation dite* ***relation de transition****.*

[3]Étant donné un ensemble fini E, une relation $R \subseteq E \times E$ est totale sur E si et seulement si, $\forall e \in E, \exists e' \in E, (e, e') \in R$.

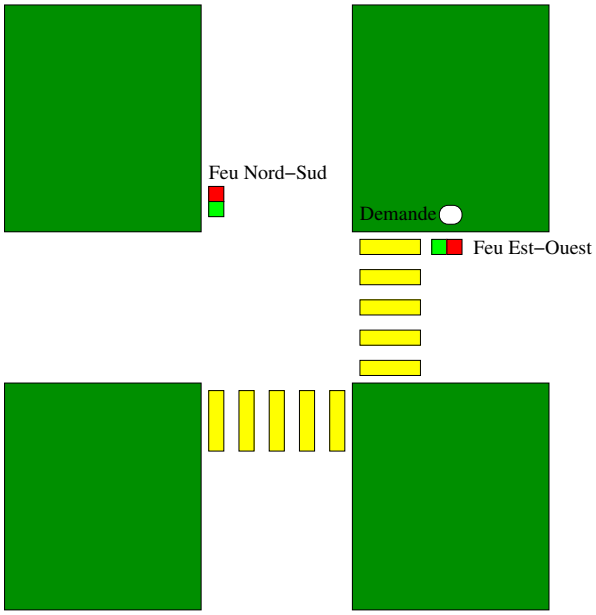


FIG. 3.1 – *Carrefour à deux feux.*

– $L : S \rightarrow 2^{AP}$ *est une fonction d'étiquetage*[4] *associant à chaque état un ensemble de propositions atomiques vraies dans cet état.*

Une structure de Kripke est dite ***totale*** *si la relation de transition T est totale.*

Intuitivement, l'ensemble S peut être vu comme une abstraction des différents états possibles du système. La fonction d'étiquetage L abstrait la validité des propriétés élémentaires pour le système dans un état considéré : une propriété élémentaire $p \in AP$ est "vraie" pour le système dans l'état s si et seulement si $p \in L(s)$. La relation de transition T, appelée aussi *relation d'accessibilité*, abstrait toutes les évolutions possibles du système : $(s, r) \in T$ abstrait le fait que le système peut évoluer de l'état s à l'état r.

Exemple 3.1 (Gestion de deux feux d'un carrefour) *Un carrefour dispose de deux voies, Nord-Sud et Est-Ouest, protégées par deux feux multicolores, et d'un passage pour les piétons. Ce passage comporte un bouton de demande de passage arrêtant la circulation dans les deux sens comme l'illustre la figure 3.1. Si le bouton n'est pas activé par un piéton, les deux feux Nord-Sud et Est-Ouest passent alternativement au vert puis au rouge. Si un piéton appuie sur le bouton, les deux feux Nord-Sud et Est-Ouest passent simultanément au rouge, ensuite le feu Nord-Sud passe au vert le premier.*

Pour modéliser le fonctionnement de ce carrefour à deux feux, on peut considérer deux variables propositionnelles nsvert et eovert qui prennent respectivement les valeurs vrai et faux (resp. faux et vrai) selon que le feu Nord-Sud (resp. Est-Ouest), soit vert ou non. Ainsi, les états et les évolutions possibles du carrefour peuvent être modélisés par un système de transitions dont l'ensemble d'états est $S_c = \{NS, EO, P\}$. À l'état NS, le feu Nord-Sud est vert ($nsvert = vrai$) tandis que le feu Est-Ouest est rouge ($eovert = faux$). À l'état EO, le feu Nord-Sud est rouge ($nsvert = faux$) tandis que celui dans la direction Est-Ouest est vert

[4] L'ensemble 2^{AP} désigne l'ensemble des parties de AP.

(eovert = vrai). D'après le fonctionnement du carrefour l'ensemble des transitions possibles est $T_c = \{(NS, EO), (EO, NS), (NS, P), (EO, P), (P, NS)\}$.

La structure de Kripke totale correspondante est la structure $K_c = (S_c, T_c, L_c)$ *sur* $\{nsvert, oevert\}$ *donnée par la figure 3.2, où la fonction d'étiquetage* $L_c : S_c \to \{\emptyset, \{nsvert\}, \{eovert\}, \{nsvert, eovert\}\}$ *est définie par* $L_c(NS) = \{nsvert\}$, $L_c(EO) = \{eovert\}$ *et* $L_c(P) = \emptyset$.

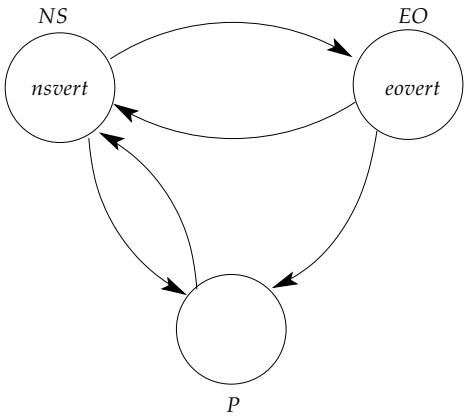


FIG. 3.2 – *Structure de Kripke totale modélisant un carrefour à deux feux*

La logique CTL est une *logique d'état*, c'est-à-dire qu'elle permet d'exprimer des propriétés sur l'ensemble des exécutions possibles d'un système à partir d'un état donné, et non pas uniquement sur une seule exécution, où une exécution est représentée par une séquence infinie d'états dont l'enchaînement est conditionné par les transitions du système. Elle permet notamment d'exprimer des propriétés du type « depuis un état donné, il est possible que le système passe par un état possédant telle propriété » par exemple « depuis un état où le feu Nord-Sud est rouge, il est possible que le système passe par un état où le feu Nord-Sud est vert ».

Dans la suite, étant donnée une structure de Kripke totale (S, T, L), on appelle *chemin* une séquence infinie d'états $\rho = s_0, s_1, \ldots$ telle que $(s_i, s_{i+1}) \in T$ pour tout $i \geq 0$. Ainsi, une exécution du système est donc représentée par un chemin. Pour un état s, on note $path(s)$ l'ensemble des chemins partant de s. Pour un chemin $\rho = s_0, s_1, \ldots$ et un indice $i \geq 0$, on note $\rho(i)$ l'état au rang i et ρ^i le chemin commençant à $\rho(i)$ et contenant tous les états $\rho(j)$, avec $i \leq j$. Un tel chemin ρ^i est appelé suffixe de ρ. On note, pour deux chemins ρ et θ, $\rho \preceq \theta$ (resp. $\rho \prec \theta$) si θ est suffixe (resp. suffixe propre, c'est-à-dire $\rho(i) \neq \rho(0)$) de ρ.

À partir des propositions atomiques AP, les opérateurs logiques de CTL permettent de construire de formules qui s'interprètent sur l'ensemble des chemins $path(s)$ partant d'un état s. On distingue les opérateurs traditionnels booléens $\neg$ et $\wedge$ et deux opérateurs temporels X, U précédé d'un quantificateur de chemin E ou A. Précisément, la syntaxe des formules CTL est donnée par la définition 3.12. Intuitivement :

– X est l'opérateur « ne$X$t » : la formule $EX\varphi$ (resp. $AX\varphi$) signifie « il existe un futur pour lequel (resp. dans n'importe quel futur), immédiatement φ va être vraie ».

– U est l'opérateur « Until » : la formule $E[\varphi U \psi]$ (resp. $A[\varphi U \psi]$) signifie « il existe un futur pour lequel (resp. dans n'importe quel futur), φ restera vraie jusqu'à ce que ψ soit vraie ».

À part ces opérateurs, il est courant d'utiliser des abréviations syntaxiques, dont nous présentons plus loin une liste non exhaustive de quelques opérateurs classiques utilisés dans la littérature.

Un large éventail de logiques temporelles ont été définies dans la littérature. Schématiquement, elles peuvent être classifiées suivant deux critères : logiques temporelles *arborescentes* qui permettent, d'exprimer des propriétés sur l'ensemble des exécutions possibles d'un système telles que la logique CTL et la logique CTL* [48], et logiques temporelles dites *linéaires*, notamment la logique LTL introduite par Pnueli en 1977 [103] et le μ-calcul linéaire [129], qui quantifient implicitement sur toutes les exécutions possibles du système, c'est-à-dire que l'on considère que le système dans un certain état à un certain instant ne peut être que dans un seul état à l'instant suivant, celui qui est donné par le chemin que l'on est en train d'étudier. Parmi ces logiques temporelles, ce sont les logiques arborescentes et notamment la logique CTL, qui a été choisie pour la vérification des exécutions des systèmes à processus comportant du non-déterminisme puisqu'elle permet de prendre en compte le branchement de l'exécution.

La logique CTL, et généralement les logiques temporelles, ont été beaucoup étudiées depuis leur introduction en vérification. Ces études ont permis de disposer d'outils pour vérifier automatiquement les propriétés qu'un système, modélisé par un système de transitions fini, doit satisfaire. Les premières études consistent à examiner de manière exhaustive l'ensemble des états du système. Les premiers algorithmes de *model checking* ont été développés au début des années 80 indépendamment par Clarke et Emerson aux États Unis [37] et par Queille et Sifakis en France [106]. Schématiquement, un algorithme de model checking prend en entrée une abstraction du comportement du système (un système de transitions fini) et une formule d'une certaine logique temporelle, et répond si l'abstraction satisfait ou non la formule. On dit alors que le système de transitions est un modèle de la formule, d'où le terme anglais de model checking.

Les premiers outils de vérification, appelés model checkers pouvaient analyser des systèmes contenant quelques dizaines de milliers d'états. De point de vue pratique, le problème principal est l'explosion du nombre d'états du système souvent rencontré dans les applications réelles, par exemple on compte déjà $2^{10\times 8}$ états pour un programme manipulant 10 variables codées sur 8 bits, ce qui a rendu les cas d'études industriels difficiles à appréhender. Le phénomène d'explosion combinatoire a deux causes distinctes : la taille du système de transitions augmente exponentiellement avec le nombre de variables (et leur taille) ou bien avec le nombre de composants du système dans le cas où le système est concurrent.

À partir des années 90, plusieurs techniques spécifiques ont été développées pour limiter ces deux causes potentielles d'explosion. Citons notamment, l'utilisation de solveurs de satisfaisabilité booléenne [28] et de

diagrammes de décisions binaires [40, 30] pour les représentations symboliques, qui permettent de représenter de manière très compacte le système de transitions, et l'utilisation les techniques d'ordres partiels [58], qui permettent elles d'alléger la vérification en ne construisant qu'une partie du système de transitions.

Des outils, intégrant ces techniques, ont été développés pour la vérification automatique des propriétés exprimées en CTL. Citons notamment NuSMV [36] qui est un model checker efficace intégrant des techniques basées sur l'utilisation des diagrammes de décision binaires pour la vérification symbolique de propriétés formulées en CTL et en LTL, et sur l'utilisation de solveurs de satisfaisabilité booléenne pour la vérification bornée des propriétés formulées en LTL [35]. Ces techniques et notamment ces outils ont rendu possibles les premières études de cas de taille industrielle, avec des systèmes de transition allant jusqu'à 10^{20} états.

Pour remédier au problème de l'explosion d'états, ce phénomène a été contré en utilisant diverses techniques de réduction de systèmes de transitions. Plusieurs relations d'équivalence sur l'ensemble des états de systèmes de transitions ont été étudiées [94, 101] afin de diminuer le coût de la vérification : au lieu de vérifier une propriété sur un système de transitions fini, on la vérifie sur le quotient de ce système de transitions modulo une relation d'équivalence convenablement choisie.

Dans la suite, nous allons présenter la logique CTL sous forme d'institution au sens de la définition 3.1.

La catégorie des signatures SIG_{CTL}. Une signature de la logique CTL est un ensemble de propositions atomiques et les morphismes de signatures sont les fonctions entre ensembles. Ainsi la catégorie SIG_{CTL} des signatures de la logique CTL est la catégorie des ensembles SET. On obtient alors la définition suivante :

Définition 3.11 (Catégorie des signatures SIG_{CTL}) *La catégorie des signatures, notée SIG_{CTL}, est la catégorie SET. Étant donnée une signature $\Sigma \in |SIG_{CTL}|$, les éléments de Σ sont appelés des **propositions atomiques**.*

Soient Σ et Σ' deux signatures de SIG_{CTL} telles que pour tout $p \in \Sigma$, $p \in \Sigma'$, alors on note $\Sigma \hookrightarrow \Sigma'$ la fonction d'inclusion.

Exemple 3.2 *La signature de la gestion de deux feux de carrefour de l'exemple 3.1 est $\Sigma_c = \{nsvert, eovert\}$.*

Imaginons maintenant que l'on dispose d'un carrefour à trois voies, Nord-Sud, Sud-Nord et Est-Ouest, protégées par trois feux multicolores, et qui dispose d'un passage pour les piétons. Le principe de fonctionnement du carrefour est le même que dans l'exemple 3.1, auquel on ajoute le fonctionnement du feu sud-nord qui est synchronisé avec le feu Nord-Sud. Une signature pour désigner les propositions atomiques peut être $\Sigma_{c'} = \{nsvert, eovert, snvert\}$, et le morphisme de signatures entre Σ_c et $\Sigma_{c'}$ sera la fonction d'inclusion $\Sigma_c \hookrightarrow \Sigma_{c'}$.

Le foncteur For_{CTL}. Nous allons maintenant définir l'ensemble des formules associées à une signature de SIG_{CTL}.

Définition 3.12 (Les formules CTL) *Soit Σ une signature de SIG_{CTL}. L'ensemble des Σ-formules, noté $For_{CTL}(\Sigma)$, est défini de la façon suivante :*

- *pour tout* $p \in \Sigma$, $p \in For_{CTL}(\Sigma)$;
- *si* $\varphi \in For_{CTL}(\Sigma)$, *alors* $\neg\varphi \in For_{CTL}(\Sigma)$;
- *si* $\varphi, \psi \in For_{CTL}(\Sigma)$, *alors* $\varphi \wedge \psi \in For_{CTL}(\Sigma)$;
- *si* $\varphi \in For_{CTL}(\Sigma)$, *alors* $EX\varphi, AX\varphi \in For_{CTL}(\Sigma)$;
- *si* $\varphi, \psi \in For_{CTL}(\Sigma)$, *alors* $E[\varphi U \psi], A[\varphi U \psi] \in For_{CTL}(\Sigma)$.

Dans la littérature, les formules CTL sont appelées des *formules d'états* et outre les opérateurs booléens usuels comme $\vee$ ou $\Rightarrow$, les abréviations suivantes sont couramment utilisées :

- $EF\varphi$ (resp. $AF\varphi$) définie par $E[trueU\varphi]$ (resp. $A[trueU\varphi]$) signifie « il existe un futur pour lequel (resp. dans n'importe quel futur), φ finira par être vraie », où la formule *true* est définie par $\neg(\neg p \wedge p)$.
- $EG\varphi$ (resp. $AG\varphi$) définie par dualité comme $E\neg F\neg\varphi$ (resp. $A\neg F\neg\varphi$)) signifie « il existe un futur pour lequel (resp. dans n'importe quel futur), φ sera tout le temps vraie ».

Nous devons maintenant caractériser la façon dont le foncteur For_{CTL} transporte les morphismes de signatures. Un morphisme de signatures $\sigma : \Sigma \to \Sigma'$ est transporté, au niveau des formules, en des fonctions qui transforment les formules en changeant tous les symboles de Σ apparaissant dans les formules de $For_{CTL}(\Sigma)$ par les symboles de Σ' correspondants.

Définition 3.13 (Prolongement de morphismes de signatures aux ensembles de formules) *Soit* $\sigma : \Sigma \to \Sigma'$ *un morphisme de signatures de* SIG_{CTL}.

Le ***prolongement de*** σ ***aux ensembles des formules*** $For_{CTL}(\sigma) : For_{CTL}(\Sigma) \to For_{CTL}(\Sigma')$ *est défini pour tout* $\varphi \in For_{CTL}(\Sigma)$ *inductivement sur la structure de* φ *de la façon suivante :*

- *si* φ *est une proposition atomique* $p \in \Sigma$, *alors* $For_{CTL}(\sigma)(p) = \sigma(p)$;
- *si* φ *est de la forme* $\neg\psi$, *alors* $For_{CTL}(\sigma)(\varphi) = \neg For_{CTL}(\sigma)(\psi)$;
- *si* φ *est de la forme* $\psi \wedge \varrho$, *alors* $For_{CTL}(\sigma)(\varphi) = For_{CTL}(\sigma)(\psi) \wedge For_{CTL}(\sigma)(\varrho)$;
- *si* φ *est de la forme* $EX\psi$ *(resp.* $AX\psi$*), alors* $For_{CTL}(\sigma)(\varphi) = EXFor_{CTL}(\sigma)(\varphi)$ *(resp.* $AXFor_{CTL}(\sigma)(\varphi)$*)* ;
- *si* φ *est de la forme* $E[\psi U \varrho]$ *(resp.* $A[\psi U \varrho]$*), alors* $For_{CTL}(\sigma)(\varphi) = E[For_{CTL}(\sigma)(\psi) U For_{CTL}(\sigma)(\varrho)]$ *(resp.* $A[For_{CTL}(\sigma)(\psi) U For_{CTL}(\sigma)(\varrho)]$*).*

On peut montrer que For_{CTL} préserve l'identité et la composition, donc For_{CTL} est bien un foncteur de SIG_{CTL} dans SET.

Le foncteur Mod_{CTL}**.** Nous allons maintenant caractériser la catégorie des modèles associée à une signature.

Définition 3.14 (Les modèles) *Soit* Σ *une signature dans* SIG_{CTL}. *Un* Σ*-modèle est une structure de Kripke totale sur* Σ.

Nous noterons $|Mod_{CTL}(\Sigma)|$ *la classe de* Σ*-modèles.*

Puisque le foncteur *Mod* d'une institution associe à chaque signature une catégorie, nous devons caractériser la catégorie des modèles associée à une signature. Pour une signature Σ de SIG_{CTL}, nous commençons par définir la notion de morphisme de modèles.

Définition 3.15 (Morphisme de modèles) *Soit Σ une signature de SIG_{CTL}. Soit $\mathcal{M} = (S, T, L)$ et $\mathcal{M}' = (S', T', L')$ deux Σ-modèles.*

Un morphisme de Σ-modèles $m : \mathcal{M} \to \mathcal{M}'$ est la donnée d'une application $m_S : S \to S'$ satisfaisant les conditions suivantes :

- *pour tous $s_1, s_2 \in S$, si $(s_1, s_2) \in T$ alors $(m_S(s_1), m_S(s_2)) \in T'$.*
- *pour tout $s \in S$, $L(s) \subseteq L'(m_S(s))$.*

$|Mod_{CTL}(\Sigma)|$ munie des morphismes de modèles définit clairement une catégorie de Σ-modèles. Nous allons maintenant définir le foncteur d'oubli associé à un morphisme de signatures.

Définition 3.16 (Foncteur d'oubli sur CTL) *Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signatures de SIG_{CTL}. Le foncteur d'oubli $Mod_{CTL}(\sigma) : Mod_{CTL}(\Sigma') \to Mod_{CTL}(\Sigma)$ est défini de la manière suivante :*

- *pour tout Σ'-modèle (S', T', L'), $Mod_{CTL}(\sigma)((S', T', L')) = (S', T', L)$, où pour tout $s \in S'$, $L(s) = \{p \in \Sigma \mid \sigma(p) \in L'(s)\}$.*
- *pour tout morphisme $m' : \mathcal{M}'_1 \to \mathcal{M}'_2$ de $Mod_{CTL}(\Sigma')$, $Mod_{CTL}(\sigma)(m') : Mod_{CTL}(\sigma)(\mathcal{M}'_1) \to Mod_{CTL}(\sigma)(\mathcal{M}'_2)$ est l'unique morphisme de $Mod_{CTL}(\Sigma)$ tel que si l'on note $\mathcal{M}'_1 = (S'_1, T'_1, L'_1)$ et $Mod_{CTL}(\sigma)(\mathcal{M}'_1) = (S'_1, T'_1, L_1)$, alors pour tout $s \in S'_1$, $Mod_{CTL}(\sigma)(m'_S)(s) = m'_S(s)$.*

Si $\Sigma \hookrightarrow \Sigma'$, on note $Mod_{CTL_{|\Sigma}}$ le foncteur d'oubli.

Intuitivement, le foncteur d'oubli associé à un morphisme de signatures $\sigma : \Sigma \to \Sigma'$ associe à tout Σ'-modèle $\mathcal{M}'$ un Σ-modèle $\mathcal{M}$ ayant le même ensemble d'états, la même relation de transitions, et pour chaque état, l'ensemble des propositions atomiques associé par L est celui associé par L' en "oubliant" toutes les propositions atomiques qui n'appartiennent pas à Σ.

Mod_{CTL} définit clairement un foncteur contravariant[5] de SIG_{CTL} dans CAT.

Exemple 3.3 *Considérons le morphisme d'inclusion $\Sigma_c \hookrightarrow \Sigma_{c'}$ de l'exemple 3.2 et soit K'_c un modèle de $\Sigma_{c'}$ donnée par la figure 3.3.*

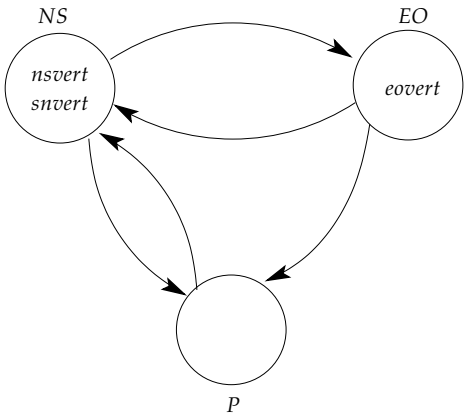


FIG. 3.3 – *K'_c : un modèle de $\Sigma_{c'}$*

[5]Voir la définition A.8 de l'annexe A pour la notion de foncteur contravariant.

Alors $Mod_{CTL_{|\Sigma}}(K'_c)$ *n'est autre que la structure de Kripke* K_c *de l'exemple 3.1.*

La relation de satisfaction $\models_{CTL}$**.** La satisfaction des formules par les modèles est définie de la manière suivante :

Définition 3.17 (La relation de satisfaction $\models_{CTL}$) *Soient* Σ *une signature de* SIG_{CTL} *et* $\mathcal{M} = (S, T, L)$ *un* Σ*-modèle. Soient* $s \in S$ *un état et* $\varphi \in For_{CTL}(\Sigma)$ *une* Σ*-formule.*

La satisfaction de φ *par* s*, notée* $\models^{ctl}_{\Sigma}$*, est définie inductivement sur la structure de* φ *de la manière suivante :*

- *si* $\varphi \in \Sigma$*, alors* $(\mathcal{M}, s) \models^{ctl}_{\Sigma} \varphi$ *ssi* $\varphi \in L(s)$ *;*
- *si* φ *est de la forme* $\neg\psi$*, alors* $(\mathcal{M}, s) \models^{ctl}_{\Sigma} \neg\psi$ *ssi* $(\mathcal{M}, s) \not\models^{ctl}_{\Sigma} \psi$ *;*
- *si* φ *est de la forme* $\psi \wedge \varrho$*, alors* $(\mathcal{M}, s) \models^{ctl}_{\Sigma} \psi \wedge \varrho$ *ssi* $(\mathcal{M}, s) \models^{ctl}_{\Sigma} \psi$ *et* $(\mathcal{M}, s) \models^{ctl}_{\Sigma} \varrho$ *;*
- *si* φ *est de la forme* $EX\psi$*, alors* $(\mathcal{M}, s) \models^{ctl}_{\Sigma} EX\psi$ *ssi* $\exists \rho \in path(s)$*,* $(\mathcal{M}, \rho(1)) \models^{ctl}_{\Sigma} \psi$ *;*
- *si* φ *est de la forme* $AX\psi$*, alors* $(\mathcal{M}, s) \models^{ctl}_{\Sigma} AX\psi$ *ssi* $\forall \rho \in path(s)$*,* $(\mathcal{M}, \rho(1)) \models^{ctl}_{\Sigma} \psi$ *;*
- *si* φ *est de la forme* $E[\psi U \varrho]$*, alors* $(\mathcal{M}, s) \models^{ctl}_{\Sigma} E[\psi U \varrho]$ *ssi* $\exists \rho \in path(s), \exists i \geq 0, (\mathcal{M}, \rho(i)) \models^{ctl}_{\Sigma} \varrho$ *et* $\forall j < i, (\mathcal{M}, \rho(j)) \models^{ctl}_{\Sigma} \psi$ *;*
- *si* φ *est de la forme* $A[\psi U \varrho]$*, alors* $(\mathcal{M}, s) \models^{ctl}_{\Sigma} A[\varphi U \psi]$ *ssi* $\forall \rho \in path(s), \exists i \geq 0, (\mathcal{M}, \rho(i)) \models^{ctl}_{\Sigma} \varrho$ *et* $\forall j < i, (\mathcal{M}, \rho(j)) \models^{ctl}_{\Sigma} \psi$ *;*

Si $(\mathcal{M}, s) \models^{ctl}_{\Sigma} \varphi$*, on dit que* s ***satisfait*** φ*, et si* $(\mathcal{M}, s) \models^{ctl}_{\Sigma} \varphi$ *pour tout* $s \in S$*, on dit que* $\mathcal{M}$ *satisfait* φ*, et on note* $\mathcal{M} \models^{ctl}_{\Sigma} \varphi$*.*

La relation de satisfaction $\models_{CTL}$ *est la famille de relations* $(\models^{ctl}_{\Sigma})_{\Sigma \in |SIG_{CTL}|}$*.*

D'une manière générale la formule $AG\neg\varphi$ permet d'exprimer la sûreté. Par exemple, la formule,

$$AG\neg(nsvert \wedge eovert)$$

se traduit par « à aucun instant les deux feux Nord-Sud et Est-Ouest sont simultanément verts ». Cette formule est bien satisfaite par K_c.

$$K_c \models^{ctl}_{\Sigma_c} AG\neg(nsvert \wedge eovert)$$

La formule $EF\varphi$ permet d'exprimer l'atteignabilité d'une propriété. Par exemple, la formule,

$$EF\neg(nsvert \vee oevert)$$

traduit le fait qu'il est possible d'atteindre un état où les deux feux sont rouges. Cette formule est bien satisfaite par K_c.

$$K_c \models^{ctl}_{\Sigma_c} EF\neg(nsvert \vee oevert)$$

Et la formule,

$$(\neg eovert \Rightarrow EFeovert)$$

se traduit par « si on passe par un état où le feu Est-Ouest n'est pas vert, alors il est possible d'atteindre un état où ce feu est vert ». Cette formule est satisfaite par K_c.

$$K_c \models^{ctl}_{\Sigma_c} \neg eovert \Rightarrow EFeovert$$

La formule $AF\varphi$ permet d'exprimer la vivacité, ou la garantie d'une propriété. Par exemple, la formule,

$$AFnsvert$$

exprime le fait qu'on finira par atteindre un état où le feu Nord-Sud est vert. Cette formule est bien satisfaite par K_c.

$$K_c \models^{ctl}_{\Sigma_c} AFnsvert$$

Néanmoins,

$$K_c \models^{ctl}_{\Sigma_c} \neg AFeovert$$

En effet, pour l'état NS et pour le chemin $\rho = NS, P, NS, P, \ldots$, ni l'état NS, ni l'état P ne vérifie la propriété *eovert*.

Théorème 3.1 *Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signatures de SIG_{CTL}. Pour tout Σ'-modèle $\mathcal{M}' \in |Mod_{CTL}(\Sigma')|$, pour toute Σ-formule $\varphi \in For_{CTL}(\Sigma)$ on a :*

$$\mathcal{M}' \models^{ctl}_{\Sigma'} For_{CTL}(\sigma)(\varphi) \Leftrightarrow Mod_{CTL}(\sigma)(\mathcal{M}') \models^{ctl}_{\Sigma} \varphi$$

Preuve. La preuve se fait par induction sur la structure des formules. Pour toute proposition atomique $p \in \Sigma$, pour tout $s \in S'$, $p \in L(s)$ si et seulement si $Mod_{CTL}(\sigma)(p) \in L'(s)$ ce qui prouve la condition de satisfaction pour les propositions atomiques et en conséquence les formules contenant des opérateurs logiques habituels. La préservation de la condition de satisfaction pour les formules contenant des quantificateurs de chemins et des opérateurs temporels, se vérifie facilement puisque l'ensemble des états et la relation d'accessibilité sont tous les deux préservés par le foncteur d'oubli $Mod_{CTL}(\sigma)$. □

D'après le théorème 3.1, la condition de satisfaction est bien vérifiée, donc le quadruplet $(SIG_{CTL}, Mod_{CTL}, For_{CTL}, \models_{CTL})$ est bien une institution.

2.3 Institution pour la logique CTL*

La logique CTL* [49, 48] est une logique temporelle qui contient des formules d'états analogues aux formules d'états de CTL et d'autres formules appelées des formules de chemins. Comme la logique CTL, elle s'applique aux systèmes de transitions non étiquetées représentés par des structures de Kripke totale. Concrètement, la logique CTL* est une extension de la logique CTL que l'on obtient en relaxant la condition qui impose que tout opérateur temporel doit être immédiatement précédé d'un quantificateur de chemin [48].

Tout comme nous l'avons fait pour la logique CTL, nous allons présenter la logique CTL* sous forme d'institution.

La catégorie des signatures SIG_{CTL^*}. La catégorie SIG_{CTL^*} de la logique CTL* est exactement la catégorie SIG_{CTL}.

Le foncteur For_{CTL^*}. Les formules associées à une signature de SIG_{CTL^*} sont formées selon les règles de la définition suivante.

Définition 3.18 (Les formules CTL*) *Soit Σ une signature de SIG_{CTL}. L'ensemble des Σ-formules, noté $For_{CTL^*}(\Sigma)$, est défini de la façon suivante :*

- *pour tout $p \in \Sigma$, $p \in For^s_{CTL^*}(\Sigma)$;*
- *si $\varphi \in For^s_{CTL^*}(\Sigma)$, alors $\neg\varphi \in For^s_{CTL^*}(\Sigma)$;*
- *si $\varphi, \psi \in For^s_{CTL^*}(\Sigma)$, alors $\varphi \wedge \psi \in For^s_{CTL^*}(\Sigma)$;*
- *si $\pi \in For^p_{CTL^*}(\Sigma)$, alors $E\pi \in For^s_{CTL^*}(\Sigma)$;*
- *si $\varphi \in For^s_{CTL^*}(\Sigma)$, alors $\varphi \in For^p_{CTL^*}(\Sigma)$;*
- *si $\pi \in For^p_{CTL^*}(\Sigma)$, alors $\neg\pi \in For^p_{CTL^*}(\Sigma)$;*
- *si $\pi, \tau \in For^p_{CTL^*}(\Sigma)$, alors $\pi \wedge \tau \in For^p_{CTL^*}(\Sigma)$;*
- *si $\pi \in For^p_{CTL^*}(\Sigma)$, alors $X\pi \in For^p_{CTL^*}(\Sigma)$;*
- *si $\pi, \tau \in For^p_{CTL^*}(\Sigma)$, alors $\pi U \tau \in For^p_{CTL^*}(\Sigma)$;*
- *si $\varphi \in For^s_{CTL^*}(\Sigma)$, alors $\varphi \in For_{CTL^*}(\Sigma)$;*
- *si $\pi \in For^p_{CTL^*}(\Sigma)$, alors $\pi \in For_{CTL^*}(\Sigma)$.*

Les formules de $For^s_{CTL^*}(\Sigma)$ sont appelées des *formules d'états*, tandis que les formules de $For^p_{CTL^*}(\Sigma)$ sont appelées des *formules de chemins*. Outre les opérateurs booléens usuels comme $\vee$ ou $\Rightarrow$, les abréviations suivantes sont couramment utilisées :

- $A\chi$ est définie par $\neg E \neg \chi$;
- $F\chi$ est définie par $trueU\chi$;
- $G\chi$ est défini par $\neg F \neg \chi$.

Le foncteur For_{CTL^*} transporte les morphismes de signatures à des fonctions qui transforment les formules. Nous n'allons pas détailler davantage puisqu'on peut le déduire facilement de ce qui a été vu dans la présentation de la logique CTL.

Le foncteur Mod_{CTL^*}. Le foncteur Mod_{CTL^*} coïncide exactement avec le foncteur Mod_{CTL}.

La relation de satisfaction $\models_{CTL^*}$. La satisfaction des formules par les modèles est définie de la manière suivante :

Définition 3.19 (La relation de satisfaction $\models_{CTL^*}$) *Soient Σ une signature de SIG_{CTL} et $\mathcal{M} = (S, T, L)$ un Σ-modèle. Soient $s \in S$ un état, ρ un chemin de $\mathcal{M}$, $\varphi \in For^s_{CTL^*}(\Sigma)$ et $\pi \in For^p_{CTL^*}(\Sigma)$ deux Σ-formules.*

La relation de satisfaction de φ par s (resp. π par ρ), notée $\models^{ctl^*}_\Sigma$, est définie inductivement sur la structure de φ (resp. π) de la manière suivante :

- *si $\varphi \in \Sigma$, alors $(\mathcal{M}, s) \models^{ctl^*}_\Sigma \varphi$ ssi $\varphi \in L(s)$;*
- *si φ est de la forme $\neg\psi$, alors $(\mathcal{M}, s) \models^{ctl^*}_\Sigma \neg\psi$ ssi $(\mathcal{M}, s) \not\models^{ctl_*}_\Sigma \psi$;*
- *si φ est de la forme $\psi \wedge \varrho$, alors $(\mathcal{M}, s) \models^{ctl^*}_\Sigma \psi \wedge \varrho$ ssi $(\mathcal{M}, s) \models^{ctl^*}_\Sigma \varphi$ et $(\mathcal{M}, s) \models^{ctl^*}_\Sigma \psi$;*
- *si φ est de la forme $E\tau$, alors $(\mathcal{M}, s) \models^{ctl^*}_\Sigma E\tau$ ssi $\exists \theta \in path(s), (\mathcal{M}, \theta) \models^{ctl^*}_\Sigma \tau$;*
- *si $\pi \in For^s_{CTL^*}(\Sigma)$, alors $(\mathcal{M}, \rho) \models^{ctl^*}_\Sigma \pi$ ssi $(\mathcal{M}, \rho(0)) \models^{ctl^*}_\Sigma \pi$;*
- *si π est de la forme $\neg\tau$, alors $(\mathcal{M}, \rho) \models^{ctl^*}_\Sigma \neg\tau$ ssi $(\mathcal{M}, \rho) \not\models^{ctl^*}_\Sigma \tau$;*
- *si π est de la forme $\tau \wedge \kappa$, alors $(\mathcal{M}, \rho) \models^{ctl^*}_\Sigma \tau \wedge \kappa$ ssi $(\mathcal{M}, \rho) \models^{ctl^*}_\Sigma \tau$ et $(\mathcal{M}, \rho) \models^{ctl^*}_\Sigma \kappa$;*
- *si π est de la forme $\tau U \kappa$, alors $(\mathcal{M}, \rho) \models^{ctl^*}_\Sigma \pi U \tau$ ssi $\exists \theta, \rho \preceq \theta, (\mathcal{M}, \theta) \models^{ctl^*}_\Sigma \kappa$ et $\forall \eta, \rho \preceq \eta \prec \theta, (\mathcal{M}, \eta) \models^{ctl^*}_\Sigma \tau$;*

Si $(\mathcal{M}, s) \models^{ctl^*}_{\Sigma} \varphi$*, on dit que* s ***satisfait*** φ*, et si* $(\mathcal{M}, s) \models^{ctl^*}_{\Sigma} \varphi$ *pour tout* $s \in S$*, on dit que* $\mathcal{M}$ *satisfait* φ*, et on note* $\mathcal{M} \models^{ctl^*}_{\Sigma} \varphi$*.*

De même, si $(\mathcal{M}, \rho) \models^{ctl^*}_{\Sigma} \pi$*, on dit que* ρ ***satisfait*** π*, et si* $(\mathcal{M}, \rho) \models^{ctl^*}_{\Sigma} \varphi$ *pour tout chemin* ρ*, on dit que* $\mathcal{M}$ *satisfait* π*, et on note* $\mathcal{M} \models^{ctl^*}_{\Sigma} \pi$*.*

La relation de satisfaction $\models_{CTL^*}$ *est la famille de relations* $(\models^{ctl^*}_{\Sigma})_{\Sigma \in |SIG_{CTL^*}|}$*.*

Théorème 3.2 *Soit* $\sigma : \Sigma \to \Sigma'$ *un morphisme de signatures de* SIG_{CTL^*}*. Pour tout* Σ'*-modèle* $\mathcal{M}' \in |Mod_{CTL^*}(\Sigma')|$*, pour toute* Σ*-formule* $\chi \in For_{CTL^*}(\Sigma)$ *on a :*

$$\mathcal{M}' \models^{ctl^*}_{\Sigma'} For_{CTL^*}(\sigma)(\chi) \Leftrightarrow Mod_{CTL^*}(\sigma)(\mathcal{M}') \models^{ctl^*}_{\Sigma} \chi$$

Preuve. La preuve est similaire à celle du théorème 3.1. □

Ainsi, la condition de satisfaction est bien vérifiée et le quadruplet $(SIG_{CTL^*}, Mod_{CTL^*}, For_{CTL^*}, \models_{CTL^*})$ est bien une institution.

2.4 Institution pour une variante de la logique CTL* : la logique FITL

Dans les systèmes de transitions que considère Clarke, Emerson et Sistla, le relation de transition entre états est supposée être totale. Comme les systèmes de transitions que nous utiliserons dans le chapitre 5, pour modéliser la dynamique des réseaux de régulation génétique, n'ont pas nécessairement cette propriété, nous adaptons le type de systèmes de transitions qui n'est plus celui des structures de kripke totales, mais plutôt des structures de Kripke quelconques. Nous adaptons en conséquence les définitions qui s'y attachent, en particulier la notion de chemins. Comme la relation de transition n'est plus supposée totale, les chemins ne seront pas nécessairement infinis. Nous précisons dans le contexte de relation de transition non totale la notion de *chemins*.

Définition 3.20 (Chemin, chemin maximal) *Soit* (S, T, L) *une structure de Kripke quelconque.*

Nous appelons chemin maximal,
- *toute séquence infinie d'états* $(s_0, \ldots, s_n, \ldots)$ *telle que* $\forall i, 0 \leq i$, $(s_i, s_{i+1}) \in T$ *ou*
- *toute séquence finie d'états* $(s_0, \ldots, s_n)$ *telle que* $\forall i, 0 \leq i < n$, $(s_i, s_{i+1}) \in T$ *et tel qu'il n'existe pas d'état* $s \in S$ *tel que* $(s_n, s) \in T$*.*

Nous appelons chemin,
- *toute séquence finie d'états* $(s_0, \ldots, s_n)$ *telle que* $\forall i, 0 \leq i < n$, $(s_i, s_{i+1}) \in T$ *ou*
- *toute séquence infinie d'états* $(s_0, \ldots, s_n, \ldots)$ *telle que* $\forall i, 0 \leq i$, $(s_i, s_{i+1}) \in T$*.*

Les états de S sont considérés comme des chemins finis de longueur 1. Pour un état s, on note $path(s)$ l'ensemble des chemins partant d'un état s. Pour un chemin $\rho = s_0, s_1, \ldots$ et un indice $i \geq 0$, on note $\rho(i)$ l'état au rang i s'il existe. Pour un indice i tel que $\rho(i)$ existe, ρ^i dénote le chemin commençant à $\rho(i)$ et contenant tous les états $\rho(j)$, avec $i \leq j$. $Indices(\rho)$ est l'ensemble des indices i tel que ρ^i est défini. Un tel chemin ρ^i est appelé suffixe de ρ. On note, pour deux chemins ρ et θ, $\rho \preceq \theta$ (resp. $\rho \prec \theta$) si θ est suffixe (resp. suffixe propre, c'est-à-dire $\rho(i) \neq \rho(0)$) de ρ.

Compte tenu de ces modifications, nous proposons une variante de la logique CTL*, que nous appelons FITL (Finite and Infinite Tree Logic). La syntaxe des formules de cette logique est la même que celle de la logique CTL*. L'interprétation des formules, la plus communément utilisée pour les structures de Kripke quelconques, se fait sur les chemins maximaux, finis ou infinis [14]. Néanmoins, une autre interprétation sur des chemins quelconques existe [43]. Nous adoptons cette interprétation pour la logique FITL parce qu'elle permet d'avoir un résultat de préservation d'une certaine classe de propriétés entre une structure de Kripke et son quotient par une relation d'équivalence particulière, la relation DBSB [43]. Nous détaillerons ces propos en section 3.2 et nous précisons que ce résultat de préservation nous sera fondamental en section 5.1 du chapitre 5.

Nous allons maintenant présenter la logique FITL en tant qu'institution.

La catégorie des signatures SIG_{FITL}. La catégorie SIG_{FITL} de la logique FITL est exactement la catégorie SIG_{CTL^*} de la logique CTL donnée dans la définition 3.11.

Le foncteur For_{FITL}. Le foncteur For_{FITL} coïncide exactement le foncteur For_{CTL^*}. Pour une signature Σ donnée, on note $For^s_{FITL}(\Sigma)$ (resp. $For^p_{FITL}(\Sigma)$) l'ensemble des formules $For^s_{CTL^*}(\Sigma)$ (resp. $For^p_{CTL^*}(\Sigma)$).

Étant donnée une signature Σ de SIG_{FITL}, nous définissons deux fragments de l'ensemble des formules $For_{FITL}(\Sigma)$ qui nous seront utiles pour modéliser les propriétés des réseaux de régulation génétique dans la partie III : le premier concerne le fragment des formules $For_{FITL}(\Sigma)$ lorsque l'opérateur « neXt » n'est pas appliqué aux formules, et le deuxième concerne les formules de ce fragment qui sont données sous sous forme normale positive, c'est-à-dire lorsque les négations ne sont appliquées qu'aux propositions atomiques.

Définition 3.21 (L'ensemble des formules sans « neXt ») *Soit Σ une signature de SIG_{FITL}. L'ensemble des formules sans « neXt » est l'ensemble de formules inclus dans $For_{FITL}(\Sigma)$ noté $For_{FITL\text{-}X}(\Sigma)$ et défini inductivement sur Σ de la façon suivante :*

- *pour tout $p \in \Sigma$, $p \in For^s_{FITL\text{-}X}(\Sigma)$;*
- *si $\varphi \in For^s_{FITL\text{-}X}(\Sigma)$, alors $\neg\varphi \in For^s_{FITL\text{-}X}(\Sigma)$;*
- *si $\varphi, \psi \in For^s_{FITL\text{-}X}(\Sigma)$, alors $\varphi \wedge \psi \in For^s_{FITL\text{-}X}(\Sigma)$;*
- *si $\pi \in For^p_{FITL\text{-}X}(\Sigma)$, alors $E\pi \in For^s_{FITL\text{-}X}(\Sigma)$;*
- *si $\varphi \in For^s_{FITL\text{-}X}(\Sigma)$, alors $\varphi \in For^p_{FITL\text{-}X}(\Sigma)$;*
- *si $\pi \in For^p_{FITL\text{-}X}(\Sigma)$, alors $\neg\pi \in For^p_{FITL\text{-}X}(\Sigma)$;*
- *si $\pi, \tau \in For^p_{FITL\text{-}X}(\Sigma)$, alors $\pi \wedge \tau \in For^p_{FITL\text{-}X}(\Sigma)$;*
- *si $\pi, \tau \in For^p_{FITL\text{-}X}(\Sigma)$, alors $\pi U \tau \in For^p_{FITL\text{-}X}(\Sigma)$;*
- *si $\varphi \in For^s_{FITL\text{-}X}(\Sigma)$, alors $\varphi \in For_{FITL\text{-}X}(\Sigma)$;*
- *si $\pi \in For^p_{FITL\text{-}X}(\Sigma)$, alors $\pi \in For_{FITL\text{-}X}(\Sigma)$.*

Définition 3.22 (L'ensemble des formules en forme normale positive) *Soit Σ une signature de SIG_{FITL}. L'ensemble des formules en forme normale positive est l'ensemble des formules inclus dans $For_{FITL}(\Sigma)$ noté $For_{nFITL\text{-}X}(\Sigma)$ et défini inductivement sur Σ de la façon suivante :*

- *pour tout $p \in \Sigma$, $p \in For^s_{nFITL\text{-}X}(\Sigma)$ et $\neg p \in For^s_{nFITL\text{-}X}(\Sigma)$;*

- *si* $\varphi, \psi \in For^{s}_{nFITL\text{-}X}(\Sigma)$, *alors* $\varphi \wedge \psi \in For^{s}_{nFITL\text{-}X}(\Sigma)$;
- *si* $\varphi, \psi \in For^{s}_{nFITL\text{-}X}(\Sigma)$, *alors* $\varphi \vee \psi \in For^{s}_{nFITL\text{-}X}(\Sigma)$;
- *si* $\pi \in For^{p}_{nFITL\text{-}X}(\Sigma)$, *alors* $E\pi \in For^{s}_{nFITL\text{-}X}(\Sigma)$;
- *si* $\varphi \in For^{s}_{nFITL\text{-}X}(\Sigma)$, *alors* $\varphi \in For^{p}_{nFITL\text{-}X}(\Sigma)$;
- *si* $\pi, \tau \in For^{p}_{nFITL\text{-}X}(\Sigma)$, *alors* $\pi \wedge \tau \in For^{p}_{nFITL\text{-}X}(\Sigma)$;
- *si* $\pi, \tau \in For^{p}_{nFITL\text{-}X}(\Sigma)$, *alors* $\pi \vee \tau \in For^{p}_{nFITL\text{-}X}(\Sigma)$;
- *si* $\pi, \tau \in For^{p}_{nFITL\text{-}X}(\Sigma)$, *alors* $\pi U \tau \in For^{p}_{nFITL\text{-}X}(\Sigma)$;
- *si* $\varphi \in For^{s}_{nFITL\text{-}X}(\Sigma)$, *alors* $\varphi \in For_{nFITL\text{-}X}(\Sigma)$;
- *si* $\pi \in For^{p}_{nFITL\text{-}X}(\Sigma)$, *alors* $\pi \in For_{nFITL\text{-}X}(\Sigma)$.

Le foncteur Mod_{FITL}**.** Nous allons maintenant caractériser la catégorie des modèles associée à une signature.

Définition 3.23 (Les modèles) *Soit* Σ *une signature dans* SIG_{FITL}*. Un* Σ*-modèle est une structure de Kripke sur* Σ*, pas nécessairement totale.*

Nous noterons $|Mod_{FITL}(\Sigma)|$ *la classe de* Σ*-modèles.*

Exemple 3.4 *Reprenons l'exemple 3.1 et supposons qu'une panne soit envisageable et fasse paralyser la circulation dans les deux sens et condamner le passage piéton. Une nouvelle signature pour la gestion de deux feux de carrefour* $\Sigma_{cp} = \{nsvert, eovert, demande\}$ *et un modèle de* Σ_{cp} *est donné par la structure de kripke* K_{cp} *de la figure 3.4.*

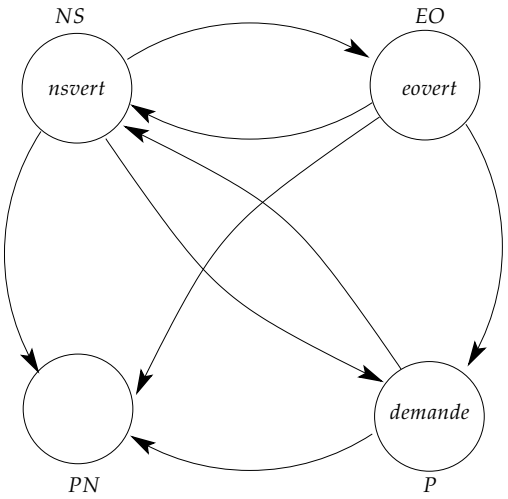


FIG. 3.4 – *La structure de Kripke* K_{cp} *modélisant un carrefour à deux feux avec panne envisageable*

$|Mod_{FITL}(\Sigma)|$ munie des morphismes définis pareillement dans la définition 3.15 définit une catégorie de Σ-modèles. En plus, à chaque morphisme de signatures $\sigma : \Sigma \rightarrow \Sigma'$, on peut associer le foncteur d'oubli $Mod_{FITL}(\sigma) : Mod_{FITL}(\Sigma') \rightarrow Mod_{FITL}(\Sigma)$ qui est défini de la manière que $Mod_{CTL}(\sigma)$ dans la définition 3.16.

Mod_{FITL} définit clairement un foncteur contravariant de SIG_{FITL} dans CAT.

La relation de satisfaction $\models_{FITL}$**.** La satisfaction des formules par les modèles est définie de la manière suivante :

Définition 3.24 (La relation de satisfaction $\models_{FITL}$) *Soient Σ une signature de SIG_{FITL} et $\mathcal{M} = (S, T, L)$ un Σ-modèle. Soient $s \in S$ un état, ρ un chemin, $\varphi \in For^{s}_{FITL}(\Sigma)$ et $\pi \in For^{p}_{FITL}(\Sigma)$ deux Σ-formules.*

La relation de satisfaction de φ par s (resp. π par ρ), notée $\models^{fitl}_{\Sigma}$, est définie inductivement sur la structure de φ (resp. π) de la manière suivante :

- *si $\varphi \in \Sigma$, alors $(\mathcal{M}, s) \models^{fitl}_{\Sigma} \varphi$ ssi $\varphi \in L(s)$;*
- *si φ est de la forme $\neg\psi$, alors $(\mathcal{M}, s) \models^{fitl}_{\Sigma} \neg\psi$ ssi $(\mathcal{M}, s) \not\models^{fitl}_{\Sigma} \psi$;*
- *si φ est de la forme $\psi \wedge \varrho$, alors $(\mathcal{M}, s) \models^{fitl}_{\Sigma} \psi \wedge \varrho$ ssi $(\mathcal{M}, s) \models^{fitl}_{\Sigma} \varphi$ et $(\mathcal{M}, s) \models^{fitl}_{\Sigma} \psi$;*
- *si φ est de la forme $E\tau$, alors $(\mathcal{M}, s) \models^{fitl}_{\Sigma} E\tau$ ssi $\exists \theta \in path(s), (\mathcal{M}, \theta) \models^{fitl}_{\Sigma} \tau$;*
- *si $\pi \in For^{s}_{fitl}(\Sigma)$, alors $(\mathcal{M}, \rho) \models^{fitl}_{\Sigma} \pi$ ssi $(\mathcal{M}, \rho(0)) \models^{fitl}_{\Sigma} \pi$;*
- *si π est de la forme $\neg\tau$, alors $(\mathcal{M}, \rho) \models^{fitl}_{\Sigma} \neg\tau$ ssi $(\mathcal{M}, \rho) \not\models^{fitl}_{\Sigma} \tau$;*
- *si π est de la forme $\tau \wedge \kappa$, alors $(\mathcal{M}, \rho) \models^{fitl}_{\Sigma} \tau \wedge \kappa$ ssi $(\mathcal{M}, \rho) \models^{fitl}_{\Sigma} \tau$ et $(\mathcal{M}, \rho) \models^{fitl}_{\Sigma} \kappa$;*
- *si π est de la forme $\tau U \kappa$, alors $(\mathcal{M}, \rho) \models^{fitl}_{\Sigma} \pi U \tau$ ssi $\exists \theta, \rho \preceq \theta, (\mathcal{M}, \theta) \models^{fitl}_{\Sigma} \kappa$ et $\forall \eta, \rho \preceq \eta \prec \theta, (\mathcal{M}, \eta) \models^{fitl}_{\Sigma} \tau$;*

*Si $(\mathcal{M}, s) \models^{fitl}_{\Sigma} \varphi$, on dit que s **satisfait** φ, et si $(\mathcal{M}, s) \models^{fitl}_{\Sigma} \varphi$ pour tout $s \in S$, on dit que $\mathcal{M}$ satisfait φ, et on note $\mathcal{M} \models^{fitl}_{\Sigma} \varphi$.*

*De même, si $(\mathcal{M}, \rho) \models^{fitl}_{\Sigma} \pi$, on dit que ρ **satisfait** π, et si $(\mathcal{M}, \rho) \models^{fitl}_{\Sigma} \varphi$ pour tout chemin ρ, on dit que $\mathcal{M}$ satisfait π, et on note $\mathcal{M} \models^{fitl}_{\Sigma} \pi$.*

La relation de satisfaction $\models_{FITL}$ est la famille de relations $(\models^{fitl}_{\Sigma})_{\Sigma \in |SIG_{FITL}|}$.

À la différence de la logique CTL*, l'interprétation des formules ne se fait pas sur des chemins infinis, mais plutôt sur des chemins quelconques, finis ou infinis vu que les modèles sont des structures de Kripke quelconques.

Tout comme la logique CTL, les formules $AG\neg\varphi$ et $EF\varphi$ permettent d'exprimer respectivement la sûreté et l'atteignabilité.

Reprenons l'exemple 3.4 et étudions la satisfiabilité de quelques formules.

- $K_{cp} \models^{fitl}_{\Sigma_{cp}} AG\neg(nsvert \wedge eovert)$
- $K_{cp} \models^{fitl}_{\Sigma_{cp}} EF\neg(nsvert \vee oevert)$
- $K_{cp} \models^{fitl}_{\Sigma_{cp}} \neg(eovert \Rightarrow EFeovert)$

Théorème 3.3 *Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signatures de SIG_{FITL}. Pour tout Σ'-modèle $\mathcal{M}' \in |Mod_{FITL}(\Sigma')|$, pour toute Σ-formule $\chi \in For_{FITL}(\Sigma)$ on a :*

$$\mathcal{M}' \models^{fitl}_{\Sigma'} For_{FITL}(\sigma)(\chi) \Leftrightarrow Mod_{FITL}(\sigma)(\mathcal{M}') \models^{fitl}_{\Sigma} \chi$$

Preuve. La preuve est similaire à la preuve du théorème 3.1. □

Ainsi, le quadruplet $(SIG_{FITL}, Mod_{FITL}, For_{FITL}, \models_{FITL})$ est bien une institution.

2.5 Institution pour les systèmes réactifs

Les systèmes réactifs sont des systèmes informatiques, logiciels ou matériels qui réagissent continuellement avec leur environnement à une vitesse déterminée par celui-ci par le biais d'actions. Ils ne calculent pas un résultat mais plutôt maintiennent une interaction avec leur environnement. Ils s'opposent, par cela, aux systèmes classiques, dits *transformationnels*, qui eux disposent au début de leur exécution d'un certain nombre d'entrées sur lesquels s'effectuent des calculs, et délivrent leurs résultats lors de leur terminaison. Les systèmes réactifs s'opposent également aux systèmes *interactifs* qui interagissent continuellement avec leur environnement, mais à leur vitesse propre, tels que les systèmes d'exploitation.

Les systèmes réactifs sont de plus en plus répandus et sont particulièrement utilisés pour le contrôle des systèmes critiques tels que les systèmes de transports, les systèmes nucléaires, les commandes de processus industriels, etc. Ce type de systèmes ne termine pas forcément et peut être concurrent, ce qui amène généralement au non-déterminisme.

Les systèmes réactifs ont été beaucoup étudiés depuis leur introduction par Harel et Pnueli [70]. Les formalismes utilisés pour les spécifier sont le plus souvent à base de systèmes de transitions. Les premiers ont été les TS (Transition System) de Keller [80] pour spécifier les programmes parallèles, où les états représentent les états du système à spécifier, et les transitions le passage d'un état à un autre. À la différence des TS, les LTS (Labeled Transition System) introduits par Milner [95] sont des systèmes de transitions où chaque transition est étiquetée par une action observable de l'environnement ou une action du système pour interagir avec son environnement. Ces systèmes ne font pas la distinction entre les actions reçues par le système de celles envoyées par celui-ci. Cette distinction a été introduite dans les IOA (Input Output Automata) proposés par Lynch et Tuttle [2] et dans IOTS (Input Output Transition Systems) proposé par Tretmans [128]. Ces systèmes de transitions réduisent le comportement d'un système à ses entrées sorties et abstraient donc son comportement interne. Pour prendre en compte le comportement interne des systèmes, une catégorie de systèmes de transitions plus riches que les IOA et les IOTS a été proposée dans [107], les EIOLTS (Extended Input Output Labelled Tansition System), qui sont des systèmes de transitions étiquetées étendues aux données. Ces systèmes de transitions permettent de spécifier, en plus des communications du système via les actions, le comportement interne du système, c'est-à-dire les évolutions possibles de son état au cours du temps.

Dans [7], nous nous sommes inspirés de ces travaux et nous avons proposé un formalisme dédié à la spécification des systèmes réactifs. Tout comme les formalismes mentionnés ci-dessus, notre formalisme se base sur les systèmes de transitions où chaque transition est étiquetée par trois éléments :

- une action de communication entre le système et son environnement ;
- une condition de franchissement de la transition, appelée *garde*, exprimées par une formule de la logique équationnelle ;

– *un ensemble d'effets de bord* où chaque effet de bord est donné par une paire de la forme (t, t'), où t et t' sont des termes clos. Un effet de bord (t, t') indique que le terme t' est affecté au terme t et décrit ainsi une modification caractérisant le changement d'état du système spécifié au travers de la transition en question.

Dans la suite de cette section, nous reprenons ce formalisme et nous introduisons une institution pour les systèmes réactifs.

La catégorie des signatures SIG_{RS}. Pour spécifier abstraitement un système réactif, on se donne une signature de la logique équationnelle au sens de la définition 1.1 et un ensemble de variables sur cette signature. Cette signature permet notamment d'exprimer les gardes et les effets de bord. On se donne aussi un ensemble de symboles pour décrire les actions de communication entre le système et son environnement.

Définition 3.25 (Signature) *Une signature est un triplet (Ω, V, C) où :*
- *$\Omega = (S, F)$ est une signature de la logique équationnelle au sens de la définition 1.1 ;*
- *V est un ensemble de variables sur Ω ;*
- *C est un ensemble de symboles dont les éléments sont appelés **actions**.*

Nous notons $|SIG_{RS}|$ la classe des signatures.

Exemple 3.5 *On s'intéresse ici à une machine appropriée pour changer automatiquement des pièces de monnaie. Ce type de machine est très utilisé dans les grandes surfaces, près des distributeurs automatiques, et dans tous les endroits ou la disponibilité de monnaie est nécessaire. Dans cet exemple, on se restreint à une configuration très simple où la machine n'accepte que des pièces de 2 euros et les échange contre des pièces de 1 euro. La machine est équipée de deux boutons « changer » et « alimenter » qui sont activés alternativement.*
- *Le bouton « changer » activé indique que l'utilisateur peut tenter de changer sa monnaie, alors quand l'utilisateur introduit une pièce de 2 euros et appuie sur le bouton « changer », deux scénarios sont possibles :*
  - *si le nombre de pièces de 1 euros dans la machine est strictement supérieur à 1, alors la machine rend la monnaie à l'utilisateur.*
  - *sinon, la machine rend la pièce introduite par l'utilisateur, puis le bouton « changer » se désactive et le bouton « alimenter » s'active.*
- *Le bouton « alimenter » activé indique donc à l'utilisateur qu'il n'y a pas suffisamment de pièces de 1 euro, alors quand l'utilisateur appuie sur ce bouton, la machine est approvisionnée de 100 pièces de 1 euro, puis le bouton « alimenter » se désactive et le bouton « changer » s'active.*

On donne ici une signature du changeur automatique de monnaie $\Sigma = (\Omega, V, C)$, où $\Omega = (S, F)$, qui va permettre de décrire un tel système :

$S = \{nat, bool, piece\}$

$$
\begin{aligned}
F = \{\ & vrai :\rightarrow bool\,, \\
& faux :\rightarrow bool\,, \\
& 0 :\rightarrow nat\,, \\
& succ : nat \rightarrow nat\,, \\
& \_ + \_ : nat \times nat \rightarrow nat\,, \\
& \_ - \_ : nat \times nat \rightarrow nat\,, \\
& \_ \leq \_ : nat \times nat \rightarrow bool\,, \\
& \_ > \_ : nat \times nat \rightarrow bool\,, \\
& 1e :\rightarrow piece\,, \\
& 2e :\rightarrow piece\,, \\
& nb : piece \rightarrow nat\,, \\
& monnaie : piece \rightarrow vrai\ \}
\end{aligned}
$$

$V = \emptyset$

$C = \{changer, alimenter\}$

La fonction nb détermine en fonction du type de la pièce le nombre de pièces disponible dans la machine. La fonction monnaie indique en fonction du type de la pièce s'il y a de la monnaie ou non : vrai s'il y a de la monnaie disponible et faux sinon.

Dans ce document, les systèmes réactifs que nous considérons manipulent des données sur des signatures différentes. Les morphismes de signatures sont définis de la manière suivante :

Définition 3.26 (Morphisme de signatures) *Soient $\Sigma = (\Omega, V, C)$ et $\Sigma' = (\Omega', V', C')$ deux signatures de SIG_{RS}, avec $\Omega = (S, F)$ et $\Omega' = (S', F')$. Un morphisme de signatures $\sigma : \Sigma \rightarrow \Sigma'$ est défini par :*

- *un morphisme de signatures $\omega : \Omega \rightarrow \Omega'$ au sens de la définition 1.2 ;*
- *une application injective $\sigma_V : V \rightarrow V'$ telle que pour tout $s \in S$, pour tout $x \in V_s$, $\sigma_V(x) \in V'_{\omega_S(s)}$;*
- *une application $\sigma_C : C \rightarrow C'$.*

Si $S \subseteq S'$, $F \subseteq F'$, $V \subseteq V'$ et $C \subseteq C'$, on note $\Sigma \hookrightarrow \Sigma'$ le morphisme d'inclusion.

La classe de signatures $|SIG_{RS}|$ munie des morphismes de signatures définit une catégorie qu'on note SIG_{RS}.

Le foncteur Mod_{RS}**.** Nous allons maintenant caractériser la catégorie des modèles associée à une signature. Un modèle associé à une signature $\Sigma = (\Omega, V, C)$ de SIG_{RS} est défini par un système de transitions dont les états représentent des Ω-algèbres au sens de la définition 1.10.

Définition 3.27 (Les modèles) *Soit $\Sigma = (\Omega, V, C)$ une signature de SIG_{RS}. Un Σ-modèle est un système de transitions $\mathcal{M} = (\mathcal{A}, R)$ où :*

- *$\mathcal{A}$ est une famille indexée par I de Ω-algèbres $(\mathcal{A}^i)_{i \in I}$, où I est un ensemble, telle que pour tout $i, j \in I$ et $s \in S$, $A^i_s = A^j_s$.*
- *R est une famille indexée par C de relations binaires sur I, c'est-à-dire pour tout $a \in C$, $R_a \subseteq I \times I$. R est appelée* ***relation d'accessibilité*** *du Σ-modèle.*

Exemple 3.6 *Pour illustrer la définition 3.27, nous considérons un exemple simple d'accès d'un processus à une section critique. Un processus, qui est dans un état inactif, peut soumettre une demande à un contrôleur de la section critique et attendre jusqu'à*

ce que celui-ci l'autorise à commencer sa section critique. À la fin de la section critique, le contrôleur peut accepter de nouvelles demandes.

Une signature Σ associée à ce système peut être $(\Omega, \emptyset, C)$, avec :
- $\Omega = \{\{bool\}, \{vrai, faux, inactif, attente, critique \rightarrow bool\}\}$
- $C = \{demande, debut, fin\}$

et un modèle sur la signature Σ est :
- $I = \{NC, A, SC\}$
- $\forall i \in I, A^i_{bool} = \{0,1\}, vrai_{\mathcal{A}^i} = 1, faux_{\mathcal{A}^i} = 0, inactif_{\mathcal{A}^i} = \begin{cases} 1 \text{ si } i = NC \\ 0 \text{ sinon} \end{cases}$

  $attente_{\mathcal{A}^i} = \begin{cases} 1 \text{ si } i = A \\ 0 \text{ sinon} \end{cases}$ et $critique_{\mathcal{A}^i} = \begin{cases} 1 \text{ si } i = SC \\ 0 \text{ sinon} \end{cases}$
- $R_{demande} = \{(NC, A)\}$, $R_{debut} = \{(A, SC)\}$ *et* $R_{fin} = \{(SC, NC)\}$

Maintenant, nous avons à définir les morphismes de modèles.

Définition 3.28 (Morphisme de modèles) *Soit $\Sigma = (\Omega, V, C)$ une signature de SIG_{RS} et soient $\mathcal{M}_1 = ((\mathcal{A}^i_1)_{i \in I_1}, R_1)$ et $\mathcal{M}_2 = ((\mathcal{A}^i_2)_{i \in I_2}, R_2)$ deux Σ-modèles.*

Un morphisme de Σ-modèles $h : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ est défini par la donnée d'une application $h_I : I_1 \rightarrow I_2$ satisfaisant les conditions suivantes :
- *$\forall i \in I_1$, il existe un homomorphisme de Ω-algèbres $h^i_I : \mathcal{A}^i_1 \rightarrow \mathcal{A}^{h(i)}_2$ au sens de la définition 1.14,*
- *$\forall i, j \in I_1$, $\forall a \in C$, $(i,j) \in R_{1_a}$ si et seulement si $(h_I(i), h_I(j)) \in R_{2_a}$.*

$|Mod_{RS}(\Sigma)|$ munie des morphismes décrits ci-dessus définit une catégorie de Σ-modèles.

Nous allons maintenant définir le foncteur d'oubli associé à un morphisme de signatures.

Définition 3.29 (Foncteur d'oubli) *Soit $\sigma : \Sigma \rightarrow \Sigma'$ un morphisme de signatures de SIG_{RS}. Le foncteur d'oubli $Mod_{RS}(\sigma) : Mod_{RS}(\Sigma') \rightarrow Mod_{RS}(\Sigma)$ est défini de la manière suivante :*
- *pour tout Σ'-modèle $((\mathcal{A}'^i)_{i \in I}, R')$, $Mod_{RS}(\sigma)((\mathcal{A}'^i)_{i \in I}, R')) = ((\mathcal{A}^i)_{i \in I}, R)$ est défini de la manière suivante :*
  - *pour tout $i \in I$, $\mathcal{A}^i = U_\omega(\mathcal{A}'^i)$ (cf. définition 1.15),*
  - *pour tout $a \in C$, $R_a = \{(i,j) \mid (i,j) \in R_{\sigma_C(a)}\}$.*
- *pour tout morphisme $h' : \mathcal{M}'_1 \rightarrow \mathcal{M}'_2$ de $Mod_{RS}(\Sigma')$ donnée par $h'_I : I'_1 \rightarrow I'_2$, $Mod_{RS}(\sigma)(h') : Mod_{RS}(\sigma)(\mathcal{M}'_1) \rightarrow Mod_{RS}(\sigma)(\mathcal{M}'_2)$ est l'unique morphisme de $Mod_{RS}(\Sigma)$ tel que pour tout $i \in I'_1$, $Mod_{RS}(\sigma)(h')_I(i) = h'_I(i)$.*

Si $\Sigma \hookrightarrow \Sigma'$, on note $Mod_{RS_{|\Sigma}}$ le foncteur d'oubli.

Intuitivement, le foncteur d'oubli associé à un morphisme de signatures $\sigma : \Sigma \rightarrow \Sigma'$ associe à tout Σ'-modèle $\mathcal{M}'$ un Σ-modèle $\mathcal{M}$ ayant comme famille de Σ-algèbres les Σ-algèbres réduites des Σ'-algèbres, états de $\mathcal{M}'$. La famille de relations de transitions du modèle s'obtient en "oubliant" les relations indexées par des actions n'appartenant pas à C.

On peut montrer que Mod_{RS} définit un foncteur contravariant de SIG_{RS} dans CAT.

Le foncteur For_{RS}**.** Les formules sont définies de la manière suivante :

Définition 3.30 (Les formules) *Soit $\Sigma = (\Omega, V, C)$ une signature de SIG_{RS}, avec $\Omega = (S, F)$. L'ensemble des Σ-formules, noté $For_{RS}(\Sigma)$, est l'ensemble défini de la façon suivante :*

- *pour tout $s \in S$, pour tout $t, t' \in T_\Omega(V)_s$, $t = t' \in For_{RS}(\Sigma)$;*
- *pour tout $\varphi \in For_{RS}(\Sigma)$, $\neg\varphi \in For_{RS}(\Sigma)$;*
- *pour tout $\varphi, \psi \in For_{RS}(\Sigma)$, $\varphi \wedge \psi, \varphi \vee \psi, \varphi \Rightarrow \psi \in For_{RS}(\Sigma)$;*
- *pour tout $x \in V$, pour tout $\varphi \in For_{RS}(\Sigma)$, $\forall x\varphi, \exists x\varphi \in For_{RS}(\Sigma)$;*
- *pour tout $\varphi \in For_{RS}(\Sigma)$, pour tout $a \in C$, $\Box_a\varphi \in For_{RS}(\Sigma)$.*

Intuitivement, l'opérateur $\Box_a$ s'interprète par « toujours après l'action a ».

Nous allons maintenant préciser la façon avec laquelle le foncteur For_{RS} prolonge les morphismes aux ensembles de formules.

Définition 3.31 (Prolongement de morphismes de signatures aux ensembles de formules) *Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signatures de SIG_{RS}.*

Le prolongement de σ aux ensembles des formules $For_{RS}(\sigma) : For_{RS}(\Sigma) \to For_{RS}(\Sigma')$ est défini pour toute formule $\varphi \in For_{RS}(\Sigma)$ inductivement sur la structure de φ de la façon suivante :

- *si φ est de la forme $t = t'$, où $t, t' \in T_\Sigma(V)_s$ et $s \in S$, alors $For_{RS}(\sigma)(\varphi)$ vaut $\ddot{\omega}(t) = \ddot{\omega}(t')$ (cf. définition 1.9) ;*
- *si φ est de la forme $\neg\psi$, alors $For_{RS}(\sigma)(\varphi)$ vaut $\neg For_{RS}(\sigma)(\psi)$;*
- *si φ est de la forme $\psi \ominus \varrho$ avec $\ominus \in \{\wedge, \vee, \Rightarrow\}$, alors $For_{RS}(\sigma)(\varphi)$ vaut $For_{RS}(\sigma)(\psi) \ominus For_{RS}(\sigma)(\varrho)$;*
- *si φ est de la forme $Qx\psi$ avec $Q \in \{\forall, \exists\}$, alors $For_{RS}(\sigma)(\varphi)$ vaut $Q\sigma_V(x)For_{RS}(\sigma)(\psi)$;*
- *si φ est de la forme $\Box_c\psi$, alors $For_{RS}(\sigma)(\varphi)$ vaut $\Box_{\sigma_C(c)}For_{RS}(\sigma)(\psi)$.*

La relation de satisfaction $\models_{RS}$. La satisfaction des formules par les modèles est définie de la manière suivante :

Définition 3.32 (La relation de satisfaction $\models_{RS}$) *Soit $\Sigma = (\Omega, V, C)$ une signature de SIG_{RS}. Soient $\mathcal{M} = (\mathcal{A}, R)$ un Σ-modèle, $\nu = (\nu_i : V \to A^i)_{i \in I}$ une famille d'interprétation des variables et $\varphi \in For_{RS}(\Sigma)$ une Σ-formule.*

Pour tout $i \in I$, la satisfaction de φ par le modèle $(\mathcal{A}, R)$ pour la famille d'interprétation ν et l'indice i, notée $\models^{i,\nu}_\Sigma$ est définie inductivement sur la structure de φ de la manière suivante :

- *si φ est de la forme $t = t'$, avec $t, t' \in T_\Sigma(V)_s$ et $s \in S$, alors $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \varphi$ ssi $\mathcal{A}^i \models^{\nu_i}_\Omega \varphi$;*
- *si φ est de la forme $\neg\psi$, alors $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \varphi$ ssi $(\mathcal{A}, R) \not\models^{i,\nu}_\Sigma \psi$;*
- *si φ est de la forme $\psi \wedge \varrho$, alors $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \varphi$ ssi $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \psi$ et $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \varrho$;*
- *si φ est de la forme $\psi \vee \varrho$, alors $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \varphi$ ssi $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \psi$ ou $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \varrho$;*
- *si φ est de la forme $\psi \Rightarrow \varrho$, alors $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \varphi$ ssi $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \psi$ alors $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \varrho$;*
- *si φ est de la forme $\forall x\psi$, alors $(\mathcal{A}, R) \models^{i,\nu}_\Sigma \varphi$ ssi pour toute famille d'interprétation $\nu' = (\nu'_i : V \to A^i)_{i \in I}$ qui vérifie pour tout $i \in I$, $\nu_i(y) = \nu'_i(y)$ pour tout $y \in V \setminus \{x\}$, $(\mathcal{A}, R) \models^{i,\nu'}_\Sigma \psi$;*

– *si* φ *est de la forme* $\exists x\psi$, *alors* $(\mathcal{A}, R) \models^{i,\nu}_{\Sigma} \varphi$ *ssi il existe une famille d'interprétation* $\nu' = (\nu'_i : V \to A^i)_{i\in I}$ *qui vérifie pour tout* $i \in I$, $\nu_i(y) = \nu'_i(y)$ *pour tout* $y \in V \setminus \{x\}$, *telle que* $(\mathcal{A}, R) \models^{i,\nu'}_{\Sigma} \psi$;
– $(\mathcal{A}, R) \models^{i,\nu}_{\Sigma} \Box_a\varphi$ *ssi* $\forall (i,j) \in R_a$, $(\mathcal{A}, R) \models^{j,\nu}_{\Sigma} \varphi$.

Si $(\mathcal{A}, R) \models^{i,\nu}_{\Sigma} \varphi$ *pour toute famille d'interprétation* ν, *on dit que* $\mathcal{M}$ *satisfait* φ *pour l'indice* i, *et on note* $\mathcal{M} \models^{i}_{\Sigma} \varphi$.

Si $(\mathcal{A}, R) \models^{i}_{\Sigma} \varphi$ *pour tout* $i \in I$, *on dit que* $\mathcal{M}$ *satisfait* φ , *et on note* $\mathcal{M} \models_{\Sigma} \varphi$.

La relation de satisfaction $\models_{RS}$ *est la famille de relations* $(\models_{\Sigma})_{\Sigma\in|SIG_{RS}|}$.

Théorème 3.4 *Soit* $\sigma : \Sigma \to \Sigma'$ *un morphisme de signatures de* SIG_{RS}. *Pour tout* Σ'*-modèle* $\mathcal{M}' \in |Mod_{RS}(\Sigma')|$, *pour toute* Σ*-formule* $\varphi \in For_{RS}(\Sigma)$ *on a :*

$$\mathcal{M}' \models_{\Sigma'} For_{RS}(\sigma)(\varphi) \Leftrightarrow Mod_{RS}(\sigma)(\mathcal{M}') \models_{\Sigma} \varphi$$

Preuve.
Soient $((\mathcal{A}'^i)_{i\in I}, R') = \mathcal{M}'$ et $((\mathcal{A}^i)_{i\in I}, R) = Mod_{RS}(\sigma)(\mathcal{M}')$. Par définition, nous avons :
– pour tout $i \in I$, $\mathcal{A}^i = U_\omega(\mathcal{A}'^i)$,
– pour tout $a \in C$, $R_a = \{(i,j) \mid (i,j) \in R_{\sigma_C(a)}\}$.
Montrons que :

$$\forall i \in I, (\mathcal{A}', R') \models^{i}_{\Sigma'} For_{RS}(\sigma)(\varphi) \Leftrightarrow Mod_{RS}(\sigma)((\mathcal{A}', R')) \models^{i}_{\Sigma} \varphi$$

($\Rightarrow$) La preuve se fait par induction structurelle sur les formules de $For_{RS}(\Sigma)$.
– φ est de la forme $t = t'$, avec $t, t' \in T_\Sigma(V)_s$ et $s \in S$: par application directe du théorème 2.6 nous avons,

$$(\mathcal{A}', R') \models^{i}_{\Sigma'} For_{RS}(\sigma)(\varphi) \Rightarrow Mod_{RS}(\sigma)((\mathcal{A}', R')) \models^{i}_{\Sigma} \varphi$$

– φ est de la forme $\Box_a\psi$: soit $(i,j) \in R_a$. Montrons que $Mod_{RS}(\sigma)((\mathcal{A}', R')) \models^{j}_{\Sigma} \psi$. Par définition $(i,j) \in R_{\sigma(a)}$, d'où $(\mathcal{A}', R') \models^{j}_{\Sigma'} For_{RS}(\sigma)(\psi)$. Par l'hypothèse d'induction nous obtenons $Mod_{RS}(\sigma)((\mathcal{A}', R')) \models^{j}_{\Sigma} \psi$.
– pour les autres formes de φ la preuve est facile.

($\Leftarrow$) La preuve se fait par analogie avec le sens $\Rightarrow$.

□

Ainsi, le quadruplet $(SIG_{RS}, Mod_{RS}, For_{RS}, \models_{RS})$ est bien une institution.

3 Exemples de logiques abstraites non institutionnelles

Dans cette section, nous présentons deux logiques abstraites qui ne vérifient pas la condition de satisfaction des institutions.

3.1 Logique équationnelle restreinte aux algèbres finiment engendrées

La logique équationnelle restreinte aux algèbres finiment engendrées consiste à ne considérer que certains modèles de la logique équationnelle : les algèbres finiment engendrées (cf. définition 1.16). Commençons tout d'abord par définir le foncteur *Gen*.

Le foncteur *Gen*. La catégorie des modèles associée à une signature Σ de SIG_{EL} se restreint à la sous-catégorie $Gen(\Sigma)$ introduite dans la définition 1.16.

Nous allons maintenant définir le foncteur d'oubli associé à un morphisme de signatures.

Définition 3.33 (Foncteur d'oubli) *Soient Σ et Σ' deux signatures. Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signatures. On définit le* ***foncteur d'oubli*** *$Gen(\sigma) : Gen(\Sigma') \to Gen(\Sigma)$ comme étant le foncteur :*

- *qui associe à chaque algèbre finiment engendrée $\mathcal{A}' \in |Gen(\Sigma')|$ l'algèbre finiment engendrée $Gen(\sigma)(\mathcal{A}') \in |Gen(\Sigma)|$ telle que l'ensemble sous-jacent $Gen(\sigma)(A')$ est défini par*[6]*,*

$$Gen(\sigma)(A')_s = \{a \in A'_{\sigma_S(s)} \mid \exists t \in (T_\Sigma)_s, t_{\mathcal{A}'} = a\}$$

  et muni pour chaque symbole d'opération $f : s_1 \times \cdots \times s_n \to s \in F$ de l'application $f_{Gen(\sigma)(\mathcal{A}')} = \sigma_F(f)_{\mathcal{A}'}$;
- *qui associe à chaque Σ'-homomorphisme d'algèbres $h' : {\mathcal{A}'}_1 \to {\mathcal{A}'}_2$ le Σ-homomorphisme d'algèbres $Gen(\sigma)(h') : Gen(\sigma)({\mathcal{A}'}_1) \to Gen(\sigma)({\mathcal{A}'}_2)$ défini pour tout $s \in S$ par $Gen(\sigma)(h')^s = {h'}^{\sigma_S(s)}$.*

Si $\Sigma \hookrightarrow \Sigma'$, on note $Gen_{|\Sigma}$ le foncteur d'oubli.

Gen définit clairement un foncteur contravariant de SIG_{EL} dans CAT.

La relation de satisfaction $\models_{FG}$**.** C'est tout simplement la restriction de $\models_{EL}$ sur $(|Gen(\Sigma)| \times For_{EL}(\Sigma))_{\Sigma \in |SIG_{EL}|}$ (cf. définition 1.18).

Théorème 3.5 *Soient Σ et Σ' deux signatures de SIG_{EL}. Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signatures. Pour toute formule $\varphi \in For_{EL}(\Sigma)$ et pour toute $\mathcal{A}' \in |Gen(\Sigma')|$,*

$$Gen(\sigma)(\mathcal{A}') \models_\Sigma \varphi \Rightarrow \mathcal{A}' \models_{\Sigma'} \overline{\sigma}(\varphi)$$

Preuve. La preuve est identique à celle du théorème 2.6 pour le même sens de l'implication. □

Pa ailleurs, l'implication réciproque de celle donnée par le théorème 3.5 n'est pas vraie en général, comme l'indique l'exemple suivant :

Exemple 3.7 *Soient $\Sigma = (\{bool\}, \{vrai \to bool\})$ et $\Sigma' = (\{bool\}, \{vrai \to bool, faux \to bool\})$ deux signatures de SIG_{EL}. Soit $\mathcal{A}'$ une Σ'-algèbre définie par :*

$$A' = \{0,1\}, vrai_{\mathcal{A}'} = 1 \text{ et } faux_{\mathcal{A}'} = 0$$

Par définition, la Σ-algèbre $Gen_{|\Sigma}(\mathcal{A}')$ au travers du morphisme d'inclusion $\Sigma \hookrightarrow \Sigma'$ est donnée par :

$$Gen_{|\Sigma}(A') = \{1\} \text{ et } vrai_{Gen_{|\Sigma}(\mathcal{A}')} = 1$$

[6] Voir section 1.2.2 du chapitre 1 pour la notation $t_{\mathcal{A}}$.

Par conséquent, $Gen_{|\Sigma}(\mathcal{A}')$ valide la formule $\forall x, x = vrai$, mais $\mathcal{A}'$ ne la valide pas.

Ainsi, bien que le quadruplet $(SIG_{EL}, Gen, For_{EL}, \models_{FG})$ soit une logique abstraite, il ne vérifie pas la condition de satisfaction des institutions.

3.2 Une variante de la logique FITL

Dans cette section, nous fournissons un deuxième exemple de logique abstraite qui ne vérifie pas la condition de satisfaction des institutions. Au travers de cet exemple, nous allons aussi détailler nos propos déjà avancés en section 2.4 : nous introduisons la relation DBSB et nous détaillons les résultats de préservation qu'elle engendre.

Étant donnée une structure de Kripke (S, T, L) sur Σ, une relation d'équivalence $\equiv$ sur S est dite *préservant les propositions atomiques* si deux états équivalents satisfont le même ensemble de propositions atomiques, c'est-à-dire :

$$\forall s, t \in S, s \equiv t \Rightarrow (\forall p \in \Sigma, ((S, T, L), s) \models_{\Sigma}^{fitl} p \Leftrightarrow ((S, T, L), t) \models_{\Sigma}^{fitl} p)$$

À partir d'une relation d'équivalence préservant les propositions atomiques, on définit naturellement le quotient d'une structure de Kripke par cette relation d'équivalence.

Définition 3.34 (Quotient d'une structure de Kripke par une relation d'équivalence préservant les propositions atomiques) *Soient Σ une signature de SIG_{FITL} et (S, T, L) un Σ-modèle. Soit $\equiv$ une relation d'équivalence sur S préservant les propositions.*

***Le quotient** de (S, T, L) par $\equiv$, noté $(S, T, L)_{/\equiv}$, est le Σ-modèle $(S_{/\equiv}, T_{/\equiv}, L_{/\equiv})$ tel que,*

- *L'ensemble des états $S_{/\equiv}$ est l'ensemble des classes d'équivalence de $\equiv$;*
- *La relation $T_{/\equiv}$ est définie par,*

$$(\lfloor s \rfloor, \lfloor t \rfloor) \in T_{/\equiv} \Leftrightarrow \exists s' \in \lfloor s \rfloor, \exists t' \in \lfloor t \rfloor, (s', t') \in T$$

- *$L_{/\equiv} : S_{/\equiv} \rightarrow 2^{\Sigma}$ est définie pour tout $\lfloor s \rfloor \in S_{/\equiv}$ par $L_{/\equiv}(\lfloor s \rfloor) = L(s)$.*

La relation DBSB (Divergence Blind Stuttering Bisimulation) est une relation d'équivalence préservant les propositions atomiques. La terminologie s'explique par le fait que deux états équivalents préservent le fragment des formules obtenu en omettant l'opérateur « neXt » (cf. définition 3.21), d'où le terme technique bégaiement (stuttering) [43]. Ce résultat de préservation entre les états équivalents induit une préservation de la satisfaction des formules par une structure de Kripke et son quotient par cette relation.

Définition 3.35 (Relation DBSB) *Soit Σ une signature de SIG_{FITL} et soit (S, T, L) un Σ-modèle. Une relation $\simeq \subseteq S \times S$ est **une relation DBSB** si et seulement si pour tout $s, t \in S$ tels que $s \simeq t$,*

1. *$L(s) = L(t)$;*
2. *Si $(s, s') \in T$ alors $\exists (t_0, t_1, \ldots, t_n) \in \rho(t_0)$ dans (S, T, L), $n \geq 0$, tel que $(t_0 = t) \wedge (\forall i, 0 \leq i < n, s \simeq t_i) \wedge s' \simeq t_n$;*
3. *$t \simeq s$, i.e. $\simeq$ est symétrique.*

Théorème 3.6 *[43] Soient Σ une signature de SIG_{FITL}, (S,T,L) un Σ-modèle, et s et t deux états de S. Soit $\simeq$ une relation DBSB sur S. Les expressions suivantes sont équivalentes :*

1. $s \simeq t$
2. *pour toute formule χ de $For_{FITL\text{-}X}(\Sigma)$, $((S,T,L),s) \models^{fitl}_{\Sigma} \chi$ ssi $((S,T,L),t) \models^{fitl}_{\Sigma} \chi$*

Théorème 3.7 *[43] Soit Σ une signature de SIG_{FITL} et soit (S,T,L) un Σ-modèle. Soit $\simeq$ une relation DBSB sur S. Pour tout $\varphi \in For_{FITL\text{-}X}(\Sigma)$,*

$$(S,T,L) \models^{fitl}_{\Sigma} \varphi \Leftrightarrow (S,T,L)_{/\equiv} \models^{fitl}_{\Sigma} \varphi$$

Notation 3.1 *Soit Σ une signature de SIG_{FITL} et soit (S,T,L) un Σ-modèle. On note F^S_{dbs} la famille des relations DBSB sur S et par $\simeq^S_{dbs} = \bigcup_{\simeq \subseteq F^S_{dbs}} \simeq$, l'union de toutes les relations DBSB sur S.*

Théorème 3.8 *Soit Σ une signature de SIG_{FITL} et soit (S,T,L) un Σ-modèle. La relation $\simeq^S_{dbs}$ est une DBSB sur S.*

Preuve. Soient $s,t \in S$ tels que $s \simeq^S_{dbs} t$, donc il existe $\simeq \subseteq F^S_{dbs}$ tel que $s \simeq t$. Ainsi, $L(s) = L(t)$. Puisque $\simeq \subseteq F^S_{dbs}$, nous obtenons aussi :
- Si $(s,s') \in T$ alors $\exists (t_0,t_1,\ldots,t_n) \in \rho(t_0)$ dans (S,T,L), $n \geq 0$, tel que $(t_0 = t) \wedge (\forall i, 0 \leq i < n, s \simeq^S_{dbs} t_i) \wedge s' \simeq^S_{dbs} t_n$;
- $t \simeq^S_{dbs} s$.

Ainsi $\simeq^S_{dbs}$ est une DBSB sur S. □

Nous allons maintenant introduire une logique abstraite $(SIG_{FITL}, For_{FITL}, Mod'_{FITL}, \models_{FITL})$ qui ne vérifie pas la condition de satisfaction des institutions. Pour définir cette logique, il suffit de définir le foncteur Mod'_{FITL}.

Le foncteur Mod'_{FITL}. Pour une signature Σ, la catégorie des modèles $Mod'_{FITL}(\Sigma)$ coïncide avec la catégorie $Mod_{FITL}(\Sigma)$.

Nous allons maintenant définir le foncteur d'oubli associé à un morphisme de signatures.

Définition 3.36 (Foncteur d'oubli) *Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signatures de SIG_{FITL}. Le foncteur d'oubli $Mod'_{FITL}(\sigma) : Mod'_{FITL}(\Sigma') \to Mod'_{FITL}(\Sigma)$ est défini de la manière suivante :*
- *pour tout Σ'-modèle (S',T',L'), $Mod'_{FITL}(\sigma)((S',T',L')) = (S'_{/\simeq^{S'}_{dbs}}, T'_{/\simeq^{S'}_{dbs}}, L)$, où pour tout $s \in S'_{/\simeq^{S'}_{dbs}}$, $L(s) = \{p \in \Sigma \mid \sigma(p) \in L'_{/\simeq^{S'}_{dbs}}(s)\}$.*
- *pour tout morphisme $m' : \mathcal{M}'_1 \to \mathcal{M}'_2$ de $Mod'_{FITL}(\Sigma')$, $Mod'_{FITL}(\sigma)(m') : Mod'_{FITL}(\sigma)(\mathcal{M}'_1) \to Mod'_{FITL}(\sigma)(\mathcal{M}'_2)$ est l'unique morphisme de $Mod'_{FITL}(\Sigma)$ tel que si l'on note $\mathcal{M}'_1 = (S'_1,T'_1,L'_1)$ et $Mod'_{FITL}(\sigma)(\mathcal{M}'_1) = (S'_{1\,/\simeq^{S'}_{dbs}}, T'_{1\,/\simeq^{S'}_{dbs}}, L_1)$, alors pour tout $s \in S'_1$, $Mod'_{FITL}(\sigma)(m')_S(\lfloor s \rfloor) = \lfloor m'_S(s) \rfloor$.*

Si $\Sigma \hookrightarrow \Sigma'$, on note $Mod'_{FITL_{|\Sigma}}$ le foncteur d'oubli.

Mod'_{FITL} définit un foncteur contravariant de SIG_{FITL} dans CAT.

Du théorème 3.7 découle le corollaire suivant :

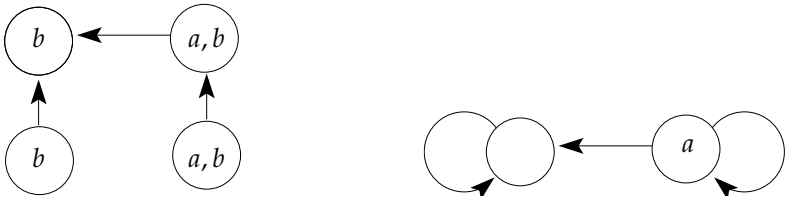


FIG. 3.5 – *Les structures de Kripke K′ (à gauche) et K (à droite).*

Corollaire 3.1 *Soit $\sigma : \Sigma \to \Sigma'$ un morphisme de signatures de SIG_{FITL}. Pour tout Σ'-modèle $\mathcal{M}' \in |Mod'_{FITL}(\Sigma')|$, pour toute formule φ de $For_{FITL\text{-}X}(\Sigma)$ on a :*

$$\mathcal{M}' \models^{fitl}_{\Sigma'} For_{FITL}(\sigma)(\varphi) \Leftrightarrow Mod'_{FITL}(\sigma)(\mathcal{M}') \models^{fitl}_{\Sigma} \varphi$$

Par ailleurs, ce corollaire est mis en cause dès que la formule φ n'est pas dans $For_{FITL\text{-}X}(\Sigma)$. En effet, considérons l'exemple suivant :

Exemple 3.8 *Soient $\Sigma = \{a\}$ et $\Sigma' = \{a, b\}$ deux signatures de SIG_{FITL} et soit le morphisme d'inclusion $\Sigma \hookrightarrow \Sigma'$. La structure de Kripke K' sur $\{a, b\}$, donnée par la figure 3.5, est un Σ'-modèle. $Mod'_{FITL_{|\Sigma}}(K')$ est la structure de Kripke K sur $\{a\}$ donnée par la même figure.*

On voit bien que :

$$K' \models^{fitl}_{\Sigma'} \neg(a \Rightarrow EXa)$$

mais,

$$K \models^{fitl}_{\Sigma} a \Rightarrow EXa$$

Ainsi, la logique abstraite $(SIG_{FITL}, Mod'_{FITL}, For_{FITL}, \models_{FITL})$ n'est pas une institution vu que la condition de satisfaction n'est pas vérifiée.

4 Spécifications abstraites structurées : modularité et complexité

Sommaire

Dans ce chapitre, nous présenterons, en s'appuyant sur le fondement des résultats avancés dans la partie I des spécifications algébriques, un cadre général pour les spécifications abstraites. L'intérêt d'un tel cadre est qu'il est :

- suffisamment général pour pouvoir exprimer les aspects de la structuration des formalismes de spécifications standards et être instancié par des formalismes de spécifications standards.
- suffisamment flexible pour permettre de généraliser les notions de modularité et de complexité que nous avons introduites dans le chapitre 2 dans le cadre des spécifications algébriques, et les étendre ainsi à d'autres formes de spécifications telles que les spécifications définies par des systèmes de transitions.
- suffisamment générique pour permettre d'unifier, de caractériser et de distinguer, dans le même cadre, les spécifications modulaires des spécifications complexes.

Pour obtenir les points ci-dessus, nous proposons alors de présenter notre cadre général de spécifications abstraites au dessus des institutions. Ceci permettra alors d'être indépendant des formalismes standards de spécifications et leurs logiques sous-jacentes.

Ce chapitre se découpe de la façon suivante : en section 1, nous présenterons notre cadre général de spécifications abstraites. Pour structurer les spécifications, nous définirons en section 2, au travers des notions de diagrammes et de colimites de la théorie des catégories[1], un connecteur architectural générique permettant de combiner des spécifications entre elles pour en construire d'autres plus larges. Enfin, en section 3, nous allons généraliser dans le cadre de ce connecteur générique les notions de modularité et de complexité que nous avons définies dans le cadre des spécifications algébriques au chapitre 2, et nous fournirons des conditions nécessaires et/ou suffisantes pour qu'un connecteur soit modulaire. Nous illustrerons toutes les notions définies dans ce chapitre au travers de deux exemples : les spécifications axiomatiques et les spécifications des systèmes réactifs.

[1] Voir la section 4 de l'annexe A pour la notion de diagrammes et de colimites.

1 Les spécifications abstraites

La spécification est reconnue comme une étape cruciale dans le développement du logiciel qui consiste en l'élaboration d'un document d'exigences et de références de correction. Ce document représente le cahier des charges du système que l'on veut développer et doit décrire les propriétés attendues de ce système. La spécification est consignée par écrit en utilisant soit le langage naturel, soit un formalisme de spécifications. D'une part, le langage naturel est souple, universel et répandu, d'autre part, les exigences en langage naturel sont réputées pour être incomplètes, incohérente et ambiguës par essence. Pour ces raisons, les spécifications formelles ont été proposées pour pallier ces déficiences. Leur principal intérêt est qu'elles sont fondées sur des bases mathématiques qui permettent de spécifier sans ambiguïté le comportement attendu du système [3, 93] et de fournir un fondement logique exploitable par des techniques de vérification formelle, de preuve ou de raffinement.

Classiquement, une spécification formelle est basée sur *une syntaxe* et *une sémantique* [68, 131] :

- la syntaxe fournit, sous forme symbolique, ce que le spécifieur est autorisé à écrire et définit la façon d'énoncer les comportements licites du système.
- la sémantique définit un ensemble de modèles, appelés *réalisations*. Ces modèles peuvent être vus comme des abstractions mathématiques du système à spécifier et dont on veut établir la correction. La correction est rendue possible par la donnée d'un ensemble de règles permettant de prouver si les comportements attendus de la spécification sont bien ceux réalisés par le(s) modèle(s) d'une spécification.

Cette vue classique des fondements de base des spécifications formelles amène à distinguer deux grandes familles de spécifications formelles : les spécifications *orientées modèles* et celles *orientées propriétés*.

Les spécifications orientées modèles. Le comportement du système est défini en construisant un unique modèle à partir des concepts mathématiques bien connus tels que les tuples, les relations, les fonctions, les ensembles, les séquences. Pour spécifier les programmes séquentiels, on trouve par exemple VDM [75], Z [117] et B [4]. Pour spécifier les systèmes concurrents et distribués, on trouve par exemple les réseaux de Petri [102], les CCS de Milner (Calculus of Communicating Systems) [95], les CSP de Hoare (Communicating Sequential Processes) [73], Unity [34], les IOA (Input Output Automatas) [2], Raise qui combine VDM et CSP [98] et Community [50].

Les spécifications orientées propriétés ou axiomatiques. Le comportement du système est défini par un ensemble de propriétés logiques appelées *axiomes*. Les spécifications orientées propriétés se définissent ainsi au dessus d'une logique donnée et les axiomes sont des formules bien construites sur cette logique. Généralement, l'ensemble des modèles d'une telle spécification est obtenu en ne conservant que les modèles qui satisfont l'ensemble des axiomes. Un système est correct par rapport à sa spé-

cification si son abstraction mathématique correspond à un modèle de la spécification.

Au départ, les principaux représentants de cette famille étaient les spécifications algébriques, puis Goguen et Burstall ont proposé d'utiliser les institutions qui offrent un cadre générique pour étudier les formalismes de spécifications orientées propriétés indépendamment de leurs logiques sous-jacentes. Dans ce cadre, étant donnée une institution $\mathcal{I} = (SIG, For, Mod, \models)$, une spécification est donnée sous la forme (Σ, Ax) où Σ est une signature de SIG et Ax est un ensemble de formules sur Σ (i.e. $Ax \subseteq For(\Sigma)$).

Prenant acte de la multiplicité des formalismes de spécifications, nous proposons dans la suite un cadre général de spécifications abstraites suffisamment général pour prendre en compte la plus large famille de formalismes de spécifications qu'ils soient orientés modèles ou orientés propriétés.

Comme nous l'avons vu plus haut, l'introduction d'un formalisme de spécifications formelles se fait par la donnée d'une syntaxe et d'une sémantique. Bien que la définition de ces différents éléments soi propre à chaque formalisme de spécifications, au même titre que tout formalisme logique, le volet syntaxique repose sur la donnée d'une signature qui représente l'interface du problème traité. De plus, un système informatique est amené à voir son comportement évoluer tout au long de son cycle de vie. Cette évolution se caractérise souvent en premier lieu par une modification de son interface (par ajout de nouvelles fonctionnalités par exemple) puis par une modification de son comportement. Pour ces raisons, la notion de signatures ainsi qu'un moyen de comparer ces signatures au niveau syntaxique puis au niveau sémantique doivent apparaître dans le cadre d'une description générale d'un formalisme de spécifications. Or, comme nous l'avons expliqué dans le chapitre 3, toutes ces caractéristiques se retrouvent dans le cadre général des institutions. Pour ces raisons, nous reprenons l'idée de Goguen et Burstall [65, 63] d'utiliser les institutions pour définir un cadre général de spécifications abstraites mais permettant de prendre en compte à la fois la plus large famille de formalismes de spécifications et non pas seulement les spécifications orientées propriétés.

Dans la suite, nous introduirons la définition de notre cadre général de spécifications abstraites au-dessus d'une institution notée $\mathcal{I} = (SIG, For, Mod, \models)$ que nous supposons fixée. Nous introduirons aussi un ensemble des notions qui s'y attachent et qui nous serviront dans la suite.

1.1 Définitions et exemples

Définition 4.1 (Langage de spécification) *Un **langage de spécification** $\mathcal{SL}$ sur $\mathcal{I}$ est la donnée d'un tuple $(SPEC, Sig, Real)$ où :*

- *$SPEC$ est une catégorie dont les objets sont appelés des **spécifications abstraites** ;*
- *$Sig : SPEC \rightarrow SIG$ est un foncteur qui, à chaque spécification $Sp \in |SPEC|$ associe une signature $\Sigma \in |SIG|$, et à chaque*

morphisme $\sigma : Sp \rightarrow Sp'$ *associe un morphisme de signatures* $Sig(\sigma) : Sig(Sp) \rightarrow Sig(Sp')$;
– $Real : |SPEC| \rightarrow |CAT|$ *est une application telle que pour tout* $Sp \in |SPEC|$, $Real(Sp)$ *est une sous-catégorie pleine de* $Mod(Sig(Sp))$. *Les objets de* $Real(Sp)$ *sont appelés les* ***réalisations*** *de* Sp.

Ainsi, les objets de $SPEC$ sont les spécifications, Sig permet d'associer à chaque spécification la signature sous-jacente, c'est-à-dire les éléments de base nécessaires à la description de la spécification, et $Real$ permet d'associer à à chaque spécification sa classe de réalisations possibles.

Nous verrons au travers de l'exemple de la section 1.1.1 qui instancie la définition 4.1 aux spécifications axiomatiques au dessus d'une institution quelconque, puis qui l'instancie particulièrement aux spécifications algébriques, que dans ce dernier cas $Real$ ne peut pas être défini comme un foncteur.

1.1.1 Exemple : Spécifications axiomatiques

1. Un langage de spécifications que l'on peut associer aux spécifications axiomatiques sur $\mathcal{I}$ est le triplet $(SPEC, Sig, Real)$ où :
   – $SPEC$ est la catégorie dont les objets sont les spécifications de la forme (Σ, Ax), où Σ est une signature de SIG et Ax est un ensemble de formules dans $For(\Sigma)$. Une application $\sigma : (\Sigma, Ax) \rightarrow (\Sigma', Ax')$ est un morphisme de $SPEC$ si et seulement s'il existe un morphisme de signatures $\Omega : \Sigma \rightarrow \Sigma'$ tel que $For(\Omega)(Ax) \subseteq Ax'$.
   – $Sig : SPEC \rightarrow SIG$ est le foncteur qui à chaque spécification (Σ, Ax) associe la signature Σ et à chaque morphisme de spécifications $\sigma : (\Sigma, Ax) \rightarrow (\Sigma', Ax')$ associe le morphisme de signatures $\Omega : \Sigma \rightarrow \Sigma'$.
   – $Real : |SPEC| \rightarrow |CAT|$ est l'application définie pour toute spécification $(\Sigma, Ax) \in |SPEC|$ par $Real((\Sigma, Ax))$, la sous-catégorie pleine de $Mod(\Sigma)$ telle que,

$$|Real((\Sigma, Ax))| = \{\mathcal{M} \in |Mod(\Sigma)| \mid \forall \varphi \in Ax, \mathcal{M} \models_\Sigma \varphi\}$$

2. Le langage de spécifications ci-dessus ayant été défini indépendamment d'une spécification donnée, il peut être instancié à toutes les institutions entre autres à $(SIG_{EL}, Alg, For_{EL}, \models_{EL})$ définie pour la logique équationnelle (cf. section 2.1 du chapitre 3). Il est aussi possible d'associer, aux spécifications algébriques, le langage de spécifications sur $(SIG_{EL}, Alg, For_{EL}, \models_{EL})$ suivant :
   – $SPEC_{EL}$ est la catégorie dont les objets sont les spécifications algébriques de la forme (Σ, Ax), où Σ est une signature de SIG_{EL} et Ax est un ensemble de formules dans $For_{EL}(\Sigma)$ (cf. définition 1.20). Une application $\sigma : (\Sigma, Ax) \rightarrow (\Sigma', Ax')$ est un morphisme de $SPEC_{EL}$ si et seulement si (Σ', Ax') est un enrichissement de (Σ, Ax) (cf. définition 2.8).
   – $Sig_{EL} : SPEC_{EL} \rightarrow SIG_{EL}$ est le foncteur qui à chaque spécification (Σ, Ax) associe la signature Σ et à chaque morphisme de spécifications $\sigma : (\Sigma, Ax) \rightarrow (\Sigma', Ax')$ associe le morphisme de signatures $\Omega : \Sigma \rightarrow \Sigma'$.

– $Real_{EL} : |SPEC_{EL}| \rightarrow |CAT|$ est l'application qui, à toute spécification (Σ, Ax), associe $Alg(Sp)$, dont la classe des algèbres valident Ax.

Nous avons vu dans le chapitre 1 que d'autres sémantiques pour les spécifications algébriques sont possibles. On peut par exemple, quand on est intéressé par des puissances de preuves supplémentaires telles que l'induction structurelle ou des résultats de décision, restreindre la classe de modèles (réalisations) aux modèles finiment engendrés. Mais dans ce cas là que le foncteur d'oubli défini sur les modèles de signatures (cf. définition 3.33 du chapitre 3) ne peut pas se restreindre aux modèles des spécifications (cf. théorème 3.5 du même chapitre). Alors, dans le cadre des spécifications algébriques, si $Real_{EL}(Sp) = Gen(Sp)$, $Real_{EL}$ n'est pas un foncteur.

1.1.2 Exemple : Spécifications des systèmes réactifs

Dans cette section, nous introduisons un langage de spécifications pour les systèmes réactifs sur $(SIG_{RS}, Mod_{RS}, For_{RS}, \models_{RS})$ (cf. section 2.5 du chapitre 3).

Comme nous l'avons expliqué en section 2.5 du chapitre 3, la spécification d'un système réactif est donnée par un système de transitions étiquetées par des actions de communication avec l'environnement. Ces actions peuvent être conditionnées par une formule de la logique équationnelle appelée garde. Elles peuvent également être suivies d'une modification de l'état du système exprimée par un ensemble d'effets de bord.

Avant de définir la catégorie des spécifications pour les systèmes réactifs, nous définissons formellement cette notion d'*effet de bord* et la sémantique qu'engendre un ensemble d'effets de bord.

Définition 4.2 (Effet de bord) *Soit $\Sigma = (\Omega, V, C)$ une signature de SIG_{RS}, où $\Omega = (S, F)$.*
*Un **effet de bord** sur Σ est une paire $(t, t') \in (T_\Omega)_s \times (T_\Omega)_s$, où $s \in S$, qu'on note $t \mapsto t'$. On note $\mathcal{EB}(\Sigma)$ l'ensemble des effets de bord sur Σ.*

Les effets de bord permettent de modifier l'état du système en modifiant les valeurs d'un ensemble de termes. Nous donnons dans la suite une sémantique pour un ensemble d'effets de bord.

Définition 4.3 (Modification d'un effet de bord) *Soit $t \mapsto t' \in \mathcal{EB}(\Sigma)$. t est nécessairement de la forme $f(t_1, \ldots, t_n)$. Soit $\mathcal{A}$ une Ω-algèbre. La **modification** engendrée par $t \mapsto t'$ sur $\mathcal{A}$ est le triplet $(f, (t_{1_\mathcal{A}}, \ldots, t_{n_\mathcal{A}}), t'_\mathcal{A})$.*

Définition 4.4 (Ensemble d'effets de bord cohérent) *Soit Δ un ensemble d'effets de bord. Δ est incohérent s'il existe une Ω-algèbre $\mathcal{A}$ et deux effets de bord de Δ qui engendrent deux modifications $(f, (a_1, \ldots, a_n), a)$ et $(f, (a_1, \ldots, a_n), b)$ sur $\mathcal{A}$ avec $a \neq b$.*

Cette condition sur l'ensemble d'effets de bord permet d'exclure le cas où une même fonction f appliquée au même tuple rendrait différentes valeurs et donc rendrait inconsistant la définition de f. Dans la suite, on suppose que tous les effets de bords qu'on définit sont cohérents.

Définition 4.5 (Sémantique des effets de bord) *Soit $\Sigma = (\Omega, V, C)$ une signature de SIG_{RS}, où $\Omega = (S, F)$, et soit $\Delta \subseteq \mathcal{EB}(\Sigma)$ un ensemble d'effets de bord sur Σ. Soit $\mathcal{A}$ une Ω-algèbre. **La transformée** de $\mathcal{A}$ par Δ, qu'on note $\tau_\Delta(\mathcal{A})$, est la Ω-algèbre $\mathcal{B}$ définie de la façon suivante :*

- *pour tout $s \in S$, $B_s = A_s$;*
- *pour tout $f : s_1 \times \cdots \times s_n \to s \in F$, pour tout $(a_1, \ldots, a_n)$ appartenant à $A_{s_1} \times \cdots \times A_{s_n}$,*
  - *s'il existe $a \in A_s$ tel que $(f, (a_1, \ldots, a_n), a)$ est engendré par un effet de bord de Δ, alors $f_\mathcal{B}(a_1, \ldots, a_n) = a$.*
  - *sinon $f_\mathcal{B}(a_1, \ldots, a_n) = f_\mathcal{A}(a_1, \ldots, a_n)$.*

La catégorie des spécifications $SPEC_{RS}$. Une spécification d'un système réactif est la donnée d'un système de transitions étiquetées. Chaque transition est étiquetée par une action conditionnée par une formule de la logique équationnelle, et par un ensemble d'effets de bord.

Définition 4.6 (Spécification d'un système réactif) *Soit $\Sigma = (\Omega, V, C)$ une signature de SIG_{RS}. Une spécification sur $\Sigma = (\Omega, V, C)$ est un système de transitions $\mathcal{S} = (Q, T)$ où :*

- *Q est un ensemble dont les éléments sont appelés **états**,*
- *$T \subseteq Q \times C \times For_{EL}(\Omega) \times 2^{\mathcal{EB}(\Sigma)} \times Q$. Les éléments de T sont les transitions du système.*

Une transition est donc un quintuplet $(q, \varphi, a, \Delta, q')$ où q et q' sont respectivement les états source et cible de la transition, φ est la condition de franchissement, a est une action de communication, et Δ est un ensemble d'effets de bord sur Σ. Une transition $(q, \varphi, a, \Delta, q')$ est représentée graphiquement de la manière suivante :

$$q \overset{a,\varphi,\Delta}{\longrightarrow} q'$$

Exemple 4.1 *On reprend ici le changeur automatique de monnaie de l'exemple 3.5 du chapitre 3. Une spécification de ce système est donnée par le système de transitions $\mathcal{S} = (Q, T)$ où :*

- $Q = \{1, 2\}$
- $T = \{ (1, changer, nb(1e) > succ(0) = vrai, \{nb(1e) \mapsto nb(1e) - succ^2(0), nb(2e) \mapsto nb(2e) + succ(0)\}, 1)$,
  $(1, changer, nb(1e) \leq succ(0) = vrai, 2)$,
  $(2, alimenter, \{nb(1e) \mapsto nb(1e) + succ^{100}(0)\}, 1)$ $\}$

Une représentation[2] graphique de cette spécification est donnée par la figure 4.1.

La catégorie $SPEC_{RS}$ est alors la catégorie dont les objets sont les spécifications des systèmes réactifs au sens de la définition 4.6 et dont les morphismes sont définis de la façon suivante.

Définition 4.7 (Morphisme de $SPEC_{RS}$) *Soient $\mathcal{S} = (Q, T)$ et $\mathcal{S}' = (Q', T')$ deux spécifications respectivement sur les signatures $\Sigma = (\Omega, V, C)$ et $\Sigma' = (\Omega', V', C')$ de SIG_{RS}. Un morphisme de spécifications $spec : \mathcal{S} \to \mathcal{S}'$ est défini par :*

- *une application $\alpha : Q \to Q'$;*

[2] Nous notons f^n l'application de la fonction f n fois.

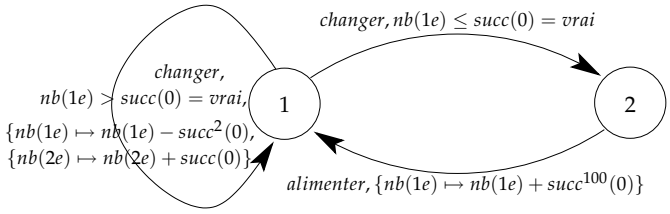


FIG. 4.1 – *Spécification d'un changeur automatique de monnaie.*

– *un morphisme de signatures* $\sigma : \Sigma \to \Sigma'$ *au sens de la définition 3.26 tels que pour toute transition* $(q, a, \varphi, \Delta, q') \in T$, *la transition* $(\alpha(q), \sigma_C(a), For_{EL}(\omega)(\varphi), \ddot{\omega}(\Delta), \alpha(q'))$ *appartient*[3] *à* T'.

Le foncteur Sig_{RS}**.** Le foncteur $Sig_{RS} : SPEC_{RS} \to SIG_{RS}$ associe, à chaque spécification $\mathcal{S}$ sur Σ, la signature Σ, et à chaque morphisme $spec : \mathcal{S} \to \mathcal{S}'$ de $SPEC_{RS}$ le morphisme de signatures $\sigma : Sig_{RS}(\mathcal{S}) \to Sig_{RS}(\mathcal{S}')$.

L'application $Real_{RS}$**.** L'application $Real_{RS} : |SPEC_{RS}| \to |CAT|$ est telle que pour toute spécification $\mathcal{S} = (Q, T)$ sur Σ, où $\Sigma = (\Omega, V, C)$, $Real_{RS}(\mathcal{S})$ est la sous-catégorie pleine de $Mod_{RS}(\Sigma)$ telle que pour tout $((\mathcal{A}^i)_{i\in I}, R) \in |Real_{RS}(\mathcal{S})|$, l'ensemble I coïncide avec Q et $((\mathcal{A}^i)_{i\in I}, R)$ satisfait les conditions suivantes :

(a) $((q, a, \varphi, \Delta, q') \in T \wedge \mathcal{A}^q \vDash_\Omega \varphi \wedge \mathcal{A}^{q'} = \tau_\Delta(\mathcal{A}^q)) \Rightarrow (q, q') \in R_a$

(b) $((q, q') \in R_a) \Rightarrow (\exists (q, a, \varphi, \Delta, q') \in T, \mathcal{A}^q \vDash_\Omega \varphi \wedge \mathcal{A}^{q'} = \tau_\Delta(\mathcal{A}^q))$

1.2 Propriétés remarquables des spécifications abstraites

Dans le cas général, *Real* ne peut pas être défini comme un foncteur (cf. l'exemple des spécifications algébriques donné en section 1.1.1). Cependant, cette propriété pour *Real* d'être un foncteur, comme nous le verrons par la suite, sera importante comme condition pour assurer la modularité des systèmes. Quand cette propriété sera vérifiée, on dit que *Real* est *compatible*.

Définition 4.8 (Compatible) *Soit* $\mathcal{SL} = (SPEC, Sig, Real)$ *un langage de spécification sur* $\mathcal{I}$. *Real est* ***compatible*** *si et seulement si Real est un foncteur contravariant de* $SPEC$ *dans* CAT.

Les formules qui sont validées par toutes les réalisations $Real(Sp)$ d'une spécification Sp seront d'une importance particulière dans la suite de cette thèse. Elles représentent concrètement l'ensemble des propriétés que le système spécifié valide dès lors que cet ensemble est validé par toutes les réalisations du système et c'est sur cet ensemble de formules qu'on va se baser pour définir la notion de modularité et de complexité des systèmes.

Définition 4.9 (Conséquences sémantiques d'une spécification) *Soit* $\mathcal{SL} = (SPEC, Sig, Real)$ *un langage de spécification sur* $\mathcal{I}$ *et soit* Sp *une spécification de* $SPEC$. *L'ensemble*

[3]Pour $\Delta = \{t_1 \mapsto t'_1, \ldots, t_k \mapsto t'_k\}$, $\ddot{\omega}(\Delta)$ dénote l'ensemble d'effets de bord $\{\ddot{\omega}(t_1) \mapsto \ddot{\omega}(t'_1), \ldots, \ddot{\omega}(t_k) \mapsto \ddot{\omega}(t'_k)\}$ sur Σ', avec $\ddot{\omega}$ est le prolongement de ω aux termes au sens de la définition 1.9.

des conséquences sémantiques *de Sp, noté $Sp^\bullet$, est l'ensemble des formules validées par $Real(Sp)$.*

$$Sp^\bullet = \{\varphi \in For(Sig(Sp)) | \forall \mathcal{R} \in |Real(Sp)|, \mathcal{R} \models_{Sig(Sp)} \varphi\}$$

Intuitivement, on pourrait penser que la classe des modèles de l'ensemble des conséquences sémantiques coïncide avec la classe des réalisations de Sp, c'est-à-dire

$$Mod(Sp^\bullet) = Real(Sp)$$

Si c'est le cas et si tout morphisme de spécifications $\sigma : Sp \to Sp'$ préserve les conséquences sémantiques, c'est-à-dire $For(Sig(\sigma))(Sp^\bullet) \subseteq Sp'^\bullet$, alors $Real$ est le foncteur tel que $Real(\sigma) = Mod(Sig(\sigma))$. En effet, en utilisant la condition de satisfaction, on peut montrer que,

$$\forall \mathcal{R}' \in |Real(Sp')|, Mod(Sig(\sigma))(\mathcal{R}') \in |Real(Sp)|$$

Dans la suite, nous introduisons trois propriétés qui nous seront utiles pour établir nos résultats sur la modularité des systèmes.

Définition 4.10 (Définissable par spécification) *Soit $\mathcal{SL} = (SPEC, Sig, Real)$ un langage de spécification sur $\mathcal{I}$. Soient Sp une spécification de $SPEC$ et $T \subseteq For(Sig(Sp))$ une théorie au dessus de $Sig(Sp)$. T est* ***définissable*** *par Sp si et seulement si $T = Sp^\bullet$.*

Définition 4.11 (Morphisme de spécifications préservant les conséquences sémantiques) *Soit $\mathcal{SL} = (SPEC, Sig, Real)$ un langage de spécification sur $\mathcal{I}$ et soit $\sigma : Sp \to Sp'$ un morphisme de $SPEC$. Si σ satisfait $For(Sig(\sigma))(Sp^\bullet) \subseteq Sp'^\bullet$, alors σ est appelé* ***morphisme de spécifications préservant les conséquences sémantiques****.*

Définition 4.12 (Morphisme de spécifications libéral) *Soit $\mathcal{SL} = (SPEC, Sig, Real)$ un langage de spécification sur $\mathcal{I}$ tel que $Real$ est compatible.*

Un morphisme de spécifications $\sigma : Sp \to Sp'$ est dit ***libéral*** *si et seulement si $Real(\sigma) : Real(Sp') \to Real(Sp)$ admet un foncteur adjoint à gauche $\mathcal{F}(\sigma) : Real(Sp) \to Real(Sp')$.*

Exemple 4.2 *Reprenons le langage de spécifications $(SPEC_{EL}, Sig_{EL}, Real_{EL})$ que nous avons associé aux spécifications algébriques (cf. section 1.1.1).*

$Real_{EL}$ est le foncteur tel que pour tout morphisme de spécifications $\sigma : Sp \to Sp'$, $Real_{EL}(\sigma) : Real_{EL}(Sp') \to Real_{EL}(Sp)$ coïncide avec $U_{|Sig_{EL}(Sp)}$. En plus, nous avons montré que si Sp' est conditionnelle positive, $Real_{EL}(\sigma)$ admet un foncteur adjoint à gauche $F : Real_{EL}(Sp) \to Real_{EL}(Sp')$ (cf. section 3.1.3 du chapitre 2).

2 Structuration des spécifications abstraites

La spécification des systèmes de grande taille représentant un certain niveau de difficulté rend indispensable la définition de moyens de structuration des spécifications. De façon générale, la structuration des spécifications est importante non seulement pour une meilleure lisibilité du

problème à spécifier mais aussi pour l'analyse formelle et la réutilisation des spécifications.

Dans la suite nous nous intéressons à la structuration des spécifications abstraites et nous introduisons un moyen de structuration, le *connecteur architectural*, qui permet de construire une nouvelle spécification à partir de plusieurs spécifications déjà définies en les composant ensemble. Mais avant d'aborder notre contribution, mettons d'abord l'accent sur des travaux existants sur la structuration de spécifications.

2.1 Approches existantes pour la structuration des spécifications

Comme nous l'avons déjà vu dans le chapitre 2, la structuration de spécifications a été bien étudiée dans le cadre de spécifications algébriques au travers des primitives de structurations telles que l'enrichissement et l'union.

Dans un cadre plus général, de nombreux travaux ont étudié la structuration au sein des spécifications axiomatiques au-dessus des institutions. Ces travaux ont été motivés par plusieurs raisons :

- le cadre général des institutions offre un bon moyen de structuration au travers de morphismes de signatures et de la condition de satisfaction.
- les études de la structuration faites dans le cadre général des institutions, permet à n'importe quel formalisme de spécifications axiomatiques de bénéficier directement des résultats qui en découlent, il suffit alors de présenter la logique sous-jacente par une institution et d'instancier les résultats.

Tout comme dans le cas des spécifications algébriques, de nombreux travaux [63, 71, 100, 29] ont étudié la structuration des spécifications axiomatiques en proposant des primitives de structuration. La primitive de structuration la plus étudiée est celle de l'enrichissement, mais dans [71, 29] les auteurs proposent un jeu de primitives plus complet comportant le renommage, l'union ou encore la restriction d'une spécification à une partie de sa signature.

Dans [71, 29], une spécification $Sp' = (\Sigma', Ax')$ est un enrichissement de $Sp = (\Sigma, Ax)$ si et seulement si on est en présence d'un morphisme d'inclusion $\Sigma \hookrightarrow \Sigma'$ et d'une inclusion d'axiomes, c'est-à-dire $Ax \subseteq Ax'$. Ainsi, le foncteur d'oubli associe à toute réalisation de Sp', c'est-à-dire tout modèle de Σ validant Ax, une réalisation de Sp.

Dans le chapitre 2, des primitives de structuration selon [132] ont été présentées dans le cadre de spécifications algébriques où la logique sous-jacente est la logique équationnelle. Si une spécification $Sp' = (\Sigma', Ax')$ est un enrichissement de $Sp = (\Sigma, Ax)$, alors on est en présence d'un morphisme d'inclusion $\Sigma \hookrightarrow \Sigma'$. De plus, l'oubli de toute réalisation de Sp' est une réalisation de Sp. Ainsi, et avec la condition de satisfaction, une formule sur Σ est valide dans une réalisation de Sp' si et seulement si elle est valide dans l'oubli de la réalisation en question.

Ces travaux génériques sur la structuration des spécifications axiomatiques au dessus des institutions ont été suivis par d'autres travaux utili-

sant principalement la notion de colimites de diagrammes[4]. Intuitivement, en théorie de catégories, un diagramme d'une catégorie permet de décrire une classe d'objets ainsi que les morphismes entre les objets. La colimite de diagramme permet de préciser, de point de vue syntaxique, le résultat de cette composition.

L'utilisation de colimites pour modéliser la classe de composants et les interconnexions qui lient ces composants a été proposée par Goguen [59, 61]. Dans cette approche, le principe est le suivant :

- les composants sont des objets d'une catégorie,
- les composants sont modélisés par un diagramme représentant la façon selon laquelle les composants sont liés,
- la composition du système s'obtient par le calcul de la colimite des diagrammes de la catégorie des composants.

2.2 Composition de spécifications abstraites

Dans cette section, nous nous intéressons à la composition de plusieurs spécifications abstraites en une seule spécification globale. Notre objectif est de proposer un moyen permettant de capturer cette composition. Nous parlerons de *connecteur architectural* pour décrire cette composition.

Le connecteur architectural que nous proposons permet de combiner plusieurs spécifications en utilisant le concept de colimite. Chaque spécification est représentée par sa signature et les morphismes de signatures permettent de caractériser les liens entre les spécifications.

Dans la suite, nous supposons que la catégorie des signatures SIG est cocomplète[5], c'est-à-dire qu'une colimite existe toujours pour tout diagramme de la catégorie des signatures.

Définition 4.13 (Connecteur architectural) *Soit $\mathcal{SL}$ un langage de spécification sur $\mathcal{I}$. Un **connecteur architectural** $c : |\mathcal{D}_{(I,SPEC)}| \rightarrow |SPEC|$ est une fonction[6] partielle telle que pour tout diagramme $\delta \in |\mathcal{D}_{(I,SPEC)}|$ tel que $c(\delta)$ est défini, $(Sig(c(\delta)), \{p_i : Sig(\delta(i)) \rightarrow Sig(c(\delta))\}_{i \in I})$ est un cocône colimite de $Sig \circ \delta$.*

Exemple 4.3 *Dans cet exemple, nous allons exprimer les primitives de structuration des spécifications axiomatiques introduites dans [29] au travers de notre notion de connecteur architectural. Considérons donc le langage de spécifications $(SPEC, Sig, Real)$ associé aux spécifications axiomatiques sur $\mathcal{I}$ vu en section 1.1.1.*

Enrichissement. *Soit I le graphe composé de deux nœuds i et j et d'une flèche $a : i \rightarrow j$. Le connecteur* Enrich *pour les spécifications axiomatiques est défini pour tout diagramme $\delta : I \rightarrow SPEC$ avec $\delta(i) = (\Sigma, Ax)$ et $\delta(j) = (\Sigma', Ax')$ telles que $For(Sig(\delta(a)))(Ax) \subseteq Ax'$, par $Enrich(\delta) = (\Sigma', Ax')$ avec $(\Sigma', \{Sig(\delta(a)), Id_{Sig(\delta(j))}\})$ le cocône colimite de $Sig \circ \delta$.*

Union. *Soit I le graphe composé de trois nœuds i, j et k et de deux flèches $a_1 : i \rightarrow j$ et $a_2 : i \rightarrow k$. Le connecteur Union est défini pour tout*

[4] Voir la définition A.15 de l'annexe A pour la notion de colimite.
[5] Voir la définition A.16 de l'annexe A.
[6] Voir la notation A.1 de l'annexe A pour la notation $\mathcal{D}_{(I,SPEC)}$.

diagramme $\delta : I \to SPEC$ *avec* $\delta(i) = (\Sigma_0, Ax_0)$, $\delta(j) = (\Sigma_1, Ax_1)$ *et* $\delta(k) = (\Sigma_2, Ax_2)$, *telles que* $For(Sig(\delta(a_1)))(Ax_0) \subseteq Ax_1$ *et* $For(Sig(\delta(a_2)))(Ax_0) \subseteq Ax_2$, *par* $Union(\delta) = (\Sigma, Ax)$ *tel que* $(\Sigma, \{f_1 \circ Sig(\delta(a_1)) : \Sigma_0 \to \Sigma, f_1 : \Sigma_1 \to \Sigma, f_2 : \Sigma_2 \to \Sigma\})$, *où* $f_1 \circ Sig(\delta(a_1)) = f_2 \circ Sig(\delta(a_2))$, *est le cocône colimite de* $Sig \circ \delta$, *et* $Ax = For(Sig(\delta(a_2)))(Ax_1) \cup For(Sig(\delta(a_2)))(Ax_2)$.

Dans [29], les deux connecteurs ci-dessus peuvent être obtenus en combinant deux connecteurs de base : le connecteur union *de deux spécifications ayant la même signature, noté* $\bigcup$, *et le connecteur* **translate _ by** σ *pour tout morphisme de signatures* σ. *Ces deux connecteurs sont définis par :*

1. *Soit I le graphe composé de deux noeuds i et j. Le connecteur* $\bigcup$ *est défini pour tout diagramme* $\delta : I \to SPEC$, *où* $\delta(i) = (\Sigma, Ax_1)$ *et* $\delta(j) = (\Sigma, Ax_2)$, *par* $\bigcup(\delta) = (\Sigma, Ax)$ *tel que* $(\Sigma, \{Id_{Sig(\delta(i))}, Id_{Sig(\delta(j))}\})$ *le cocône colimite de* $Sig \circ \delta$, *et* $Ax = Ax_1 \cup Ax_2$.
2. *Soit I le graphe composé de deux noeuds k et l. Le connecteur* **translate _ by** σ *où* $\sigma : \Sigma \to \Sigma'$ *est un morphisme de signatures, est défini pour tout diagramme* $\delta : I \to SPEC$, *où* $\delta(k) = (\Sigma, Ax)$ *et* $\delta(l) = (\Sigma', For(\sigma)(Ax))$, *par* **translate _ by** $\sigma(\delta) = (\Sigma', For(\sigma)(Ax))$.

Dans [29], $\bigcup(\delta)$ *et* **translate _ by** $\sigma(\delta)$ *sont respectivement exprimés par* $\delta(i) \bigcup \delta(j)$ *et* **translate** $\delta(k)$ **by** σ.

Dans la suite de ce chapitre, on se place dans un langage de spécification $\mathcal{SL} = (SPEC, Sig, Real)$ sur une institution $\mathcal{I} = (SIG, For, Mod, \models)$ où la catégorie SIG est cocomplète.

Les connecteurs architecturaux peuvent être combinés pour fournir des spécifications plus larges.

Définition 4.14 (Combinaison de connecteurs) *Soient* $c : |\mathcal{D}_{I,SPEC}| \to |SPEC|$ *et* $c' : |\mathcal{D}_{(I',SPEC)}| \to |SPEC|$ *deux connecteurs architecturaux. Soient* $i' \in |I'|$ *un objet. Soit* $I' \circ_{i'} I$ *la catégorie définie par :*

- $|I' \circ_{i'} I| = |I| \coprod |I'|$
- *pour tout* $k, l \in |I' \circ_{i'} I|$, $Hom_{I' \circ_{i'} I}(k, l)$ *est défini inductivement de la façon suivante*[7] *:*
  - $k, l \in |I'| \Rightarrow Hom_{I'}(k, l) \subseteq Hom_{I' \circ_{i'} I}(k, l)$
  - $k, l \in |I| \Rightarrow Hom_I(k, l) \subseteq Hom_{I' \circ_{i'} I}(k, l)$
  - *pour tout* $i \in |I|$, *on introduit la flèche* q_i *dans* $Hom_{I' \circ_{i'} I}(i, i')$.
  - $Hom_{I' \circ_{i'} I}$ *est close par composition.*

Notons $c' \circ_{i'} c : |\mathcal{D}_{(I' \circ_{i'} I, SPEC)}| \to |SPEC|$ *le connecteur architectural défini par :* [8] [9]

$$\delta \mapsto \begin{cases} c'(\delta_{|_{I'}}) & \text{si } c(\delta_{|_I}) \text{ est défini, } \delta(i') = c(\delta_{|_I}) \\ & \text{et } \forall i \in |I|, \delta(q_i) \text{ est le morphisme dans } SPEC \\ & \text{dont l'image par } Sig \text{ est le composant} \\ & p_i \text{ du cocône } (Sig(c(\delta_{|_I})), \{p_i : Sig(\delta(i)) \to Sig(c(\delta_{|_I}))\}_{i \in I}) \\ & \text{colimite de } Sig(c(\delta_{|_I}) \\ \text{non défini} & \text{sinon} \end{cases}$$

[7] Voir la définition A.1 de l'annexe A pour la notation $Hom_{I' \circ_{i'} I}(k, l)$.

[8] q_i est la flèche introduite dans $Hom_{I' \circ_{i'} I}(i, i')$.

[9] Voir la définition A.1 de l'annexe A pour la notation $\delta_{|_I}$.

Exemple 4.4 *L'enrichissement des spécifications axiomatiques vu dans l'exemple 4.3 peut être exprimé par la combinaison de* **translate** *et* $\cup$ *de la manière suivante : soit* δ *un diagramme de* $\mathcal{D}_{(I,SPEC)}$*, où* I *est la catégorie d'index du connecteur Enrich,* $\delta(i) = (\Sigma, Ax)$ *et* $\delta(j) = (\Sigma', Ax')$

$$Enrich(\delta) = \bigcup \circ_i \textbf{translate\_by} \delta'(p_i)(\delta')$$

où δ' *est le diagramme de* $\mathcal{D}_{(I''\circ_i I', SPEC)}$ *où* I'' *(resp.* I'*) est la catégorie d'index du connecteur* $\cup$ *(resp.* **translate***), défini par :* $\delta'(k) = \delta(i)$*,* $\delta'(i) = \textbf{translate}\delta'(k)\textbf{by}\delta'(p_i) = (\Sigma', For(Sig(p_i))(Ax))$ *et* $\delta'(j) = (\Sigma', Ax' \setminus Ax)$*.*

3 Structuration modulaire, structuration complexe

Nous allons généraliser ici les premiers résultats que nous avons obtenus dans le chapitre 2 de la partie I de ce document qui étudiait la modularité et la complexité des spécifications algébriques.

En effet, lorsque la spécification d'un système évolue par composition de spécifications par un connecteur architectural, on doit s'assurer que certaines propriétés des spécifications composantes sont maintenues au niveau de la spécification globale résultante de la composition et d'autres sont maintenues au niveau des spécifications composantes. Il est possible que certaines propriétés d'une certaine spécification composante ne soient pas des propriétés de la spécification globale et vice-versa.

Les propriétés que nous proposons alors d'étudier sont intuitivement de l'une des deux natures suivantes :

- Les *propriétés émergentes* qui sont des propriétés attendues de la spécification globale mais qui ne sont pas inférées des propriétés attendues des spécifications composantes.
- Les *propriétés de non-conformité* qui sont des propriétés attendues des spécifications composantes mais qui ne sont pas des propriétés de la spécification globale.

Formellement, les propriétés attendues d'une spécification sont naturellement dénotées par son ensemble de conséquences sémantiques.

Définition 4.15 (Modularité orientée propriétés, complexité) *Soient* $c : |\mathcal{D}_{(I,SPEC)}| \to |SPEC|$ *un connecteur architectural et* δ *un diagramme de* $\mathcal{D}_{(I,SPEC)}$ *tels que* $c(\delta)$ *est défini.*

Soit $(Sig(c(\delta)), \{p_i : Sig(\delta(i)) \to Sig(c(\delta))\}_{i\in I})$ *le cocône colimite de* $Sig \circ \delta$*.*

$c(\delta)$ *est dit* ***complexe*** *si et seulement si au moins l'une des propriétés suivantes n'est pas satisfaite :*

1. ***Conformité.***
   $\forall i \in I, \forall \varphi \in For(Sig(\delta(i))), \varphi \in \delta(i)^\bullet \iff For(p_i)(\varphi) \in c(\delta)^\bullet$
2. ***Non vraie émergence.*** $\forall \varphi \in c(\delta)^\bullet, \bigcup_{i\in I} For(p_i)(\delta(i)^\bullet) \vDash_{Sig(c(\delta))} \varphi$

   Une formule φ *qui ne satisfait pas une des deux conditions ci-dessus est appelée une* ***propriété émergente*** *pour* $c(\delta)$*.*

*Si $c(\delta)$ n'est pas complexe, alors il est dit **modulaire par propriétés**.*

Comme nous l'avons fait dans le chapitre 2, nous donnons aussi le pendant sémantique de la définition 4.15

Définition 4.16 (Modularité orientée modèles) *Supposons que Real soit compatible.*

Soient $c : |\mathcal{D}_{(I,SPEC)}| \to |SPEC|$ un connecteur architectural et δ un diagramme de $\mathcal{D}_{(I,SPEC)}$ tels que $c(\delta)$ est défini.

Soit $(Sig(c(\delta)), \{p_i : Sig(\delta(i)) \to Sig(c(\delta))\}_{i\in I})$ le cocône colimite de $Sig \circ \delta$.

*$c(\delta)$ est dit **modulaire par modèles** si et seulement si la propriété suivante est satisfaite :*

- *$\forall i \in I, \forall \mathcal{M} \in |Real(\delta(i))|, \exists \mathcal{M}' \in |Real(c(\delta))|$ tel que $Real(p'_i)(\mathcal{M}') = \mathcal{M}$, avec p' défini par $Sig \circ p'_i = p \circ Sig$.*

Le résultat qui suit établit la relation entre la modularité orientée modèles et celle orientée propriétés. En effet, sous la condition de la préservation de la satisfaction des formules par changement de réalisations induit pas les morphismes de $SPEC$, la modularité orientée modèles implique l'absence de propriétés de non-conformité.

Théorème 4.1 *Supposons que Real soit compatible.*

Soient $c : |\mathcal{D}_{(I,SPEC)}| \to |SPEC|$ un connecteur architectural et δ un diagramme de $\mathcal{D}_{(I,SPEC)}$ tels que $c(\delta)$ est défini.

Soit $(Sig(c(\delta)), \{p_i : Sig(\delta(i)) \to Sig(c(\delta))\}_{i\in I})$ le cocône colimite de $Sig \circ \delta$.

Supposons que $c(\delta)$ soit modulaire par modèles et que Real satisfasse la condition suivante : $\forall i \in I, \forall \varphi \in For(Sig(\delta(i))), \forall \mathcal{M}' \in |Real(c(\delta))|$,

$$Real(p'_i)(\mathcal{M}') \models_{Sig(\delta(i))} \varphi \Leftrightarrow \mathcal{M}' \models_{Sig(c(\delta))} For(p_i)(\varphi) \tag{4.1}$$

alors $c(\delta)$ n'admet pas de propriétés de non-conformité.

Preuve.

($\Rightarrow$) Soient $i \in I$ et $\varphi \in For(Sig(\delta(i)))$ tel que $\varphi \in \delta(i)^\bullet$. Montrons que $For(p_i)(\varphi) \in c(\delta)^\bullet$.
Soit $\mathcal{M}' \in |Real(c(\delta))|$. $c(\delta)$ est modulaire par modèles alors $Real(p'_i)(\mathcal{M}') \in |Real(\delta(i))|$, d'où $Real(p'_i)(\mathcal{M}') \models_{Sig(\delta(i))} \varphi$. Par la condition (4.1) on a $\mathcal{M}' \models_{Sig(c(\delta))} For(p_i)(\varphi)$, donc $For(p_i)(\varphi) \in c(\delta)^\bullet$.

($\Leftarrow$) Soit $i \in I, \varphi \in For(Sig(\delta(i)))$ tel que $For(p_i)(\varphi) \in c(\delta)^\bullet$. Montrons que $\varphi \in \delta(i)^\bullet$.
Soit $\mathcal{M} \in |Real(\delta(i))|$. $c(\delta)$ est modulaire par modèles alors $\exists \mathcal{M}' \in |Real(c(\delta))|$ tel que $Real(p'_i)(\mathcal{M}') = \mathcal{M}$. $\mathcal{M}' \in |Real(c(\delta))|$ alors $\mathcal{M}' \models_{Sig(c(\delta))} For(p_i)(\varphi)$. Par la condition (4.1) on a $Real(p'_i)(\mathcal{M}') \models_{Sig(\delta(i))} \varphi$, donc $\varphi \in \delta(i)^\bullet$.

□

4 Conditions de modularité

Après avoir exprimé la structuration des spécifications au moyen du connecteur architectural, il est intéressant d'étudier des conditions qui garantissent la modularité orientée propriétés et donc l'absence de propriétés émergentes au sens de la définition 4.15. Dans cette section, nous fournissons des conditions suffisantes pour une bonne modularité.

Le théorème 4.2 fournit une condition suffisante pour garantir l'absence de vraies propriétés émergentes.

Théorème 4.2 *Soient $c : |\mathcal{D}_{(I,SPEC)}| \to |SPEC|$ un connecteur architectural et $\delta \in |\mathcal{D}_{(I,SPEC)}|$ un diagramme tel que $c(\delta)$ est défini.*

Soit $(Sig(c(\delta)), \{p_i : Sig(\delta(i)) \to Sig(c(\delta))\}_{i\in I})$ le cocône colimite de $Sig \circ \delta$.

Si $T = (\bigcup_{i\in I} For(p_i)(\delta(i)^\bullet))$ est une théorie au dessus de $Sig(c(\delta))$, alors T est définissable par $c(\delta)$ si et seulement si,

(a) l'ensemble de vraies propriétés émergentes est vide, et

(b) pour tout $i \in I$, le morphisme p'_i défini par $Sig \circ p'_i = p_i \circ Sig$ est un morphisme de spécifications préservant les conséquences sémantiques.

Preuve.

($\Rightarrow$) (a) Montrons que l'ensemble de vraies propriétés émergentes est vide.

$T = (\bigcup_{i\in I} For(p_i)(\delta(i)^\bullet))$ est définissable par $c(\delta)$, ainsi par définition et du fait que T est une théorie au dessus de $Sig(c(\delta))$, nous avons $c(\delta)^\bullet = (\bigcup_{i\in I} For(p_i)(\delta(i)^\bullet))^\bullet$, donc l'ensemble de vraies propriétés émergentes est vide.

(b) Montrons que pour tout $i \in I$, p_i est un morphisme de spécifications préservant les conséquences sémantiques : le résultat est immédiat du fait que $c(\delta)^\bullet = (\bigcup_{i\in I} For(p_i)(\delta(i)^\bullet))$.

($\Leftarrow$) L'ensemble de vraies propriétés émergentes est vide, donc nous avons :

$$c(\delta)^\bullet \subseteq (\bigcup_{i\in I} For(p_i)(\delta(i)^\bullet))^\bullet$$

En plus, pour tout $i \in I$, p_i est un morphisme de spécifications préservant les conséquences sémantiques, donc :

$$(\bigcup_{i\in I} For(p_i)(\delta(i)^\bullet))^\bullet \subseteq c(\delta)^\bullet$$

Ainsi, $c(\delta)^\bullet = (\bigcup_{i\in I} For(p_i)(\delta(i)^\bullet))^\bullet$.

Puisque T est une théorie au dessus de $Sig(c(\delta))$, alors $T = T^\bullet$, donc $(\bigcup_{i\in I} For(p_i)(\delta(i)^\bullet))$ est définissable par $c(\delta)$.

□

Dans le théorème 4.3, nous donnons des conditions supplémentaires pour empêcher la présence de propriétés de non-conformité.

Théorème 4.3 *Supposons que $\mathcal{I}$ est close par isomorphismes et que Real est compatible.*

Soient $c : |\mathcal{D}_{(I,SPEC)}| \rightarrow |SPEC|$ un connecteur architectural et $\delta \in |\mathcal{D}_{(I,SPEC)}|$ un diagramme tel que $c(\delta)$ est défini.

Soit $(Sig(c(\delta)), \{p_i : Sig(\delta(i)) \rightarrow Sig(c(\delta))\}_{i \in I})$ le cocône colimite de $Sig \circ \delta$.

Supposons que $T = (\bigcup_{i \in I} For(p_i)(\delta(i)^\bullet_{Sig(\delta(i))}))$ est une théorie au dessus de $Sig(c(\delta))$ définissable par $c(\delta)$.

Si les trois conditions suivantes sont satisfaites :

(a) pour tout p_i, $i \in I$, p'_i défini par $Sig \circ p'_i = p_i \circ Sig$ est libéral,

(b) pour toute réalisation $\mathcal{M}$ de $Real(\delta(i))$, le morphisme d'adjonction $I_{\mathcal{M}} : \mathcal{M} \rightarrow Real(p'_i)(\mathcal{F}(p'_i)(\mathcal{M}))$ est un isomorphisme,

(c) $\forall \varphi \in For(Sig(\delta(i))), \forall \mathcal{M} \in Real(c(\delta))$,

$$\mathcal{M} \models_{Sig(c(\delta))} For(p_i)(\varphi) \Rightarrow Real(p'_i)(\mathcal{M}) \models_{Sig(\delta(i))} \varphi$$

alors, $c(\delta)$ est modulaire.

Preuve.

- Montrons que l'ensemble de vraies propriétés émergentes est vide : conséquence immédiate du théorème 4.2.
- Montrons que l'ensemble de propriétés de non-conformité est vide.
  ($\Rightarrow$) Comme $(\bigcup_{i \in I} For(p_i)(\delta(i)^\bullet))$ est une théorie au dessus de $Sig(c(\delta))$ définissable par $c(\delta)$. Par le théorème 4.2, pour tout p_i, $i \in I$, p'_i défini par $Sig \circ p'_i = p_i \circ Sig$ est un morphisme de spécifications préservant les conséquences sémantiques, donc $\forall\, i \in I, \delta(i)^\bullet \subseteq c(\delta)^\bullet$.
  ($\Leftarrow$) Soit $\varphi \in For(Sig(\delta(i)))$ tel que $For(p_i)(\varphi) \in c(\delta)^\bullet$ et soit $\mathcal{M} \in Real(\delta(i))$. Comme $\mathcal{F}(p'_i)$ est le foncteur adjoint à gauche à $Real(p'_i)$, avec p'_i défini par $Sig \circ p'_i = p_i \circ Sig$, nous avons $\mathcal{F}(p'_i)(\mathcal{M}) \models_{Sig(c(\delta))} For(p_i)(\varphi)$. Par la condition (c), nous avons $Real(p'_i)(\mathcal{F}(p'_i)(\mathcal{M})) \models_{Sig(\delta(i))} \varphi$. Comme le morphisme d'adjonction $I_{\mathcal{M}} : \mathcal{M} \rightarrow Real(p'_i)(\mathcal{F}(p'_i)(\mathcal{M}))$ associé à $\mathcal{M}$ est un isomorphisme, alors $\mathcal{M} \models_{Sig(\delta(i))} \varphi$.

□

Le théorème 4.3 généralise, sur tout connecteur architectural, les conditions de modularité étudiées dans le cas de la structuration des spécifications algébriques conditionnelles positives par enrichissement, comme nous l'avons détaillé en section 3.1 du chapitre 2.

5 Application aux systèmes réactifs

Avant d'aborder le résultat de modularité de spécifications des systèmes réactifs, nous avons tout d'abord besoin de définir la manière de structurer les spécifications pour ce type de systèmes. Dans ce manuscrit, nous nous restreignons à la structuration au travers de *produit synchronisé*.

5.1 Produit synchronisé

Dans notre formalisme de spécifications des systèmes réactifs tel que nous l'avons défini en section 1.1.2, nous avons vu que la spécification est donnée par un système de transitions. Nous allons maintenant expliquer comment on peut composer différentes spécifications de sous-systèmes pour obtenir un système plus grand. Cette structuration, appelé *produit synchronisé*, qui est inspirée du produit synchronisé défini dans [95], fait intervenir une description formelle des interactions entre les divers sous-systèmes au travers les actions. La condition fondamentale que présuppose la définition de ce produit synchronisé est que dans le système global, tous les sous-systèmes exécutent simultanément les transitions étiquetées par la même action. Dans la suite, pour simplifier, nous considérons le produit synchronisé de deux systèmes de transitions. La description formelle du produit synchronisé est donnée ci-dessous.

Définition 4.17 (Produit synchronisé) *Soient $\mathcal{S}_1 = (Q_1, T_1)$ et $\mathcal{S}_2 = (Q_2, T_2)$ deux spécifications de $SPEC_{RS}$ respectivement sur les signatures $\Sigma_1 = (\Omega_1, V_1, C_1)$ et $\Sigma_2 = (\Omega_2, V_2, C_2)$ de SIG_{RS}.*

*Le **produit synchronisé** de $\mathcal{S}_1$ et $\mathcal{S}_2$, noté $\mathcal{S}_1 \otimes \mathcal{S}_2$, est la spécification (Q, T) de $SPEC_{RS}$ sur $\Sigma = (\Omega_1 \cup \Omega_2, V_1 \cup V_2, C_1 \cup C_2)$ définie par :*

- *$Q = Q_1 \times Q_2$;*
- *si $a \in C_1 \cap C_2$, $(q_1, a, \varphi_1, \Delta_1, q_1') \in T_1$ et $(q_2, a, \varphi_2, \Delta_2, q_2') \in T_2$, alors $((q_1, q_2), a, \varphi_1 \wedge \varphi_2, \Delta_1 \cup \Delta_2, (q_1', q_2')) \in T$;*
- *pour tout $i \in \{1, 2\}$, si $a \in C_i \setminus C_{3-i}$ et $(q_i, a, \varphi_i, \Delta_i, q_i') \in T_i$, alors pour tout $q_{3-i} \in Q_{3-i}$, $((q_i, q_{3-i}), a, \varphi_i, \Delta_i, (q_i', q_{3-i})) \in T$.*

Le produit synchronisé de deux spécifications de $SPEC_{RS}$ est bien une spécification de $SPEC_{RS}$. Donc sous cette condition, il est bien possible d'étudier la modularité des spécifications structurées au travers de produit synchronisé.

5.2 Conditions de modularité

Après avoir abordé la notion de structuration au travers de produit synchronisé, nous allons présenter les résultats de modularité qui s'y attachent. Comme les propriétés émergentes sont naturelles, puisqu'elles sont la conséquence de l'union des deux signatures associées à chacune des deux spécifications composant le produit synchronisé, nous allons étudier surtout les propriétés de non-conformité et nous allons fournir des conditions qui assurent la non-existence de telles propriétés.

L'un des sens de la conformité est cependant satisfait par le produit synchronisé. Plus précisément, nous allons montrer que ($\Rightarrow$) est satisfaite. Mais avant d'aborder ce résultat, nous allons introduire la notion de *projection* d'un produit synchronisé qui sera fondamental pour obtenir ce résultat.

Définition 4.18 (Projection) *Soient $\mathcal{S}_1 = (Q_1, T_1)$ et $\mathcal{S}_2 = (Q_2, T_2)$ deux spécifications de $SPEC_{RS}$ respectivement sur les signatures $\Sigma_1 = (\Omega_1, V_1, C_1)$ et $\Sigma_2 = (\Omega_2, V_2, C_2)$ de SIG_{RS}. Soit $\mathcal{S} = \mathcal{S}_1 \otimes \mathcal{S}_2$ le produit synchronisé de $\mathcal{S}_1$ et $\mathcal{S}_2$ et soit $(\mathcal{A}, R) \in |Real_{RS}(\mathcal{S})|$ une réalisation de $\mathcal{S}$.*

*La **projection** de $(\mathcal{A}, R)$ sur Σ_i, où $i \in \{1,2\}$, est le Σ_i-modèle défini de la façon suivante,*

- *pour tout $q_i \in Q_i$, pour tout $q_{3-i} \in Q_{3-i}$, $\mathcal{A}_i^{q_i} = \mathcal{A}_{|\Omega_i}^{(q_1,q_2)}$,*
- *pour tout $a \in C_i$, $R_{i_a} = \{(q_i, q_i') \mid \exists((q_1,q_2),(q_1',q_2')) \in R_a\}$*

On note $(\mathcal{A}, R)_{/\Sigma_i}$ la projection de $(\mathcal{A}, R)$ sur Σ_i.

Théorème 4.4 *Soient $\mathcal{S}_1 = (Q_1, T_1)$ et $\mathcal{S}_2 = (Q_2, T_2)$ deux spécifications de $SPEC_{RS}$ respectivement sur les signatures $\Sigma_1 = (\Omega_1, V_1, C_1)$ et $\Sigma_2 = (\Omega_2, V_2, C_2)$ de SIG_{RS}. Soit $\mathcal{S} = (Q, T)$ le produit synchronisé de $\mathcal{S}_1$ et $\mathcal{S}_2$. Alors, pour tout $i \in \{1,2\}$, si $(\mathcal{A}, R) \in |Real_{RS}(\mathcal{S})|$ alors $(\mathcal{A}, R)_{/\Sigma_i} \in |Real_{RS}(\mathcal{S}_i)|$.*

Preuve. Montrons que $\forall i \in \{1,2\}, (\mathcal{A}_i, R_i) = (\mathcal{A}, R)_{/\Sigma_i}$ est dans $Real_{RS}(\mathcal{S}_i)$.

(a) Montrons que $\forall i \in \{1,2\}$,

$$((q_i, a, \varphi_i, \Delta_i, q_i') \in T_i \wedge \mathcal{A}_i^{q_i} \models_{\Omega_i} \varphi_i \wedge \mathcal{A}_i^{q_i'} = \tau_{\Delta_i}(\mathcal{A}_i^{q_i})) \Rightarrow (q_i, q_i') \in R_{i_a}$$

Supposons que $a \in C_1 \cap C_2$ (la preuve des autres cas se fait par analogie). Par construction du produit synchronisé $\mathcal{S}$, il existe une transition $((q_1,q_2), a, \varphi, \Delta, (q_1', q_2')) \in T$ telle que ou bien $\varphi = \varphi_i$ et $\Delta = \Delta_i$, ou bien $\varphi = \varphi_i \wedge \varphi_{3-i}$ et $\Delta = \Delta_i \cup \Delta_{3-i}$. Supposons que $\varphi = \varphi_i$ et $\Delta = \Delta_i$ (la démonstration de l'autre cas se fait par analogie). Par le théorème 2.6, nous avons $\mathcal{A}^{(q_1,q_2)} \models_\Omega \varphi$, avec $\Omega = \Omega_1 \cup \Omega_2$. De plus, nous avons $\mathcal{A}^{(q_1',q_2')} = \tau_\Delta(\mathcal{A}^{(q_1,q_2)})$, d'où $((q_1,q_2),(q_1',q_2')) \in R_a$. Ainsi, par la définition 4.18, nous obtenons $(q_i, q_i') \in R_{i_a}$.

(b) Montrons que $\forall i \in \{1,2\}$,

$$((q_i, q_i') \in R_{i_a}) \Rightarrow (\exists(q_i, a, \varphi_i, \Delta_i, q_i') \in T_i, \mathcal{A}_i^{q_i} \models_{\Omega_i} \varphi_i \wedge \mathcal{A}_i^{q_i'} = \tau_{\Delta_i}(\mathcal{A}_i^{q_i}))$$

Supposons que $a \in C_1 \cap C_2$ (la preuve des autres cas se fait par analogie). $(q_i, q_i') \in R_{i_a}$, donc par la définition 4.18, il existe $((q_1,q_2),(q_1',q_2')) \in R_a$. Ainsi, il existe une transition $((q_1,q_2), a, \varphi, \Delta, (q_1', q_2')) \in T$, et par conséquent $(q_i, a, \varphi_i, \Delta_i, q_1') \in T_i$ et $(q_{3-i}, a, \varphi_{3-i}, \Delta_{3-i}, q_{3-i}') \in T_{3-i}$ telles que ou bien $\varphi = \varphi_i$ et $\Delta = \Delta_i$, ou bien $\varphi = \varphi_i \wedge \varphi_{3-i}$ et $\Delta = \Delta_i \cup \Delta_{3-i}$. Supposons que $\varphi = \varphi_i$ et $\Delta = \Delta_i$ (la démonstration de l'autre cas se fait par analogie). Puisque $((q_1,q_2),(q_1',q_2')) \in R_a$, donc $\mathcal{A}^{(q_1,q_2)} \models_\Omega \varphi_i$ et $\mathcal{A}^{(q_1',q_2')} = \tau_{\Delta_i}(\mathcal{A}^{(q_1,q_2)})$, d'où $\mathcal{A}_i^{q_i} \models_{\Omega_i} \varphi_i$ (cf. théorème 2.6) et $\mathcal{A}_i^{q_i'} = \tau_{\Delta_i}(\mathcal{A}_i^{q_i})$.

□

Théorème 4.5 *Soient $\mathcal{S}_1 = (Q_1, T_1)$ et $\mathcal{S}_2 = (Q_2, T_2)$ deux spécifications de $SPEC_{RS}$ respectivement sur les signatures $\Sigma_1 = (\Omega_1, V_1, C_1)$ et $\Sigma_2 = (\Omega_2, V_2, C_2)$ de SIG_{RS}. Soit $\mathcal{S} = (Q, T)$ le produit synchronisé de $\mathcal{S}_1$ et $\mathcal{S}_2$.*

Pour tout $i \in \{1,2\}$, pour toute formule $\varphi \in For_{RS}(\Sigma_i)$ et pour toute réalisation $(\mathcal{A}, R) \in |Real_{RS}(\mathcal{S})|$,

$$(\mathcal{A}, R)_{/\Sigma_i} \models_{\Sigma_i} \varphi \Rightarrow (\mathcal{A}, R) \models_\Sigma \varphi$$

où $\Sigma = (\Omega_1 \cup \Omega_2, V_1 \cup V_2, C_1 \cup C_2)$.

Preuve. Montrons que :

$$\forall q_i \in Q_i, \forall q_{3-i} \in Q_{3-i}, (\mathcal{A}, R)_{/\Sigma_i} \models_{\Sigma_i}^{q_i} \varphi \Rightarrow (\mathcal{A}, R) \models_{\Sigma}^{(q_1,q_2)} \varphi$$

La preuve se fait par induction structurelle sur les formules de $For_{RS}(\Sigma_i)$.

- si φ est de la forme $t = t'$, avec $t, t' \in T_\Sigma(V)_s$ et $s \in S_i$, alors par application directe du théorème 2.6 nous avons,

$$(\mathcal{A}, R)_{/\Sigma_i} \models_{\Sigma_i}^{q_i} \varphi \Rightarrow (\mathcal{A}, R) \models_{\Sigma}^{(q_1,q_2)} \varphi$$

- si φ est de la forme $\Box_a \psi$, alors, soit $((q_1, q_2), (q_1', q_2')) \in R_a$ et montrons que $(\mathcal{A}, R) \models_{\Sigma}^{(q_1',q_2')} \psi$. Par la définition 4.18, $(q_i, q_i') \in R_{i_a}$, d'où $(\mathcal{A}, R)_{/\Sigma_i} \models_{\Sigma_i}^{q_i'} \psi$. Par l'hypothèse d'induction nous obtenons $(\mathcal{A}, R) \models_{\Sigma}^{(q_1',q_2')} \psi$.
- pour les autres formes de φ la preuve est facile.

□

Par les théorèmes 4.4 et 4.5, on peut déduire que les conséquences sémantiques de $\mathcal{S}_1$ et $\mathcal{S}_2$ sont préservées par le produit synchronisé $\mathcal{S}_1 \otimes \mathcal{S}_2$.

Théorème 4.6 *Soient $\mathcal{S}_1 = (Q_1, T_1)$ et $\mathcal{S}_2 = (Q_2, T_2)$ deux spécifications de $SPEC_{RS}$ respectivement sur les signatures $\Sigma_1 = (\Omega_1, V_1, C_1)$ et $\Sigma_2(\Omega_2, V_2, C_2)$ de SIG_{RS}. Soit $\mathcal{S} = \mathcal{S}_1 \otimes \mathcal{S}_2$ le produit synchronisé de $\mathcal{S}_1$ et $\mathcal{S}_2$, alors $\mathcal{S}_i^\bullet \subseteq (\mathcal{S}_1 \otimes \mathcal{S}_2)^\bullet$.*

Preuve. Soient $\varphi \in \mathcal{S}_i^\bullet$ et $(\mathcal{W}, R) \in |Real_{RS}(\mathcal{S}_1 \otimes \mathcal{S}_2)|$. D'après le théorème 4.4, $(\mathcal{A}, R)_{/\Sigma_i} \in |Real_{RS}(\mathcal{S}_i)|$, ainsi $(\mathcal{A}, R)_{/\Sigma_i} \models_{\Sigma_i} \varphi$. Par le théorème 4.5, nous obtenons $(\mathcal{A}, R) \models_\Sigma \varphi$. □

Dans la suite, nous allons étudier des conditions suffisantes fondées sur l'adjonction vue au chapitre 2 pour garantir l'autre sens de la conformité, c'est-à-dire ($\Leftarrow$).

Rappelons tout d'abord la notion de produit de deux algèbres.

Définition 4.19 (Produit de deux algèbres) *Soit $\Omega = (S, F)$ une signature de SIG_{EL} et soient $\mathcal{A}_1$ et $\mathcal{A}_1$ deux Ω-algèbres. Le produit de $\mathcal{A}_1$ et $\mathcal{A}_1$, noté $\mathcal{A}_1 \times \mathcal{A}_2$, est la Ω-algèbre définie par :*

- *l'ensemble sous-jacent est tel que pour tout $s \in S$,*

$$(A_1 \times A_2)_s = (A_1)_s \times (A_1)_s$$

- *pour tout $f : s_1 \times \cdots \times s_n \to s \in F$, pour tout $((a_{1,1}, a_{2,1}), \ldots, (a_{1,n}, a_{2,n}))$ dans $(A_1 \times A_2)_{s_1} \times (A_1 \times A_2)_{s_n}$,*

$$f_{\mathcal{A}_1 \times \mathcal{A}_2}((a_{1,1}, a_{2,1}), \ldots, (a_{1,n}, a_{2,n})) = (f_{\mathcal{A}_1}(a_{1,1}, \ldots, a_{1,n}), f_{\mathcal{A}_2}(a_{2,1}, \ldots, a_{2,n}))$$

Fait 4.1 *[39] Soit Ω une signature de SIG_{EL}. Pour toute formule $\varphi \in For_{EL}(\Omega)$ conditionnelle positive et pour toutes algèbres $\mathcal{A}_1$ et $\mathcal{A}_2$ de $|Alg(\Omega)|$,*

$$\forall i \in \{1, 2\}, \mathcal{A}_i \models_\Omega \varphi \Leftrightarrow \mathcal{A}_1 \times \mathcal{A}_2 \models_\Omega \varphi$$

Définition 4.20 (Extension) *Soient $\mathcal{S}_1 = (Q_1, T_1)$ et $\mathcal{S}_2 = (Q_2, T_2)$ deux spécifications de $SPEC_{RS}$ respectivement sur les signatures $\Sigma_1 = (\Omega_1, V_1, C_1)$ et $\Sigma_2 = (\Omega_2, V_2, C_2)$ de SIG_{RS} telles que $For_{EL}(\Omega_1 \cup \Omega_2)$ est restreint aux formules conditionnelles positives. Soit $\mathcal{S} = \mathcal{S}_1 \otimes \mathcal{S}_2$ le produit synchronisé de $\mathcal{S}_1$ et $\mathcal{S}_2$.*

Pour tout $i \in \{1,2\}$, pour tout $(\mathcal{A}_{3-i}, R_{3-i}) \in |Real_{RS}(\mathcal{S}_{3-i})|$, on définit l'application $\mathcal{F}_{(\mathcal{A}_{3-i},R_{3-i})} : |Real_{RS}(\mathcal{S}_i)| \to |Mod_{RS}(\Sigma)|$ qui à tout $(\mathcal{A}_i, R_i) \in |Real_{RS}(\mathcal{S}_i)|$ associe $\mathcal{F}_{(\mathcal{A}_{3-i},R_{3-i})}((\mathcal{A}_i, R_i)) \in |Mod_{RS}(\Sigma)|$ défini par :

- *pour tout $q_i \in Q_i$, pour tout $q_{3-i} \in Q_{3-i}$,*

$$\mathcal{A}^{(q_1,q_2)} = F_{|\Omega_i}(\mathcal{A}_i^{q_i}) \times F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}})$$

  où :
  - *$F_{|\Omega_i} : Alg(Sp_i) \to Alg(Sp)$ (resp. $F_{|\Omega_{3-i}} : Alg(Sp_{3-i}) \to Alg(Sp)$) est le foncteur de synthèse donné par le théorème 2.4 ;*
  - *$Sp_i = (\Omega_i, Th(\mathcal{A}_i^{q_i}))$ (resp. $Sp_{3-i} = (\Omega_{3-i}, Th(\mathcal{A}_{3-i}^{q_{3-i}}))$),*
  - *$Sp = (\Omega_1 \cup \Omega_2, Th(\mathcal{A}_i^{q_i}) \cup Th(\mathcal{A}_{3-i}^{q_{3-i}}))$.*
- *pour tout $a \in C_1 \cap C_2$, $R_a = \{((q_1, q_2), (q_1', q_2')) \mid (q_1, q_1') \in R_{1_a}$ et $(q_2, q_2') \in R_{2_a}\}$,*
- *pour tout $a \in C_i \setminus C_{3-i}$, $R_a = \{((q_1, q_2), (q_1', q_2')) \mid (q_i, q_i') \in R_{i_a}\}$.*

*L'**extension** de $(\mathcal{A}_i, R_i)$ sur Σ, avec $i \in \{1,2\}$, est le Σ-modèle $\mathcal{F}_{(\mathcal{A}_{3-i},R_{3-i})}((\mathcal{A}_i, R_i))$.*

Théorème 4.7 *Soient $\mathcal{S}_1 = (Q_1, T_1)$ et $\mathcal{S}_2 = (Q_2, T_2)$ deux spécifications de $SPEC_{RS}$ respectivement sur les signatures $\Sigma_1 = (\Omega_1, V_1, C_1)$ et $\Sigma_2 = (\Omega_2, V_2, C_2)$ de SIG_{RS} telles que $For_{EL}(\Omega_1 \cup \Omega_2)$ est restreint aux formules conditionnelles positives. Soit $\mathcal{S} = (Q, T)$ le produit synchronisé de $\mathcal{S}_1$ et $\mathcal{S}_2$.*

Si pour tout $i \in \{1,2\}$, pour tout $(\mathcal{A}_i, R_i) \in |Real_{RS}(\mathcal{S}_i)|$, pour tout $(\mathcal{A}_{3-i}, R_{3-i}) \in |Real_{RS}(\mathcal{S}_{3-i})|$, pour tout $q_i \in Q_i$, pour tout $q_{3-i} \in Q_{3-i}$, les morphismes d'adjonction $I_{\mathcal{A}_i^{q_i}} : \mathcal{A}_i^{q_i} \to U_{|\Omega_i}(F_{|\Omega_i}(\mathcal{A}_i^{q_i}))$ et $I_{\mathcal{A}_{3-i}^{q_{3-i}}} : \mathcal{A}_{3-i}^{q_{3-i}} \to U_{|\Omega_{3-i}}(F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}))$ sont des isomorphismes, alors,

$$\mathcal{F}_{(\mathcal{A}_{3-i},R_{3-i})}((\mathcal{A}_i, R_i)) \in |Real_{RS}(\mathcal{S})|$$

Preuve. Montrons que $\forall i \in \{1,2\}$, $\mathcal{F}_{(\mathcal{A}_{3-i},R_{3-i})}((\mathcal{A}_i, R_i)) = (\mathcal{A}, R)$ est dans $|Real_{RS}(\mathcal{S})|$.

Soit $F'_{|\Omega_i} : Alg(Sp'_i) \to Alg(Sp')$ (resp. $F'_{|\Omega_{3-i}} : Alg(Sp'_{3-i}) \to Alg(Sp')$) le foncteur de synthèse donné par le théorème 2.4, avec :

- $Sp'_i = (\Omega_i, Th(\mathcal{A}_i^{q_i'}))$ (resp. $Sp'_{3-i} = (\Omega_{3-i}, Th(\mathcal{A}_{3-i}^{q'_{3-i}}))$),
- $Sp' = (\Omega_1 \cup \Omega_2, Th(\mathcal{A}_i^{q_i'}) \cup Th(\mathcal{A}_{3-i}^{q'_{3-i}}))$.

(a) Montrons que $\forall i \in \{1,2\}$,

$$(((q_1, q_2), a, \varphi, \Delta, (q_1', q_2')) \in T \wedge F_{|\Omega_i}(\mathcal{A}_i^{q_i}) \times F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}) \vDash_\Omega \varphi$$

$$\wedge F'_{|\Omega_i}(\mathcal{A}_i^{q_i'}) \times F'_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q'_{3-i}}) = \tau_\Delta(F_{|\Omega_i}(\mathcal{A}_i^{q_i}) \times F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}})))$$

$$\Rightarrow ((q_1, q_2), (q_1', q_2')) \in R_a$$

Supposons que $a \in C_1 \cap C_2$ (la preuve des autres cas se fait par analogie). Par construction du produit synchronisé, ou bien φ (resp.

Δ) est sous la forme $\varphi_i \wedge \varphi_{3-i}$ (resp. $\Delta_i \cup \Delta_{3-i}$), ou bien $\varphi \in For_{EL}(\Omega_i)$ (resp. $\Delta \in \mathcal{EB}(\Sigma_i)$), ou bien $\varphi \in For_{EL}(\Omega_{3-i})$ (resp. $\Delta \in \mathcal{EB}(\Sigma_{3-i})$). Supposons que $\varphi = \varphi_i \wedge \varphi_{3-i}$ et $\Delta = \Delta_i \cup \Delta_{3-i}$ (la preuve des autres cas se fait par analogie). Par construction du produit synchronisé, il existe deux transitions

$$(q_i, a, \varphi_i, \Delta_i, q_i') \in T_i \text{ et } (q_{3-i}, a, \varphi_{3-i}, \Delta_{3-i}, q_{3-i}') \in T_{3-i}$$

Par hypothèse, nous avons $F_{|\Omega_i}(\mathcal{A}_i^{q_i}) \times F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}) \vDash_\Omega \varphi_i \wedge \varphi_{3-i}$, ainsi par le fait 4.1 et par le théorème 2.6, nous avons :

$$U_{|\Omega_i}(F_{|\Omega_i}(\mathcal{A}_i^{q_i})) \vDash_{\Omega_i} \varphi_i \text{ et } U_{|\Omega_{3-i}}(F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}})) \vDash_{\Omega_{3-i}} \varphi_{3-i}$$

Comme les morphismes d'adjonction $I_{\mathcal{A}_i^{q_i}} : \mathcal{A}_i^{q_i} \to U_{|\Omega_i}(F_{|\Omega_i}(\mathcal{A}_i^{q_i}))$ et $I_{\mathcal{A}_{3-i}^{q_{3-i}}} : \mathcal{A}_{3-i}^{q_{3-i}} \to U_{|\Omega_{3-i}}(F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}))$ sont des isomorphismes, alors,

$$\mathcal{A}_i^{q_i} \vDash_{\Omega_i} \varphi_i \text{ et } \mathcal{A}_{3-i}^{q_{3-i}} \vDash_{\Omega_{3-i}} \varphi_{3-i}$$

De plus, par construction,

$$\tau_{\Delta_i}(\mathcal{A}_i^{q_i}) = \mathcal{A}_i^{q_i'} \text{ et } \tau_{\Delta_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}) = \mathcal{A}_{3-i}^{q_{3-i}'}$$

Ainsi, $(q_i, q_i') \in R_{i_a}$ et $(q_{3-i}, q_{3-i}') \in (R_{3-i})_a$. Donc par la définition 4.20, nous avons $((q_1, q_2), (q_1', q_2')) \in R_a$.

(b) Montrons que $\forall i \in \{1, 2\}$,

$$\begin{aligned} ((q_1, q_2), (q_1', q_2')) \in R_a \quad \Rightarrow \quad & ((q_1, q_2), a, \varphi, \Delta, (q_1', q_2')) \in T, \\ & F_{|\Omega_i}(\mathcal{A}_i^{q_i}) \times F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}) \vDash_\Omega \varphi \wedge \\ & F'_{|\Omega_i}(\mathcal{A}_i^{q_i'}) \times F'_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}'}) = \tau_\Delta(F_{|\Omega_i}(\mathcal{A}_i^{q_i}) \times F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}))) \end{aligned}$$

Supposons que $a \in C_1 \cap C_2$ (la preuve des autres cas se fait par analogie). $((q_1, q_2), (q_1', q_2')) \in R_a$, donc par la définition 4.20, $(q_1, q_1') \in R_{a_1}$ et $(q_2, q_2') \in R_{a_2}$. Par construction du produit synchronisé, il existe une transition $((q_1, q_2), a, \varphi, \Delta, (q_1', q_2')) \in T$, $(q_1, q_1') \in R_{a_1}$ et $(q_2, q_2') \in R_{a_2}$ telle que, ou bien φ (resp. Δ) est sous la forme $\varphi_i \wedge \varphi_{3-i}$ (resp. $\Delta_i \cup \Delta_{3-i}$), ou bien $\varphi \in For_{EL}(\Omega_i)$ (resp. $\Delta \in \mathcal{EB}(\Sigma_i)$), ou bien $\varphi \in For_{EL}(\Omega_{3-i})$ (resp. $\Delta \in \mathcal{EB}(\Sigma_{3-i})$). Supposons que $\varphi = \varphi_i \wedge \varphi_{3-i}$ et $\Delta = \Delta_i \cup \Delta_{3-i}$ (la preuve des autres cas se fait par analogie). Donc, il existe deux transitions

$$(q_i, a, \varphi_i, \Delta_i, q_i') \in T_i \text{ et } (q_{3-i}, a, \varphi_{3-i}, \Delta_{3-i}, q_{3-i}') \in T_{3-i}$$

Puisque $(q_i, q_i') \in R_{a_i}$ et $(q_{3-i}, q_{3-i}') \in R_{a_{3-i}}$, donc,

$$\mathcal{A}_i^{q_i} \vDash_{\Omega_i} \varphi_i \text{ et } \mathcal{A}_{3-i}^{q_{3-i}} \vDash_{\Omega_{3-i}} \varphi_{3-i}$$

Par conséquent,

$$F_{|\Omega_i}(\mathcal{A}_i^{q_i}) \vDash_{\Omega_i} \varphi_i \text{ et } F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}) \vDash_{\Omega_{3-i}} \varphi_{3-i}$$

Ainsi, par le fait 4.1, nous avons :

$$F_{|\Omega_i}(\mathcal{A}_i^{q_i}) \times F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}) \vDash_\Omega \varphi_i \wedge \varphi_{3-i}$$

De plus, nous avons :

$$\tau_{\Delta_i}(\mathcal{A}_i^{q_i}) = \mathcal{A}_i^{q_i'} \text{ et } \tau_{\Delta_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}) = \mathcal{A}_{3-i}^{q_{3-i}'}$$

Par construction, nous obtenons :

$$F'_{|\Omega_i}(\mathcal{A}_i^{q_i'}) \times F'_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}'}) = \tau_\Delta(F_{|\Omega_i}(\mathcal{A}_i^{q_i}) \times F_{|\Omega_{3-i}}(\mathcal{A}_i^{q_{3-i}}))$$

□

Théorème 4.8 *Soient $\mathcal{S}_1 = (Q_1, T_1)$ et $\mathcal{S}_2 = (Q_2, T_2)$ deux spécifications de $SPEC_{RS}$ respectivement sur les signatures $\Sigma_1 = (\Omega_1, V_1, C_1)$ et $\Sigma_2 = (\Omega_2, V_2, C_2)$ de SIG_{RS} telles que $For_{EL}(\Omega_1 \cup \Omega_2)$ est restreint aux formules conditionnelles positives. Soit $\mathcal{S} = (Q, T)$ le produit synchronisé de $\mathcal{S}_1$ et $\mathcal{S}_2$.*

Si pour tout $i \in \{1, 2\}$, pour tout $(\mathcal{A}_i, R_i) \in |Real_{RS}(\mathcal{S}_i)|$, pour tout $(\mathcal{A}_{3-i}, R_{3-i}) \in |Real_{RS}(\mathcal{S}_{3-i})|$, pour tout $q_i \in Q_i$, pour tout $q_{3-i} \in Q_{3-i}$, les morphismes d'adjonction $I_{\mathcal{A}_i^{q_i}} : \mathcal{A}_i^{q_i} \to U_{|\Omega_i}(F_{|\Omega_i}(\mathcal{A}_i^{q_i}))$ et $I_{\mathcal{A}_{3-i}^{q_{3-i}}} : \mathcal{A}_{3-i}^{q_{3-i}} \to U_{|\Omega_{3-i}}(F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}))$ sont des isomorphismes, alors,

$$(\mathcal{S}_1 \otimes \mathcal{S}_2)^\bullet \cap For_{RS}(\Sigma_i) \subseteq \mathcal{S}_i^\bullet$$

Preuve. Soit $\varphi \in (\mathcal{S}_1 \otimes \mathcal{S}_2)^\bullet \cap For_{RS}(\mathcal{S}_i)$ et soit $(\mathcal{A}_i, R_i) \in Real_{RS}(\mathcal{S}_i)$. Par le théorème 4.7, pour tout $(\mathcal{A}_{3-i}, R_{3-j}) \in Real_{RS}(\mathcal{S}_j)$, nous avons $\mathcal{F}_{(\mathcal{A}_{3-i}, R_{3-i})}((\mathcal{A}_i, R_i)) \models_\Sigma \varphi$. Montrons alors que,

$$\mathcal{F}_{(\mathcal{A}_{3-i}, R_{3-i})}((\mathcal{A}_i, R_i)) \models_\Sigma \varphi \Rightarrow (\mathcal{A}_i, R_i) \models_{\Sigma_i} \varphi$$

Montrons par induction structurelle sur les formules de $For_{RS}(\Sigma_i)$ que :

$$\forall q_i \in Q_i, \forall q_{3-i} \in Q_{3-i}, \mathcal{F}_{(\mathcal{A}_{3-i}, R_{3-i})}((\mathcal{A}_i, R_i)) \models_\Sigma^{(q_1, q_2)} \varphi \Rightarrow (\mathcal{A}_i, R_i) \models_{\Sigma_i}^{q_i} \varphi$$

- si φ est de la forme $t = t'$, avec $t, t' \in T_{\Omega_i}(V_i)_s$ et $s \in S_i$, alors,
  $F_{|\Omega_i}(\mathcal{A}_i^{q_i}) \times F_{|\Omega_{3-i}}(\mathcal{A}_{3-i}^{q_{3-i}}) \models_\Omega \varphi \Rightarrow \mathcal{A}_i^{q_i} \models_{\Omega_i} \varphi$
  $\Rightarrow (\mathcal{A}_i, R_i) \models_{\Sigma_i}^{q_i} \varphi$
- si φ est de la forme $\Box_a \psi$, alors, soit $(q_i, q_i') \in R_{i_a}$ et montrons que $(\mathcal{A}_i, R_i) \models_{\Sigma_i}^{q_i'} \psi$. Par la définition 4.20, $((q_1, q_2), (q_1', q_2')) \in R_a$, d'où $\mathcal{F}_{(\mathcal{A}_{3-i}, R_{3-i})}((\mathcal{A}_i, R_i)) \models_\Sigma^{(q_1, q_2)} \psi$. Par l'hypothèse d'induction nous obtenons $(\mathcal{A}_i, R_i) \models_{\Sigma_i}^{q_i'} \psi$.
- pour les autres formes de φ la preuve est facile.

□

6 Conclusion

Dans ce chapitre nous avons présenté un cadre général pour les spécifications que nous avons appelé de façon générique spécifications abstraites. Ces spécifications étant définies indépendamment de toute logique, dans le cadre général des institutions.

Dans ce cadre, nous avons proposé une présentation générique de la notion de spécifications structurées par l'intermédiaire des connecteurs

architecturaux. Ce cadre général nous a permis aussi de distinguer les spécifications modulaires des spécifications complexes.

Pour illustrer l'intérêt de notre approche, nous avons modélisé les systèmes réactifs à l'aide d'une institution dédiée, et d'un connecteur représentant le produit synchronisé usuel. Nous avons mis à jour le résultat attendu que les spécifications résultantes ne sont modulaires que sous des conditions restrictives très fortes.

7 Avant propos sur les réseaux de régulation génétique

Dans la partie qui suit, et comme indiqué dans l'introduction de ce document, nous nous intéresserons à la problématique de l'émergence des propriétés au sein des systèmes biologiques, particulièrement au sein de la modélisation discrète des réseaux de régulation génétique.

Les évolutions que connaît la biologie de nos jours laissent entrevoir la possibilité de collecter des informations partielles sur les réseaux de régulations génétiques. Toutefois, il est communément accepté que les propriétés de ces réseaux ne découlent pas des propriétés de leurs composants individuels. Ainsi, les propriétés étudiées au niveau d'un sous-réseau ne se retrouvent pas systématiquement au niveau du réseau l'englobant.

Néanmoins, les biologistes ont souvent pour habitude d'étudier une entité en l'isolant de son contexte (une cellule,organelle, réseau de régulation). Il est donc essentiel de savoir sous quelles conditions les propriétés associées à l'entité prise en isolation restent encore valides lorsqu'on considère l'organisme englobant l'entité concernée.

Par ailleurs, des études ont été faites pour exprimer les propriétés biologiques en formules de la logique temporelle dans la modélisation discrète introduite de R. Thomas [22]. C'est dans ce cadre discret que nous allons étudier la préservation des propriétés en plongeant un sous-réseau dans un réseau l'englobant.

Ainsi, le connecteur que nous allons considérer dans le cadre biologique s'apparente au connecteur de l'enrichissement (défini par un morphisme d'inclusion) pour les spécifications axiomatiques. De plus, il s'agira d'un enrichissement très simple, car seul un morphisme d'inclusion sera considéré, sans qu'il lui soit adjoint de nouveaux axiomes à considérer. Le plongement des réseaux fournira le morphisme d'inclusion des signatures, et aucune formule temporelle ne sera prise en compte spécifiquement dans la construction du connecteur.

Troisième partie

Etude de la préservation des propriétés temporelles des réseaux de régulation génétique au travers de leur plongement

Dans cette partie de la thèse, et comme indiqué dans l'introduction de ce document, nous nous intéresserons à la problématique de l'émergence des propriétés au sein des systèmes biologiques, particulièrement au sein de la modélisation discrète des réseaux de régulation génétique.

Les biologistes ont souvent pour habitude d'étudier une entité en l'isolant de son contexte (une cellule,organelle, réseau de régulation). Il est donc essentiel de savoir sous quelles conditions les propriétés associées à l'entité prise en isolation restent encore valides lorsqu'on considère l'organisme englobant l'entité concernée.

Par ailleurs, des études ont été faites pour exprimer certaines propriétés des réseaux de régulation génétique en formules de la logique temporelle dans la modélisation discrète introduite de R. Thomas [22]. C'est dans ce cadre discret que nous allons étudier la préservation des propriétés en plongeant un sous-réseau dans un réseau l'englobant.

Ainsi, le connecteur que nous allons considérer dans le cadre biologique s'apparente au connecteur de l'enrichissement (défini par un morphisme d'inclusion) pour les spécifications axiomatiques. De plus, il s'agira d'un enrichissement très simple, car seul un morphisme d'inclusion sera considéré, sans qu'il lui soit adjoint de nouveaux axiomes à considérer. Le plongement des réseaux fournira le morphisme d'inclusion des signatures, et aucune formule temporelle ne sera prise en compte spécifiquement dans la construction du connecteur.

Réseaux de régulation génétique

5

Sommaire

Dans une première partie de ce chapitre, quelques notions biologiques simples mais fondamentales, qui permettront au lecteur non biologiste de comprendre la problématique seront présentées. Puis, nous étudierons la représentation des réseaux de régulation génétique dans la modélisation discrète introduite par R. Thomas [125, 124] et dans un cadre logique où les propriétés biologiques sont exprimées en logique temporelle [22].

1 Le contexte biologique

La notion de réseau de régulation génétique résulte de l'observation, faite dans le cadre de la biologie moléculaire, d'une action coordonnée de différents gènes d'un organisme. En effet, les gènes ne sont pas des entités indépendantes les unes des autres, mais constituent une collectivité structurée dont l'organisation et le fonctionnement dépendent en partie de l'expression de ces gènes : des protéines régulatrices activent des gènes particuliers, ainsi l'expression des gènes est régulée par des protéines issues de l'expression d'autres gènes. L'ensemble de ces interactions génétiques est appelé réseau de régulation génétique.

La modélisation de ces réseaux et l'étude de leur dynamique seront le thème fondamental de cette partie. Mais avant d'en aborder la modélisation mathématique suivie dans le manuscrit, commençons par décrire succinctement le principe de la régulation génétique. Pour cela, nous allons introduire certaines notions de base, destinées essentiellement aux lecteurs non familiers de la biologie cellulaire. Pour une description plus détaillée de ces phénomènes, on pourra se référer à [86, 11, 130].

Les cellules possèdent un programme génétique et les moyens permettant son utilisation

Ce programme génétique constitué par l'ensemble des gènes, *le génome*, définit les caractères héréditaires d'une espèce et de chaque individu. Les expériences menées par Griffith et Avery ont montré que le support physique du programme génétique était une longue molécule, l'acide désoxyribonucléique ou ADN. La structure moléculaire de l'ADN, a été découverte en 1953 par Watson et Crick. C'est une macromolécule constituée de deux brins qui s'enroulent l'un autour de l'autre, le tout formant une structure caractéristique en double hélice. Chaque brin est lui-même constitué d'un enchaînement de quatre types de nucléotides. Les nucléotides de l'ADN sont composés d'un sucre, le désoxyribose, auquel est attaché un seul phosphate et une base azotée. Les bases possibles sont au nombre de quatre : l'adénine (A), la cytosine (C), la guanine (G) et la thymine (T). Les deux brins sont reliés par leurs bases selon une complémentarité stricte : A en face de T, G en face de C. L'ordre des bases A, T, G ou C le long de la molécule d'ADN varie d'un individu à l'autre, et leur nombre est caractéristique d'une espèce. La question qui se pose est comment la cellule décode-t-elle et utilise-t-elle ces informations ?

Les cellules utilisent l'ADN pour synthétiser des protéines, qui sont les molécules constituées d'acides aminés et responsables, au final, d'un caractère ou d'une propriété biologique. En effet, dans une molécule d'ADN on distingue certaines zones appelées *gènes*, caractérisées par leur séquence nucléotidique. En effet, dès que la structure de l'ADN a été décrite, il était évident que la structure des acides animés d'une protéine était déterminée par la séquence des nucléotides de l'ADN d'un gène. Intuitivement, chaque gène contient l'information nécessaire à la synthèse d'une *protéine* par la cellule. On dit qu'il code pour une protéine. La question devient alors de savoir comment les gènes qui sont conservés sous forme de séquences de nucléotides peuvent-ils exercer leurs fonctions sous la forme de protéines ?

Le transfert de l'information depuis l'ADN vers les protéines s'effectue au travers de deux mécanismes distincts : *la transcription* et *la traduction*.

La transcription

La synthèse des protéines implique la copie des régions spécifiques de l'ADN (les gènes) en polynucléotides connus sous le nom d'acide ribonucléique ou ARN. L'ARN, comme l'ADN, est composé d'une séquence linéaire de nucléotides, mais il présente deux petites différences chimiques : les nucléotides de l'ARN contiennent du ribose au lieu du désoxyribose, et la thymine (T) est remplacée par l'uracile (U), une base très voisine qui s'apparie également avec A. La structure de l'ARN est différente de celle de l'ADN : l'ARN est toujours composé d'un brin unique. La synthèse de l'ARN est réalisée par des enzymes, appelées ARN polymérases. Ces enzymes rencontrent aléatoirement les brins d'ADN auxquels ils n'adhèrent que faiblement. Cependant, ils se lient étroitement lorsqu'ils rencontrent une séquence spécifique de l'ADN, *le promoteur*, qui contient le site d'initiation de la synthèse de l'ARN et indique où celle-ci doit commencer. Les ARN polymérases ne sont pas capables d'identifier les promoteurs par leurs propres moyens, elles ont besoin d'autres protéines, appelées *facteurs de transcription* qui se fixent à des séquences spécifiques de l'ADN. Une fois bien positionnée, l'ARN polymérase ouvre une courte région de la double hélice en exposant ainsi les nucléotides et permettant la formation de bases complémentaires. Un des deux brins d'ADN sert de matrice sur laquelle les capacités d'appariement des nouveaux nucléotides sont contrôlées. De cette façon, la chaîne d'ARN en croissance s'allonge d'un nucléotide à la fois.

Les cellules synthétisent plusieurs types d'ARN :
- ARN messager ou ARNm qui portent le code nécessaire à la synthèse des protéines,
- ARN de transfert ou ARNt qui sont des molécules qui servent d'adaptateur lors la synthèse des protéines,
- ARN ribosomique ou ARNr qui sont des constituants des ribosomes.

La traduction

Les acides nucléiques et les protéines ressemblent à deux langues écrites avec des symboles différents. C'est pourquoi la synthèse de la protéine est appelée traduction. La traduction exige que l'information codée dans la séquence nucléotique d'un ARNm soit décodée et utilisée pour contrôler l'assemblage progressif des acides aminés de la chaîne polypeptidique.

La synthèse protéique demande l'incorporation de 20 acides aminés selon une séquence précise dictée par un message codé écrit dans un langage qui utilise 4 symboles différents. Un code génétique permet le passage d'une séquence nucléotidique (écrites sur 4 symboles) à une séquence polypeptidique composée d'acides aminés (écrites sur vingt symboles). Si chaque acide aminé est déterminé par un groupe de nucléotides, deux nucléotides ne donnant que 16 possibilités, et on est vite passé à un code à 3 bases permettant 64 combinaisons, c'est-à-dire qu'une protéine est codée par une séquence de *codons* formés de triplets de nucléotides. Plusieurs

codons ont la même signification ou certains codons ne correspondent à aucun acide aminé.

Le décodage de l'information d'un ARNm est réalisé par les ARN de transfert, qui jouent le rôle d'adaptateurs. D'un côté, chaque ARNt est uni à un acide aminé, tandis que de l'autre côté, le même ARNt est capable de reconnaître un codon particulier de l'ARNm par complémentarité des bases.

La synthèse protéique est l'activité la plus complexe de la cellule. L'assemblage d'une protéine a besoin de tous les ARNt différents avec leurs acides aminés attachés, des ribosomes, de l'ARN messager. On peut diviser la synthèse d'une chaîne polypeptidique en trois activités assez distinctes : l'*initiation* de la chaîne, son *élongation* et sa *terminaison*. La synthèse commence dès que le ribosome s'attache à un site précis de l'ARNm, le codon d'initialisation, caractérisé par AUG. L'union à ce codon place automatiquement le ribosome dans le bon cadre de lecture pour lire correctement le message. Ce processus est contrôlé par des protéines appelées facteurs d'initiation de la traduction. L'ARNr permet de lier les acides aminés produits par des liaisons peptidiques. La protéine terminée est libérée du ribosome dès qu'un codon stop est atteint (UAA, UAG, UGA).

La figure 5.1 résume grossièrement les mécanismes de transfert de l'information de l'ADN à l'ARN puis aux protéines.

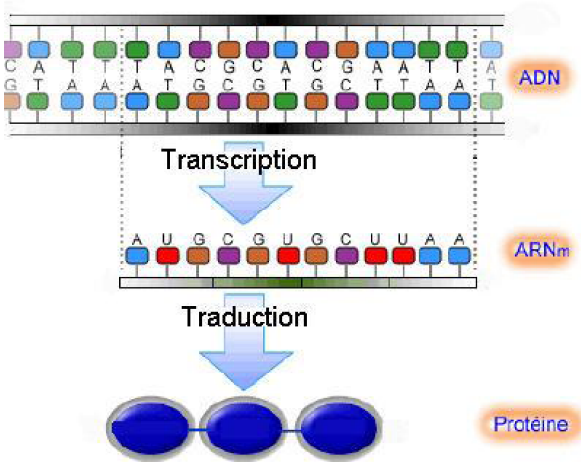


FIG. 5.1 – *Le transfert d'information de l'ADN à la protéine. Source [1].*

Stratégie de contrôle de l'expression des gènes

Les cellules d'un organisme multicellulaire se différencient les unes des autres, car elles synthétisent et stockent des ensembles différents d'ARN et de protéines, c'est-à-dire qu'elles expriment des gènes différents. Elles le font sans changement de leur ADN. La question est de savoir comment la cellule détermine parmi les milliers de gènes qu'elle contient ceux devant être exprimés.

L'expression de la majorité des gènes est contrôlée de manière prédominante au niveau de l'initiation de la transcription, bien que chaque étape de l'expression d'un gène puisse en principe être contrôlée. La transcription de chaque gène est contrôlée par des facteurs de transcription qui

sont des protéines particulièrement importantes à l'initiation de la transcription. Autrement dit, la transcription d'un gène est activée ou inhibée par ces protéines régulatrices. Il existe un petit nombre de motifs structuraux de liaison à l'ADN et bien que chaque protéine régulatrice ait des caractéristiques particulières, elles se lient toutes à l'ADN en utilisant l'un des motifs. La séquence précise d'acides aminés qui participe au domaine de liaison à l'ADN détermine la séquence l'ADN qui sera reconnue et transcrite. La protéine régulatrice peut avoir alors deux effets, on la qualifiera :

- d'activateur lorsque la protéine facilite la liaison avec l'ARN polymérase permettant l'initiation de la transcription,
- de répresseur ou inhibiteur quand la protéine se lie à une séquence régulatrice et bloque l'accès de l'ARN polymérase.

Les analyses génétiques, réalisées chez la bactérie dans les années 50 ont fourni les premières preuves de l'existence de protéines régulatrices de l'expression des gènes, qui induisent ou supprime spécifiquement la transcription des gènes. Un des régulateurs, le *répresseur de lambda* est codé par un virus bactérien, le *bactériophage lambda*. Le répresseur empêche la transcription des gènes viraux codant pour les constituants protéiques de nouvelles particules virales. Le répresseur de lambda fut l'une des premières protéines régulatrices caractérisées et demeure l'une des plus comprises comme on le verra dans les exemples plus loin.

Plus généralement, les protéines synthétisées par les mécanismes de transcription et puis traduction d'un gène donné peuvent à leur tour activer ou inhiber la transcription d'autres gènes. Pour simplifier, on dira qu'un gène active ou inhibe un autre gène. Les gènes d'une cellule se régulent ainsi mutuellement et forment ce qu'on appelle un *réseau de régulation génétique*.

Devant la complexité de tels réseaux, qui peuvent inclure un très grand nombre de gènes, la modélisation et la simulation apparaissent comme des outils importants de compréhension.

2 Modélisation des réseaux de régulation génétique

Généralement, les biologistes ont recours à des schémas pour représenter un réseau génétique impliqué dans les processus et fonctions biologiques qu'ils étudient. Ces schémas sont très compliqués, intégrant plusieurs processus biologiques de différentes natures (régulation transcriptionnelle, voies métaboliques, etc.). Dans le cadre des réseaux de régulation génétique, ils peuvent être abstraits sous forme de graphe orienté dont les sommets représentent des composants biologiques et les arcs les différents types de régulations entre ces composants. Ils se trouvent ainsi avec des graphes ayant une forte hétérogénéité et de grande dimension, nécessitant la mise en place d'une formalisation mathématique pour être correctement exploités.

Ainsi, une des premières questions à se poser concerne le choix de la modélisation pour représenter un réseau de régulation génétique. Cette

question est, bien sûr intimement liée à la problématique du phénomène étudié : la modélisation est-elle discrète ou continue ? Est-elle déterministe, ou y a-t-il une part de stochastique ? Les réponses à ces questions forment un sujet de recherche à elles seules, et si nous ne prétendons pas y répondre ici, il est évident que nous devrons tenir compte de ces questions, en particulier pour la première étape concernant le choix du formalisme des réseaux de régulation génétique.

Un réseau génétique décrit les différentes interactions (régulations) entre les gènes. Un même gène peut être régulateur de plusieurs gènes et avoir un effet activateur ou inhibiteur selon la cible considérée. Que ce soit discret ou continu, un réseau de régulation est en général modélisé par un graphe orienté signé appelé *graphe d'interactions* [114, 42, 119], noté $G = (V, F)$, où l'ensemble des sommets, $V = \{1, \ldots, n\}$, représente les n gènes du réseau, et l'ensemble des arcs F représente les régulations : $(i, j) \in F$ si la protéine synthétisée à partir du gène i régule l'expression du gène j. Les arcs sont étiquetés par un signe positif, (+) dans le cas d'une activation, quand la protéine codée par i favorise l'expression du gène j, et d'un signe négatif (-) dans le cas d'une inhibition, quand la protéine codée par i bloque l'expression de j.

Un tel graphe d'interactions représente l'aspect statique d'un réseau de régulation génétique. Néanmoins, il permet d'expliquer au niveau de la régulation génétique un processus biologique donné : les variations temporelles des niveaux d'expression des gènes sont les conséquences des régulations génétiques. Les niveaux d'expressions des gènes à un instant donné représentent un état du réseau qui peut être modélisé par un n-uplet $s = (s_1, \ldots, s_n)$ où s_i représente le niveau d'expression du gène i.

Dès lors que l'on souhaite représenter la dynamique des réseaux de régulation, c'est-à-dire l'évolution des niveaux d'expression des gènes au cours du temps, il s'avère nécessaire de préciser la manière dont la combinatoire des différentes interactions influence l'expression d'un gène. Au sein d'un large spectre de formalismes, on distingue deux types d'approches :

- continu [42] : $s_i \in [0, +\infty[$ et l'évolution globale du système est donné par un ensemble d'équations différentielles partielles ou d'équations linéaires par morceaux.
- discret [125, 124, 115, 74] : $s_i \in \{0, \ldots, b_i\} \subset \mathbb{N}$, où b_i est le niveau d'expression maximum du gène b_i et la dynamique est donnée par une relation $D : \prod_{i=1}^{n} \{0, \ldots, b_i\} \rightarrow \prod_{i=1}^{n} \{0, \ldots, b_i\}$

Les approches continues permettent de faire des prédictions numériques précises sur les propriétés dynamiques des réseaux de régulation génétique. Cependant, dans la plupart des cas, l'application d'équations différentielles est loin d'être évidente. En effet, les mécanismes de régulation restent complexes et ne sont pas connus de façon complète, ce qui complique la formulation des modèles. En plus, les données quantitatives sur les paramètres cinétiques et les concentrations moléculaires sont généralement absentes, même dans le cas de systèmes bien connus, ce qui rend les méthodes continues standards difficiles à appliquer.

Toutefois, il est intéressant d'observer que les graphes de régulation

sont souvent déduits de façon qualitative à partir de données hétérogènes, et que les biologistes les utilisent souvent pour construire des modèles dynamiques, typiquement dans un cadre discret. En effet, comme nous l'avons déjà évoqué, les données expérimentales sont souvent insuffisantes pour établir une dynamique exacte et ces données montrent que la régulation génétique est un phénomène à effet de seuils. Dans un second temps, la pertinence du modèle considéré est ensuite évaluée en comparant des simulations numériques avec les données expérimentales. En effet, il a été montré que les régulations entre les gènes sont presque toujours non-linéaires et sont généralement associées à des courbes sigmoïdales caractérisées par une valeur seuil, qui est l'abscisse du point d'inflexion, et par le plafonnement de leur effet (figure 5.2 à gauche). Prenons l'exemple d'un réseau composé des deux gènes i et j où i active j. La figure 5.2 à gauche représente la variation du niveau d'expression de j en fonction de i qui est donnée par une sigmoïde : au dessous d'un certain niveau d'expression "seuil" τ_i, le gène activateur i n'a aucun effet sur le gène j. Par contre au dessus du seuil τ_i, le niveau d'expression du gène j atteint son maximum. Ainsi, il semble naturel d'assimiler la fonction sigmoïde à une fonction à seuil [56, 76, 77, 78, 79], et de discrétiser le niveau de l'expression du gène i en considérant qu'au dessous d'un certain seuil, le gène i n'a pas d'effet sur j, et au dessus de ce seuil, il a un effet maximal sur j. La figure 5.2 à droite représente la discrétisation de l'expression du gène j.

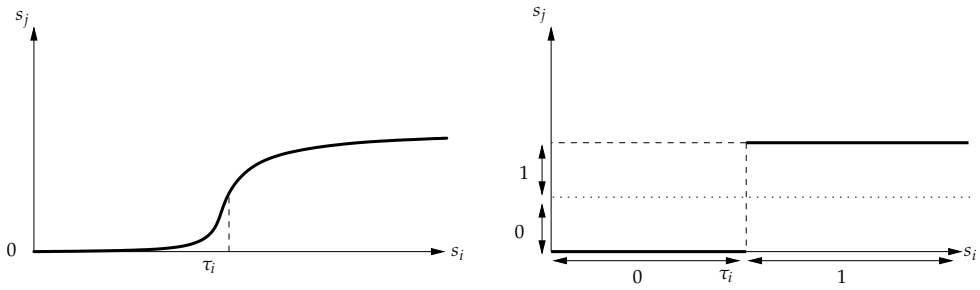


FIG. 5.2 – *Illustration des approches continue et discrète.*

En raison de la nature qualitative de nombreuses données expérimentales et de la nature de la régulation caractérisée par un effet de seuils, de nombreux chercheurs ont proposé d'étudier les réseaux de régulation dans un cadre discret [74, 57]. L'approche discrète la plus utilisée est *la méthode logique généralisée* développée par le biologique R. Thomas et ses collègues depuis les années 70 [125, 124, 127]. Cette méthode a prouvé sa pertinence dans l'étude de plusieurs réseaux de régulation génétique, comme le lambda phage, infection de la bactérie Escherichia coli [121, 125, 122, 120], l'embryogenèse de Drosophila melanogaster [111, 109, 110], la morphogenèse des fleurs chez l'Arabidopsis thaliana [91, 90] et la réponse immunitaire [79, 78, 97].

Dans cette approche, des variables discrètes, dites *logiques ou multivaluées*, abstraient les niveaux d'expressions réels des protéines. La structure d'un réseau est représentée par un graphe d'interactions. À chaque arc (i, j) du graphe est associé une valeur discrète qui modélise le seuil à partir duquel i agit sur j. Une dynamique possible du réseau est déterminée

par la donnée *des paramètres logiques* donnant pour chaque gène i et pour chaque état du réseau (ou plutôt pour chaque état de l'ensemble des prédécesseurs de i dans le graphe d'interactions), la valeur vers laquelle tend i. Un comportement global du réseau peut être alors représenté par *un graphe de transitions asynchrone*. Dans ce graphe, les sommets représentent les états possibles du réseau et les arcs indiquent les changements d'états qui peuvent avoir lieu au cours du temps.

Lorsque le graphe d'interactions du réseau est connu, déterminer sa dynamique suivant l'approche de R. Thomas consiste principalement à ajuster la valeur de chaque paramètre logique de sorte que la dynamique obtenue ne soit pas en contradiction avec les observations expérimentales portant sur le réseau. Ceci n'est pas évident puisque les paramètres logiques sont généralement inconnus, nombreux et difficiles à mesurer. Néanmoins, les paramètres ne peuvent prendre qu'un nombre fini de valeurs permettant ainsi d'énumérer l'ensemble des dynamiques possibles. De plus, comme chaque dynamique est représentée par un graphe de transitions asynchrone lui aussi fini, l'usage des logiques temporelles et du model checking permet d'automatiser la sélection de ces dynamiques valides. Succinctement, les logiques temporelles permettent de traduire formellement les observations biologiques portant sur la dynamique du réseau et les algorithmes de model checking sont alors utilisés afin de vérifier automatiquement si une dynamique vérifie les observations traduites en formules temporelles.

Cette étape de validation est d'autant plus importante que les réseaux de régulation à modéliser sont complexes et à paramètres inconnus, et pourtant le problème de validation de propriétés dynamiques n'a attiré jusqu'à présent que peu d'attention. Un groupe de chercheurs ont effectivement utilisé le model checking pour analyser systématiquement un grand nombre de modèles simples [22]. Ces auteurs utilisent l'approche logique généralisée, ou plus exactement une reformulation plus adaptée à un traitement informatique de celle-ci, en combinaison avec du model checking, pour faire la sélection de modèles. Cette approche permet de sélectionner parmi un ensemble de modèles potentiels les modèles satisfaisant un ensemble de contraintes sur les valeurs discrètes des niveaux d'expression des gènes, formulées en CTL. L'approche a été implémentée en un prototype de logiciel *SMBioNet*, utilisant le model checker NuSMV [108]. La méthode a été appliquée pour l'analyse d'un ensemble de réseaux potentiels, tel que l'étude des gènes impliqués dans la production de mucus chez le Pseudomonas aeruginosa [22] et le contrôle de l'immunité chez le phage λ [108], et a permis de renforcer la crédibilité de certaines hypothèses biologiques. Cette approche est cependant limitée par le phénomène d'explosion combinatoire à deux niveaux. D'une part, l'espace des états, qu'il est nécessaire de construire avant de vérifier la satisfiabilité d'une formule CTL, peut être important. D'autre part, le nombre de modèles à construire est lui-même énorme.

Dans ce travail, nous proposons de développer une méthode formelle pour l'étude de la préservation des propriétés dynamiques au travers les morphismes d'inclusion. Pour représenter formellement les réseaux de régulation génétique, nous proposons une reformulation de l'approche pro-

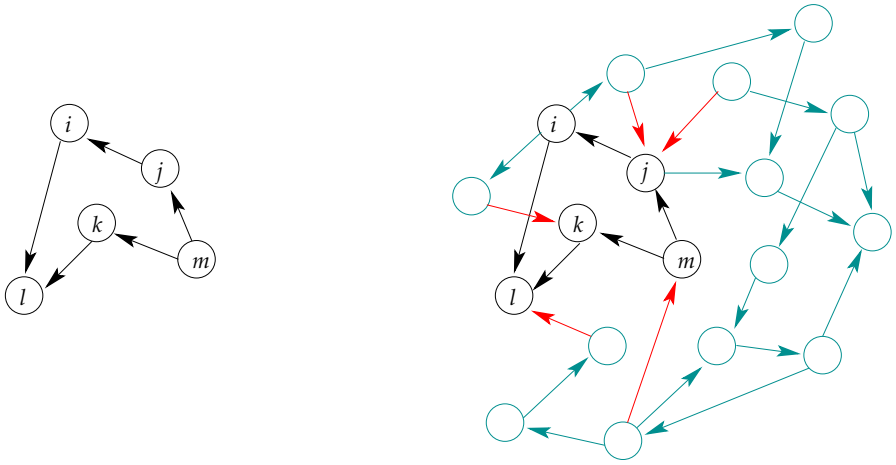


FIG. 5.3 – *Plongement du réseau P (à gauche) dans le réseau L (à droite).*

posée dans [22]. En exploitant les outils mathématiques disponibles pour cette approche et les informations qualitatives sur les valeurs des paramètres qui peuvent habituellement être déduites de la littérature expérimentale, nous nous sommes focalisés sur la vérification de propriétés dans un cadre discret fondé sur une abstraction des réseaux de régulation dans le cadre général des institutions.

3 Schéma général

Un réseau de régulation génétique est souvent composé d'un grand nombre de gènes, et pour en simplifier l'étude de la dynamique il est tentant de supposer qu'il existe des sous-réseaux qui peuvent être étudiés séparément pour déduire la dynamique du réseau global. Seulement, les réseaux de régulation ont une structure et un fonctionnement complexe puisque les propriétés étudiées au niveau d'un sous-réseau ne se retrouvent pas systématiquement au niveau du réseau l'englobant. Il est donc essentiel de savoir sous quelles conditions les propriétés associées à un sous-réseau pris en isolation restent valides lorsqu'on plonge le sous-réseau concerné dans le réseau global (la figure 5.3 représente le plongement d'un réseau P dans un réseau L). Les questions suivantes sont donc à considérer : quels types de plongement peuvent assurer la préservation des propriétés du sous-réseau dans le réseau étendu ? En particulier, étant donné un sous-réseau P, dans quels cas peut-on utiliser les connaissances acquises sur celui-ci pour inférer les propriétés du réseau étendu L obtenu par plongement du sous-réseau P ? Il est vraisemblable que la préservation des propriétés de P dans le réseau global L, dépend de l' absence/présence de nouvelles régulations des gènes de P par des nouveaux gènes de L.

Dans la suite de la thèse, nous allons tenter de répondre à ces questions. Précisément, nous allons identifier des conditions de plongement sous lesquelles les propriétés sont systématiquement préservées. Ces conditions seront étudiées en détail dans le chapitre 6 et les résultats de préservation en sections 4 et 5 de ce chapitre. Ces résultats sont l'extension des résultats publiés dans [89].

4 Une modélisation discrète des réseaux de régulation génétique

Dans ce chapitre et dans le chapitre qui suit, nous allons présenter dans le cadre abstrait des institutions, au sens de la définition 3.1, la modélisation des réseaux de régulation génétique (RRG) en s'inspirant de l'approche proposée dans [22].

4.1 Signature

Comme nous l'avons déjà souligné, la structure d'un RRG est décrite symboliquement par un graphe d'interactions. Il s'agit d'un graphe fini, orienté et étiqueté $G = \langle V, F, Sn, Sl \rangle$ où l'ensemble des noeuds V, appelés *variables*, représente l'ensemble des gènes du réseau, et l'ensemble des arcs F représente leurs interactions : il y a un arc $(i, j) \in F$ d'une variable i vers une variable j si le gène i est régulateur du gène j. Un arc est signé positivement dans le cas d'une *activation*, c'est à dire quand la protéine codée par le gène i favorise l'expression du gène j, et il est signé négativement dans le cas d'une *inhibition*, c'est à dire quand la protéine codée par le gène i ralentit ou stoppe l'expression du gène j. Tout arc $(i, j) \in F$ est étiqueté par un seuil logique de déclenchement $Sl(i, j)$, c'est-à-dire le niveau d'expression abstrait du gène i à partir duquel il est effectif. Si un gène i agit sur n gènes, au plus $n + 1$ niveaux d'expression abstraits sont considérés, chaque valeur est comprise entre 0 et n, éventuellement moins car le même seuil peut concerner deux cibles différentes.

Les seuils logiques ne correspondent pas directement aux valeurs biologiques, qui sont souvent difficiles à mesurer expérimentalement, mais ils donnent plutôt l'ordre de ces seuils. Dans un graphe d'interactions G, si une variable i a b_i seuils sortants alors pour tout $l \in \{1, \ldots, b_i\}$ il existe une variable j du réseau tel que (i, j) soit dans F et tel que $Sl(i, j) = l$.

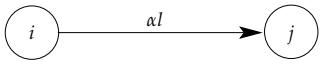
Dans la méthode de Thomas [124], ainsi que dans les travaux de Bernot et al. [22], il n'est pas possible d'avoir plusieurs interactions d'un sommet à un autre. Sous cette hypothèse, on définit, aux notations près, la signature d'un RRG. Néanmoins, cette hypothèse est assez forte puisque, dans la réalité, il n'est pas rare q'un régulateur agisse différemment sur une même cible suivant le contexte. Un exemple bien connu est celui de la régulation de l'opéron arabinose chez l'Escherichia coli qu'on peut trouver en détail dans [99].

Définition 5.1 (RRG-signature) *Une RRG-signature G est un graphe orienté étiqueté donné par le quadruplet $\langle V, F, Sn, Sl \rangle$ où :*

1. *$V = \{1, \ldots, n\}$ est un ensemble fini dont les éléments sont appelés des variables.*
2. *$F \subseteq V \times V$ dénote l'ensemble des arcs.*
3. *Sn est une application de F dans $\{+, -\}$.*
4. *Sl est une application de F dans $\mathbb{N}^*$ telle que :*

$$\forall (i, j) \in F, Sl(i, j) > 1 \Rightarrow \exists (i, k) \in F, Sl(i, k) = Sl(i, j) - 1$$

Si $(i,j) \in F$, $Sl(i,j) = l$ et $Sn(i,j) = \alpha$, on dit que G contient une interaction de i à j de seuil l et de signe α qui est représentée graphiquement de la manière suivante :



On dit aussi que i est une ***source*** *de j dans G, et que j est une* ***cible*** *de i dans G. Pour toute variable i de G, on note G_i^- (resp. G_i^+) l'ensemble des sources (resp. cibles) de i dans G, et par b_i le cardinal de l'ensemble des seuils des arcs (i,j), avec j une variable de V.*

$$G_i^+ = \{j \in V \mid (i,j) \in F\},\ G_i^- = \{j \in V \mid (j,i) \in F\},\ b_i = |\{Th(i,j) \mid j \in G_i^+\}|$$

Pour illustrer la définition 5.1, prenons l'exemple de la régulation génétique du bactériophage Lambda.

Exemple 5.1 (Mode de croissance d'un bactériophage Lambda dans une cellule Escherichia coli)
Le bactériophage Lambda est un virus qui infecte la bactérie Escherichia coli. Une fois dans la cellule hôte, le phage Lambda est sujet à deux cycles d'évolution possibles : le premier est le cycle lytique et le second et le cycle lysogénique. Le cycle lytique conduit à la mort de la cellule hôte et à la libération des phages fabriqués à l'intérieur de celle-ci. Chacun des phages peut alors infecter une autre cellule hôte, de sorte que quelques cycles lytiques pourraient détruire une population entière. Cependant, ces virus sont capables de s'intégrer dans l'ADN de l'E. coli où ils sont répliqués automatiquement chaque fois que la bactérie se divise : c'est le cycle lysogène. L'ADN viral s'insère dans le chromosome bactérien et exprime un répresseur, la protéine cI, qui active sa propre synthèse et bloque l'expression de tous les autres gènes viraux. L'ADN viral subit des réplications en même temps que le reste du génome bactérien. Ce cycle aboutit généralement au bout d'un certain temps à un cycle lytique et à l'excision du génome phagique hors du génome bactérien.

Nous ne pouvons pas ici entrer dans les détails de ce système régulateur complexe, mais nous allons donner un aperçu de quelques-unes de ses propriétés générales et des régulateurs principaux de cI. Une synthèse de mécanismes de régulation de ce système est donnée dans [104]. Lorsque l'ADN de Lambda pénètre dans une cellule hôte, les voies lytique et lysogénique débutent de la même façon. Elles ont toutes deux besoin de l'expression des gènes précoces immédiats et des gènes précoces retardés, mais ensuite, elles divergent : le développement lytique continue si les gènes tardifs sont exprimés, et la lysogénie se poursuit au contraire si le répresseur cI est synthétisé.

Lambda possède seulement deux gènes précoces immédiats, transcrits indépendamment par l'ARN polymérase de l'hôte :

- *Le gène N qui engendre la transcription des gènes précoces retardés.*
- *Le gène cro qui empêche la synthèse du répresseur cI (une action nécessaire si le cycle lytique doit se poursuivre) et interrompt l'expression des gènes précoces immédiats et le gène précoce retardé cII.*

Les gènes précoces retardés possèdent des fonctions opposées :

- *Le gène cII qui est nécessaire pour débuter la synthèse du répresseur cI, et donc pour que le phage s'engage dans la lysogénie.*
- *Les autres gènes précoces retardés interviennent plutôt dans le déroulement du cycle lytique.*

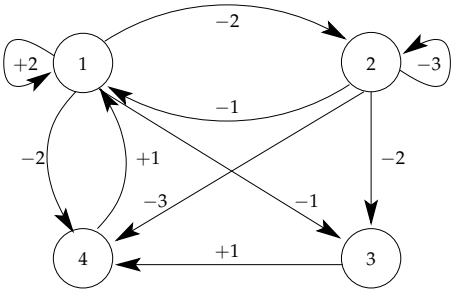


FIG. 5.4 – *Signature du réseau de régulation correspondant à la réplication du bactériophage Lambda*

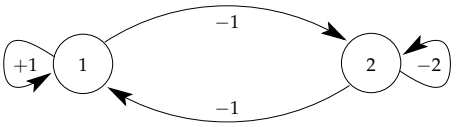


FIG. 5.5 – *RRG-signature G_1 pour le réseau de régulation génétique cI-cro.*

Ainsi, les régulateurs principaux de cI sont cI lui même et les trois protéines virales cro, cII et N. Un réseau de régulation formé par ces quatre gènes a été étudié par D. Thieffry et R. Thomas [120] en s'appuyant sur les données expérimentales recueillies et triées dans [105]. La signature de ce réseau de régulation représentée par le graphe de la figure 5.4 est donnée par le quadruplet $\langle V, F, Sn, Sl\rangle$ où :

- $V = \{1,2,3,4\}$, *où* 1, 2, 3 *et* 4 *dénotent respectivement les gènes cI, cro, N et cII.*
- $F = \{(1,1),(1,4),(1,2),(1,3),(4,1),(2,1),(2,4),(2,2),(2,3),(3,4)\}$
- $Sn = \{(1,1) \to +,\ (1,4) \to -,(1,2) \to -,\ (1,3) \to -,\ (4,1) \to +,\ (2,1) \to -,\ (2,4) \to -,\ (2,2) \to -,\ (2,3) \to -,\ (3,4) \to +\}$
- $Sl = \{(1,1) \to 2,\ (1,4) \to 2,\ (1,2) \to 2,\ (1,3) \to 1,\ (4,1) \to 1,\ (2,1) \to 1,\ (2,4) \to 3,\ (2,2) \to 3,\ (2,3) \to 2,\ (3,4) \to 1\}$

Exemple 5.2 *Dans cet exemple, on considère un réseau de régulation à deux gènes* 1 *et* 2 *extrait du réseau à* 4 *gènes de l'exemple 5.1. Cet exemple nous servira dans la suite.*

4.2 Les formules

En pratique, lorsqu'on étudie la dynamique d'un réseau de régulation génétique donné par sa signature, et donc par son graphe d'interactions, on cherche à donner un modèle valide, c'est à dire un modèle qui ne contredit pas les propriétés dynamiques observées expérimentalement. Dans notre cas, et comme nous le verrons par la suite, un modèle est *un graphe de transitions asynchrone* non déterministe qui peut être étendu à une structure de Kripke non nécessairement totale. Dans le contexte biologique qui est le notre, il est essentiel de pouvoir exprimer des « possibilités dans le futur » et non nécessairement des « possibilités dans le futur immédiat» , ainsi la logique temporelle arborescente FITL (cf. section 2.4 du chapitre 3) restreinte aux formules sans « ne*X*t », qu'on appelle dans la suite la logique FITL-X, est bien adaptée pour exprimer naturellement telles propriétés dynamiques très diverses surtout qu'une logique temporelle linéaire par exemple ne permet d'exprimer ces possibilités. Pour l'ensemble des propositions atomiques, on se contente des propositions atomiques qui s'ex-

priment sous la forme $(x_i \sim l)$, avec i une variable du réseau, l un entier naturel dans $\{0, \ldots, b_i\}$ et $\sim$ un opérateur de comparaison.

Définition 5.2 (Formules) *Soit G une RRG-signature. L'ensemble des propositions atomiques de G, noté $Atome(G)$, est l'ensemble défini de la façon suivante :*

$$Atome(G) = \{x_i \sim l \mid i \in V, \sim \in \{=, <, >\} \text{ et } 0 \leq l \leq b_i\}$$

L'ensemble des formules de G, noté $For(G)$, est l'ensemble $For_{FITL\text{-}X}(Atome(G))$ donné par la définition 3.21.

Néanmoins, nous indiquons l'existence d'autres logiques temporelles qui ont été utilisées dans le contexte des réseaux de régulation comme la logique CTL qui a été utilisée dans [22] et le λ-calcul [83] qui a été utilisé avec la logique CTL dans [19].

Pour montrer la diversité des questions biologiquement pertinentes sur la dynamique d'un réseau que l'on peut formuler à l'aide de la logique FITL-X, nous donnons quelques exemples de propriétés, en précisant leur signification biologique et leur formalisation en FITL. Les propriétés fréquemment traduites en FITL-X concernent l'atteignabilité d'une propriété. On utilise donc pour cela l'opérateur EF. Par exemple, la formule

$$(x_i = 0) \Rightarrow EF(x_j > 1)$$

se traduit par « depuis un état où le niveau d'expression de i est nul, il est possible d'atteindre un état où le niveau d'expression de j est strictement supérieur à 1 ». L'opérateur AF permet d'exprimer des propriétés inévitables. Par exemple, la formule

$$(x_i = 1) \Rightarrow AF[(x_i = 1) \wedge (x_j = 0)]$$

se traduit par « toutes les trajectoires qui partent de l'état où le niveau de d'expression de i est égal à 1 passent par un état où le niveau d'expression de i garde cette valeur et le niveau d'expression de j est nul ». Ce type de propriété permet d'identifier des « étapes obligatoires » par lesquelles le système passe nécessairement. L'opérateur AG permet d'exprimer des invariants. La formule

$$(x_i > 0) \Rightarrow AG(x_i > 0)$$

se traduit par « depuis un état où le niveau d'expression de i est non nul, pour tous les trajectoires qui partent de cet état, le niveau d'expression de i est non nul tout au long de chacune de ces trajectoires ». Les opérateurs EU et AU permettent de décrire des enchaînements de propriétés. Par exemple,

$$(x_i = 0) \Rightarrow E[(x_j > 0) U (x_i = 1)]$$

se traduit par « depuis un état où le niveau d'expression de i est nul, il existe un futur où le niveau d'expression de j est non nul jusqu'à ce que le niveau d'expression de i soit égal à 1 ». La formule

$$(x_i = 0) \Rightarrow A[(x_i = 0) U (x_j = 0)]$$

se traduit par « depuis un état où le niveau d'expression de i est nul, celui de j finira par l'être aussi, et tant qu'il ne l'est pas, le niveau de i reste nul ».

Exemple 5.3 *Reprenons l'exemple 5.1 du mode de croissance d'un bactériophage Lambda dans une cellule Escherichia coli. Lorsque l'ADN de lambda pénètre dans une cellule hôte, les protéines virales n'existent pas encore. On suppose donc que* $(0,0,0,0)$ *est l'état initial. En FITL-X, cet état est caractérisé par la formule init suivante :*

$$init = (x_1 = 0 \wedge x_2 = 0 \wedge x_4 = 0 \wedge x_3 = 0)$$

On traduit ensuite les voies lytique et lysogénique par la présence de deux états « attracteurs » atteignables depuis l'état initial.

La réponse lyrique est caractérisée par une forte expression de cro l'inhibiteur du gène cI, et par une absence des protéines cI, cII et N. Nous allons supposer que le niveau d'expression de cro est supérieur ou égal à 2 *lors de la réponse lyrique. Ceci s'exprime en FITL-X par la formule lytique suivante :*

$$lytique = (x_1 = 0 \wedge x_2 \geq 2 \wedge x_4 = 0 \wedge x_3 = 0)$$

La réponse lysogénique est caractérisée par une forte expression de cI et par une absence des autres protéines. Nous allons donc supposer que le niveau d'expression de cI est égal à deux lors de la réponse lysogénique. Ceci s'exprime en FITL-X par la formule lyso suivante :

$$lyso = (x_1 = 2 \wedge x_2 \geq 2 \wedge x_4 = 0 \wedge x_3 = 0)$$

Enfin, il faut préciser que les deux voies lytique et lysogénique sont atteignables depuis l'état initial avec la formule voies suivante.

$$voies = init \Rightarrow (EF(lytique) \wedge (EF(lyso)))$$

4.3 Les états

Supposons qu'une variable i régule une variable j à partir du seuil $Sl(i,j)$, ainsi qu'une variable k à partir d'un second seuil $Sl(i,k)$ supérieur au premier, alors trois situations qualitatives distinctes apparaissent clairement : soit le niveau d'expression de i est inférieure aux deux seuils et dans ce cas l'effet de i est minimal sur ses deux cibles, soit le niveau d'expression de i est entre les deux seuils et dans ce cas l'effet de i est maximal sur j et minimal sur k, soit le niveau d'expression de i est supérieure aux deux seuils et dans ce cas, l'effet de i est maximal sur ses deux cibles. Ainsi, trois niveaux logiques doivent être utilisés pour décrire qualitativement comment i régule ses cibles.

D'une façon plus générale, les seuils d'une variable divisent son intervalle de niveaux d'expression en domaines où elle a un effet constant sur chacune de ses cibles. Ainsi, et en considérant l'exemple 5.2, deux niveaux d'expression peuvent être associés à la variable 1 : 0 pour le domaine où la variable 1 n'agit sur aucune cible, 1 pour le domaine où elle agit sur les deux cibles. De même, trois niveaux d'expression peuvent être associés à la variable 2 : 0 dans le cas où la variable 2 ne régule aucune cible, 1 dans le cas où elle régule seulement la variable 1 et 2 dans le cas où elle régule les deux cibles. Plus généralement, une variable i, ayant b_i seuils différents, a $(b_i + 1)$ niveaux d'expression différents qui sont $\{0, ..., b_i\}$. *Un état* du réseau est la donnée du niveau d'expression de chaque gène à un instant fixé.

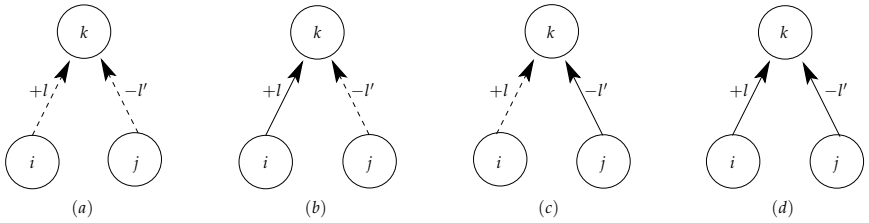


FIG. 5.6 – *Lorsque le niveau d'expression d'une source de k ne dépasse pas son seuil d'activation/inhibition, l'interaction est représentée en pointillé : (a) j, inhibiteur absent, est une ressource de k, (b) i, activateur présent, et j, inhibiteur absent, sont ressources de k, (c) Niveau basal, (d) i, activateur présent, est une ressource de k.*

Définition 5.3 (État) *Soit $G = \langle V, F, Sn, Sl \rangle$ une RRG-signature tel que $V = \{1, \ldots, n\}$. Un état s est un n-uplet $(s_1, \ldots, s_n) \in \prod_{i=1}^{n} \{0, \ldots, b_i\}$ où s_i représente le niveau d'expression de la variable i. L'espace d'états relatif à G est noté S_G.*

Exemple 5.4 *Si on reprend l'exemple 5.2, les variables* 1 *et* 2 *ont respectivement* 2 *et* 3 *niveaux d'expression :* 0 *ou* 1 *pour la variable* 1*, et* 0*,* 1 *ou* 2 *pour la variable* 2*. D'où l'espace d'états sera donc $S_{G_1} = \{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2)\}$.*

4.4 Les ressources

Pour définir la dynamique associée à un réseau de régulation génétique G, nous avons commencé par définir un état du système comme étant la donnée des niveaux d'expression de chacun des gènes. Supposons que le réseau se retrouve à un état s, et que (i, j) soit une interaction de G, alors i est *une ressource* de j à l'état s si :

- $s_i > Sl(i, j)$ et $Sn(i, j) = +$
- $s_i < Sl(i, j)$ et $Sn(i, j) = -$

Ainsi, pour un état donné, l'ensemble des ressources d'une variable, c'est-à-dire l'ensemble des sources susceptibles d'augmenter son niveau d'expression peut être déduit des seuils de chacun de ses arcs entrants ainsi que de l'état considéré.

La figure 5.6 illustre comment sont calculées les ressources lorsqu'il y a des activateurs et des inhibiteurs : l'absence d'un inhibiteur joue un rôle similaire à la présence d'un activateur.

Définition 5.4 (Ressources) *Soit G une RRG-signature. L'ensemble des ressources d'une variable i dans un état $s \in S_G$, noté par $R_{G,i}(s)$, est donné par :*

$$R_{G,i}(s) = \begin{cases} \{j \in G_i^- | (Sn(j,i) = + \text{ et } s_j \geq Sl(j,i))\} \\ \cup \\ \{j \in G_i^- | (Sn(j,i) = - \text{ et } s_j < Sl(j,i))\} \end{cases}$$

Exemple 5.5 *On déduit dans la figure 5.7 l'ensemble des ressources des deux variables* 1, 2 *dans chaque état de S_{G_1} du réseau donné dans l'exemple 5.2. Considérons par exemple la première ligne, à l'état $(0,0)$:*

- *la variable* 1 *est un activateur non effectif pour lui même, et la variable* 2 *est un inhibiteur non effectif pour la variable* 1*, donc $R_{G,1}((0,0)) = \{2\}$.*
- *la variable* 1 *est un inhibiteur non effectif pour la variable* 2*, de même la variable* 2 *est un inhibiteur non effectif pour lui même, donc $R_{G,2}((0,0)) = \{1,2\}$.*

1	2	$R_{G_1,1}$	$R_{G_1,2}$
0	0	{2}	{1,2}
0	1	∅	{1,2}
0	2	∅	{1}
1	0	{1,2}	{2}
1	1	{1}	{2}
1	2	{1}	∅

FIG. 5.7 – *Ressources de* 1 *et* 2 *dans chacune des états de* S_{G_1}.

4.5 Modèle

Étant donné une RRG-signature G, le niveau d'expression d'une variable i à un état s tend vers un niveau d'expression dépendant seulement de l'ensemble des ressources de i dans s. Pour définir cet effet on définit une application p qui associe à chaque variable i et à chaque ensemble de ressource w de i un poids attracteur donnant le niveau d'expression vers lequel évolue le niveau d'expression de i ayant w comme ressources. Dans la littérature, on appelle les $p(i,w)$ *les paramètres logiques* associés à i [125, 124, 22]. Nous appelons p un G-modèle de G.

Définition 5.5 (G-modèle) *Soit G une RRG-signature et soit $\kappa = \{(i,w) \mid i \in V \wedge w \subseteq G_i^-\}$. Un G-modèle est une application $p : \kappa \to \mathbb{N}$ telle que pour tout (i,w) dans κ, on a $p(i,w) \in \{0,\ldots,b_i\}$.*

À une RRG-signature G, on peut donc associer autant de G-modèles que d'applications p. Prenons par exemple la RRG-signature G_1 de l'exemple 5.2, les paramètres logiques associés à la variable 1 : $p(1,\emptyset)$, $p(1,\{1\})$, $p(1,\{2\})$ et $p(1,\{1,2\})$ prennent leurs valeurs dans $\{0,1\}$, alors que les paramètres logiques associés à la variable 2 : $p(2,\emptyset)$, $p(2,\{1\})$, $p(2,\{2\})$ et $p(2,\{1,2\})$ prennent leur valeur dans $\{0,1,2\}$. Ainsi, G_1 admet $2^4 * 3^4$ G_1-modèles possibles.

À une RRG-signature G on peut associer la catégorie des modèles de G.

Définition 5.6 (Catégorie des modèles associé à une RRG-signature) *Soit G une RRG-signature. La catégorie des modèles de G, notée $Mod(G)$, est la catégorie dont les objets sont les G-modèles et dont les morphismes sont restreints aux morphismes identité.*

Exemple 5.6 *Deux G_1-modèles p_1 et p_1' sont donnés par la figure 5.8.*

ressources ω	$p_1(1,\omega)$	$p_1(2,\omega)$
∅	0	0
{1}	1	1
{2}	1	1
{1,2}	1	1

ressources ω	$p_1'(1,\omega)$	$p_1'(2,\omega)$
∅	0	0
{1}	0	2
{2}	1	1
{1,2}	1	1

FIG. 5.8 – *Deux G_1-modèles : p_1 à gauche et p_1' à droite.*

Ainsi, étant donnés une RRG-signature G et un G-modèle p, $p(i, R_{G,i}(s))$ $(i = 1,\ldots,n)$ donne la valeur vers laquelle tend à évoluer le niveau d'expression de i quand le réseau est à l'état s et s'interprète comme suit :

- Si $s_i < p(i, R_{G,i}(s))$, alors cela signifie que le taux de synthèse de i est suffisamment important pour que le niveau d'expression de i augmente.
- Si $s_i > p(i, R_{G,i}(s))$, alors cela signifie que le taux de synthèse de i est trop faible pour maintenir le niveau d'expression de i à sa valeur actuelle. Puisque les entités biologiques considérées se dégradent, le niveau de i est amené à diminuer.
- Si $s_i = p(i, R_{G,i}(s))$, alors le taux de de synthèse de i est tel que la production de i compense exactement la dégradation de i, le niveau d'expression de i reste alors constant.

En résumé, à l'état s, on considère que le niveau de i est en augmentation si $s_i < p(i, R_{G,i}(s))$, en diminution si $s_i > p(i, R_{G,i}(s))$, et stable si $s_i = p(i, R_{G,i}(s))$. Suivant cette interprétation, et sous l'hypothèse que les niveaux d'expression évoluent de façon synchrone, c'est à dire simultanément, un G-modèle p définit de façon triviale un système de transitions, appelé *système de transitions synchrone*, où pour chaque état s il existe au plus une transition vers l'état $s' = (p(1, R_{G,1}(s)), \ldots, p(n, R_{G,n}(s)))$, appelé *état focal*, qui peut être vu comme un état attracteur qui définit la tendance d'évolution du réseau. Par exemple, dans l'exemple 5.6, $p_1(1, R_{G_1,1}((0,0))) = 1$ et $p_1(2, R_{G_1,2}((0,0))) = 1$, donc l'état focal de $(0,0)$ est l'état $(1,1)$ (cf. figure 5.9 à gauche).

Cependant, cette dynamique synchrone présente au moins deux inconvénients comme l'ont remarqué R. Thomas et M. Kauffman [125, 126] :

- considérons par exemple la transition diagonale de l'état $(0,0)$ à l'état $(1,1)$ dans le système de transitions synchrone de la figure 5.9. Cette transition change simultanément le niveau d'expression des deux variables 1 et 2. Cela implique que les délais de production de 1 et 2 sont les mêmes. Cette hypothèse est peu probable, car l'activation ou l'inhibition d'un gène est due à différents phénomènes de synthèse et de dégradation des protéines qui prennent un certain temps. La transition diagonale de l'état $(0,0)$ à l'état $(1,1)$ n'est donc pas représentative de la réalité biologique. En plus, il n'est pas possible de dire laquelle des deux variables 1 ou 2 passera son seuil en premier.
- considérons maintenant la transition de l'état $(1,2)$ à l'état $(1,0)$ dans le système de transitions synchrone de la figure 5.9. Cette transition implique que le niveau d'expression de la variable 2 diminue de 2 unités en une seule étape.

Pour ces raisons, une autre dynamique, appelée dynamique asynchrone, a été introduite [125, 124, 126]. Dans cette dynamique, toute transition diagonale est remplacée par la l'ensemble des transitions qui modifient seulement l'une des variables impliquées à la fois, et dans laquelle chaque transition fait évoluer les niveaux d'expression de façon graduelle.

4.6 Dynamique asynchrone

En dynamique asynchrone, au plus un gène met à jour son niveau d'expression d'une unité : à un état s donné, pour chaque variable i dont le niveau d'expression s_i est différent de $p(i, R_{G,i}(s))$, il existe une transition permettant au système d'atteindre l'état que l'on obtient :

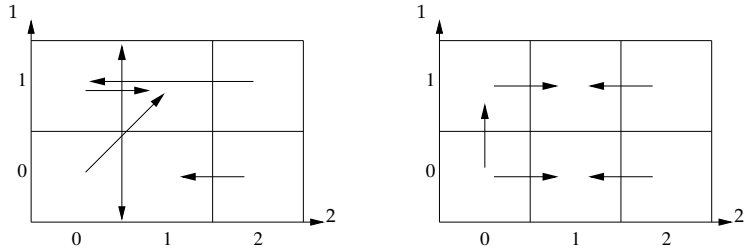


FIG. 5.9 – *Dynamiques de G_1 déduites de p_1 : à gauche la dynamique synchrone et à droite la dynamique asynchrone.*

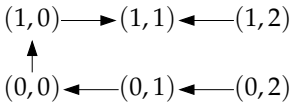


FIG. 5.10 – *Le graphe de transitions asynchrone relatif au modèle p_1.*

- en augmentant d'une unité la $i$ème composante s_i et en laissant inchangées les autres composantes si $s_i < p(i, R_{G,i}(s))$,
- en diminuant d'une unité la $i$ème composante s_i et en laissant inchangées les autres composantes si $s_i > p(i, R_{G,i}(s))$

Par exemple et comme le montre la dynamique asynchrone de la figure 5.9, à l'état $(0,0)$, le système peut passer à l'état $(1,0)$ ou à l'état $(0,1)$, et à l'état $(1,2)$ il peut passer à l'état $(1,1)$.

La dynamique asynchrone est représentée par un graphe, appelé *graphe de transitions asynchrone* et contient tous les chemins possibles d'évolution du réseau, et donc un nombre de transitions important. Dès que le nombre de gènes est assez important, le graphe de transitions asynchrone devient très grand, et est donc difficilement manipulable.

Définition 5.7 (Graphe de transitions asynchrone) *Soit G une RRG-signature et soit p un G-modèle. Un graphe de transitions asynchrones de p est un graphe orienté $GTA(G,p) = (S_G, T)$ tel que :*

$\forall s, s' \in S_G$, $(s, s') \in T$ si et seulement si :

- *il existe $i \in V$, tel que*

$$s'_i = \begin{cases} s_i + 1 \text{ et } s_i < p(i, R_{G,i}(s)) \\ ou \\ s_i - 1 \text{ et } s_i > p(i, R_{G,i}(s)) \end{cases}$$

- *pour tout $j \in V \setminus \{i\}$ $s'_j = s_j$.*

Exemple 5.7 *La figure 5.10 représente le graphe de transitions asynchrone relatif au G_1-modèle p_1 donné dans l'exemple 5.6.*

Lorsque plusieurs transitions partent d'un même état, elles sont concurrentes et chacune d'entre elles peut être choisie. Les états focaux du système de transitions synchrone restent des états focaux du graphe de transitions asynchrone, mais les chemins partant de l'état courant en direction de son état focal diffèrent. Dans le cas synchrone, une seule transition existe et elle atteint directement l'état focal. Dans le cas asynchrone, plusieurs transitions peuvent coexister, elles permettent d'atteindre seulement des états voisins qui eux-mêmes ont leurs propres états focaux. Le

comportement de la dynamique asynchrone diffère ainsi de celui de la dynamique synchrone (voir Figure 5.9). Alors qu'un état focal dans une dynamique synchrone est toujours atteignable, il ne l'est pas toujours dans la dynamique asynchrone.

Ainsi, le choix de la dynamique n'est pas anodin. Dans cette thèse, et pour les raisons déjà avancées, nous choisissons d'étudier les réseaux de régulation avec une dynamique asynchrone de façon à examiner tous les cas possibles d'évolution des réseaux.

4.7 Formulation de graphe de transitions asynchrone en une structure de Kripke

Considérons une RRG-signature G et un G-modèle p. Pour faire du graphe de transitions asynchrone $GTA(G,p)$ une structure de Kripke, il suffit de munir l'espace d'états S_G d'une fonction d'étiquetage qui indique pour chaque état $s \in S_G$ les propositions atomiques qui sont vraies dans cet état. Les propositions atomiques que nous considérons portent sur les niveaux d'expressions des variables de G et qui sont définies dans la définition des formules 5.2.

Définition 5.8 (Fonction d'étiquetage) *Soit G une RRG-signature et soit p un G-modèle. La fonction d'étiquetage associée au graphe de transitions asynchrone $GTA(G,p)$ est la fonction $L : S_G \to 2^{Atome(G)}$ définie par :*

$$\forall s \in S_G, L(s) = \{(x_i \sim l) \in Atome(G) \mid s_i \sim l\}$$

La structure de Kripke associée au graphe de transitions asynchrone $GTA(G,p)$, avec $GTA(G,p) = (S_G, T)$, est alors donnée par (S_G, T, L).

4.8 Relation de satisfaction

Nous avons vu en section 4.7 comment faire d'un graphe de transitions asynchrone une structure de Kripke. Nous avons aussi vu en section 4.2 comment exprimer des propriétés sur la dynamique d'un réseau de régulation génétique en formules FITL-X. Ainsi, étant donnés une RRG-signature G, un G-modèle p, et une formule φ dans $For(G)$, il est maintenant possible d'utiliser la relation de satisfaction pour FITL-X, donnée en définition 3.24, pour définir la relation de satisfaction de la formule χ par le G-modèle p.

Définition 5.9 (Relation de satisfaction) *Soient G une RRG-signature, p un G-modèle et (S_G, T, L) la structure de Kripke relative au graphe de transitions asynchrone de p et soit $\varphi \in For(G)$. La relation de satisfaction de φ par p, notée $\models_G$, est définie par :*

$$p \models_G \chi \iff (S_G, T, L) \models^{fitl}_{Atome(G)} \chi$$

Lorsque le contexte sera clair, on notera $\models^{fitl}_{Atome(G)}$ par $\models$.

4.9 Exploitation

Nous avons vu dans les sections précédentes qu'à un réseau de régulation génétique, on peut associer plusieurs modèles possibles vérifiant ou non les observations biologiques, et qu'à partir de chaque modèle, on peut associer une structure de Kripke. Nous avons également vu qu'il est possible d'exprimer les observations biologiques en formules des logiques temporelles, notamment FITL-X. Une fois que ces observations sont exprimées en logique temporelle, elles peuvent alors être vérifiées automatiquement sur un modèle possible à l'aide des algorithmes de *model checking*. L'étape de la validation d'une dynamique peut donc être automatisée, ce qui a ramené Bernot et collègues à développer un prototype, appelé SMBionet [22, 108]. SMBionet prend en entrée le graphe d'interactions du réseau de régulation génétique en question et une formule de CTL, et retourne l'ensemble des dynamiques associées au réseau qui vérifient la formule. La génération des dynamiques est faite suivant l'énumération de tous les paramètres logiques, et la confrontation des dynamiques avec les formules CTL est réalisée avec le model checker NuSMV [25]. La génération de chaque système de transitions asynchrone est réalisée non pas par SMBionet, mais plutôt par NuSMV qui utilise une représentation symbolique plutôt qu'explicite du système de transitions à analyser, notamment à l'aide de diagrammes de décision binaires, ce qui permet de diminuer parfois considérablement la taille de l'espace mémoire nécessaire [31] et de contourner ainsi le problème d'explosion de nombre d'états qui est surtout lié à la grande taille des réseaux de régulation étudiés. SMbionet permet donc de vérifier des propriétés sur des réseaux de grande taille, bien que dans ce cas l'énumération de toutes les dynamiques est impossible. En effet, si on reprend l'exemple 5.4 composé seulement de quatre gènes, on trouve à peu près $3^8 * 4^4 * 2^8 * 2^4 = 6879707136$ modèles possibles. Néanmoins, l'outil a été utilisé pour l'analyse d'un ensemble de réseaux potentiels, par exemple la régulation de la sécrétion de mucus chez le Pseudomonas aeruginosa [22] et a prouvé son efficacité pour ce type de réseaux.

4.10 Dynamique et structure des graphes de régulation

Pour étudier les réseaux de régulation génétique, d'autres travaux de recherches se sont focalisés aux liens qui peuvent exister entre la structure des réseaux de régulation génétique et leurs propriétés dynamiques. De tels liens entre la dynamique et les graphes d'interactions existent : les règles de R. Thomas en sont un bel exemple.

Au début des années 80, R. Thomas a énoncé deux conjectures permettant de faire un lien entre le graphe d'interactions d'un réseau et ses propriétés dynamiques [123]. La première stipule que la présence d'un circuit positif, c'est à dire contenant un nombre pair d'interactions négatives, dans le graphe d'interactions d'un réseau est une condition nécessaire pour la présence de plusieurs états stables. La seconde conjecture de Thomas stipule que la présence d'un circuit négatif, c'est à dire contenant un nombre impair d'interactions négatives, dans le graphe d'interactions du réseau, est une condition nécessaire pour la présence d'oscillations stables.

Ces deux types de propriétés dynamiques correspondent à d'importants phénomènes biologiques : la différenciation cellulaire dans le premier cas et l'homéostasie ou des comportements périodiques (comme le cycle cellulaire ou le rythme cardiaque) dans le second cas.

Partant de ce constat, pour étudier les propriétés d'un réseau de régulation génétique, nous avons choisi de revenir aux graphes d'interactions et de chercher des liens susceptibles d'exister entre la structure de ces graphes et leurs propriétés dynamiques.

Dans le chapitre qui suit, nous allons étudier des conditions sous lesquelles lorsqu'on plonge un sous-réseau dans un réseau global, les propriétés associées au sous-réseau pris en isolation restent valides dans le réseau l'englobant.

Plongement des réseaux de régulation et préservation de propriétés

6

Sommaire

Dans ce chapitre, nous allons étudier la structuration des réseaux de régulation génétique par le plongement d'un réseau dans un autre plus large. Ce choix de structuration est entièrement guidé par la volonté d'identifier des conditions de plongement sous lesquelles la préservation des propriétés, telles que nous les avons définies en section 4.2 du chapitre 5, est assurée au travers de ce plongement.

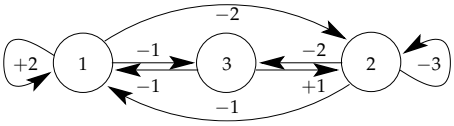


FIG. 6.1 – *RRG-signature G_4.*

1 PLONGEMENT

Le plongement d'une RRG-signature G dans une RRG-signature G' que nous considérons n'est pas une simple inclusion de graphes. En effet, de nouvelles variables peuvent être ajoutées. Cet ajout a pour conséquence immédiate de modifier les valeurs des seuils dans G'. Pour cette raison, nous choisissons non pas de préserver les seuils par plongement, mais plutôt de préserver l'ordre des seuils. Ceci est assuré par les conditions 3 et 4 de la définition 6.1.

Définition 6.1 (Plongement) *Soient G et G' deux RRG-signatures. On dit que G est **plongée** dans G', noté par $G \hookrightarrow G'$, si :*

1. $V \subseteq V'$ *et* $F = F' \cap V \times V$
2. $\forall (i,j) \in F$, $Sn(i,j) = Sn'(i,j)$
3. $\forall i \in V, \forall j,k \in G_i^+$, $Sl(i,j) = Sl(i,k) \Leftrightarrow Sl'(i,j) = Sl'(i,k)$
4. $\forall i \in V, \forall j,k \in G_i^+$, $Sl(i,j) < Sl(i,k) \Leftrightarrow Sl'(i,j) < Sl'(i,k)$

Exemple 6.1 *La figure 6.1 présente une RRG-signature G_4 qui partage les deux variables 1 et 2 avec la RRG-signature G_1 de l'exemple 5.2 et qui a une variable N de plus. Il est facile de voir que $G_1 \hookrightarrow G_4$. En effet, les conditions 1 et 2 sont clairement vérifiées puisque toutes des variables de G_1 sont dans G_4 et que tous les arcs de G_4 qui représentent les interactions entre des variables de G_1 sont aussi présents dans G_1, en plus les arcs préservent les signes. La condition 3 impose que l'égalité des seuils d'arcs sortants de chaque variable dans G_1 soit préservée dans G_4. Dans G_1, on a $Sl_1(1,1) = Sl_1(1,2)$, ce qui est préservé dans G_4 puisque $Sl_4(1,1) = Sl_4(1,2)$. Finalement, la condition 4 impose que l'ordre des seuils d'arcs sortants de chaque variable dans G_1 soit préservé dans G_4. Par exemple, dans G_1, 2 a deux arcs sortants $(2,1)$ et $(2,2)$ tels que $Sl_1(2,1) < Sl_1(2,2)$. Dans G_4, l'ordre est préservé puisque $Sl_4(2,1) < Sl_4(2,2)$.*

Définition 6.2 (Catégorie des RRG-signatures) *La catégorie des RRG-signatures, notée Sig est la catégorie dont les objets sont les RRG-signatures et dont les morphismes sont les plongements de signatures.*

Nous allons voir en section 5 que le plongement tel qu'il est défini dans la définition 6.1 ne nous permet pas d'avoir des résultats de préservation de propriétés. Pour ce, nous avons imposé des conditions supplémentaires sur le plongement et nous avons défini deux types de plongement : *le plongement strict* et *le plongement monotone*. Le choix de plongement strict est surtout motivé par une intuition biologique stipulant que l'absence de nouvelles ressources, donné par la condition donnée dans la définition 6.3, n'affecte pas le comportement du réseau, alors que le choix de plongement monotone est motivé par le fait que l'ajout d'une variable i par plongement d'un réseau G dans un réseau G' préserve un certain comportement de G et G' si,

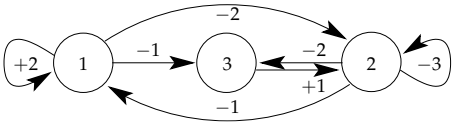


FIG. 6.2 – *RRG-signature G_2.*

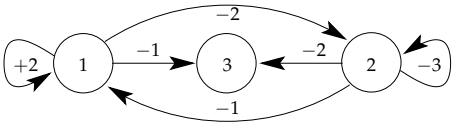


FIG. 6.3 – *RRG-signature G_3.*

- i a une régulation positive (+) sur toutes ses cibles de G' qui sont aussi des variables de G et son niveau d'expression est minimal, c'est à dire égal à 0, ou
- i a une régulation négative (-) sur toutes ses cibles de G' qui sont aussi des variables de G et son niveau d'expression est maximal, c'est-à-dire égal à b'_i,

Définition 6.3 (Plongement strict et monotone) *Soit G et G' deux RRG-signatures telles que G est plongée dans G'. Le plongement $G \hookrightarrow G'$ est dit*

- *strict si*

$$\forall (i,j) \in F', \qquad j \in V \Rightarrow i \in V.$$

- *monotone si*

$$\forall i \in V' \setminus V,\ \forall j,k \in V, \quad (i,j) \in F' \wedge (i,k) \in F' \Rightarrow Sn'(i,j) = Sn'(i,k).$$

Ainsi, un plongement strict $G \hookrightarrow G'$ impose l'absence de nouvelles régulations sur les variables de G dans G', alors qu'un plongement monotone autorise l'introduction de nouvelles régulations sur les variables de G dans G' à condition que ces régulations soient de même type (inhibition ou activation).

Exemple 6.2 *Si on considère la RRG-signature G_2 illustrée par la figure 6.2 et la RRG-signature G_1 de l'exemple 5.2, le plongement $G_1 \hookrightarrow G_2$ est non strict puisqu'il existe une variable 3 de G_2 n'appartenant pas G_1 qui régule la variable 2 de G_1. Néanmoins, puisque la variable 3 a seulement un arc sortant, ce plongement est monotone. Maintenant, étant donnée la RRG-signature G_3 illustrée par la figure 6.3 et qui est obtenue en omettant de G_2 l'arc sortant de 3, le plongement $G_1 \hookrightarrow G_3$, est clairement strict.*

Maintenant, si on considère la RRG-signature G_4 illustrée par la figure 6.1, le plongement $G_1 \hookrightarrow G_4$ n'est pas monotone, puisque la condition du plongement monotone est violée. En effet la variable 3 agit sur la variable 2 en l'activant ($Sl(3,2) = +$) et sur la variable 1 en l'inhibant ($Sl(3,1) = -$).

Remarque 6.1 *Tout plongement strict est monotone.*

2 Renommage des formules par plongement de signatures

Étant donné un plongement $G \hookrightarrow G'$, ce plongement induit un renommage de propositions atomiques et puis de formules intuitivement en prenant en considération le changement des seuils par le plongement.

Définition 6.4 (Renommage des propositions atomiques par plongement des signatures) *Soient G et G' deux RRG-signatures telles que G est plongée dans G'. Pour tout $i \in V$, on définit l'application injective $\sigma_i : \{0, 1, \ldots, b_i\} \rightarrow \{0, 1, \ldots, b'_i\}$ de la façon suivante :*

- $\sigma_i(0) = 0$,
- *Pour tout $l \neq 0$, $\sigma_i(l) = Sl'(i, j)$, où j une variable de G_i^+ telle que $Sl(i, j) = l$.*

Le renommage des propositions atomiques par $G \hookrightarrow G'$, *noté $\sigma : Atome(G) \rightarrow For(G')$ est définie pour toute proposition atomique $a \in Atome(G)$ de la façon suivante :*

- *si a vaut $(x_i = l)$ avec $l \neq b_i$, alors $\sigma(a)$ vaut $(x_i \geq \sigma_i(l) \wedge x_i < \sigma_i(l + 1))$*
- *si a vaut $(x_i = b_i)$, alors $\sigma(a)$ vaut $(x_i \geq \sigma_i(b_i))$*
- *si a vaut $(x_i > l)$, alors $\sigma(a)$ vaut $(x_i \geq \sigma_i(l + 1))$*
- *si a vaut $(x_i < l)$, alors $\sigma(a)$ vaut $(x_i < \sigma_i(l))$*

Le renommage des formules par plongement découle directement du renommage des propositions atomiques.

Définition 6.5 (Renommage des formules par plongement des signatures) *Soient G et G' deux RRG-signatures telles que G est plongée dans G' et soit $\sigma : Atome(G) \rightarrow For(G')$ le renommage des propositions atomiques par ce plongement.*

Le renommage des formules par $G \hookrightarrow G'$, *noté $\bar{\sigma} : For(G) \rightarrow For(G')$, est défini pour toute formule χ inductivement sur la forme de χ de la façon suivante :*

- *si χ est une proposition atomique $p \in Atome(G)$, alors $\bar{\sigma}(\chi) = \sigma(p)$;*
- *si $\chi \in For^s_{FITL\text{-}X}(Atome(G))$, alors,*
  - *si χ est de la forme $\neg\varphi$, avec $\varphi \in For^s_{FITL\text{-}X}(Atome(G))$, alors $\bar{\sigma}(\chi) = \neg\bar{\sigma}(\varphi)$;*
  - *si χ est de la forme $\varphi \wedge \psi$, avec $\varphi, \psi \in For^s_{FITL\text{-}X}(Atome(G))$, alors $\bar{\sigma}(\chi) = \bar{\sigma}(\varphi) \wedge \bar{\sigma}(\psi)$;*
  - *si χ est de la forme $E\pi$, avec $\pi \in For^p_{FITL\text{-}X}(Atome(G))$, alors $\bar{\sigma}(\chi) = E\bar{\sigma}(\pi)$.*
- *si $\chi \in For^p_{FITL\text{-}X}(Atome(G))$, alors,*
  - *si $\chi \in For^s_{FITL\text{-}X}(Atome(G))$, alors $\bar{\sigma}(\chi)$ est déjà défini ;*
  - *si χ est de la forme $\neg\pi$, avec $\pi \in For^p_{FITL\text{-}X}(Atome(G))$, alors $\bar{\sigma}(\chi) = \neg\bar{\sigma}(\pi)$;*
  - *si χ est de la forme $\pi \wedge \tau$, avec $\pi, \tau \in For^p_{FITL\text{-}X}(Atome(G))$, alors $\bar{\sigma}(\chi) = \bar{\sigma}(\pi) \wedge \bar{\sigma}(\tau)$;*
  - *si χ est de la forme $\pi U \tau$, avec $\pi, \tau \in For^p_{FITL\text{-}X}(Atome(G))$, alors $\bar{\sigma}(\chi) = \bar{\sigma}(\pi) U \bar{\sigma}(\tau)$;*

Exemple 6.3 *Considérons le plongement $G_1 \hookrightarrow G_2$ de l'exemple 6.2. Nous avons $\sigma_1(1) = 2$ et $\sigma_1(2) = 3$. Donc $\sigma(x_1 = 1)$ vaut $x_1 \geq 2 \wedge x_i < 3$. En fait, pour ce plongement, le seuil 1 est changé en intervalle $[2; 2]$ contenant l'unique valeur de*

2. *Pour la variable* 2, *nous avons* $\sigma_2(1) = 1$ *et* $\sigma_2(2) = 3$. *Ainsi* $\sigma(x_2 = 1)$ *vaut* $x_1 \geq 1 \wedge x_i < 3$ *qui correspond à l'intervalle* $[1;2]$ *contenant les deux valeurs* 1 *et* 2.

Fait 6.1 *σ_i est strictement croissante : si $l < l'$ alors $\sigma_i(l) < \sigma_i(l')$.*

Preuve. C'est évident pour $l = 0$. Pour $l \neq 0$, par définition même de la signature, il existe (i,j) et (i,k) dans F tels que $l = Sl(i,j)$ et $l' = Sl(i,k)$. Alors, $\sigma_i(l) = Sl'(i,j)$ et $\sigma_i(l') = Sl'(i,k)$, et il suffit de remarquer que, par définition du plongement des signatures, si $Sl(i,j) < Sl(j,k)$ alors $Sl'(i,j) < Sl'(i,k)$. □

On peut maintenant définir le foncteur *For* qui associe donc à chaque RRG-signature son ensemble de formules, et à chaque plongement de signatures le renommage des formules par ce prolongement.

Définition 6.6 (Foncteur *For*) *$For : Sig \to Set$ est le foncteur :*
- *qui associe à chaque RRG-signature $G \in |Sig|$ l'ensemble $For(G)$ des formules sur G ;*
- *qui associe à chaque plongement $G \hookrightarrow G'$ le renommage des formules $\bar{\sigma} : For(G) \to For(G')$ par ce plongement.*

3 Modèle réduit

Étant donné un plongement $G \hookrightarrow G'$, ce plongement est transporté sur les modèles en un foncteur $Mod(G') \to Mod(G)$ qui permet de construire un modèle p de G à partir d'un modèle p' de G' en "oubliant" les seuils des arcs qui sont dans $F' \setminus F$. La construction de chaque $p(i,w)$ se fait en fonction de $p'(i,w)$ de la façon suivante :
- si $p'(i,w)$ est égal au seuil d'un certain arc sortant de i, c'est à dire $Sl'(i,j)$, avec $j \in V'$, alors
  - si $(i,j) \in F$, alors $p(i,w)$ est égale à $Sl(i,j)$,
  - Sinon, on cherche l'arc $(i,j) \in F$ tel que $Sl'(i,j)$ soit le plus grand seuil inférieur à $p'(i,w)$. Si cet arc existe alors $p(i,w)$ sera alors égale à $Sl(i,j)$, sinon $p(i,w)$ est égale à 0.
- sinon $p(i,w)$ est égale à 0.

Définition 6.7 (Modèle réduit) *Soient G et G' deux RRG-signatures telles que G est plongée dans G', et soit p' un G'-modèle. Le* G-modèle réduit *p de p' est défini de la façon suivante*[1] *:*

$$\forall i \in V, \forall w \subseteq G_i^-, \quad p(i,w) = \max\{l \in \{0,\dots,b_i\} \mid \sigma_i(l) \leq p'(i,w)\}$$

Exemple 6.4 *Les figures 6.4 et 6.5 donnent, respectivement, le G_1-modèle réduit de G_2-modèle p_2, noté par p_{12}, et le G_1-modèle réduit de G_3-modèle p_3, noté par p_{13}, par les plongements respectifs $G_1 \hookrightarrow G_2$ et $G_1 \hookrightarrow G_3$ (voir exemple 6.2).*

La figure 6.6 illustre de gauche à droite les graphes de transitions asynchrones associés respectivement au G_1-modèle p_{12}, et au G_2-modèle p_2, alors que la figure 6.7 illustre de gauche à droite les graphes de transitions asynchrones associés respectivement au G_1-modèle p_{13}, et au G_3-modèle p_3

[1] Étant donné un ensemble fini, la fonction $max(E)$ donne simplement le plus grand élément de E.

ressources ω'	$p_2(1,\omega')$	$p_2(2,\omega')$	$p_2(3,\omega')$
$\emptyset$	0	0	0
{1}	2	2	1
{2}	2	1	1
{3}		1	
{1,2}	2	2	1
{1,3}		3	
{2,3}		3	
{1,2,3}		3	

ressources ω	$p_{12}(1,\omega)$	$p_{12}(2,\omega)$
$\emptyset$	0	0
{1}	1	1
{2}	1	1
{1,2}	1	1

FIG. 6.4 – *À gauche un G_2-modèle p_2 et à droite son modèle réduit G_1-modèle p_{12}.*

ressources ω'	$p_3(1,\omega')$	$p_3(2,\omega')$	$p_3(3,\omega')$
$\emptyset$	0	0	0
{1}	2	2	0
{2}	2	1	0
{1,2}	2	3	0

ressources ω	$p_{13}(1,\omega)$	$p_{13}(2,\omega)$
$\emptyset$	0	0
{1}	1	1
{2}	1	1
{1,2}	1	2

FIG. 6.5 – *À gauche un G_3-modèle p_3 et à droite son modèle réduit G_1-modèle p_{13}.*

4 Préservation des chemins

Dans cette section, nous considérons deux RRG-signatures G et G' telles que $G \hookrightarrow G'$, nous considérons aussi un G'-modèle p' et son G-modèle réduit p, et nous allons étudier l'équivalence de traces des deux modèles p et p'. Autrement dit, nous allons étudier la préservation des chemins entre les graphes de transitions asynchrones $GTA(G', p')$ et $GTA(G, p)$. Ceci n'est pas simple puisque les deux graphes sont différents en états, et en transitions. Pour ce faire, commençons tout d'abord par définir une partition sur l'espace d'états de $GTA(G', p')$.

4.1 Partition

Tout au long de cette section, G et G' dénotent deux RRG-signatures telles que G est plongée dans G'. Nous allons construire une partition de l'espace d'états de $GTA(G', p')$ en régions dans lesquelles tous les états d'une même région satisfont les mêmes formules obtenues par renommage des propositions atomiques $Atome(G)$ par le plongement $G \hookrightarrow G'$.

Définition 6.8 (Partition) *Une partition de l'ensemble d'états S'_G est donnée par l'application $B : S_G \to 2^{S_{G'}}$ définie de la façon suivante :*
pour tout $s \in S_G$, $B(s)$ est l'ensemble d'états s' dans $S_{G'}$ vérifiant, pour tout

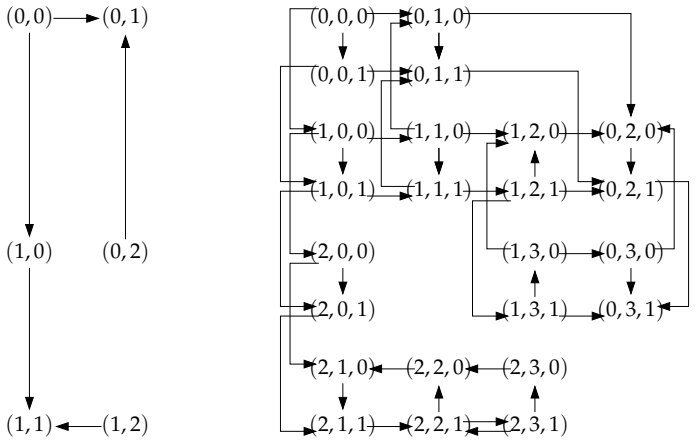


FIG. 6.6 – *Graphes de transitions asynchrones $GTA(G_1, p_{12})$ (à gauche) et $GTA(G_2, p_2)$ (à droite).*

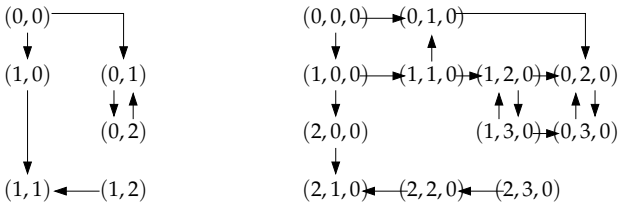


FIG. 6.7 – *Graphes de transitions asynchrones $GTA(G_1, p_{13})$ (à gauche) et $GTA(G_3, p_3)$ (à droite).*

i dans V :

$$\sigma_i(s_i) \leq s'_i < \sigma_i(s_i + 1)$$

si $s_i < b_i$, *et*

$$\sigma_i(s_i) \leq s'_i$$

sinon.

Remarque 6.2 *La définition de B est inspirée du renommage des propositions atomiques $Atome(G)$ par plongement de sorte que,*
$\forall s \in S_G,\ \forall s' \in B(s),\ \forall \varphi \in Atome(G),$

$$((S_G, T, L), s) \models \varphi \iff ((S_{G'}, T', L'), s') \models \bar{\sigma}(\varphi)$$

Proposition 6.1 *L'application B définit une partition de $S_{G'}$.*

Preuve. Pour commencer nous montrons que, pour tout $s, t \in S_G$, si $s \neq t$ alors $B(s) \cap B(t) = \emptyset$. En effet, si $s \neq t$ il existe $i \in V$ telle que $s_i \neq t_i$. Supposons que $s_i < t_i$. Alors, $s_i + 1 \leq t_i$ et d'après le fait 6.1,

$$\sigma_i(s_i + 1) \leq \sigma_i(t_i)$$

Si $s' \in B(s)$ alors

$$s'_i < \sigma_i(s_i + 1) \leq \sigma_i(t_i)$$

ainsi $s' \notin B(t)$. Si $s' \in B(t)$ alors

$$\sigma_i(s_i + 1) \leq \sigma_i(t_i) \leq s'_i$$

ainsi $s' \notin B(s)$. Donc $B(s) \cap B(t)$ est vide. Il reste à prouver que

$$\cup_{s \in S_G} B(s) = S_{G'}$$

Puisque $B(s) \subseteq S_{G'}$ pour tout $s \in S_G$, il suffit d'observer que, étant donné un état s' dans $S_{G'}$, nous avons $s' \in B(s)$ pour l'état $s \in S_G$ défini par, pour tout $i \in V$,

$$s_i = \max\{l \in \{0, \ldots, b_i\} \mid \sigma_i(l) \leq s'_i\}$$

□

De plus, pour tout $s' \in B(s)$, si (r, i) est arc de F, alors r est une ressource de i dans s' si et seulement si elle l'est dans s.

Proposition 6.2 *Pour tout $s \in S_G$, pour tout $s' \in B(s)$ et pour tout $(r, i) \in F$:*

$$r \in R_{G,i}(s) \iff r \in R_{G',i}(s')$$

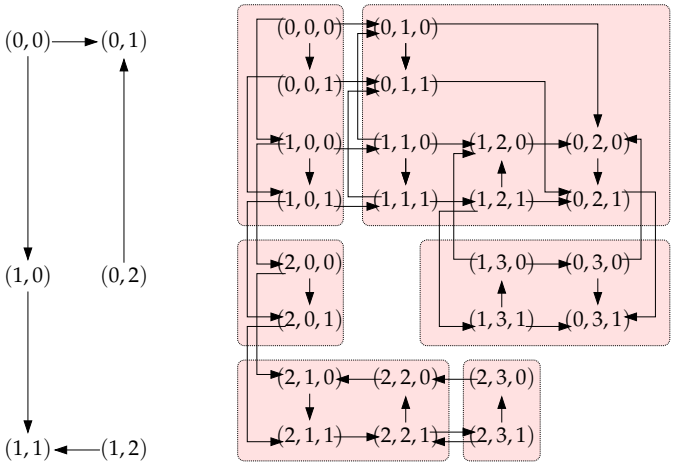


FIG. 6.8 – *Les boites colorées représentent les classes d'équivalence de* $GTA(G_2, p_2)$ *par* $\simeq$.

Preuve. Puisque $Sn(r,i) = Sn'(r,i)$, il suffit de prouver que

$$Sl(r,i) \leq s_r \iff Sl'(r,i) \leq s'_r$$

Supposons que

$$Sl(r,i) \leq s_r$$

Alors, par définition de σ_r, le fait 6.1 et $s' \in B(s)$, nous pouvons déduire que

$$Sl'(r,i) = \sigma_r(Sl(r,i)) \leq \sigma_r(s_r) \leq s'_r$$

Maintenant, supposons que $Sl(r,i) > s_r$. Alors

$$Sl(r,i) \geq s_r + 1$$

et par définition de σ_r, le fait 6.1 et le fait que $s' \in B(s)$, nous déduisons que

$$Sl'(r,i) = \sigma_r(Sl(r,i)) \geq \sigma_r(s_r + 1) > s'_r$$

□

En utilisant la partition B, on peut définir une relation d'équivalence sur l'espace d'états $S_{G'}$.

Proposition 6.3 *La relation binaire* $\simeq$ *sur* $S_{G'}$ *définie de la façon suivante :*

$$\forall s'_1, s'_2 \in S_{G'}, \quad s'_1 \simeq s'_2 \iff B^{-1}(s'_1) = B^{-1}(s'_2)$$

est une relation d'équivalence.

Preuve. Il est simple de vérifier que $\simeq$ est réflexive, symétrique et transitive. □

Exemple 6.5 *Les quotients par* $\simeq$ *de* (S_{G_2}, T_2) *et* (S_{G_3}, T_3) *associés respectivement à* $GTA(G_2, p_2)$ *et* $GTA(G_3, p_3)$ *de l'exemple 6.4 sont donnés par les figures 6.8 et 6.9.*

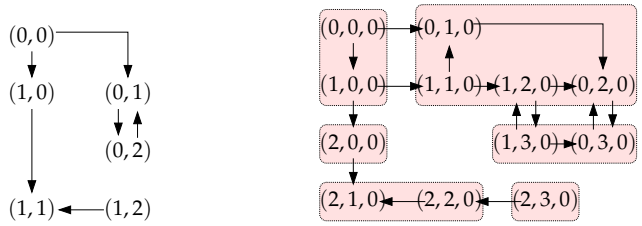


FIG. 6.9 – *Les boites colorées représentent les classes d'équivalence de* $GTA(G_3, p_3)$ *par* $\simeq$.

Nous allons maintenant étudier la relation entre les deux graphes de transitons asynchrones $GTA(G, p)$ et $GTA(G', p')$ dans la but d'établir la préservation de chemins entre ces deux graphes. Nous allons procéder de la manière suivante :

- nous allons tout d'abord prouver que si le plongement $G \hookrightarrow G'$ est monotone, et si $(s_0, s_1, \ldots, s_n)$ est un chemin de $GTA(G, p)$, alors $GTA(G', p')$ contient un chemin passant successivement par les classes $B(s_0), B(s_1), \ldots, B(s_n)$. Ceci implique que tout chemin fini de $GTA(G, p)$ est préservé dans le graphe quotient $GTA(G', p')_{/\simeq}$.
- ensuite, nous allons montrer un résultat plus fort dans le cas où le plongement $G \hookrightarrow G'$ est strict. En effet, nous allons prouver dans ce cas que, si $(s_0, s_1, \ldots, s_n)$ est un chemin de $GTA(G, p)$, alors $GTA(G', p')$ contient, pour tout état $s' \in B(s_0)$, un chemin commençant en s' et passant successivement par les classes $B(s_0), B(s_1), \ldots, B(s_n)$. En outre, nous allons prouver que les transitions entre les différents états de $GTA(G, p)$ et $GTA(G', p')_{/\simeq}$ sont préservées.

4.2 Préservation des chemins par plongement monotone

Dans cette section, nous allons étudier la préservation des chemins par plongement monotone. Rappelons que pour un plongement monotone $G \hookrightarrow G'$, pour chaque variable $i \in V'$ qui n'appartient pas à V, les arcs sortants de i, et ayant pour cibles des variables de V' qui sont aussi dans V, ont tous le même signe. Si ce signe est positif (resp. négatif), alors i n'est ressource d'aucune variable de G si son niveau d'expression est minimal (resp. maximal). En d'autres termes, il est toujours possible de fixer un niveau d'expression de chaque variable agissant sur les variables de G' qui sont aussi des variables de G de sorte que toutes les interactions correspondantes soient neutres.

Tout d'abord, définissons, pour chaque $s \in S_G$, un sous-ensemble $B'(s)$ de $B(s)$ dans lequel les variables ajoutées par plongement et agissant sur des variables de G', qui sont aussi dans G, ne sont ressources d'aucune de ces variables.

Définition 6.9 *Soit $G \hookrightarrow G'$ un plongement monotone. L'application $B' : S_G \to 2^{S_{G'}}$ définie de la façon suivante : pour tout $s \in S_G$, $B'(s)$ est l'ensemble d'états $s' \in B(s)$ tel que pour tout $(i, j) \in F'$ avec $i \in V' \setminus V$ et $j \in V$,*

$$s'_i = \begin{cases} 0 & si\ Sn'(i,j) = + \\ b'_i & si\ Sn'(i,j) = - \end{cases}$$

Remarque 6.3 *Si le plongement $G \hookrightarrow G'$ est strict alors pour tout $s \in S_G$, $B'(s) = B(s)$.*

Lemme 6.1 *Si $G \hookrightarrow G'$ est un plongement monotone, alors pour tout $s \in S_G$, pour tout $s' \in B'(s)$ et pour tout $i \in V$, on a $R_{G',i}(s') = R_{G,i}(s)$.*

Preuve. D'après la proposition 6.2, nous avons

$$R_{G,i}(s) = R_{G',i}(s') \cap V \tag{6.1}$$

Il suffit de prouver que $R_{G',i}(s') \subseteq V$. Soit $r \in R_{G',i}(s')$. Si $Sn'(r,i) = +$ alors $s'_r \geq Sl'(r,i) > 0$, et puisque $s' \in B'(s)$, nous déduisons que $r \in V$. Donc, si $Sn'(r,i) = -$, nous avons $s'_r < Sl'(r',i) \leq b'_r$, et puisque $s' \in B'(s)$, nous déduisons que $r \in V$. □

Lemme 6.2 *Soit $G \hookrightarrow G'$ un plongement monotone, soit p' un G'-modèle et soit p le G-modèle réduit de p'.*

Pour tout $s \in S_G$, pour tout $s' \in B'(s)$ et pour tout $i \in V$,

$$s_i < p(i, R_{G,i}(s)) \Rightarrow s'_i < p'(i, R_{G',i}(s'))$$

et

$$s_i > p(i, R_{G,i}(s)) \Rightarrow s'_i > p'(i, R_{G',i}(s'))$$

Preuve. Supposons que $s_i < p(i, R_{G,i}(s))$, ainsi,

$$s_i + 1 \leq p(i, R_{G,i}(s))$$

Par le fait 6.1 et la définition de p,

$$\sigma_i(s_i + 1) \leq \sigma_i(p(i, R_{G,i}(s))) \leq p'(i, R_{G,i}(s))$$

Par le lemme 6.1 nous déduisons que

$$\sigma_i(s_i + 1) \leq \sigma_i(p(i, R_{G,i}(s))) \leq p'(i, R_{G',i}(s'))$$

et puisque $s' \in B(s)$ nous obtenons

$$s'_i < \sigma_i(s_i + 1) \leq \sigma_i(p(i, R_{G,i}(s))) \leq p'(i, R_{G',i}(s'))$$

Maintenant, supposons que

$$p(i, R_{G,i}(s)) < s_i$$

Par définition de p et du fait que $s \in B(s)$ on a

$$p'(i, R_{G,i}(s)) < \sigma_i(s_i) \leq s'_i$$

En utilisant le lemme 6.1 nous obtenons

$$p'(i, R_{G',i}(s')) < \sigma_i(s_i) \leq s'_i$$

□

Maintenant, nous pouvons énoncer le résultat fondamental de la préservation des chemins par plongement monotone.

Lemme 6.3 *Soit $G \hookrightarrow G'$ un plongement monotone, soit p' un G'-modèle et soit p le G-modèle réduit de p'.*

Si (s,t) est une transition de $GTA(G,p)$, alors pour tout $s' \in B'(s)$ il existe $t' \in B'(t)$ telle que $GTA(G',p')$ a un chemin allant de s' à t' dont les états, excepté t', appartiennent à $B'(s)$.

Preuve. Soit (s,t) une transition de $GTA(G,p)$. Par définition, il existe $i \in V$ telle que $s_j = t_j$ pour tout $j \in V \setminus \{i\}$ et

$$t_i = \begin{cases} s_i + 1 \text{ et } s_i < p(i, R_{G,i}(s)) \\ s_i - 1 \text{ et } s_i > p(i, R_{G,i}(s)) \end{cases}$$

Supposons que $s_i < p(i, R_{G,i}(s))$, la preuve est similaire pour les autres cas. Pour tout $s' \in B'(s)$, soit t' un état de $S_{G'}$ défini par :

$$t'_i = \sigma_i(t_i) \qquad \text{et} \qquad t'_j = s'_j \text{ pour tout } j \in V' \setminus \{i\}$$

Il est évident que $t' \in B'(t)$. En effet, puisque $s' \in B'(s)$ et puisque pour tout $j \in V' \setminus V$ nous avons $t'_j = s'_j$, pour prouver que $t' \in B'(t)$ il suffit de prouver que $t' \in B(t)$. Maintenant, puisque $s' \in B(s)$, et puisque pour tout $j \in V \setminus \{i\}$ nous avons $t'_j = s'_j$ et $t_j = s_j$, pour montrer que $t' \in B(t)$, il suffit de montrer que $\sigma_i(t_i) \leq t'_i$ et, si $t_i < b_i$, alors $t'_i < \sigma_i(t_i + 1)$. Puisque $\sigma_i(t_i) = t'_i$ il suffit de remarquer que, si $t_i < b_i$, alors d'après le fait 6.1, $\sigma_i(t_i) < \sigma_i(t_i + 1)$.

Ainsi pour prouver le lemme, il suffit de prouver que, pour tout $s' \in B'(s)$, il existe un chemin allant de s' à t' dont les états, excepté t', appartiennent à $B'(s)$. On procède par induction sur $d(s') = t'_i - s'_i$.

1. *Cas de base :* $d(s') = 1$. Par le lemme 6.2,

   $$s'_i < p'(i, R'_{G',i}(s'))$$

   Par $d(s') = 1$ et par définition de t', nous déduisons que (s', t') est une transition de $GTA(G', p')$.

2. *Pas de récurrence :* $d(s') > 1$. Soit s'' un état de $S_{G'}$ défini par

   $$s''_i = s'_i + 1 \quad \text{et} \quad s''_j = s'_j \text{ pour tout } j \in V' \setminus \{i\}$$

   Il est clair que, $s'' \in B'(s)$. En effet, puisque $s' \in B'(s)$ et puisque pour tout $j \in V' \setminus V$ nous avons $s''_j = s'_j$, pour prouver que $s'' \in B'(s)$ il suffit de prouver que $s'' \in B(s)$. Maintenant, puisque $s' \in B(s)$, et puisque pour tout $j \in V \setminus \{i\}$ nous avons $t'_j = s'_j$, pour prouver $s'' \in B(s)$, il suffit de prouver que $\sigma_i(s_i) \leq s''_i < \sigma_i(s_i + 1)$ (nous avons $s_i < t_i \leq b_i$). C'est évident : puisque $s' \in B(s)$, $\sigma_i(s_i) \leq s'_i < s'_i + 1 = s''_i$ et, puisque $d(s') > 1$, $s''_i = s'_i + 1 < t'_i = \sigma_i(t_i) = \sigma_i(s_i + 1)$. $d(s'') < d(s')$ et $t'' = t'$, donc, par hypothèse d'induction, il existe un chemin allant de s'' à t' dont les états, excepté t', appartiennent à $B'(s)$. Donc, par le lemme 6.2,

   $$s'_i < p'(i, R'_{G',i}(s'))$$

   et par définition de s'', (s', s'') est une transition de $GTA(G', p')$. Ainsi, il existe un chemin allant de s' à t' dont les états excepté t' appartiennent à $B'(s)$.

□

En appliquant plusieurs fois le lemme précédent, on obtient le théorème suivant.

Théorème 6.1 *Soit $G \hookrightarrow G'$ un plongement monotone, soit p' un G'-modèle et soit p le G-modèle réduit de p'.*

Si ρ est un chemin de $GTA(G,p)$, alors pour tout $s' \in B'(\rho(0))$, il existe un chemin ρ' dans $GTA(G',p')$ vérifiant :

- $\rho'(0) = s'$,
- $\forall i \in Indices(\rho)$, $\exists k_i \in Indices(\rho')$ *tel que la suite* $(k_i)_{i \in Indices(\rho')}$ *est strictement croissante, c'est-à-dire,*

$$\forall i, i' \in Indices(\rho), i < i' \Rightarrow k_i < k_{i'}$$

- $\forall j \in Indices(\rho')$,

$$(\exists i \in Indices(\rho), k_i \leq j < k_{i+1} \Rightarrow \rho'(j) \in B'(\rho(i)))$$

ou

$$(\forall i \in Indices(\rho), k_i \leq j) \Rightarrow \rho'(j) \in B'(\rho(max(Indices(\rho))))$$

On note $B'(\rho)^{s'}$ ce chemin et on note $B'(\rho)$ l'ensemble des chemins $B'(\rho)^{s'}$, avec $s' \in B'(\rho(0))$.

$$B'(\rho) = \{B'(\rho)^{s'} \mid s' \in B'(\rho(0))\}$$

Par analogie, on note $B(\rho)$ l'ensemble de tous les chemins ρ' tel que pour tout indice i de $Indices(\rho)$, il existe un indice k_i tels que,

$$\forall i, i' \in Indices(\rho), i < i' \Rightarrow k_i < k_{i'}$$

et $\forall j \in Indices(\rho')$,

$$(\exists i \in Indices(\rho), k_i \leq j < k_{i+1} \Rightarrow \rho'(j) \in B(\rho(i)))$$

ou

$$(\forall i \in Indices(\rho), k_i \leq j) \Rightarrow \rho'(j) \in B(\rho(max(Indices(\rho))))$$

4.3 Préservation de chemins par plongement strict

Maintenant, nous allons étudier la préservation de chemins par plongement strict. Intuitivement, le plongement strict ne perturbe pas le comportement du réseau plongé puisque les variables de V' qui ne sont pas dans V n'ont pas d'influences sur les variables de V (absence d'arcs sortants).

Lemme 6.4 *Soit $G \hookrightarrow G'$ un plongement strict, soit p' un G'-modèle et soit p le G-modèle réduit de p'.*

Si (s,t) est une transition de $GTA(G,p)$ alors, pour tout $s' \in B(s)$, il existe $t' \in B(t)$ tel que $GTA(G',p')$ a un chemin allant de s' à t' dont les états, excepté t', appartiennent à $B(s)$.

Preuve. Conséquence immédiate des remarques 6.1 et 6.3 et du lemme 6.3. □

En plus, pour un plongement strict, les transitions entre différents états de $GTA(G,p)$ sont préservés dans $GTA(G',p')_{/\simeq}$.

Théorème 6.2 *Soit $G \hookrightarrow G'$ un plongement strict. Soit p' un G'-modèle et soit p le G-modèle réduit de p'. Alors, (s,t) est une transition de $GTA(G,p)$ si et seulement si $(B(s),B(t))$ est une transition de $GTA(G',p')_{/\simeq}$.*

Preuve. Soit (s,t) une transition de $GTA(G,p)$, alors d'après le lemme 6.4, ils existent $s' \in B(s)$ et $t' \in B(t)$ tels que (s',t') est une transition de $GTA(G',p')$, ainsi $(B(s),B(t))$ est une transition de $GTA(G',p')_{/\simeq}$. Il reste à prouver que si $(B(s),B(t))$ est une transition de $GTA(G',p')_{/\simeq}$ alors (s,t) est une transition de $GTA(G,p)$. Maintenant, soit $(B(s),B(t))$ une transition de $GTA(G',p')_{/\simeq}$. Ils existent $s' \in B(s)$ et $t' \in B(t)$ tels que (s',t') est une transition de $GTA(G',p')$. Par définition, il existe $i \in V'$ tel que $s'_j = t'_j$ pour tout $j \in V' \setminus \{i\}$ et

$$t'_i = \begin{cases} s'_i + 1 \text{ et } s'_i < p'(i, R_{G',i}(s')) \\ s'_i - 1 \text{ et } s'_i > p'(i, R_{G',i}(s')). \end{cases}$$

Clairement, $i \in V$ car sinon s' et t' seront dans la même classe d'équivalence. Maintenant, supposons que

$$s'_i < p'(i, R_{G',i}(s')),$$

la preuve est similaire pour les autres cas. Soit $j \in V$. Si $t_j < s_j$ alors

$$t_j + 1 \leq s_j$$

ainsi, par le fait 6.1,

$$\sigma_j(t_j + 1) \leq \sigma_j(s_j)$$

et puisque $s' \in B(s)$ et $t' \in B(t')$ nous obtenons

$$t'_j < \sigma_j(t_j + 1) \leq \sigma_j(s_j) \leq s'_j,$$

ce qui est contradictoire. Ainsi, $s_j \leq t_j$ pour tout $j \in V$. Avec des arguments similaires, nous montrons que $s_j \geq t_j$ pour tout $j \in V \setminus \{i\}$. Par conséquent,

$$s_i \leq t_i \quad \text{et} \quad s_j = t_j \text{ pour tout } j \in V \setminus \{i\}$$

Puisque $B(s) \neq B(t)$ nous déduisons que $s_i < t_i$. Ainsi $s_i + 1 \leq t_i$ et d'après le fait 6.1,

$$\sigma_j(s_j + 1) \leq \sigma_j(t_j)$$

Puisque $s' \in B(s)$ et $t' \in B(t')$ nous obtenons

$$s'_i < \sigma_i(s_i + 1) \leq \sigma_i(t_i) \leq t'_i = s'_i + 1.$$

D'où,

$$\sigma_i(s_i + 1) = \sigma_i(t_i)$$

D'après le fait 6.1, σ_i est une injection, ainsi

$$s_i + 1 = t_i$$

et pour prouver que (s,t) est une transition de $GTA(G,p)$ il reste à prouver que

$$s_i < p(i, R_{G,i}(s)) \tag{6.2}$$

Puisque

$$s'_i < p'(i, R_{G',i}(s'))$$

nous avons

$$t'_i \leq p'(i, R_{G',i}(s'))$$

et puisque $t' \in B(t)$ nous obtenons

$$\sigma_i(t_i) \leq t'_i \leq p'(i, R_{G',i}(s'))$$

nous déduisons donc des remarques 6.1 et 6.3 et du lemme 6.1 que

$$\sigma_i(t_i) \leq p'(i, R_{G,i}(s))$$

Par définition de p, nous avons

$$t_i \leq p(i, R_{G,i}(s))$$

et puisque $s_i < t_i$ l'inégalité (6.2) est prouvée.

□

Théorème 6.3 *Soit $G \hookrightarrow G'$ un plongement strict. Soit p' un G'-modèle et soit p le G-modèle réduit de p'. Soient s et t deux états de $GTA(G,p)$ et soient s'_1 , s'_2, et r'_1 des états de $GTA(G',p')$ tels que $s'_1, s'_2 \in B(s)$, $r'_1 \in B(r)$ et (s'_1, r'_1) est une transition de $GTA(G',p')$. Alors, il existe $r'_2 \in B(r)$ telle que $GTA(G',p')$ a un chemin allant de s'_2 à r'_2 dont les états, excepté r'_2, appartiennent à $B(s)$.*

Preuve. Puisque (s'_1, r'_1) est une transition de $GTA(G',p')$, $(B(s), B(r))$ est une transition de $GTA(G',p')_{/\simeq}$, et nous déduisons du théorème 6.2 que (s,r) est une transition de $GTA(G,p)$. Alors, d'après le lemme 6.4, il existe $r'_2 \in B(r)$ telle que $GTA(G',p')$ a un chemin allant de s'_2 à r'_2 dont les états, excepté r'_2, appartiennent à $B(s)$. □

5 Préservation de formules par plongement

Dans cette section, nous allons prouver, en nous basant sur les résultats de préservation de chemins obtenus en section 4, deux résultats de préservation de formules selon que le plongement soit strict ou monotone.

En section 5.1, nous prouvons que par plongement strict touts les RRG-formules sont préservées, tant dis qu'en section 5.2 nous prouvons que par plongement monotone, seules les formules de $For_{nFITL\text{-}X}(Atome(G))$ (cf. définition 3.22) sont préservées.

5.1 Préservation de formules par plongement strict

Théorème 6.4 *Soient $G \hookrightarrow G'$ un plongement strict, p' un G'-modèle, et p le G-modèle réduit de p'. Pour toute formule $\chi \in For(G)$,*

$$p' \models_{G'} \bar{\sigma}(\chi) \quad \Longleftrightarrow \quad p \models_G \chi$$

Preuve. Soit $GTA(G', p') = (S_{G'}, T')$, soit $GTA(G, p) = (S_G, T)$ et soient L et L' les fonctions d'étiquetage associées respectivement à $GTA(G, p)$ et à $GTA(G', p')$ (cf. définition 5.8). Maintenant, considérons l'application $\mathfrak{L}' : S_{G'} \to 2^{\bar{\sigma}(Atome(G))}$ définie par :

$$\mathfrak{L}'(s') = \{\bar{\sigma}(\varphi) | \varphi \in L(s), s \in S_G, s' \in B(s)\}.$$

Dans la suite, on va montrer que la relation d'équivalence $\simeq$ définie dans la proposition 6.3 est une relation DBSB (cf. définition 3.35).

En effet, considérons la structure de Kripke $(S_{G'}, T', \mathfrak{L}')$ sur $\bar{\sigma}(Atome(G))$. Pour montrer que $\simeq$ est une relation DBSB sur $(S_{G'}, T', \mathfrak{L}')$, il suffit de montrer qu'elle vérifie les trois conditions de la définition 3.35. La première condition est assurée par la définition de $\mathfrak{L}'$ ($\mathfrak{L}'(s')$ dépend seulement de la classe d'équivalence $B(s)$ de $\simeq$ contenant s'), la deuxième condition est assurée par le théorème 6.3 alors que la troisième est assurée par le fait que $\simeq$ soit une relation d'équivalence, d'où $\simeq$ est une DBSB sur $(S_{G'}, T', \mathfrak{L}')$. Par conséquent, en appliquant le théorème 3.7, nous pouvons déduire que $(S_{G'}, T', \mathfrak{L}')$ et son quotient par $\simeq$ préservent les formules de $For(G')$ construites sur $\bar{\sigma}(Atome(G))$, c'est à dire :

$$\forall \varphi \in For(G), \qquad (S_{G'}, T', \mathfrak{L}') \models \bar{\sigma}(\varphi) \iff (S_{G'}, T', \mathfrak{L}')_{/\simeq} \models \bar{\sigma}(\varphi). \quad (6.3)$$

Maintenant, d'après la remarque 6.2 et la définition de $\mathfrak{L}'$, pour tout $s \in S_G$, $s' \in B(s)$ et $\varphi \in Atome(G)$, nous avons :

$$\begin{array}{lll} ((S_{G'}, T', L'), s') \models \bar{\sigma}(\varphi) & \iff & \varphi \in L(s) \\ & \iff & \bar{\sigma}(\varphi) \in \mathfrak{L}'(s') \\ & \iff & \bar{\sigma}(\varphi) \in \mathfrak{L}'_{/\simeq}(B(s)). \end{array} \quad (6.4)$$

D'une part, nous déduisons de ceci que,

$$\forall \varphi \in Atome(G), \qquad (S_{G'}, T', L') \models \bar{\sigma}(\varphi) \iff (S_{G'}, T', \mathfrak{L}') \models \bar{\sigma}(\varphi).$$

Il est facile de montrer, par induction sur la structure des formules, que

$$\forall \varphi \in For(G), \qquad (S_{G'}, T', L') \models \bar{\sigma}(\varphi) \iff (S_{G'}, T', \mathfrak{L}') \models \bar{\sigma}(\varphi). \quad (6.5)$$

D'autre part, nous déduisons de (6.4) que,

$$\forall \varphi \in Atome(G), \qquad (S_{G'}, T', \mathfrak{L}')_{/\simeq} \models \bar{\sigma}(\varphi) \iff (S_G, T, L) \models \varphi.$$

Ainsi, par le théorème 6.2 nous savons que les transitions entre les différents états de $GTA(G, p)$ et les différents états de $GTA(G', p')_{/\simeq}$ sont préservées, et il est facile de montrer par induction sur la structure des formules, que

$$\forall \varphi \in For(G), \qquad (S_{G'}, T', \mathfrak{L}')_{/\simeq} \models \bar{\sigma}(\varphi) \iff (S_G, T, L) \models \varphi. \quad (6.6)$$

En utilisant (6.5), (6.3) et (6.6) nous obtenons :

$$\forall \varphi \in For(G), \qquad (S_{G'}, T', L') \models \bar{\sigma}(\varphi) \iff (S_G, T, L) \models \varphi.$$

□

FIG. 6.10 – *Deux RRG-signatures : G (à gauche) et G′ (à droite).*

ressources ω'	$p'((1,\omega'))$	$p'((2,\omega'))$
$\emptyset$	0	0
$\{1\}$	1	
$\{2\}$	1	
$\{1,2\}$	1	

ressources ω	$p(1,\omega)$
$\emptyset$	0
$\{1\}$	1

FIG. 6.11 – *Un G′-modèle p′ et son modèle réduit G-modèle p.*

5.2 Préservation de formules par plongement monotone

Avant d'étudier la préservation de propriétés par plongement monotone des signatures, nous montrons par un simple exemple qu'il existe des RRG-formules qui ne sont préservées par plongement monotone. En effet, considérons les deux RRG-signatures G et G' de la figure 6.10. Il est bien clair que G est plongée dans G' et que ce plongement est monotone, mais non strict. Soit p' un G'-modèle de G' donné par la figure 6.11 (à gauche) et soit p le G-modèle réduit p de p', aussi donné par la figure 6.11 (à droite), ainsi que les graphes de transitions asynchrones relatifs à p' (à gauche) et à p (à droite). Il est clair que les deux modèles p' et p ne préservent pas les mêmes RRG-formules. Par exemple, la formule $EF(x_1 = 0)$, qui traduit le fait que le système passe éventuellement par un état où $x_1 = 0$, est satisfaite par p mais non par p' puisque l'état $(1,0)$ ne satisfait pas cette formule.

Cependant, nous avons montré par le théorème 6.1 que tout chemin de $GTA(G,p)$ est plongé à au moins un chemin de $GTA(G',p')_{/\simeq}$. À partir de ce résultat on peut énoncer le résultat de préservation de formules suivant :

Lemme 6.5 *Soit $G \hookrightarrow G'$ un plongement monotone, soit p' un G'-modèle, et soit p le G-modèle réduit de p'. Soit $GTA(G',p') = (S_{G'},T')$ et $GTA(G,p) = (S_G,T)$ les graphes de transitions asynchrones associés respectivement à p' et p. Soient $s \in S_G$ et $\rho \in path(s)$ un chemin de $GTA(G,p)$. Pour toute formule $\chi \in For_{nFITL\text{-}X}(Atome(G))$ on a :*

(a) si $\chi \in For^s_{nFITL\text{-}X}(Atome(G))$ alors,

$$(p,s) \models_G \chi \Rightarrow \forall s' \in B'(s),\ (p',s') \models_{G'} \bar{\sigma}(\chi)$$

(b) si $\chi \in For^p_{nFITL\text{-}X}(Atome(G))$ alors,

$$(p,\rho) \models_G \chi \Rightarrow \forall \rho' \in B'(\rho),\ (p',\rho') \models_{G'} \bar{\sigma}(\chi)$$

Preuve. Considérons tout d'abord les formules de $For^s_{nFITL\text{-}X}(Atome(G))$.

(1) $\chi = a$, avec $a \in Atome(G)$: puisque $s' \in B'(s)$, par la remarque 6.2 nous avons $(p,s) \models_G \chi$ ssi $(p',s') \models_{G'} \bar{\sigma}(\chi)$.
(2) $\chi = \neg a$, avec $a \in Atome(G)$: déduit de (1).
(3) $\chi = \varphi \wedge \varphi'$: nous avons bien $(p,s) \models_G \varphi \wedge \varphi' \Rightarrow \forall\, s' \in B'(s), (p',s') \models_{G'} \bar{\sigma}(\varphi \wedge \varphi')$ puisque nous avons par hypothèse d'induction $(p,s) \models_G \varphi \Rightarrow \forall s' \in B'(s), (p',s') \models_{G'} \bar{\sigma}(\varphi)$ et $(p,s) \models_G \varphi' \Rightarrow \forall s' \in B'(s), (p',s') \models_{G'} \bar{\sigma}(\varphi')$ et puisque $\bar{\sigma}(\varphi) \wedge \bar{\sigma}(\varphi') = \bar{\sigma}(\varphi \wedge \varphi')$.
(4) $\chi = \varphi \vee \varphi$. La preuve se fait par analogie avec (3).
(5) $\chi = E\pi$: supposons que $(p,s) \models_G E\pi$. Alors il existe un chemin $\rho \in path(s)$ dans $GTA(G,p)$ tel que $(p,\rho) \models_G \pi$. Par hypothèse d'induction, $\forall \rho' \in B'(\rho),\ (p',\rho') \models_{G'} \bar{\sigma}(\pi)$. Puisque $B'(\rho)$ contient au moins un chemin commençant par un état de $B'(\rho(0))$, ceci implique $\forall s' \in B'(s)$, il existe un chemin $\rho' \in path(s')$ vérifiant $(p',\rho') \models_{G'} \bar{\sigma}(\pi)$. Par définition ceci implique $\forall s' \in B'(s), (p',s') \models_{G'} \bar{\sigma}(E\pi)$.

Considérons ensuite les formules de $For^{p}_{nFITL\text{-}X}(Atome(G))$.

(6) $\chi = \varphi$: par définition, $(p,\rho) \models_G \chi$ ssi $(p,s) \models_G \varphi$. Par hypothèse d'induction, nous avons $\forall s' \in B'(s), (p',s') \models_{G'} \bar{\sigma}(\varphi)$. Par définition ceci implique $\forall s' \in B'(s),\ \forall \rho' \in path(s'),\ (p',\rho') \models_{G'} \bar{\sigma}(\varphi)$. Ainsi, $\forall \rho' \in B'(\rho),\ (p',\rho') \models_{G'} \bar{\sigma}(\varphi)$.
(7) $\chi = \pi \Delta \pi'$ avec $\Delta \in \{\wedge, \vee\}$. La preuve se fait par analogie avec (3).
(8) $\chi = \pi U \pi'$. Supposons que $(p,\rho) \models_G \pi U \pi'$. Alors il existe $j, 0 \leq j \leq n$ tel que $(p,\rho^j) \models_G \pi'$ et pour tout $i, 0 \leq i < j$, $(p,\rho^i) \models_G \pi$. Soit ρ' un chemin de $B'(\rho)$ et soit $(k_i)_{i \in Index(\rho)}$ la suite d'indices définie par le théorème 6.1. Par induction, $(p',\rho'^{k_j}) \models_{G'} \bar{\sigma}(\pi')$ et pour tout $k, 0 \leq k < k_j$, $(p',\rho'^{k}) \models_{G'} \bar{\sigma}(\pi)$. Ainsi $\forall\, \rho' \in B'(\rho), (p',\rho') \models_{G'} \bar{\sigma}(\pi) U \bar{\sigma}(\pi')$.

□

En utilisant le fait que $B'(s) \subseteq B(s)$, nous pouvons déduire le théorème suivant :

Théorème 6.5 *Soit $G \hookrightarrow G'$ un plongement monotone, soit p' un G'-modèle, et soit p le G-modèle réduit de p'. Soit $GTA(G',p') = (S_{G'}, T')$ et $GTA(G,p) = (S_G, T)$ les graphes de transitions asynchrones associés respectivement à p' et p. Soient $s \in S_G$ et $\rho \in path(s)$ un chemin de $GTA(G,p)$. Pour toute formule $\chi \in For_{nFITL\text{-}X}(Atome(G))$ on a :*

(a) si $\chi \in For^{s}_{nFITL\text{-}X}(Atome(G))$ alors,

$$(p,s) \models_G \chi \Rightarrow \exists s' \in B(s),\ (p',s') \models_{G'} \bar{\sigma}(\chi)$$

(b) si $\chi \in For^{p}_{nFITL\text{-}X}(Atome(G))$ alors,

$$(p,\rho) \models_G \chi \Rightarrow \exists \rho' \in B(\rho),\ (p',\rho') \models_{G'} \bar{\sigma}(\chi)$$

L'implication inverse du théorème 6.5 n'est pas vrai puisqu'on peut avoir des formules $\chi \in For_{nFITL\text{-}X}(Atome(G))$ qui sont satisfaites par un état s, mais il existe $s' \in B(s)$ qui satisfait $\bar{\sigma}(\chi)$. Par exemple, on peut constater en figure 6.8 que la formule $((0 \leq x_1 < 2 \wedge x_2 = 0) \Rightarrow EF(x_2 = 3))$ est satisfaite par $((GTA(G',p'),(0,0,0))$ mais que $((x_1 = 0 \wedge x_2 = 0) \Rightarrow EF(x_2 = 2))$ est non satisfaite par $(GTA(G,p),(0,0))$.

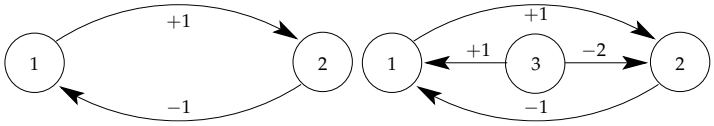


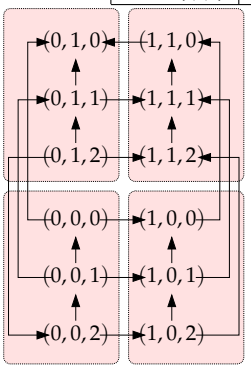
FIG. 6.12 – *Contre-example.*

5.3 Contre-exemple justifiant la notion de plongement monotone

Dans cette section, nous donnons un contre-exemple pour illustrer l'utilité de la condition de monotonie de plongement donné par la définition 6.3. Considérons les deux RRG-signatures G et G' de la figure 6.12. Nous pouvons définir un plongement $G \hookrightarrow G'$ satisfaisant les conditions de la définition 6.1, et violant celles de la définition 6.3 du plongement monotone. Soit p' un G'-modèle de G' illustré par la figure 6.13 (à gauche) et soit p le G-modèle réduit p de p' donné par la même figure (à droite), et considérons les graphes de transitions asynchrones associés respectivement à p et p'.

ressources ω'	$p'((1,\omega'))$	$p'((2,\omega'))$	$p'((3,\omega'))$
$\emptyset$	0	0	0
{1}		1	
{2}	1		
{3}	1	1	
{1,2}			
{1,3}		1	
{2,3}	1		
{1,2,3}			

ressources ω	$p(1,\omega)$	$p(2,\omega)$
$\emptyset$	0	0
{1}		1
{2}	1	
{1,2}		



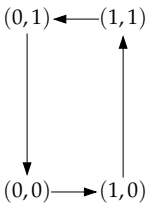


FIG. 6.13 – *Un G'-modèle p' (à gauche) et son modèle réduit G-modèle p (à droite).*

Il est clair que le résultat du théorème 6.5 n'est plus valide. Par exemple la formule $\varphi = ((x_1 = 1 \wedge x_2 = 1) \Rightarrow (EF(x_1 = 0 \wedge x_2 = 0)))$ est satisfaite par $(p, (1,1))$ mais il n'existe pas d'état $s' \in B((1,1))$ tel que (p', s') satisfait $\bar{\sigma}(\varphi) = ((x_1 = 1 \wedge x_2 = 1) \Rightarrow (EF(x_1 = 0 \wedge x_2 = 0)))$.

6 Convergence avec d'autres travaux

Dans [24], Bernot et al. se sont aussi intéressés à l'étude du comportement des réseaux de régulation génétique au travers de leur plongement. Pour avoir des résultats de préservation, les auteurs n'ont pas étudié les conditions sur la nature du plongement, mais plutôt ils ont étudié des conditions sur les modèles. Leur notion de préservation de comportement impose une condition sur les graphes de transitions asynchrones. Ils ont défini la notion de *plongement préservant de comportement* de la façon suivante :

Définition 6.10 (Plongement préservant le comportement) *Un plongement $G \hookrightarrow G'$ préserve le comportement de G si et seulement si pour toute transition (s', t') de $GTA(G', p')$ telle que $B^{-1}(s') = B^{-1}(t')$ et pour tout $j \in V' \setminus V$, $(s'_{[j \leftarrow l]}, t'_{[j \leftarrow l]}) \in GTA(G', p')$, où $l \in \{0, \ldots, b_j\}$*[2].

Néanmoins, cette condition sémantique de comportement est forte puisqu'elle implique nécessairement la préservation des chemins. En effet, nous avons pu démontrer qu'une telle condition implique les théorèmes 6.6 et 6.7 qui suivent et qui sont similaires respectivement aux théorèmes 6.3 et 6.2, d'où la préservation des chemins.

Théorème 6.6 *Soit $G \hookrightarrow G'$ un plongement préservant le comportement de G. Soit p' un G'-modèle et soit p le G-modèle réduit de p'. Soient s et t deux états de $GTA(G, p)$ et soient s'_1 , s'_2, et r'_1 des états de $GTA(G', p')$ tels que $s'_1, s'_2 \in B(s)$, $r'_1 \in B(r)$ et (s'_1, r'_1) est une transition de $GTA(G', p')$. Alors, il existe $r'_2 \in B(r)$ telle que $GTA(G', p')$ a un chemin allant de s'_2 à r'_2 dont les états, excepté r'_2, appartiennent à $B(s)$.*

Preuve. Voir section 1 de l'annexe B. □

Théorème 6.7 *Soit $G \hookrightarrow G'$ un plongement préservant le comportement de G. Soit p' un G'-modèle et soit p le G-modèle réduit de p'. Alors, (s, t) est une transition de $GTA(G, p)$ si et seulement si $(B(s), B(t))$ est une transition de $GTA(G', p')_{/\simeq}$.*

Preuve. Voir section 2 de l'annexe B. □

Cette définition de préservation de comportement est difficile à vérifier sur un graphe de transitions asynchrone de grande taille, par conséquent les auteurs l'ont traduit par une condition suffisante sur les modèles. Cette condition se traduit dans notre cadre par, $\forall i \in V, \forall w \subseteq G_i^-, \forall j \in G_i'^- \setminus G_i^-$:

- $p'(i, w \cup \{j\})) \geq 0$ si $p(i, w) = 0$,
- $p'(i, w \cup \{j\})) \geq \sigma_i(b_i)$ si $p(i, w) = b_i$,
- $\sigma_i(p(i, w)) \leq p'(i, w \cup \{j\})) < \sigma_i(p(i, w) + 1)$ sinon.

Il est vrai que cette condition assure la préservation de propriétés au travers du plongement tel que nous l'avons défini (cf. 6.1). Néanmoins, en imposant des conditions sur les modèles, on risque de rejeter des modèles à tort. À l'opposé, notre démarche consiste à étudier des conditions sur le plongement qui permettent d'avoir des résultats de préservation sur des classes de formules tout en garantissant qu'aucun modèle n'est rejeté à tort. Une telle démarche nous a permis, dans le cas de plongement monotone, d'avoir des résultats de préservation d'une classe de formules sans pour autant que les modèles satisfont la condition de la définition 6.10.

7 Conclusion

Dans le cadre de la modélisation discrète introduite par R. Thomas, nous avons étudié le plongement d'un réseau de régulation dans un réseau plus

[2] Soit G une RRG-signature et soit s un état de S_G, $s_{[j \leftarrow l]}$, où $l \in \{0, \ldots, b_j\}$, dénote l'état $s' \in S_G$ tel $s'_j = l$ et $s'_i = s_i$ pour tout i différent de j.

grand et nous avons fourni des conditions sur le plongement pour assurer la préservation de propriétés biologiques au travers de plongement. Nous avons obtenu deux résultats selon que le plongement est strict ou monotone :

- le choix de plongement strict est surtout motivé par une intuition biologique stipulant que l'absence de nouvelles ressources n'affecte pas le comportement du réseau. Nous avons montré que tous les chemins des graphes de transitions asynchrones sont préservés au travers de plongement strict, ce qui rejoint la restriction de [24] sur les modèles. Cette préservation des chemins garantit une préservation totale des propriétés temporelles des réseaux de régulation.
- le choix de plongement monotone est fondé sur la structure du graphe d'interaction et la nature des interactions rentrantes. Nous avons montré que seul un chemin des graphes de transitions asynchrones est préservé au travers de ce type de plongement. À partir de ce résultat, nous avons identifié une sous-classe de formules préservées.

Les conditions que nous avons proposées restent des conditions restrictives. Plusieurs suites peuvent être envisagées pour relâcher ces conditions comme l'étude d'autres familles de plongement et d'autres classes de propriétés des réseaux de régulation génétique.

Conclusion

Dans ce dernier chapitre de la thèse, nous faisons le bilan des travaux réalisés, ensuite nous indiquons plusieurs directions de travail futur.

Bilan

Dans ce travail, nous avons proposé une caractérisation formelle générique et unificatrice de la notion de systèmes complexes et de systèmes modulaires centrée sur la notion de composant/propriété. Cette caractérisation a abouti à la définition d'un cadre général pour les spécifications des systèmes informatiques au-dessus des institutions. Dans ce cadre, nous avons défini un connecteur générique comme moyen de composition d'un système global à partir de sous-systèmes et nous avons défini des conditions nécessaires et/ou suffisantes pour que les propriétés des sous-systèmes soient héritées par le système global.

La première partie a été consacrée à la présentation des résultats de modularité des spécifications algébriques structurées. Nous avons tout d'abord présenté un état de l'art des spécifications algébriques et de leurs primitives de structuration. Ensuite, nous avons proposé une contribution concernant la modularité et la complexité au travers de la structuration, par enrichissement ou par union. Nous avons fourni des conditions sous lesquelles la modularité d'une spécification algébrique structurée, par enrichissement ou par union, est assurée.

Dans la seconde partie, nous avons présenté une généralisation des notions de structuration, modularité et complexité à des spécifications abstraites. Nous avons défini un cadre générique de spécifications abstraites. Pour structurer les spécifications, nous avons défini au travers des notions catégorielles de diagrammes et de colimites, un connecteur architectural générique permettant de combiner des spécifications pour en construire d'autres plus larges. Les notions de la modularité et de la complexité que nous avons définies dans le cadre des spécifications algébriques ont été généralisées dans le cadre de ce connecteur générique. Dans ce cadre, nous avons fourni des conditions nécessaires et/ou suffisantes pour qu'un connecteur soit modulaire. Nous avons illustré toutes les notions définies dans cette partie au travers de spécifications axiomatiques et de spécifications des systèmes réactifs.

La troisième partie a été consacrée à la problématique de l'émergence des propriétés au sein des réseaux de régulation génétique. Nous avons tout d'abord présenté les réseaux de régulation génétique dans la modélisation discrète introduite par R. Thomas. Ensuite, nous avons défini le plongement d'un réseau de régulation dans un réseau de régulation plus

large. Enfin, nous avons proposé des conditions suffisantes sur le plongement pour assurer la préservation d'une certaine classe de propriétés, exprimées sous forme de logique temporelle.

Perspectives

Les travaux effectués dans cette thèse ouvrent de nouveaux horizons pour des études futures. Les principales perspectives en vue de poursuivre ces travaux sont :

Concernant le cadre général de spécifications abstraites :

- Instancier notre cadre par des formalismes de spécifications, autres que ceux présentés dans cette thèse. Plus précisément, on peut se demander si les formalismes de spécifications proposés dans divers travaux peuvent être aisément réexprimés au travers de notre cadre général. Dans cette perspective, on peut s'intéresser aux interactions de services d'autant que les travaux de Gaston [55] ont permis de spécifier les systèmes à base de services dans le cadre de spécifications axiomatiques (Σ, Ax) où Σ dénote une signature d'un service et Ax l'ensemble d'axiomes caractérisant le comportement attendu de ce service.
- Faire suivre notre cadre de spécifications abstraites par un guide méthodologique sur le raffinement de spécifications. En effet, quand on est confronté à un système complexe (i.e. un système ayant de l'émergence), on ne peut pas bénéficier de la structure du système au travers des connecteurs utilisés dans sa conception pour aborder la validation/vérification du système. Ainsi, tout nouvel ajout d'un composant de spécification induit alors un nouveau système qu'il faut étudier dans son ensemble. On proposerait alors, comme ceci a déjà été fait dans [5, 54], d'utiliser la technique du raffinement algébrique pour obtenir une nouvelle méthode de conception incrémentale. Ainsi, tout système possédant de l'émergence serait abstrait en un système plus simple pour lequel on étudierait/résoudrait cette émergence. Par des résultats généraux que nous devrons établir et qui montreraient que toute propriété émergente à un niveau abstrait est préservée à un niveau plus concret, nous permettraient alors de nous concentrer sur l'émergence liée à la concrétisation des systèmes.
- Dans les travaux de Krob [81, 82], la complexité d'un système est définie de façon « flou et subjective ». L'auteur ne se base pas sur une définition formelle mais sur une « réalité pragmatique » à laquelle sont confrontés les ingénieurs. Il serait intéressant de regarder dans quelles mesures les systèmes de Krob sont un cas particulier de notre cadre général.

Concernant les réseaux de régulation génétique :

- Les conditions de plongement des réseaux de régulation que nous avons proposées restent des conditions restrictives et des travaux futurs peuvent être envisagés pour relâcher ces conditions pour pou-

voir étudier d'autres familles de plongement et d'autres classes de propriétés des réseaux de régulation génétique.

Quatrième partie

Annexes

Notions sur la théorie des catégories

A

Cette annexe rappelle les outils principaux de la théorie des catégories qui ont été intensivement utilisés dans cette thèse.

1 Catégorie, Foncteur

Définition A.1 (Catégorie) *Une catégorie $\mathcal{C}$ est la donnée de :*

- *une classe, dont les éléments sont appelés d'objets, notée $|\mathcal{C}|$,*
- *une classe $Hom_{\mathcal{C}}(A, B)$, pour toute paire d'objets A et B, dont les éléments sont appelés morphismes (ou flèches). On utilise usuellement la notation*

$$f : A \to B$$

pour indiquer que $f \in Hom_{\mathcal{C}}(A, B)$, l'objet A est appelé source (domaine) de f et l'objet B est appelé cible (codomaine),
- *un morphisme $id^{\mathcal{C}}_A : A \to A$, pour tout objet A, appelé identité sur A,*
- *une opération binaire*

$$- \circ_{\mathcal{C}} - : Hom_{\mathcal{C}}(B, C) \times Hom_{\mathcal{C}}(A, B) \to Hom_{\mathcal{C}}(A, C)$$

appelée composition, qui à tous morphismes $f : A \to B$ et $g : B \to C$ associe le morphisme

$$g \circ_{\mathcal{C}} f : A \to C$$

qui satisfait les axiomes suivants :

- *associativité : pour tous objets A , B, C et D et pour tous morphismes*

$$f : A \to B, g : B \to C \text{ et } h : C \to D$$

l'égalité

$$(h \circ_{\mathcal{C}} g) \circ_{\mathcal{C}} f = h \circ_{\mathcal{C}} (g \circ_{\mathcal{C}} f)$$

est vérifiée,
- *unicité : pour tous objets A et B et tout morphisme $f : A \to B$, les égalités*

$$id^{\mathcal{C}}_B \circ_{\mathcal{C}} f = f = f \circ_{\mathcal{C}} id^{\mathcal{C}}_A$$

sont vérifiées.

Dans la suite, on omettra l'indice $\mathcal{C}$ de Hom, id, $\circ$, etc. lorsque la catégorie dont il s'agit peut être aisément déduite du contexte.

Définition A.2 (Catégorie CAT) *On note CAT la catégorie dont les objets sont les catégories et dont les morphismes entre deux objets $\mathcal{C}$ et $\mathcal{D}$ sont les foncteurs $F : \mathcal{C} \to \mathcal{D}$.*

Définition A.3 (Catégorie SET) *On note SET la catégorie dont les objets sont les ensembles et dont les morphismes sont les fonctions.*

Définition A.4 (Sous-catégorie) *Soit $\mathcal{C}$ une catégorie. Une sous-catégorie $\mathcal{D}$ de $\mathcal{C}$ est une catégorie telle que :*
- $|\mathcal{D}| \subseteq |\mathcal{C}|$,
- *pour toute paire d'objets A et B de $\mathcal{D}$, $Hom_{\mathcal{D}}(A, B) \subseteq Hom_{\mathcal{C}}(A, B)$,*
- *pour tout objet A de $\mathcal{D}$, $id^{\mathcal{C}}_A \in Hom_{\mathcal{D}}(A, A)$,*
- *pour tous objets A, B et C de $\mathcal{D}$ et toute paire de morphismes $f \in Hom_{\mathcal{D}}(A, B)$ et $g \in Hom_{\mathcal{D}}(B, C)$, $g \circ_{\mathcal{C}} f \in Hom_{\mathcal{D}}(A, C)$.*

Une sous-catégorie $\mathcal{D}$ de $\mathcal{C}$ est dite pleine lorsque pour toute paire d'objets A et B de $\mathcal{D}$, $Hom_{\mathcal{D}}(A, B) = Hom_{\mathcal{C}}(A, B)$.

Définition A.5 (Catégorie duale) *La catégorie duale d'une catégorie $\mathcal{C}$, notée $\mathcal{C}^{op}$ est la catégorie définie par :*
- $|\mathcal{C}^{op}| = |\mathcal{C}|$,
- *pour toute paire d'objets A et B de $\mathcal{C}^{op}$, $f \in Hom_{\mathcal{C}^{op}}(A, B)$ si, et seulement si, $f \in Hom_{\mathcal{C}}(B, A)$,*
- *pour tout objet A de $\mathcal{C}^{op}$, $id^{\mathcal{C}^{op}}_A = id^{\mathcal{C}}_A$,*
- *pour tous objets A, B et C de $\mathcal{C}^{op}$ et toute paire de morphismes $f \in Hom_{\mathcal{C}^{op}}(A, B)$ et $g \in Hom_{\mathcal{C}^{op}}(B, C)$, $g \circ_{\mathcal{C}^{op}} f = f \circ_{\mathcal{C}} g$.*

Définition A.6 (Objet initial, objet final) *Un objet* initial *dans une catégorie $\mathcal{C}$, est un objet de $\mathcal{C}$, usuellement noté 0, tel que pour tout objet A de $\mathcal{C}$, il existe un unique morphisme $0_A \in Hom_{\mathcal{C}}(0, A)$. De façon duale, un objet est* final *dans une catégorie $\mathcal{C}$, usuellement noté 1, si et seulement s' il est initial dans $\mathcal{C}^{op}$, c'est-à-dire que 1 est final dans $\mathcal{C}$ si est seulement si, pour tout objet A de $\mathcal{C}$, il existe un unique morphisme $1_A \in Hom_{\mathcal{C}}(A, 1)$.*

On va définir maintenant la notion de *foncteur*. Intuitivement, un foncteur est un morphisme de catégories.

Définition A.7 (Foncteur) *Un foncteur F entre deux catégories $\mathcal{C}$ et $\mathcal{D}$, noté $F : \mathcal{C} \to \mathcal{D}$, est la donnée de :*
- *d'une fonction $F : |\mathcal{C}| \to |\mathcal{D}|$ qui à tout objet A de $\mathcal{C}$, associe un objet $F(A)$ de $\mathcal{D}$,*
- *d'une fonction $F_{A,B} : Hom_{\mathcal{C}}(A, B) \to Hom_{\mathcal{D}}(F(A), F(B))$, simplement noté F, pour toute paire d'objets A et B, qui à tout morphisme $f : A \to B$ de $\mathcal{C}$ associe un morphisme $F(f) : F(A) \to F(B)$ de $\mathcal{D}$,*

qui satisfont les axiomes suivants :
- *préservation de l'identité : pour tout objet A de $\mathcal{C}$, l'égalité*

$$F_{A,A}(id^{\mathcal{C}}_A) = id^{\mathcal{D}}_{F(A)}$$

  est vérifiée,
- *préservation de la composition : pour tous objets A, B et C de $\mathcal{C}$, et tous*

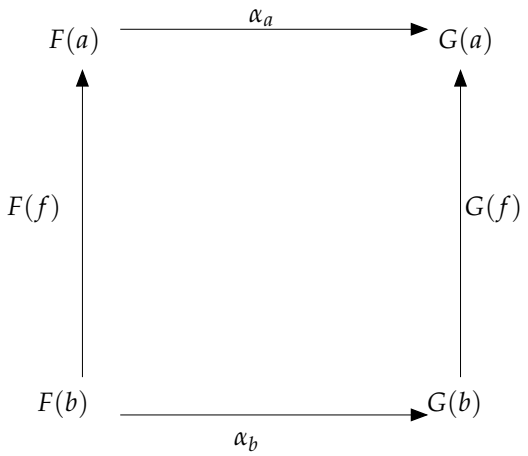
morphismes $f : A \to B$ et $g : B \to C$ de $\mathcal{C}$, l'égalité

$$F_{A,C}(g \circ_{\mathcal{C}} f) = F_{B,C}(g) \circ_{\mathcal{D}} F_{A,B}(f)$$

est vérifiée.

Définition A.8 (Foncteur contravariant) *Soient $\mathcal{C}$ et $\mathcal{D}$ deux catégories. Un foncteur $F : \mathcal{C} \to \mathcal{D}^{op}$ est appelé foncteur contravariant de $\mathcal{C}$ dans $\mathcal{D}$.*

Définition A.9 (Transformation naturelle) *Soient $\mathcal{C}$ et $\mathcal{D}$ deux catégories et soient $F, G : \mathcal{C} \to \mathcal{D}$ deux foncteurs. Une transformation naturelle $\alpha : F \to G$ est la donnée pour chaque objet a d'un morphisme $\alpha_a : F(a) \to G(a)$ tel que pour tout morphisme $f : a \to b$ de $\mathcal{C}$, le diagramme suivant commute :*



2 Propriétés de morphismes

Définition A.10 (Épimorphisme, monomorphisme) *Soit $\mathcal{C}$ une catégorie. Un morphisme $f : A \to B$ de $\mathcal{C}$ est un épimorphisme si pour tout objet C de $\mathcal{C}$, pour tous morphismes $g, h : B \to C$ de $\mathcal{C}$,*

$$g \circ f = h \circ f \quad implique \quad g = h$$

De façon duale, $f : A \to B$ est un monomorphisme si pour tout objet C de $\mathcal{C}$, pour tous morphismes $g, h : C \to A$ de $\mathcal{C}$,

$$f \circ g = f \circ h \quad implique \quad g = h$$

Définition A.11 (Isomorphisme) *Soit $\mathcal{C}$ une catégorie. Un morphisme $f : A \to B$ de $\mathcal{C}$ est un isomorphisme (ou encore est inversible) s'il existe un morphisme $g : B \to A$ de $\mathcal{C}$ tel que :*

$$g \circ f = id_A \quad et \quad f \circ g = id_B$$

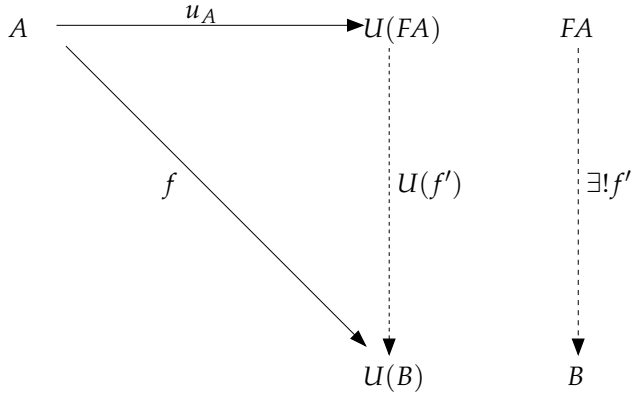
*Un tel morphisme g, lorsqu'il existe, est déterminé de façon unique par f et est appelé l'inverse de f, usuellement notée f^{-1}. A et B sont dits **isomorphes**.*

Fait A.1 *Soit $\mathcal{C}$ une catégorie. Si un morphisme de $\mathcal{C}$ est un isomorphisme, alors il est aussi un épimorphisme et un monomorphisme.*

3 Adjonction

Définition A.12 (Objet libre, morphisme universel) *Soient $U : \mathcal{D} \to \mathcal{C}$ un foncteur et A un objet de $\mathcal{C}$. Un objet FA de $\mathcal{D}$ est libre en A s'il existe un morphisme $u_A : A \to U(FA)$ tel que pour tout objet B de $\mathcal{D}$, pour tout morphisme $f : A \to U(B)$, il existe un unique morphisme $f' : FA \to B$ tel que $U(f') \circ u_A = f$. Le morphisme u_A est appelé morphisme universel de A vers U.*

La définition précédente signifie en fait que chaque morphisme f de $\mathcal{C}$ vers U se factorise de façon unique au travers du morphisme u comme indiqué sur le diagramme ci-dessous.



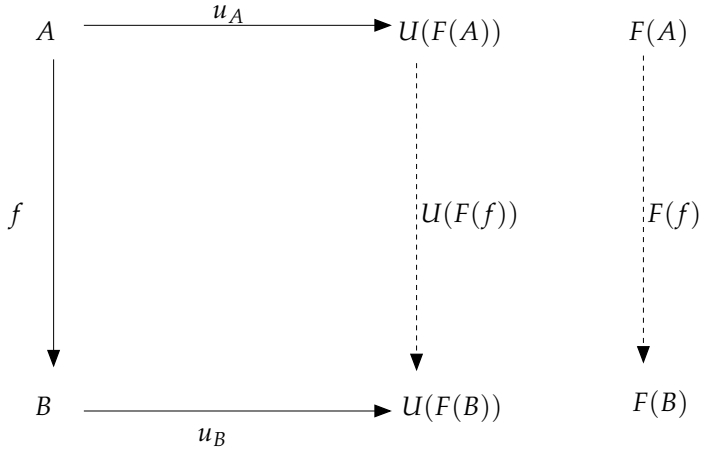
Définition A.13 (Adjonction) *Soit $U : \mathcal{D} \to \mathcal{C}$ un foncteur. Si pour tout objet A dans $\mathcal{C}$, il existe un objet libre FA en A, avec u_A le morphisme universel correspondant, alors le foncteur $F : \mathcal{C} \to \mathcal{D}$ défini de la façon suivante :*

- *pour tout objet A dans $\mathcal{C}$ associe l'objet libre FA en A dans $\mathcal{D}$,*
- *pour tout morphisme $f : A \to B$ dans $\mathcal{C}$ associe le morphisme $F(f)$ défini par :*

$$U(F(f)) \circ u_A = u_B \circ f$$

avec u_A et u_B les morphismes universels respectifs de A et de B vers U, est appelé adjoint à gauche de U, le foncteur U est appelé adjoint à droite de F. La paire U, F est appelée adjonction et pour tout A dans $\mathcal{C}$, l'homomorphisme u_A est appelé homomorphisme d'adjonction associé à A.

La définition précédente signifie en fait que le diagramme ci-dessous commute.



4 Catégorie des diagrammes, Cocône, colimite

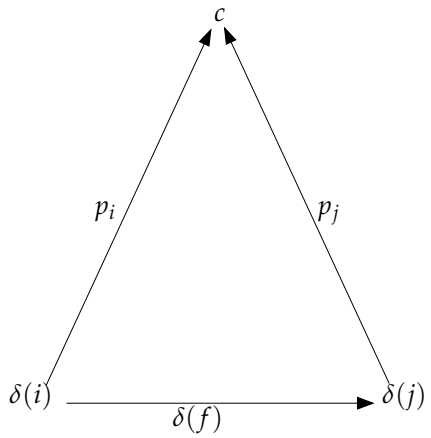
Notation A.1 (Catégorie des diagrammes) *Soient I et C deux catégories.*

On note $\mathcal{D}_{(I,\mathcal{C})}$ la catégorie des diagrammes dans $\mathcal{C}$ avec shape I, c'est-à-dire la catégorie dont les objets sont les foncteurs $\delta : I \to \mathcal{C}$, les diagrammes dans $\mathcal{C}$ avec shape I, et les morphismes sont les transformations naturelles entre foncteurs.

Soit I' une sous-catégorie de I. Soit δ un diagramme de $\mathcal{D}_{(I,\mathcal{C})}$. On note $\delta_{|_{I'}}$ le diagramme de $\mathcal{D}_{(I',\mathcal{C})}$ obtenu par la restriction de δ à I'.

Définition A.14 (Cocône) *Soient I et $\mathcal{C}$ deux catégories et soit $\delta : I \to \mathcal{C}$ un diagramme de $\mathcal{D}_{(I,\mathcal{C})}$.*

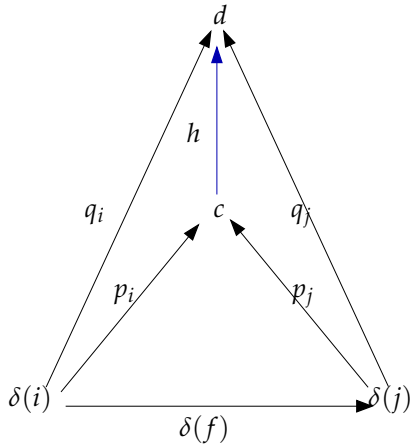
*On appelle **cocône** la donnée d'un objet $c \in |\mathcal{C}|$ et d'une famille de morphismes $\{p_i : \delta(i) \to c\}_{i \in I}$ tel que pour tout morphisme $f : i \to j$ de I, on a $p_j \circ \delta(f) = p_i$, c'est-à-dire que le diagramme ci-dessous commute :*



On note $(c, \{p_i\}_{i \in I})$ un tel cocône.

Définition A.15 (Colimite) *Étant données deux catégories I et $\mathcal{C}$ deux catégories et un diagramme $\delta : I \to \mathcal{C}$ de $\mathcal{D}_{(I,\mathcal{C})}$.*

*Une **colimite** est un cocône $(c, \{p_i\}_{i \in I})$ tel que pour tout autre cocône $(d, \{q_i\}_{i \in I})$, il existe un unique morphisme $h : c \to d$ tel que pour tout $i \in I$, on a $h \circ p_i = q_i$, c'est-à-dire que le diagramme ci-dessous commute :*



Définition A.16 (Cocomplète) *Une catégorie $\mathcal{C}$ est* cocomplète *si pour tout diagramme dans $\mathcal{C}$ admet une colimite.*

Preuves

B

1 Preuve du théorème 6.6

Preuve. Soit (s,t) une transition de $GTA(G,p)$. Par définition, il existe $i \in V$ tel que $s_j = t_j$ pour tout $j \in V \setminus \{i\}$ et

$$t_i = \begin{cases} s_i + 1 \text{ et } s_i < p(i, R_{G,i}(s)) \\ s_i - 1 \text{ et } s_i > p(i, R_{G,i}(s)) \end{cases}$$

Supposons que $s_i < p(i, R_{G,i}(s))$, la preuve est similaire dans les autres cas. Soit $s' \in B'(s)$ telle que pour tout $k \in G_i'^{-} \setminus G_i^{-}$, $s'_k = 0$ si $Sl'(k,i) = +$, et $s'_k = b_i$ si $Sl'(k,i) = -$, donc $R'_{G',i}(s') = R_{G,i}(s)$. Soit t' l'état de $S_{G'}$ défini par :

$$t'_i = \sigma_i(t_i) \qquad \text{et} \qquad t'_j = s'_j \text{ for all } j \in V' \setminus \{i\}$$

Par construction, $t' \in B(t)$. Il existe un chemin allant de s' jusqu'au t' dont les états, sauf t', appartiennent à $B'(s)$, nécessairement cette propriété est vérifié par condition (forcer des flèches) pour tout $s'' \in B(s)$ telle $s''_j = s'_j$ pour tout $j \in V$. Ainsi, pour montrer le théorème, il suffit de montrer qu'il existe un chemin allant de s' jusqu'au t' dont les états, sauf t', appartiennent à $B'(s)$. Montrons-le par induction sur

$$d(s') = t'_i - s'_i$$

1. *Cas de base :* $d(s') = 1$.
   Nous avons $s_i < p(i, R_{G,i}(s))$, d'où

   $$s_i + 1 \leq p(i, R_{G,i}(s))$$

   Par monotonie de σ_i et par définition de p,

   $$\sigma_i(s_i + 1) \leq \sigma_i(p(i, R_{G,i}(s))) \leq p'(i, R_{G,i}(s))$$

   Or $R'_{G',i}(s') = R_{G,i}(s)$, donc

   $$\sigma_i(s_i + 1) \leq \sigma_i(p(i, R_{G,i}(s))) \leq p'(i, R_{G',i}(s'))$$

   et puisque $s' \in B(s)$ on obtient

   $$s'_i < \sigma_i(s_i + 1) \leq \sigma_i(p(i, R_{G,i}(s))) \leq p'(i, R_{G',i}(s'))$$

   $$s'_i < p'(i, R'_{G',i}(s'))$$

   D'où nous déduisons que (s', t') est une transition de $GTA(G', p')$.

2. *Pas de récurrence :* $d(s') > 1$. Soit s'' un état de $S_{G'}$ défini par

$$s''_i = s'_i + 1 \quad \text{et} \quad s''_j = s'_j \text{ pour tout } j \in V' \setminus \{i\}$$

Clairement, $s'' \in B'(s)$, $d(s'') < d(s')$ et $t'' = t'$. Donc, par hypothèse d'induction, il existe un chemin allant de s'' à t' dont les états, sauf t', sont dans $B'(s)$. En plus, nous avons montré que,

$$s'_i < p'(i, R'_{G',i}(s'))$$

d'où (s', s'') est une transition de $GTA(G', p')$. Par la suite, il existe un chemin allant de s' jusqu'au t' dont les états, excepté t', appartiennent à $B'(s)$.

□

2 Preuve du théorème 6.7

Preuve. Si(s, t) est une transition de $GTA(G, p)$, alors il existe $s' \in B(s)$ et $t' \in B(t)$ tel que (s', t') est une transition de $GTA(G', p')$, donc $(B(s), B(t))$ est une transition de $GTA(G', p')_{/\simeq}$. Montrons alors que si $(B(s), B(t))$ est une transition de $GTA(G', p')_{/\simeq}$ alors (s, t) est une transition de $GTA(G, p)$. Soit $(B(s), B(t))$ une transition de $GTA(G', p')_{/\simeq}$. Il existe $s' \in B(s)$ telle que pour tout $k \in G_i'^{-} \setminus G_i^{-}$, $s'_k = 0$ si $Sl'(k, i) = +$, et $s'_k = b_i$ si $Sl'(k, i) = -$, et il existe $t' \in B(t)$ telles que (s', t') est une transition de $GTA(G', p')$. Par définition, il existe $i \in V'$ tel que $s'_j = t'_j$ pour tout $j \in V' \setminus \{i\}$ et

$$t'_i = \begin{cases} s'_i + 1 \text{ et } s'_i < p'(i, R_{G',i}(s')) \\ s'_i - 1 \text{ et } s'_i > p'(i, R_{G',i}(s')). \end{cases}$$

Clairement, $i \in V$ (autrement s' et t' seront dans la même classe d'équivalence). Supposons que

$$s'_i < p'(i, R_{G',i}(s')),$$

la preuve est similaire dans les autres cas. Soit $j \in V$. Si $t_j < s_j$ alors

$$t_j + 1 \leq s_j$$

donc, par monotonie de σ_j,

$$\sigma_j(t_j + 1) \leq \sigma_j(s_j)$$

et puisque $s' \in B(s)$ et $t' \in B(t')$ nous obtenons

$$t'_j < \sigma_j(t_j + 1) \leq \sigma_j(s_j) \leq s'_j,$$

ce qui est contradictoire. Donc, $s_j \leq t_j$ pour tout $j \in V$. Pareillement, nous montrons que $s_j \geq t_j$ pour tout $j \in V \setminus \{i\}$. Par conséquent,

$$s_i \leq t_i \quad \text{et} \quad s_j = t_j \text{ pour tout } j \in V \setminus \{i\}$$

et puisque $B(s) \neq B(t)$ nous déduisons $s_i < t_i$. Donc $s_i + 1 \leq t_i$ et par monotonie de σ_j,

$$\sigma_j(s_j + 1) \leq \sigma_j(t_j)$$

Puisque $s' \in B(s)$ et $t' \in B(t')$ nous obtenons

$$s'_i < \sigma_i(s_i + 1) \leq \sigma_i(t_i) \leq t'_i = s'_i + 1.$$

Donc,

$$\sigma_i(s_i + 1) = \sigma_i(t_i)$$

σ_i est injective, donc

$$s_i + 1 = t_i$$

et pour montrer que (s, t) est une transition $GTA(G, p)$ il suffit de montrer que

$$s_i < p(i, R_{G,i}(s)) \tag{B.1}$$

Puisque

$$s'_i < p'(i, R_{G',i}(s'))$$

on a

$$t'_i \leq p'(i, R_{G',i}(s'))$$

et puisque $t' \in B(t)$ nous obtenons

$$\sigma_i(t_i) \leq t'_i \leq p'(i, R_{G',i}(s'))$$

et puisque $R'_{G',i}(s') = R_{G,i}(s)$, nous déduisons

$$\sigma_i(t_i) \leq p'(i, R_{G,i}(s))$$

et par définition de p, nous obtenons

$$t_i \leq p(i, R_{G,i}(s))$$

et puisque $s_i < t_i$ l'inégalité (B.1) est prouvée. □

Bibliographie

[1] http://fr.academic.ru/dic.nsf/frwiki/1378907. (Cité pages ix et 109.)

[2] Lynch N. A. et M. R. Tuttle : An introduction to input/output automata. *CWI Quarterly*, 2:219–246, 1989. (Cité pages 67 et 79.)

[3] J. R. Abrial : Spécifier ou comment matérialiser l'abstrait = specification or how to make abstraction real. *Technique et Science Informatique*, 3(3):201–219, 1984. (Cité page 79.)

[4] J. R. Abrial : *The B-book : assigning programs to meanings*. Cambridge University Press, New York, NY, USA, 1996. (Cité page 79.)

[5] M. Aiguier : Étoile-specifications : An object-oriented algebraic formalism with refinement. *Journal of Logic and Computation*, 14(2):145, 2004. (Cité page 150.)

[6] M. Aiguier, J. Benzakki, G. Bernot, S. Beroff, D. Dupont, L. Freund, M. Israel et F. Rousseau : ECOS : A Generic Codesign Environment for the Prototyping of Real Time Applications. *Hardware/Software Co-Design and Co-Verification, Edited by BERGÉ, J. et al. Kluwer Academic Publishers, The Netherlands*, pages 23–58, 1997. (Cité page 3.)

[7] M. Aiguier, P. Le Gall et M. Mabrouki : Emergent properties in reactive systems. *In APSEC*, pages 273–280. IEEE, 2008. (Cité pages 7 et 67.)

[8] M. Aiguier, P. Le Gall et M. Mabrouki : A formal definition of complex software. *In ICSEA*, pages 415–420. IEEE Computer Society, 2008. (Cité page 7.)

[9] M. Aiguier, P. Le Gall et M. Mabrouki : Complex software systems : Formalization and applications. *International Journal on Advances in Software*, 2(1):47–62, 2009. (Cité page 7.)

[10] Y. Aït Ameur, R. Delmas et V. Wiels : Un cadre formel pour la spécification multivue de systèmes avioniques. *Technique et Science Informatiques*, 25(1):43–72, 2006. (Cité page 3.)

[11] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts et Watson J.D. : *Molecular biology of the cell, Fourth Edition*. Garland Publishing, 2002. (Cité page 107.)

[12] R. Allen : *A Formal Approach to Software Architecture*. Thèse de doctorat, Carnegie Mellon, School of Computer Science, Junuary 1997. Issued as CMU Technical Report CMU-CS-97-144. (Cité page 6.)

[13] R. Allen et D. Garlan : A formal basis for architectural connectors. *ACM TOSEM*, 6(3):213–249, 1997. (Cité page 6.)

[14] A. Arnold : *Systèmes de transitions finis et sémantique des processus communicants*. 1992. (Cité page 64.)

[15] A. Arnould et M. C. Gaudel : Test à partir de spécifications de structures bornées : une théorie du test, une méthode de sélection, un outil d'assistance à la sélection= Testing from bounded data type specifications a testing theory, a selection method, a tool for selection assistance. 1997. (Cité page 3.)

[16] E. Astesiano : *Algebraic Foundations of Systems Specification*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999. (Cité page 53.)

[17] E. Astesiano, M. Bidoit, H. Kirchner, B. Krieg-Brückner, P. D. Mosses, D. Sannella et A. Tarlecki : Casl : the common algebraic specification language. *Theor. Comput. Sci.*, 286(2):153–196, 2002. (Cité page 25.)

[18] F. Barbier : *Résultats de théorie abstraite des modèles dans le cadre des institutions : vers la combinaison de logiques*. Thèse de doctorat, Université d'Evry-Val d'Essonne, France, 2005. (Cité page 53.)

[19] G. Batt, D. Ropers, H. De Jong, J. Geiselmann, R. Mateescu, M. Page et D. Schneider : Validation of qualitative models of genetic regulatory networks by model checking : Analysis of the nutritional stress response in Escherichia coli. *Bioinformatics*, 21(Suppl 1):i19, 2005. (Cité page 118.)

[20] G. Bernot et M. Bidoit : Proving the correctness of algebraically specified software : Modularity and observability issues. *In AMAST '91 : Proceedings of the Second International Conference on Methodology and Software Technology*, pages 216–239, London, UK, 1992. Springer-Verlag. (Cité page 23.)

[21] G. Bernot et J.P. Comet : On the use of temporal formal logic to model gene regulatory networks. *In 6th International Meeting on Computational Intelligence Methods for Bioinformatics and biostatistics, CIBB'2009*, volume 6160, pages 112–138, 2010. (Cité page 5.)

[22] G. Bernot, J.P. Comet, A. Richard et J. Guespin : Application of formal methods to biological regulatory networks : extending thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347, 2004. (Cité pages 5, 99, 103, 105, 113, 114, 115, 118, 121 et 125.)

[23] G. Bernot, S. Coudert et P. Le Gall : Towards heterogeneous formal specification. *In AMAST*, volume 1101 de *Lecture Notes in Computer Science*, pages 458–472. Springer, 1996. (Cité page 3.)

[24] G. Bernot et F. Tahi : Behaviour Preservation of a Biological Regulatory Network when Embedded into a Larger Network. *Fundamenta Informaticae*, 91(3):463–485, 2009. (Cité pages 145 et 147.)

[25] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri et P. Traverso : NuSMV 2 : An Open Source Tool for Symbolic Model Checking. *In Proc. of International Conference on Computer-Aided Verification*, 2002. (Cité page 125.)

[26] M. Bidoit : The stratified loose approach : A generalization of initial and loose semantics. *In ADT*, volume 332 de *Lecture Notes in Computer Science*, pages 1–22. Springer, 1987. (Cité pages 2, 23 et 30.)

[27] M. Bidoit, D. Sannella et A. Tarlecki : Architectural specifications in CASL. *In Proc. 7th Intl. Conference on Algebraic Methodology and Software Technology (AMAST'98)*, volume 1548 de *Lecture Notes in Computer Science*, pages 341–357. Springer, 1999. (Cité page 25.)

[28] A. Biere, A. Cimatti, E. M. Clarke et Y. Zhu : Symbolic model checking without bdds. *In TACAS '99 : Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 193–207, London, UK, 1999. Springer-Verlag. (Cité page 56.)

[29] T. Borzyszkowski : Logical systems for structured specifications. *Theor. Comput. Sci.*, 286(2):197–245, 2002. (Cité pages 1, 3, 30, 33, 86, 87 et 88.)

[30] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill et L. J. Hwang : Symbolic model checking : 1020 states and beyond. *Inf. Comput.*, 98(2):142–170, 1992. (Cité page 57.)

[31] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill et L. J. Hwang : Symbolic model checking : 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992. (Cité page 125.)

[32] R. M. Burstall et J. A. Goguen : The semantics of clear, a specification language. *In Proceedings of the Abstract Software Specifications, 1979 Copenhagen Winter School*, pages 292–332, London, UK, 1980. Springer-Verlag. (Cité page 25.)

[33] E.J. Cameron et H. Velthuijsen : Feature interactions in telecommunications systems. *Communications Magazine, IEEE*, 31(8):18–23, 2002. (Cité page 3.)

[34] K. M. Chandy et M. Misra : *Parallel Program Design : A Foundation*. Addison Wesley Publishing Company, Inc., Reading, MA, USA, 1988. (Cité page 79.)

[35] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani et A. Tacchella : Nusmv 2 : An opensource tool for symbolic model checking. *In CAV '02 : Proceedings of the 14th International Conference on Computer Aided Verification*, pages 359–364, London, UK, 2002. Springer-Verlag. (Cité page 57.)

[36] A. Cimatti, Clarke E. M., F. Giunchiglia et M. Roveri : Nusmv : a reimplementation of smv. *In STTT'98 : Proceeding of the International Workshop on Software Tools for Technology Transfer*, pages 25–31, 1998. (Cité page 57.)

[37] E. M. Clarke et E. A. Emerson : Design and synthesis of synchronization skeletons using branching-time temporal logic. *In Logic of Programs, Workshop*, pages 52–71, London, UK, 1982. Springer-Verlag. (Cité pages 51 et 56.)

[38] E. M. Clarke, E. A. Emerson et A. P. Sistla : Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986. (Cité page 53.)

[39] R. Cori et D. Lascar : Logique mathématique, tome 2 : Fonctions récursives, théorème de Gödel, théorie des ensembles. *Dunod, January*, 13, 2003. (Cité page 95.)

[40] O. Coudert, J. C. Madre et C. Berthet : Verifying temporal properties of sequential machines without building their state diagrams. *In CAV '90 : Proceedings of the 2nd International Workshop on Computer Aided Verification*, pages 23–32, London, UK, 1991. Springer-Verlag. (Cité page 57.)

[41] R. I. Damper : Emergence and levels of abstraction. *International Journal of Systems Science*, 31(7):811–818, 2000. Editorial for the Special Issue on 'Emergent Properties of Complex Systems'. (Cité page 4.)

[42] H. De Jong : Modeling and simulation of genetic regulatory systems : a literature review. *Journal of computational biology*, 9(1):67–103, 2002. (Cité page 111.)

[43] R. De Nicola : Three logics for branching bisimulation. *Journal of the ACM (JACM)*, 42(2):458–487, 1995. (Cité pages 64, 74 et 75.)

[44] R. Diaconescu, J. Goguen et P. Stefaneas : Logical support for modularisation. *Logical Environments*, pages 83–130, 1993. (Cité page 2.)

[45] M. Doche et V. Wiels : Extended institutions for testing. *In AMAST '00 : Proceedings of the 8th International Conference on Algebraic Methodology and Software Technology*, pages 514–528, London, UK, 2000. Springer-Verlag. (Cité page 48.)

[46] A. C. Ehresmann et J. P. Vanbremeersch : *Memory Evolutive Systems : Hierarchy, Emergence, Cognition*. Elsevier Science, 2007. (Cité page 4.)

[47] E. A. Emerson et E. M. Clarke : Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program.*, 2(3):241–266, 1982. (Cité pages 51 et 53.)

[48] E. A. Emerson et J. Y. Halpern : "sometimes" and "not never" revisited : on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986. (Cité pages 51, 56 et 61.)

[49] E. A Emerson et A. P. Sistla : Deciding full branching time logic. Rapport technique, Austin, TX, USA, 1985. (Cité pages 51 et 61.)

[50] J. L. Fiadeiro et A. Lopes : Community on the move : Architectures for distribution and mobility. *In* F. S. de Boer, M. M. Bonsangue, S. Graf et W. P. de Roever, éditeurs : *FMCO*, volume 3188 de *Lecture Notes in Computer Science*, pages 177–196. Springer, 2003. (Cité pages 6 et 79.)

[51] J. L. Fiadeiro et T. S. E. Maibaum : Categorical semantics of parallel program design. *Science of Computer Programming*, 28(2-3):111–138, 1997. (Cité page 6.)

[52] K. Futatsugi, J. A. Goguen, J. P. Jouannaud et J. Meseguer : Principles of obj2. *In POPL '85 : Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 52–66, New York, NY, USA, 1985. ACM. (Cité page 25.)

[53] D. Garlan, R. T. Monroe et D. Wile : Acme : An architecture description interchange language. *In CASCON 1997 : Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research*, pages 169–183. IBM Press, 1997. (Cité page 6.)

[54] C. Gaston, M. Aiguier et P. Le Gall : Algebraic treatment of feature-oriented systems, 2000. (Cité page 150.)

[55] Ch. Gaston : *Une description générique formelle des systèmes à base de services : Application du raffinement et de l'enrichissement algébrique au problème des interactions de services*. Thèse de doctorat, Université d'Evry-Val d'Essonne, France, 2002. (Cité pages 3 et 150.)

[56] L. Glass et S.A. Kauffman : Co-operative components, spatial localization and oscillatory cellular dynamics. *Journal of theoretical biology*, 34(2):219–237, 1972. (Cité page 112.)

[57] L. Glass et S.A. Kauffman : The logical analysis of continuous nonlinear biochemical control networks. *J. Theor. Biol.*, 39:103–129, 1973. (Cité page 112.)

[58] P. Godefroid : On the costs and benefits of using partial-order methods for the verification of concurrent systems. *In POMIV '96 : Proceedings of the DIMACS workshop on Partial order methods in verification*, pages 289–303, New York, NY, USA, 1997. AMS Press, Inc. (Cité page 57.)

[59] J. Goguen : *Advances in Cybernetics and Systems Research*, chapitre Categorical Foundations for General Systems Theory, pages 121–130. Transcripta Books, 1973. (Cité page 87.)

[60] J. Goguen, C. Kirchner, J. Meseguer, H. Kirchner, T. Winkler et A. Megrelis : An introduction to obj 3. *In 1st international workshop on Conditional Term Rewriting Systems*, pages 258–263, London, UK, 1988. Springer-Verlag. (Cité page 25.)

[61] J. A. Goguen : Sheaf semantics for concurrent interacting objects. *Mathematical Structures in Computer Science*, 2(2):159–191, 1992. (Cité page 87.)

[62] J. A. Goguen, J. W. Thatcher, E. G. Wagner et Wright J. B. : Abstract data types as initial algebras and the correctness of data representations. pages 89–93, 1975. (Cité page 11.)

[63] J.A. Goguen et Burstall R.M. : Institutions : Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992. (Cité pages 2, 48, 80 et 86.)

[64] J.A. Goguen, J. W. Thatcher et E. G. Wagner : An initial algebra approach to the specification, correctness, and implementation of abstract data types. pages 80–149, 1978. (Cité pages 11 et 34.)

[65] Joseph A. Goguen et Rod M. Burstall : Introducing institutions. *In Proceedings of the Carnegie Mellon Workshop on Logic of Programs*, pages 221–256, London, UK, 1984. Springer-Verlag. (Cité pages 2, 48, 49 et 80.)

[66] J. V. Guttag : *The specification and application to programming of abstract data types*. Thèse de doctorat, Toronto, Ont., Canada, Canada, 1975. (Cité page 11.)

[67] J. V. Guttag et J. J. Horning : The algebraic specification of abstract data types. *Acta Inf.*, 10:27–52, 1978. (Cité pages 11 et 38.)

[68] J. V. Guttag, J. J. Horning et J. M. Wing : Some remarks on putting formal specifications to productive use. *SCP*, 2(1):53–68, 1982. (Cité page 79.)

[69] J. V. Guttag, J. J. Horning et J. M. Wing : The larch family of specification languages. *IEEE Softw.*, 2(5):24–36, 1985. (Cité page 25.)

[70] D. Harel et A. Pnueli : On the development of reactive systems. pages 477–498, 1985. (Cité page 67.)

[71] R. Harper, D. Sannella et A. Tarlecki : Structured theory presentations and logic representations. *Annals of Pure and Applied Logic*, 67:113–160, 1994. (Cité pages 30, 33 et 86.)

[72] R. Hennicker et M. Wirsing : Proof systems for structured algebraic specifications : An overview, 1997. (Cité page 31.)

[73] C. A. R. Hoare : *Communicating Sequential Processes*. Prentice-Hall, 1985. (Cité page 79.)

[74] J. Jacob, F. nd Monod : Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology*, 3:318–356, 1961. (Cité pages 111 et 112.)

[75] C. B. Jones : *Systematic software development using VDM (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990. (Cité page 79.)

[76] S.A. Kauffman et K. Glass : The logical analysis of continuous, nonlinear biochemical control networks. *Journal of Theoretical Biology*, 39:103–129, 1973. (Cité page 112.)

[77] M. Kaufman : Role of multistationarity in an immune response model : a combined discrete and continuous approach. *Theoretical Immunology, Addison Wesley, New York*, pages 199–222, 1988. (Cité page 112.)

[78] M. Kaufman et R. Thomas : Model analysis of the bases of multistationarity in the humoral immune response. *Journal of theoretical biology*, 129(2):141–162, 1987. (Cité page 112.)

[79] M. Kaufman, J. Urbain et R. Thomas : Towards a logical analysis of the immune response. *Journal of theoretical biology*, 114(4):527–561, 1985. (Cité page 112.)

[80] Robert M. Keller : Formal verification of parallel programs. *Commun. ACM*, 19(7):371–384, 1976. (Cité page 67.)

[81] D. Krob : Eléments d'architecture des systèmes complexes. (Cité page 150.)

[82] D. Krob : Modelling of complex software systems : a reasoned overview. *Formal Techniques for Networked and Distributed Systems-FORTE 2006*, pages 1–22, 2006. (Cité page 150.)

[83] O. Kupferman, M.Y. Vardi et P. Wolper : An automata-theoretic approach to branching-time model checking. *Journal of the ACM (JACM)*, 47(2):312–360, 2000. (Cité page 118.)

[84] P. Le Gall et A. Arnould : Formal specifications and test : Correctness and oracle. *In COMPASS/ADT*, pages 342–358, 1995. (Cité page 48.)

[85] P. Le Gall et A. Arnould : Formal specifications and test : Correctness and oracle. *Recent Trends in Data Type Specification*, pages 342–358, 1996. (Cité page 3.)

[86] B. Lewin : *Genes VIII*. Prentice Hall, 2004. (Cité pages 4 et 107.)

[87] B. Liskov et S. Zilles : Programming with abstract data types. *In Proceedings of the ACM SIGPLAN symposium on Very high level languages table of contents*, pages 50–59. ACM New York, NY, USA, 1974. (Cité page 11.)

[88] J. Loeckx, H. D. Ehrich et M. Wolf : *Specification of Abstract Data Types*. New York, USA : Wiley, 1996. (Cité page 13.)

[89] M. Mabrouki, M. Aiguier, J.P. Comet et P. Le Gall : Property preservation along embedding of biological regulatory networks. *Lecture Notes in Computer Science*, 5147:125–138, 2008. (Cité pages 7 et 114.)

[90] L. Mendoza et E.R. Alvarez-Buylla : Genetic regulation of root hair development in arabidopsis thaliana : a network model. *Journal of Theoretical Biology*, 204(3):311–326, 2000. (Cité page 112.)

[91] L. Mendoza, D. Thieffry et E.R. Alvarez-Buylla : Genetic control of flower morphogenesis in arabidopsis thaliana : a logical analysis. *Bioinformatics*, 15(7):593–606, 1999. (Cité page 112.)

[92] J. Meseguer : General logics. 1989. (Cité page 3.)

[93] B. Meyer : On formalism in specifications. *IEEE Software*, 2(1):6–26, 1985. (Cité page 79.)

[94] R. Milner : An algebraic definition of simulation between programs. Rapport technique, Stanford, CA, USA, 1971. (Cité page 57.)

[95] R. Milner : *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982. (Cité pages 67, 79 et 93.)

[96] T. Mossakowski et A. Tarlecki : Heterogeneous logical environments for distributed specifications. *Recent Trends in Algebraic Development Techniques*, pages 266–289, 2009. (Cité page 3.)

[97] E. Muraille, D. Thieffry, O. Leo et M. Kaufman : Toxicity and neuroendocrine regulation of the immune response : a model analysis. *Journal of theoretical biology*, 183(3):285–305, 1996. (Cité page 112.)

[98] M. Nielsen, K. Havelund, K. R. Wagner et C. George : The raise language, methods and tools. *Formal Aspects of Computing*, 1(1):85–114, 1989. (Cité page 79.)

[99] S. Ogden, D. Haggerty, C. M. Stoner, D. Kolodrubetz et R. Schleif : The escherichia coli l-arabinose operon : binding sites of the regulatory proteins and a mechanism of positive and negative regulation. *PNA*, 77(6):3346–3350, 1980. (Cité page 115.)

[100] F. Orejas : Structuring and modularity. *In on Algebraic Foundations of Systems Specification, Chapter 6*, pages 159–200. Springer, 1996. (Cité pages 2, 3, 30 et 86.)

[101] D. Park : Concurrency and automata on infinite sequences. *In Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pages 167–183, London, UK, 1981. Springer-Verlag. (Cité page 57.)

[102] James L. Peterson : Petri nets. *ACM Comput. Surv.*, 9(3):223–252, 1977. (Cité page 79.)

[103] A. Pnueli : The temporal logic of programs. *In FOCS*, pages 46–57. IEEE, 1977. (Cité page 56.)

[104] M. Ptashne : *A genetic switch : Phage [lambda] and higher organisms*. Blackwell Science Inc, 1992. (Cité page 116.)

[105] M. Ptashne : *A genetic switch : Phage [lambda] and higher organisms*. Blackwell Science Inc, 1992. (Cité page 117.)

[106] J. P. Queille et J. Sifakis : Specification and verification of concurrent systems in cesar. *In Proceedings of the 5th Colloquium on International Symposium on Programming*, pages 337–351, London, UK, 1982. Springer-Verlag. (Cité page 56.)

[107] N. Rapin, C. Gaston, Lapitre A. et J. P. Gallois : Behavioural unfolding of formal specifications based on communicating extended automata. *In Workshop ATVA*, Tapei, Taiwan. (Cité page 67.)

[108] A. Richard : *Modèle formel pour les réseaux de régulation génétique et influence des circuits de rétroaction*. Thèse de doctorat, Université d'Evry-Val d'Essonne, France, 2006. (Cité pages 113 et 125.)

[109] L. Sanchez et D. Thieffry : A logical analysis of the drosophila gap-gene system. *Journal of theoretical Biology*, 211(2):115–141, 2001. (Cité page 112.)

[110] L. Sanchez et D. Thieffry : Segmenting the fly embryo : a logical analysis of the pair-rule cross-regulatory module. *Journal of theoretical Biology*, 224(4):517–537, 2003. (Cité page 112.)

[111] L. Sanchez, D. van Helden et D. Thieffry : Establishment of the dorso-ventral pattern during embryonic development of drosophila melanogaster : a logical analysis. *J. Theor. Biol*, 189:337–389, 1997. (Cité page 112.)

[112] D. Sannella : The Common Framework Initiative for algebraic specification and development of software. *In Proc. 3rd Intl. Conf. on Perspectives of System Informatics (PSI'99)*, volume 1755 de *Lecture Notes in Computer Science*, pages 1–9. Springer, 2000. (Cité page 25.)

[113] D. Sannella et A. Tarlecki : Specifications in an arbitrary institution. *Inf. Comput.*, 76(2/3):165–210, 1988. (Cité page 48.)

[114] T. Schlitt et A. Brazma : Current approaches to gene regulatory network modelling. *BMC bioinformatics*, 8(Suppl 6):S9, 2007. (Cité page 111.)

[115] E. Simao, E. Remy, D. Thieffry et C. Chaouiya : Qualitative modelling of regulated metabolic pathways : application to the tryptophan biosynthesis in e. coli. *Bioinformatics*, 21(2):190–196, 2005. (Cité page 111.)

[116] H. A. Simon : The science of the artificial. *MIT Press*, 1967. (Cité page 4.)

[117] J. M. Spivey : *Understanding Z : a specification language and its formal semantics*. Cambridge University Press, New York, NY, USA, 1988. (Cité page 79.)

[118] A. Tarlecki : *Algebraic Foundations of Systems Specification*, chapitre Institutions : An abstract Framework for Formal Specifications, pages 105–131. IFIP State-of-the-Art Reports. Springer, 1999. (Cité pages 13 et 48.)

[119] D. Thieffry : From global expression data to gene networks. *BioEssays*, 21(11):895–899, 1999. (Cité page 111.)

[120] D. Thieffry et R. Thomas : Dynamical behaviour of biological regulatory networks - ii. immunity control in bacteriophage lambda. *Bull. Math. Biol.*, 57(2):277–297, 1995. (Cité pages 112 et 117.)

[121] R. Thomas : Kinetic logic : A boolean approach to the analysis of complex regulatory systems, vol. 29 of. *Lecture Notes in Biomathematics*, 1979. (Cité page 112.)

[122] R. Thomas : Some biological examples. *Lecture Notes in Biomathematics*, 29:354–401, 1979. (Cité page 112.)

[123] R. Thomas : On the relation between the logical structure of systems and their ability to generate multiple steady states and sustained oscillations. Series in Synergetics, 9 : 180-193, 1981. (Cité page 125.)

[124] R. Thomas : Regulatory networks seen as asynchronous automata : a logical description. *Journal of Theoretical Biology*, 153(1):1–23, 1991. (Cité pages 5, 8, 105, 111, 112, 115, 121 et 122.)

[125] R. Thomas et R. D'Ari : *Biological feedback*. CRC, 1990. (Cité pages 8, 105, 111, 112, 121 et 122.)

[126] R. Thomas et M. Kaufman : Multistationarity, the basis of cell differentiation and memory. ii. logical analysis of regulatory networks in terms of feedback circuits. *Chaos : An Interdisciplinary Journal of Nonlinear Science*, 11:180–195, 2001. (Cité page 122.)

[127] R. Thomas, D. Thieffry et M. Kaufman : Dynamical behaviour of biological regulatory networks - i. biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bulletin of Mathematical Biology*, 57(2):247–276, 1995. (Cité page 112.)

[128] J. Tretmans : Conformance testing with labelled transition systems : Implementation relations and test generation. *Computer Networks and ISDN Systems*, 29(1):49–79, 1996. (Cité page 67.)

[129] M. Y. Vardi : A temporal fixpoint calculus. *In POPL '88 : Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 250–259, New York, NY, USA, 1988. ACM. (Cité page 56.)

[130] J.D. Watson, T.A. Baker, S.P. Bell, A. Gann, M. Levine et R. Losick : *Molecular Biology of the Gene, Fifth Edition*. Cold Spring Harbor Laboratory Press, 2003. (Cité page 107.)

[131] J. M. Wing : A specifier's introduction to formal methods. *Computer*, 23(9):8–23, 1990. (Cité page 79.)

[132] M. Wirsing : Structured specifications : Syntax, semantics and proof calculus. *Logic and Algebra of Specification*, 94:411–442. (Cité pages 1, 27, 30, 31, 33 et 86.)