Sémantique des systèmes réactifs : raffinement, bisimulations et sémantique opérationnelle structurée dans les systèmes de transitions asynchrones

Juan Vicente Echagüe Zappettini

THÈSE

présentée par

Juan Vicente Echagüe Zappettini

pour obtenir le titre de Docteur
de l'Institut National Polytechnique de Grenoble

(Arrêté ministériel du 30 mars 1992)

Spécialité: Informatique

Sémantique des Systèmes Réactifs: Raffinement, Bisimulations et Sémantique Opérationnelle Structurée dans les Systèmes de Transitions Asynchrones

Thèse soutenue le 14 Juin 1993 devant le jury:

J. Sifakis	président
Ph. Darondeau R. Gorrieri	rapporteurs
Ph. Jorrand Ph. Schnoebelen	examinateurs

A Virginia
1m78 de espléndida
mujer envuelta en
piel dorada.

Remerciements

Je tiens à remercier les membres du jury:

- Joseph Sifakis qui m'a fait l'honneur de présider ce jury,
- Philippe Darondeau et Roberto Gorrieri qui ont accepté d'être rapporteurs. Cette tâche habituellement ingrate était encore alourdie par l'état inachevé du manuscrit et par mes demandes très strictes quant aux délais.

Je voudrais de même remercier Philippe Jorrand, qui a encadré cette thèse, avec le difficile rôle paternel que cette tâche suppose.

Et particulièrement Philippe Schnoebelen, qui a beaucoup collaboré à ma formation. Quelques unes des questions essentielles à ce travail, ont surgi de discussions avec lui. Pour finir, mais sans être le moins important, il m'a prêté une partie substantielle de sa collection de BD.

Je me sens aussi reconnaissant à Sophie Pinchinat et à Zineb Habbas. Les résultats de cette thèse sont dans son écrasante majorité le produit d'un effort commun. Dans le cadre d'une difficile comparaison, je crois être le plus entêté des trois, et je revendique donc le fait d'être le seul responsable des erreurs qui puissent subsister dans ce texte.

Le fait de travailler avec Zineb et Sophie fut une expérience remplie de plaisir, stimulante, pleine d'énergie. Sans elles, cette thèse ne serait jamais arrivée à son état actuel.

Je veux de même manifester ma reconnaissance aux membres de l'équipe parallelisme, anciens ou actuels, par leur gentillesse et tout l'intérêt qu'ils m'ont porté. François Laroussinie a lu et corrigé certaines parties de ce travail à des moments clés.

A tous les membres du LIFIA, pour la construction commune d'un lieu de travail. Laurence Neault et Brigitte Vivion, qui, dans l'étape finale m'ont aidé avec leur compétence et leurs sourires.

Aussi Ricardo Caferra, par le fait de m'avoir orienté à mon arrivée. Et lui faire parvenir l'importance de son appui et l'affection qui m'unit à lui, à Marta et à Gabriel.

Hier et aujourd'hui, Omar Macadar m'a appris plus que ce que j'ai pu apprendre sur la science, et m'a permis de découvrir combien j'aime mes défauts. Ruben Budelli m'a montré une façon de travailler que j'essaye de cultiver, et il m'a appris à claquer les doigts lorsque je marche dans les couloirs. Le travail passioné de Jorge Vidart m'a permis de m'imaginer

consacré à l'informatique, de me rapprocher de Grenoble et de vivre l'expérience de l'ESLAI et de l'Argentine. Alicia Golijov m'a poussé à danser et a rendu possible le fait que je découvre qui je suis. Et Pierre Arel m'a écouté.

Je veux manifester toute mon affection à mes familles. A Virginia, ma femme et beaucoup plus. A ma famille "de sang", la première qu'on choisit, qui m'a soutenu tout au long de ces longues années de éloignement. Elle fût près de moi lorsqu'un de mes yeux commença à passer de l'autre côté. Et aux autres familles.

Les membres de mes familles et mes amis sont aujourd'hui en Allemagne, en Angleterre, en Argentine, au Canada, aux Etats Unis, en France, en Hollande, en Italie, au Mexique, en Suède et en Uruguay. Je veux de même laisser ici empreints les noms de Alejandra, Alfredo, Ana y Herve, Daniel, Dominique, Fabrice, Frédéric, Jean-Jacques, Margit, Ma. Noel, Nathalie, Olivier, Pedro, Sami, Sergio, Sophie, Stephane, Valerie et Zineb.

Et bien sur les noms de Agustín, Inés, Cecilia, Elena, Florencia, Jeremias, Joaquín, Maite, Nicolás, Pedro et Victoria.

Sommaire

Liste des Figures

Chapitre 1

Introduction

Cette thèse s'inscrit dans le cadre de l'étude de la *sémantique formelle des systèmes réactifs.*

Une sémantique formelle est une *fonction* qui relie chaque expression écrite dans un certain langage de programmation ou de spécification, à une structure (où *modèle*) dans un domaine mathématique (dit aussi *domaine sémantique*).

Cette association nous permet de plonger nos programmes et nos spécifications dans un univers de discours très riche, contenant toute la mathématique. Les retombées de cette démarche sont immenses: on hérite de toute la machinerie développée depuis des siècles, et de son mode d'emploi: il faut "faire de la mathématique". Cette voie nous permet de disposer d'*expressions formelles* pour décrire les propriétés de nos programmes, mais aussi de la notion de *preuve mathématique* comme façon de justifier des affirmations.

Ce que nous attendons de cette approche mathématique est une façon d'aborder le *problème de la correction des programmes* qui est au cœur de l'informatique: *Le programme fait-il bien ce que l'on attend de lui?.*

Nous pourrons donner des expressions formelles pour représenter les programmes et les spécifications, pour formaliser les propriétés sur lesquelles on s'interroge[1] et utiliser l'existence des preuves mathématiques pour nous convaincre qu'un programme possède cette propriété.

L'espoir derrière ce recours à la mathématique est fondé sur l'expérience des démarches semblables dans les autres branches de la science.

En plus, cette approche a déjà fait ses preuves dans la famille des programmes qui acceptent une entrée et calculent une sortie, sans d'autres interactions avec l'extérieur. On les appelle sou-

[1] comme "le résultat est un nombre inférieur à 10", "le programme s'arrête toujours", "le programme attend 15 secondes qu'on lui délivre notre mot de passe" ou "pour un mot de passe de n caractères, l'action α s'exécute $2 \times n$ fois"

vent les *systèmes transformationnels.* On dispose déjà de techniques bien établies, par exemple, celles fondées sur le λ-calcul, la réécriture, la théorie des domaines, les spécifications algébriques et les logiques de Hoare (pour une vision de l'ensemble, voir [Lee90]).

La situation est moins satisfaisante quand l'interaction du programme avec son environnement est plus complexe que dans le cas des systèmes transformationnels. C'est le cas par exemple des interfaces basées sur l'usage de souris et plusieurs fenêtres, de programmes de contrôle de processus industriels ou, simplement, des systèmes d'exploitation.

Il est difficile pour ces programmes, qui ont comme vocation d'entretenir une certaine interaction avec l'environnement, de les réduire au calcul d'une relation d'entrée-sortie. Pnueli [Pnu85] appelle *systèmes réactifs* ce type de programmes, par opposition aux systèmes transformationnels.

Une multitude de modèles pour les systèmes réactifs existent dans la littérature. Puisqu'il n'y a pas de notion formelle de "comportement réactif élémentaire" qui soit universellement acceptée, ces modèles ne sont pas tous réductibles les uns aux autres. Le débat sur la pertinence de l'un ou de l'autre (ou d'une famille de modèles ou d'une autre) s'est développé dans la communauté scientifique.

Il y a des modèles qui représentent des automates, d'autres qui sont plutôt des comportements de ces automates. Il y a des modèles qui représentent "toutes les exécutions possibles" comme l'ensemble des exécutions, d'autres donnent à cet ensemble une structure (par exemple, un arbre). Il y a des modèles qui considèrent qu'un système peut être engagé dans plusieurs activités au même instant, d'autres qui considèrent que cette activité ne peut être qu'unique. Ces activités peuvent avoir une durée, cette durée peut être munie d'une structure algébrique. On peut avoir des activités "invisibles"... Et la liste d'options est loin d'être achevée.

1.1 Les questions abordées

Dans cette thèse nous travaillons sur une classe particulière de modèles de systèmes réactifs: les *Systèmes de Transitions Asynchrones* , et nous nous intéressons à trois questions: le *raffinement d'actions*, les *équivalences sémantiques*, et particulièrement les bisimulations compatibles avec le raffinement d'actions, et la technique de *Sémantique Opérationnelle Structurée* pour la description de fonctions sémantiques.

Les Systèmes de Transitions Asynchrones

Les *Systèmes de Transitions Asynchrones (STA)* on été introduits de façon indépendante par Bednarczyk [Bed87] et Shields [Shi85b, Shi85a], comme une généralisation simple des *Systèmes de Transitions (ST)* [Kel76].

Les ST sont les modèles de systèmes réactifs les mieux connus. Ce sont des graphes orientés et étiquetés. Les sommets du graphe représentent les *états* du système et les flèches représentent les *transitions* entre les états, munies d'une étiquette prise dans un certain *alphabet d'actions.*

Les STA généralisent les ST; les étiquettes des transitions sont des éléments d'un ensemble d'*événements*, muni d'une relation d'*indépendance.* Intuitivement, deux événements indépendants ne partagent pas de ressources, autrement dit, ils peuvent s'exécuter en parallèle. Dans la

famille des modèles de systèmes réactifs, les STA se placent parmi les *systèmes*[2], *arborescents*[3] et *non-entrelacement*[4]. Nous renvoyons à Sassone, Nielsen et Winskel [SNW93] pour une classification des différentes familles de modèles.

Nous n'essaierons pas de convaincre le lecteur de l'importance des STA en disant que la représentation du comportement qu'il permet d'exprimer est *adéquate* par rapport à l'intuition, ou *suffisamment riche* ou *suffisamment abstraite...* si telle est notre conviction après des mois de travail, seule l'expérience de la communauté au fil du temps pourra peut être utilisée comme fondement de telles affirmations[5].

Nous en choisissons cependant une justification issue de la tradition mathématique: Les STA généralisent de façon simple (mais non-triviale) d'autres modèles bien connus dans la littérature, en permettant de généraliser certaines constructions intéressantes.. En plus, ces modèles permettent de nouvelles constructions qui répondent à de nouvelles questions.

Des constructions catégoriques pour les STA

Des constructions catégoriques ont été proposées dans la littérature pour les STA.

Dans [Bed87] et [WN92] les constructions sont utilisées pour donner une caractérisation de la composition parallèle et de la somme non-déterministe des STA comme produits et sommes dans la catégorie Ces constructions permettent aussi de faire le lien entre les STA et d'autres modèles de systèmes réactifs. Dans ces deux travaux, les morphismes considérés ne préservent pas les transitions.

Dans [MN92] des morphismes pour les STA (appelés "strong foldings") sont proposés. Ces morphismes préservent les transitions et l'indépendance, en généralisant des définitions connues de morphismes entre ST (voir [DDM88] et[AD89]).

Le raffinement d'actions

Le *raffinement d'actions* est une opération ayant pour effet de remplacer les transitions que l'on considère élémentaires à un certain niveau d'abstraction, par des structures plus complexes à un niveau plus bas. Le raffinement est donc un moyen d'établir des liens entre les descriptions d'un système à différents niveaux d'abstraction, il reflète ainsi le développement de programmes par raffinement successifs[6].

Le raffinement d'actions dans les modèles de systèmes réactifs a été introduit dans [CMP87], où les auteurs montrent qu'une forme très naturelle du raffinement d'actions n'est pas compatible avec l'approche d'entrelacement. L'étroite relation entre l'approche d'entrelacement et l'atomicité des transitions est étudiée aussi par Lamport [Lam86] et Pratt [Pra86][7].

---

[2] i.e. des automates, par opposition à des *comportements* de ces automates.

[3] les exécutions sont structurées en un arbre, où l'information sur les points de choix est représentée, par opposition aux modèles *linéaires*, où cette structure est simplement un ensemble.

[4] l'information sur le parallélisme est représentée directement, et n'est pas modélisée comme une construction dérivée du non-déterminisme et de la composition séquentielle.

[5] nous nous rappelons d'Umberto Eco, "Io ci faccio la tesi, uno che fa la tesi sulla sifilide finisce par amare anche la spirocheta pallida" (Il Pendolo di Foucault).

[6] une version moins optimiste de cette remarque nous dit que nous ne pourrons jamais comprendre le développement par raffinement successifs si on ne comprend pas le raffinement d'actions.

[7] "exactly what is interleaved depends on which events of a process one takes to be atomic", [Pra86].

Une approche possible pour le raffinement consiste à définir un *raffinement atomique* [Bou89, GMM90, Gor91], où les actions sont considérées comme atomiques, et le raffinement doit, dans un certain sens, préserver cette atomicité.

Nous suivrons ici une approche plus libérale qui n'est pas fondée sur l'atomicité. Cette approche a été étudiée sur les ST, plusieurs variantes des Structures d'Evénements [NPW81, Win89a], les arbres Causaux [DD89], et quelques variantes des Réseaux de Petri [Pet62, Rei85], voir [GW89, BDE, BDKP91, DD90, GGR93, GG89a, GG89c, Gla90a, GG89b, GG90b, Gla90b, GG90a, Vog90b, Vog90a, Gor91, DG92, CR93].

La technique est quasiment identique pour tous ces modèles: chaque fonction de l'ensemble des actions dans le domaine sémantique (satisfaisant certains contraintes) détermine un opérateur de raffinement.

Les bisimulations

L'étude d'un domaine sémantique est indissociable de celle des *relations d'équivalence* dans ce domaine.

Les bisimulations sont des relations d'équivalence particulièrement fines qui prennent en compte la structure arborescente du comportement. La bisimulation (classique) a été introduite par Milner [Mil80] et Park [Par81] pour les ST, et elle est compatible avec le raffinement d'actions sur les ST. Elle peut être caractérisée par les morphismes sur les ST (voir [AD89]).

La bisimulation classique a été étendue par Boudol et Castellani [BC87] (sous le nom de *pomset bisimulation*) sur des modèles de non-entrelacement. Mais la pomset bisimulation n'est pas compatible avec le raffinement ([GG89a]).

Deux familles de bisimulations compatibles avec le raffinement sur les systèmes non-entrelacement ont été étudiées: les *ST-bisimulations*[8] de van Glabbeek, Vaandrager et Vogler [GV87, Gla90b, Vog90a] et la *behaviour structure bisimulation* de Rabinovich et Trachtenbrot [RT88] [9].

Les ST-bisimulations prennent en compte la notion d'état, mais aussi celle de transitions concurrentes qui sont en train d'être exécutées. La version ST de la bisimulation forte est la plus large équivalence sur les Structures d'Événements compatible avec le raffinement.

La history preserving bisimulation, dans sa caractérisation comme équivalence d'ordre mixte (mo-equivalence) préserve l'ordre partiel qui représente la causalité, et un ordre total qui reflète une linéarisation de l'exécution.

La Sémantique Opérationnelle Structurée

Plotkin [Plo81] a proposé une méthode générale pour définir la sémantique opérationnelle des langages de programmation dans les ST. Cette méthode, connue sous le nom de *SOS* (pour "Structural Operational Semantics"), consiste à définir des règles structurelles sur les termes du langage. Les règles forment un système de déduction naturelle qui permet de prouver les transitions d'états d'un ST, où les états sont les termes du langage.

De nos jours, la méthode SOS est très largement utilisée, et on compte de nombreux langages munis d'une sémantique SOS, e.g. pour CCS voir [Mil80, Mil89].

[8] ST pour "Stellen", "Transitionen".

[9] qui coïncide avec la *history preserving bisimulation* de van Glabbeek et Goltz [GG89a], la *fully concurrent bisimulation* de Best, Devillers, Kiehn et Pomello [BDKP91], l'*équivalence d'ordre mixte* de Degano, De Nicola et Montanari [DDM89], l'*équivalence causale* de Darondeau et Degano [DD89] et la *pomset bisimulation avant-arrière (sur le calcul)* de Cherief [Che92a].

Badouel et Darondeau [BD92] proposent une méthode générale pour extraire à partir des preuves de transitions, l'information de transitions indépendantes. Ils considèrent des spécifications SOS d'un format particulier (appelé *SOS de base*), à partir des quelles ils engendrent des Automates de Traces [Sta89b, Sta89a]. A notre connaissance, la technique pour extraire une information à partir des preuves de transitions est due à [BC88], où une sémantique non-entrelacement pour un sous-ensemble de CCS est donnée. Depuis, cette approche s'est répandue dans la littérature (voir par exemple [CGM90, Fer90, Yan93]).

Un des travaux les plus intéressants sur les SOS se trouve dans [GV88] où les auteurs démontrent que la bisimulation classique sur les ST est une congruence pour tous les combinateurs décrits au moyen de règles SOS respectant un certain format syntaxique. Ce résultat établit une nouvelle technique de preuve: pour prouver la propriété de congruence par rapport à un combinateur, il suffit de montrer qu'il peut être décrit au moyen de règles SOS respectant ce format.

1.2 Notre contribution.

Dans cette thèse, nous proposons:

• Une nouvelle définition de *morphisme* entre STA, les structurant en une catégorie. Ces morphismes seront des outils de base dans une partie de notre travail[10].
Ceci fait l'objet de la Section 2.4 du Chapitre 2.

• Une définition du *raffinement d'actions* dans les STA muni d'une caractérisation comme foncteur dans la catégorie.
Nos résultats sur le raffinement, présentées dans le Chapitre 4 généralisent par exemple, ceux de van Glabbeek et Weijland [GW89] sur les Systèmes de Transition et ceux de van Glabbeek et Goltz [GG89c] sur les Structures d'Evénements.

• Plusieurs *équivalences* entre STA. On s'intéresse notamment aux bisimulations compatibles avec le raffinement d'actions.
Nous adaptons aux STA l'équivalence d'ordre mixte (*mo-équivalence*) de Degano, De Nicola et Montanari [DDM89] qui outre la structure arborescente, préserve l'ordre total dans le calcul et l'ordre partiel reflétant la causalité. Nous montrons que la mo-équivalence est compatible avec le raffinement.
Puisque l'ordre partiel est, dans les STA, une notion dérivée de celle d'indépendance, nous nous intéressons aux bisimulations qui préservent l'indépendance, plutôt que l'ordre partiel. Un premier pas dans cette direction (à notre connaissance le seul dans la littérature) a été fait par Mukund et Nielsen [MN92] en proposant une bisimulation (appelée ici *cI-bisimulation*) qui préserve l'indépendance le long des calculs. Nous montrons que la cI-bisimulation est compatible avec le raffinement d'actions.
Nous cherchons des bisimulations définies directement sur l'automate, et non sur ses calculs. Dans cette direction, nous proposons d'abord la *gI-bisimulation*, qui préserve globalement l'indépendance. Ce premier essai n'est pas satisfaisant, puisque la gI-bisimulation n'est pas

[10] même si on ne suit pas la position extrême présentée par Goguen dans [Gog91] sur l'utilisation de la Thèorie des Catégories.

compatible avec le raffinement.

Nous introduisons ensuite la *dI-bisimulation*, définie à partir de nos morphismes. La dI-bisimulation est définie sur l'automate et préserve l'indépendance. En plus, nous montrons que la dI-bisimulation est compatible avec le raffinement, avec une preuve basée sur la caractérisation catégorique du raffinement.

Nous montrons que ces quatre équivalences sont différentes dans les STA. Dans la famille des STA qui représentent les Structures d'Événements, la mo-équivalence et la cI-bisimulation coïncident, et dans celle qui représente les ST, les quatre coïncident avec la bisimulation classique.

Le Chapitre 5 est consacré à l'étude des bisimulations sur les STA.

• L'*adaptation* de la technique de Badouel et Darondeau [BD92, BD93] pour obtenir une variante de la *Sémantique Opérationnelle Structurée* de Plotkin [Plo81] qui nous permet d'associer à un langage une sémantique formelle sur les STA. Dans le Chapitre 6 nous nous intéressons à cette variante, et on montre aussi le résultat suivant.

• Une *condition suffisante* sur les spécifications opérationnelles pour que la *mo-équivalence*, la *cI-bisimulation* et la *gI-bisimulation* soient *compatibles avec les constructeurs du langage.* Ce travail s'inscrit dans la lignée des travaux de De Simone [Sim85], Bloom, Istrail et Meyer [BIM88], et Groote et Vaandrager [GV89] sur les formats de règles SOS.

Une retombée de ce résultat est la preuve que *le raffinement d'actions ne peut être représenté* par des spécifications satisfaisant cette condition.

Afin de mieux situer nos résultats par rapports aux autres approches de la littérature, nous rappelons dans le Chapitre 3 les liens entre les STA et d'autres modèles de systèmes réactifs. Ces résultats sont extraits du travail de Winskel et Nielsen [WN92].

Le Chapitre Chapitre 7 conclut ce travail en ajoutant quelque remarques et suggestions pour les travaux futurs.

Chapitre 2

Systèmes de Transitions Asynchrones

Dans ce chapitre, nous donnons une présentation du modèle auquel nous nous intéressons principalement dans cette thèse: les *Systèmes de Transitions Asynchrones*, notés plus concisement STA, dans la suite.

Les STA, introduits de façon indépendante par Bednarczyk [Bed87] et Shields [Shi85a, Shi85b] généralisent très simplement les *Systèmes de Transitions (ST)* [Kel76].

Les ST sont les modèles de systèmes réactifs les mieux connus. Ce sont des graphes orientés et étiquetés. Les sommets du graphe représentent les *états* du système (parmi lesquels on distingue un état initial) et les flèches représentent les *transitions* entre les états, munies d'une étiquette prise dans un *alphabet d'actions*. Quand une transition a lieu, l'action exécutée peut éventuellement être observée de l'extérieur. La notion de base du comportement d'un ST est la transition. Une exécution dans un ST est alors modélisée par un chemin dans le graphe, dont l'observation extérieure est une séquence d'actions. On va s'intéresser ici aux ST dits *extensionnels*, c'est à dire pour lesquels deux transitions différentes, partant du même état et arrivant au même état, doivent avoir des étiquettes différentes. Une façon naturelle d'abstraire le codage d'objets mathématiques est de définir l'isomorphisme entre objets. L'isomorphisme préserve la structure essentielle de l'objet et ignore les détails syntaxiques. Une condition première de cohérence est que deux objets isomorphes soient indistinguables au sein de la théorie.

Un isomorphisme de ST est un couple de bijections, entre les états, et entre les transitions, qui préservent l'état initial, la relation d'incidence et l'étiquetage. On peut alors représenter graphiquement un ST (de même qu'un STA) par un graphe où les états sont des petits cercles et les transitions reliant deux états sont des flèches étiquetées par les actions. On indiquera l'état initial par une petite flèche incidente.

Les STA généralisent les ST; les étiquettes des transitions sont des éléments d'un ensemble d'*événements*, muni d'une relation d'*indépendance*. Intuitivement, deux événements indépendants ne partagent les mêmes ressources, autrement dit, ils peuvent s'exécuter en parallèle. Une relation d'indépendance est une relation binaire I, symétrique et irréflexive, telle que tout couple d'événements $(a, b) \in I$, la relation de transition satisfait les deux axiomes suivantes:

- *Stabilité en avant*: Si un état s est l'origine de deux transitions étiquetées par a et b, alors il est à l'origine de deux séquences de transitions étiquetées par $\langle a, b\rangle$ et $\langle b, a\rangle$ aboutissant à un même état u.
- *Commutativité*: Si un état s est à l'origine d'une séquence de transitions étiquetés par $\langle a, b\rangle$, alors il est à l'origine aussi d'une séquence étiquetée par $\langle b, a\rangle$, et les deux séquences aboutissent à un même état u.

En somme, quand deux transitions étiquetées par des événements indépendants partent du même état ou sont en séquence, elles forment un *diamant*, comme le représente la Figure 2.1.

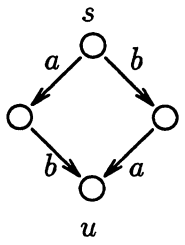


Figure 2.1: Un premier diamant (aIb)

La relation de transition satisfait aussi l'axiome de *déterminisme*: tout état est l'origine d'au plus une transition étiquetée par un événement donné. Le déterminisme nous dit que les événements sont des objets de bas niveau d'abstraction. On étiquette les événements par des actions visibles de l'extérieur et on obtient ainsi un niveau d'abstraction plus élevé.

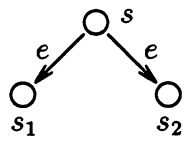
Dans les STA, les chemins partant de l'état initial représentent des exécutions. Comme le STA est déterministe par rapport aux événements, cette notion d'exécution, que l'on appelle *calcul*, est univoquement définie par une séquence d'événements. On peut aussi définir une notion plus abstraite d'exécution, puisque d'après la commutativité, toute permutation d'événements indépendants dans un calcul définit encore un calcul[1]. Pour cette deuxième notion d'exécution, on identifie tous les calculs qui diffèrent par permutation d'événements indépendants. Cette construction est classiquement appelée *Trace de Mazurkiewicz* [Maz77, Maz89]. Cependant, plutôt que d'utiliser le terme "trace", nous préférons le terme *configuration* qui appartient à la littérature des Structures d'Événements [NPW81, Win89b]. Ceci pour deux raisons: d'une part nous allons montrer que les STA sont des généralisations des Structures d'Événements, et d'autre part, nous nous intéressons à la structure arborescente de notre modèle, c.-à-d. le moment des choix, qui n'est pas une notion primitive dans les langages de traces.

Ce chapitre est organisé en sept sections. Les deux premières introduisent les STA (2.1) ainsi que les définitions formelles d'exécutions: le calcul et la configuration (2.2). Dans la

[1] On dit que l'ensemble des calculs est fermé par rapport à la permutation d'événements indépendants.

Section 2.3, on se consacre à présenter plus finement l'intuition qu'il y a derrière la notion de diamant. Les morphismes des STA sont l'objet de la Section 2.4. La Section 2.5 présente une première abstraction sur les STA, par dépliage. Puis, quelques combinateurs classiques sont définis sur les STA dans la Section 2.6. Nous terminons le chapitre par quelques considérations bibliographiques dans la Section 2.7.

2.1 Les Systèmes de Transitions Asynchrones



implique $s_1 = s_2$

Figure 2.2: $\rightarrow$ est déterministe

Soit un alphabet d'actions $Act = \{\alpha, \beta, \gamma \ldots\}$.

Définition 2.1 *Un* Système de Transitions Asynchrones (STA) (étiqueté sur *Act*) *est une structure* $G = \langle Q, i, E, I, \rightarrow, l \rangle$, *où*

- $Q = \{i, s, u, s_1, s_2 \ldots\}$ *est un* ensemble d'états, *i est l'*état initial *de G.*
- $E = \{a, b, e \ldots\}$ *est un* ensemble d'événements.
- $I \subseteq E \times E$ *est une relation irréflexive et symétrique, appelée* relation d'indépendance.
- $\rightarrow \subseteq S \times E \times S$, *est la* relation de transition *(nous noterons* $s \xrightarrow{e} s_1$ *pour* $(s, e, s_1) \in \rightarrow$*) telle que:*
  - Déterminisme. *Si* $s \xrightarrow{e} s_1$ *et* $s \xrightarrow{e} s_2$ *alors* $s_1 = s_2$ *(Figure 2.2).*
  - Stabilité en avant. *Si* aIb, $s \xrightarrow{a} s_1$ *et* $s \xrightarrow{b} s_2$ *alors* $\exists u, s_1 \xrightarrow{b} u \wedge s_2 \xrightarrow{a} u$ *(Figure 2.3).*
  - Commutativité. *Si* aIb *et* $s \xrightarrow{a} s_1 \xrightarrow{b} u$ *alors* $\exists s_2, s \xrightarrow{b} s_2 \xrightarrow{a} u$ *(Figure 2.4).*
- $l : E \mapsto Act$ *est la* fonction d'étiquetage [2]

On note $\boldsymbol{A} = \{G, G_1, G_2 \ldots\}$ *la classe des STA.*

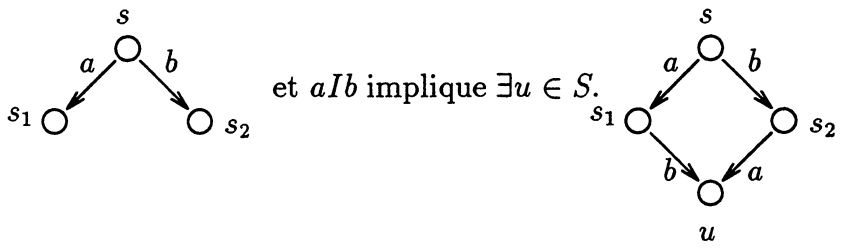


Figure 2.3: $\rightarrow$ est stable en avant

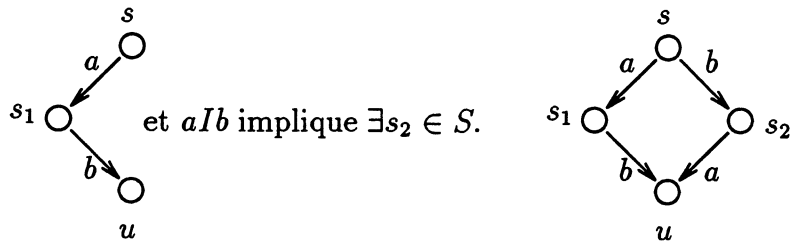


Figure 2.4: $\rightarrow$ est commutative

Nous appelons *e-transition* un élément $s \xrightarrow{e} u$. Nous notons $s \rightarrow$ lorsque $s \xrightarrow{e} u$ pour une certaine action e et un état u. Un état s n'ayant pas de successeur ($s \not\rightarrow$) est appelé *état terminal.*

La Figure 2.5 contient un exemple d'un STA G, sur l'alphabet $\{\alpha, \beta, \gamma\}$ avec 6 états, $E = \{a, b, c, d\}$ et $I = \{(a, b), (b, a)\}$ (noté plus concisément aIb). Nous représentons graphiquement l'état initial par une flèche incidente. Parfois nous écrirons les noms des événements ainsi que leur étiquette ($a : \alpha$, où $a \in E$ et $\alpha = l(a)$).

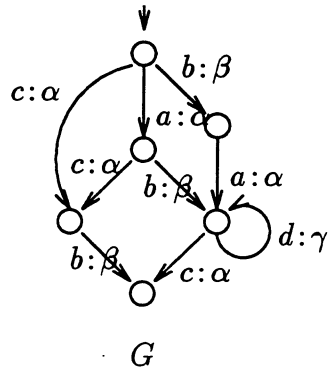


Figure 2.5: Un exemple de STA (aIb)

2.2 Les exécutions: calculs et configurations

Nous introduisons à présent deux notions d'*exécution* dans un STA: les chemins partants de l'état initial, que l'on appelle *calculs*, et une abstraction sur les calculs, les *configurations*, où l'on identifie les calculs qui diffèrent par permutation d'événements indépendants.

Définition 2.2 *Soit $s \in Q$. Un* chemin partant de s *dans G est une séquence finie d'événements $\pi = \langle e_1, e_2, \ldots, e_n \rangle$ telle que il existe $s_1, s_2, \ldots, s_n \in S$ et $s \xrightarrow{e_1} s_1 \xrightarrow{e_2} s_2 \ldots \xrightarrow{e_n} s_n$.*

D'après la condition du déterminisme de la Définition 2.1, les états s_j ($1 \leq j \leq n$) sont complètement déterminés par s et π. On appelle l'état s_n la cible *de π (noté cible(π)). Un chemin π est* complet *ssi cible(π) $\not\rightarrow$.*

Soit $\pi = \langle e_1, e_2, \ldots, e_n \rangle$ un chemin, on note $\pi(j)$ l'événement e_j, et $|\pi| = n$ la taille de π.

[2] *On note les fonctions avec $\mapsto$, et les transitions avec $\rightarrow$.*

Définition 2.3 *Un* calcul *dans G est un chemin partant de i, l'état initial de G. On note $Calc(G) = \{\sigma, \rho, \rho', \ldots\}$ l'ensemble des calculs de G, et $CalcC(G)$ l'ensemble de calculs complets ($CalcC(G) \subseteq Calc(G)$).*

La séquence d'événements vide λ, $\langle a\rangle$, $\langle a,b\rangle$, $\langle a,c\rangle$, $\langle a,b,d,d\rangle$, $\langle c,b\rangle$, $\langle a,b,c\rangle$, et $\langle b,a,c\rangle$ sont de calculs du STA G de la Figure 2.5. Le trois derniers sont des calculs complets.

Pour notre deuxième notion d'exécution, les *configurations*, on identifie tous les calculs qui ne diffèrent que dans l'ordre des événements indépendants. Le terme configuration est tiré de la littérature sur les Structures d'Evénements (SE). Dans une SE, chaque événement ne peut apparaître qu'une fois dans une exécution. Les configurations des SE sont donc des ensembles.

Dans le cadre plus riche des STA, un même événement peut apparaître plusieurs fois dans une exécution. Pour représenter cette configuration, on utilisera des multi-ensembles, munis d'un ordre partiel ("partially ordered multiset", ou "pomset" après [Pra86]).

Soit A un alphabet. Soient $\tilde{X} = \langle X, \leq_X, L_X : X \mapsto A\rangle$ et $\tilde{Y} = \langle Y, \leq_Y, L_Y : Y \mapsto A\rangle$ deux ordres partiels étiquetés sur A. Un *isomorphisme* de $\tilde{X}$ vers $\tilde{Y}$ est une bijection $f : X \mapsto Y$ qui préserve et reflète la relation d'ordre et la fonction d'étiquetage. Un *pomset sur A* est une classe d'isomorphismes d'ordres partiels étiquetés sur A.

Pour simplifier la lecture, nous écrirons $\tilde{X}$ (ou encore X lorsqu'il n'y a pas d'ambiguïté) pour la classe d'isomorphisme de $\tilde{X} = \langle X, \leq_X, L_X\rangle$.

Soit le pomset $\tilde{X} = \langle X, \leq_X, L_X\rangle$, $x, y \in X$. Nous dirons que x et y sont *concurrents dans* $\tilde{X}$ (noté $x\, co_{\tilde{X}}\, y$) ssi $x \not\leq_X y$ et $y \not\leq_X x$.

Soient les pomsets $\tilde{X} = \langle X, \leq_X, L_X : X \mapsto A\rangle$ et $\tilde{Y} = \langle Y, \leq_Y, L_Y : Y \mapsto A\rangle$, nous disons que $\tilde{X}$ est un *préfixe* de $\tilde{Y}$ (noté $\tilde{X} \subseteq \tilde{Y}$) ssi il existe un sous-ensemble $Y' \subseteq Y$, fermé a gauche par $\leq_Y$ (c-à-d. pour tout $j \leq_Y k$, $k \in Y'$ implique $j \in Y'$), tel que $\tilde{X}$ et $\langle Y', \leq_Y |_{Y' \times Y'}, L_Y|_{Y'}\rangle$ sont isomorphes.

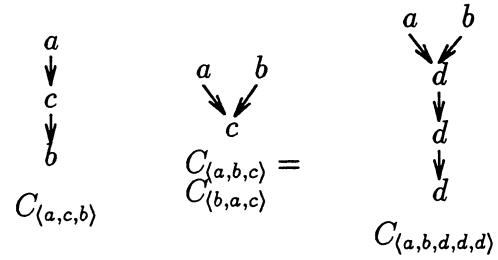


Figure 2.6: Quelques configurations de G. (aIb)

En utilisant l'ordre linéaire sur les occurrences d'événements dans les calculs et la relation d'indépendance entre les événements, on peut aisément définir les configurations comme des pomsets.

Définition 2.4 *Soit $G = \langle Q, i, E, I, \rightarrow, l\rangle \in \mathbf{A}$. A chaque chemin π on associe un pomset étiqueté sur E, noté $C_\pi = \langle F_\pi, \leq_\pi, L_\pi\rangle$, tel que $F_\pi \stackrel{def}{=} \{1 \ldots \mid \pi \mid\}$, $\leq_\pi$ est le plus petit ordre partiel sur F_π qui contient tous les couples (j,k) t.q. $j < k$ et $\neg(\pi(j) I \pi(k))$, et $L_\pi(j) \stackrel{def}{=} \pi(j)$.*

Quand $\sigma \in Calc(G)$, C_σ est appelée configuration *dans G. On note Conf(G) l'ensemble des configurations dans G.*

La Figure 2.6 montre quelques configurations dans le STA G de la Figure 2.5, en utilisant la représentation classique pour les pomsets (où on note $a \leq_C c$ comme $a \rightarrow c$).

Remarque 1 *Soit σ un calcul dans G, $j, k \in F_\sigma$. D'après la Définition 2.4, $j co_{C_\sigma} k$ (noté $j co_\sigma k$ dans la suite) implique $\sigma(j) I \sigma(k)$. L'inverse n'est pas vrai dans le cas général, comme le montre la première configuration de la Figure 2.6. Dans le calcul $\langle a, c, b \rangle$, $\rho(1) I \rho(3)$ mais $\neg(1 co_{\langle a,c,b \rangle} 3)$.*

2.3 Les états, les transitions, les diamants: les ST-états

Dans cette section nous discutons le sens du diamant comme représentation d'une exécution en parallèle d'événements. Nous basons notre présentation sur la définition de *ST-état* (en suivant l'approche de [GV87]) et sur l'intuition de la structure géométrique que cette définition induit (en suivant [Shi85b] et [Pra91]).

La principale question est: Etant donné le STA de la Figure 2.7, où aIb et $\neg(aIc)$, comment différencier le diamant de côtés a, b de celui de côtés a, c?

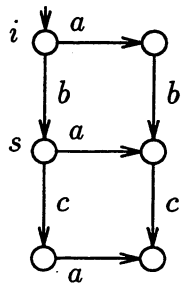


Figure 2.7: Un vrai diamant et un faux diamant $(aIb, \neg(aIc))$.

Imaginons que les événements ne soient pas instantanés. Alors, à partir de l'état initial, l'événement a peut *commencer à s'exécuter*. Est-ce que l'événement b peut commencer à son tour, sans attendre la fin de a? Nous savons que dans l'état initial, b peut s'exécuter. Puisque a est indépendante de b ces deux événements ne partagent pas de ressources et b peut commencer sans attendre la fin de a.

Il est facile de généraliser cette construction à n'importe quel nombre d'événements, à condition qu'ils soient deux à deux indépendants, et que tous se trouvent sur des transitions partant d'un même état. C'est l'intuition qu'il y a derrière les *ST-états*.

Définition 2.5 *Soit $G = \langle Q, i, E, I, \rightarrow, l \rangle$. Un* ST-état *de G est un couple $\langle s, \{e_1, \ldots e_n\} \rangle \in Q \times \mathcal{P}(E)$ tel que pour tous les $e_j, e_k \in \{e_1, \ldots e_n\}$, $s \xrightarrow{e_j}$ et $j \neq k$ implique $e_j I e_k$.*

La cible *d'un ST-état $(s, \{e_1, \ldots e_n\})$ (noté $cible(s, \{e_1, \ldots e_n\})$) est l'etat u tel que $s \xrightarrow{e_1} \cdots \xrightarrow{e_n} u$.*

La taille *d'un ST-état $(s, \{e_1, \ldots e_n\})$ (notée $|(s, \{e_1, \ldots e_n\})|$) est n.*

Soit $Diam(G)$ l'ensemble des ST-états de G.

Les constructions ST[3] ont été introduites par [GV87] sur les Réseaux de Petri [Pet62, Rei85]. Elles sont essentiellement une généralisation du marquage dans le réseau, qui contient des places mais aussi des transitions.

Revenons maintenant à la Figure 2.7: quelle est la différence entre les diamants étiquetés a, b, et a, c? La réponse est: seul le premier représente un ST-état, à savoir $\langle i, \{a, b\}\rangle$. Ce n'est pas le cas pour l'autre diamant, puisque $\neg(aIc)$. Dans la Figure 2.8 chaque ST-état est désigné comme une surface.

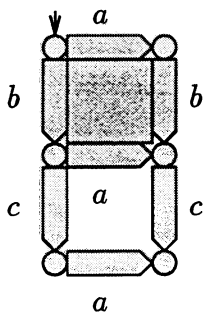


Figure 2.8: Une autre façon de representer le STA de la figure 2.7

Les ST-états sont une représentation plus générale des états (par exemple $i = \langle i, \emptyset\rangle$), des transitions ($i \xrightarrow{b} s = \langle i, \{b\}\rangle$) et des diamants engendrés par des événements indépendants ($\langle i, \{a, b\}\rangle$).

Puisque les transitions sont aussi des ST-états, il est légitime de se demander quelles sont alors les transitions reliant ces ST-états. Cette démarche a été suivie, par exemple, dans [GV87, Vog90a, Vog91, Dev90, Hen91, Gla90c, Gla90b], et donne lieu à des constructions très intéressantes, notamment, la famille des *ST-bisimulations.*

Pratt [Pra91] procède autrement. Il appelle *élément d-dimensionnel* (ou d-cellule) un ST-état $\langle s, \epsilon\rangle$ quand ϵ contient exactement d événements. Nos anciens états, transitions et diamants correspondent respectivement à des 0, 1 et 2-cellules. Les d-cellules sont disposées dans un espace où les relations de contiguïté et d'emboîtement sont définies. Le concept de d-cellule nous donne un niveau d'abstraction très agréable car les états et les transitions sont de même nature.

Le lecteur est invité à explorer la suite de cette construction dans [Pra91].

2.4 Les morphismes des STA

Un morphisme de G_1 vers G_2 est une façon de particulière de replier G_1 sur G_2, définie a partir de deux fonctions: l'une sur les états et l'autre sur les événements.

Définition 2.6 *Soient $G_1, G_2 \in \boldsymbol{A}$. Un* STA-morphisme, *ou plus simplement un* morphisme, *entre G_1 et G_2 est un couple $h = (h_S, h_E)$ tel que*

[3] De l'allemand "Stellen" (place) et "Transitionen" (transition)

- $h_S : S_1 \mapsto S_2$, *préserve l'état initial (i.e.* $h_S(i_1) = i_2$*),*
- $h_E : E_1 \mapsto E_2$, *préserve l'étiquetage (i.e.* $\forall e \in E_1, l_2(h_E(e)) = l_1(e)$*),*
- h, *étendu de façon canonique aux ST-états en* $h_D(s, \{\ldots e_j \ldots\}) = (h_S(s), \{\ldots h_E(e_j)\ldots\})$, *est tel que*
  - h_D : $Diam(G_1) \mapsto Diam(G_2)$, h_D *est surjectif (i.e.* $h_D(Diam(G_1)) = Diam(G_2)$*), préserve les cibles (i.e.* $cible(h_D(s,P)) = h_S(cible(s,P))$*) et la taille (i.e.* $|h_D(s,P)| = |s,P|$*).*
  - h_D *reflète la croissance des ST-états, i.e.* $h_D(s_1,P_1) = (s_2,P_2)$ *et* $(s_2, P_2 \cup \widetilde{P_2}) \in Diam(G_2)$ *implique* $\exists (s_1, P_1 \cup \widetilde{P_1}) \in Diam(G_1)$, $h_D(s_1, P_1 \cup \widetilde{P_1}) = (s_2, P_2 \cup \widetilde{P_2})$.

Un exemple de morphisme des STA ($h : G_1 \mapsto G_2$) est présenté dans la Figure 2.9 (aIb). Quelques couples de h_S sont représentés en pointillé, et $h_E = Id|_{\{a,b,c,d\}} \cup \{(e,c)\}$.

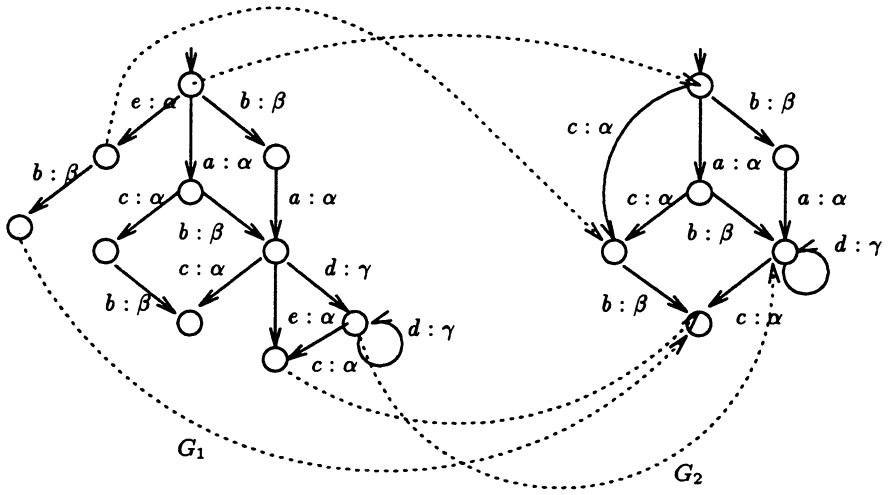


Figure 2.9: Un morphisme $h : G_1 \mapsto G_2$ (aIb)

Lemme 2.7 *La classe* $\mathbf{A}$ *munie des STA-morphismes est une catégorie.*

La preuve est simple.

Lemme 2.8 *Soient* $G_1, G_2, G_0 \in \mathbf{A}$, $h^1 : G_1 \mapsto G_0$ *et* $h^2 : G_2 \mapsto G_0$. *Il existe* $G_3 \in \mathbf{A}$, *et des morphismes* j^1, j^2 *tels que* $j^1 : G_3 \mapsto G_1$ *et* $j^2 : G_3 \mapsto G_2$

Preuve. Le bon candidat est $G_3 \stackrel{\text{def}}{=} \langle S_3, i_3, E_3, I_3, \rightarrow_3, l_3\rangle$ défini par $S_3 \stackrel{\text{def}}{=} \{(s_1,s_2) \in S_1 \times S_2 : h^1_S(s_1) = h^2_S(s_2)\}$, $i_3 \stackrel{\text{def}}{=} (i_1, i_2)$, $E_3 \stackrel{\text{def}}{=} \{(e_1,e_2) \in E_1 \times E_2 : h^1_E(e_1) = h^2_E(e_2)\}$, $(a_1,a_2)I_3(b_1,b_2) \stackrel{\text{def}}{\Leftrightarrow} a_1 I_1 b_1$ *et* $a_2 I_2 b_2$, $(s_1,s_2) \stackrel{(e_1,e_2)}{\rightarrow}_3 (u_1,u_2) \stackrel{\text{def}}{\Leftrightarrow} s_1 \stackrel{e_1}{\rightarrow}_1 u_1$ *et* $s_2 \stackrel{e_2}{\rightarrow}_2 u_2$, et $l_3((e_1,e_2)) \stackrel{\text{def}}{=} l_1(e_1)$, avec $j^i \stackrel{\text{def}}{=} (j^i_S, j^i_E)$, $j^i_S(s_1,s_2) \stackrel{\text{def}}{=} s_i$, et $j^i_E(e_1,e_2) \stackrel{\text{def}}{=} e_i$. □

La réciproque de ce lemme n'est pas vraie dans le cas général. La Figure 2.10 montre un exemple. Il existe $j^1 : G_3 \mapsto G_1$ et $j^2 : G_3 :\mapsto G_2$, mais il n'existe pas $G_0 \in \mathbf{A}$, $h^1 : G_1 \mapsto G_0$ $h^2 : G_2 \mapsto G_0$.

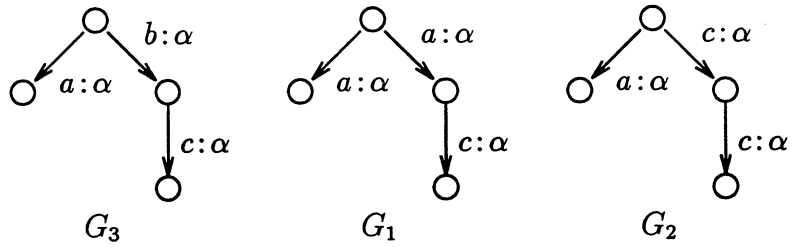


Figure 2.10: La "reduction par les morphismes" n'est pas confluente. ($I = \emptyset$)

2.5 Dépliage d'un STA

Le *dépliage* nous donne une première notion de comportement, et donc un premier niveau d'abstraction sur les STA, en identifiant ceux qui ont le même dépliage.

L'isomorphisme des STA[4] ne nous permet pas d'identifier les trois STA de la Figure 2.11. Pourtant, ces STA se comportent de la même façon: ils sont tous capables d'exécuter la même suite d'événements a, puis choisir d'exécuter c, puis de reprendre l'exécution des événements a. Le dépliage d'un STA permet une première formalisation de l'idée "d'avoir le même comportement".

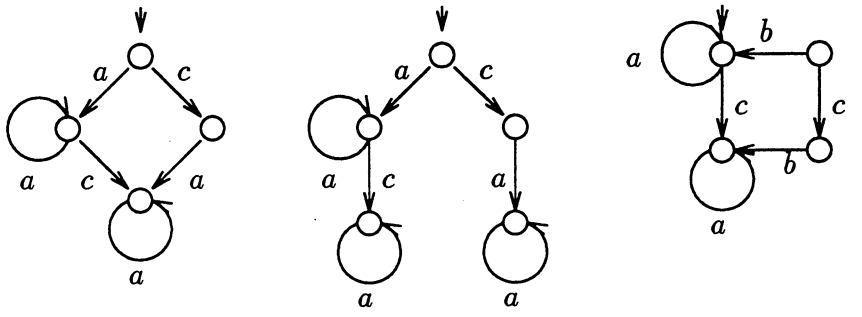


Figure 2.11: Trois STA non isomorphes, mais ayant "le même comportement". ($\neg(aIc), bIc$)

Le dépliage associe à tout STA un autre STA, sans cycles, sans états ni événements ni couples d'événements indépendantes non atteignables, ayant les mêmes exécutions, les mêmes moments de choix.

Définition 2.9 *Soit* $G = \langle Q, i, E, I, \rightarrow, l\rangle \in \boldsymbol{A}$, *le* dépliage *de* G, *noté* $déplie(G)$, *est défini par* $déplie(G) \stackrel{def}{=} \langle Conf(G), \emptyset, E_d, I_d, \rightarrow_d, l_d\rangle$ *où*

- $E_d \stackrel{def}{=} \{e \in E : \exists \rho e \in Calc(G)\}$,
- $aI_db \stackrel{def}{\Leftrightarrow} aIb \wedge \exists \rho ab \in Calc(G)$,
- $x \stackrel{e}{\rightarrow_d} y \stackrel{def}{\Leftrightarrow} x \leq y \wedge y \setminus x = \{e\}$, *et*

[4] construit d'une façon classique a partir des morphismes

- $l_d \stackrel{def}{=} l|_{E_d}$.

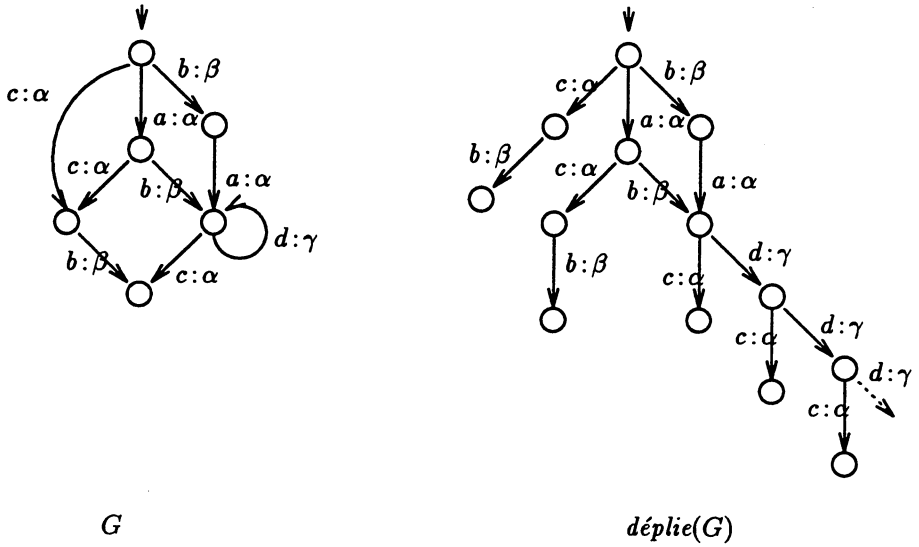


Figure 2.12: G et *déplie*(G). (aIb)

La Figure 2.12 montre un STA G (avec aIb) à gauche et un début de son dépliage à droite. Remarquons que le dépliage préserve les diamants engendrés par les événements indépendants, et déplie les "faux" diamants.

A partir du dépliage, nous définissons une première relation d'équivalence non triviale sur les STA, qui identifie les STA de mêmes dépliages. Soit $G_1, G_2 \in \mathbf{A}$, on pose.

$$G_1 =_{déplie} G_2 \stackrel{\text{def}}{\Leftrightarrow} déplie(G_1) = déplie(G_2)$$

L'équivalence $=_{déplie}$ peut être vue comme une sorte de "borne inférieure" pour toutes les notions de comportement, car elle préserve les exécutions et les moments de choix. C'est à dire que toute propriété préservée par le dépliage est de nature "comportementale". Par contre, les propriétés non préserveés par le dépliage font référence à la "structure interne" du système.

2.6 Quelques combinateurs de base sur les STA

Dans cette section on définit sur les STA quelques combinateurs classiques.

Des ensembles de combinateurs sur les STA assez complets (contenant par exemple ceux de CCS [Mil80, Mil89] et de CSP [Hoa78, BHR84, Hoa85]) sont présentés dans [Bed87, WN92]. Notre objectif ici est d'introduire certains combinateurs sur les STA, que nous utiliserons ensuite pour décrire nos exemples. Dans le chapitre 6 une famille de combinateurs très riche est étudiée.

On introduit le STA *nul* ($\emptyset$), la *composition séquentielle* (;), la *composition parallèle* (sans communication, aussi appelée *produit*, $\otimes$) et la *somme non-déterministe* ($+$). A la fin de la section, nous montrons que les combinateurs sont bien définis sur le comportement au sens

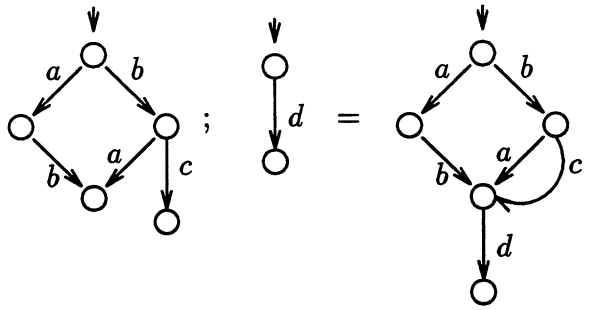


Figure 2.13: $G_1; G_2$

de la Section 2.5: les STA qui ont le même dépliage ont le même comportement vis-à-vis des combinateurs.

Le plus simple des STA est celui qui n'a aucune transition. On le note $\emptyset \stackrel{\text{def}}{=} \langle \{i\}, i, \emptyset, \emptyset, \emptyset, \emptyset \rangle$.

La *composition séquentielle* de G_1 et G_2 s'obtient en identifiant les états s *terminaux* de G_1 (ceux qui satisfont $s \not\rightarrow$), avec l'état initial de G_2. Soient $G_1 = \langle Q_1, i_1, E_1, I_1, \rightarrow_1, l_1 \rangle, G_2 = \langle Q_2, i_2, E_2, I_2, \rightarrow_2, l_2 \rangle \in \mathbf{A}$, $Q_1 \cap Q_2 = \emptyset$, $E_1 \cap E_2 = \emptyset$, sa composition séquentielle est le STA $G_1; G_2$ défini par

$$\begin{aligned}
G_1; G_2 &\stackrel{\text{def}}{=} \langle Q_3, i_3, E_1 \cup E_2, I_1 \cup I_2, \rightarrow_3, l_1 \cup l_2 \rangle \\
Q_3 &\stackrel{\text{def}}{=} (Q_1 \setminus \{s : s \in Q_1 \wedge s \not\rightarrow_1\}) \cup Q_2 \\
i_3 &\stackrel{\text{def}}{=} \text{ si } i_1 \rightarrow_1 \text{ alors } i_1 \text{ sinon } i_2 \\
\rightarrow_3 &\stackrel{\text{def}}{=} (\rightarrow_1 \setminus \{s \stackrel{e}{\rightarrow}_1 u : u \not\rightarrow_1\}) \cup \rightarrow_2 \cup \{s \stackrel{e}{\rightarrow}_3 i_2 : s \stackrel{e}{\rightarrow}_1 u \not\rightarrow_1\}
\end{aligned}$$

Un exemple de composition séquentielle est donné Figure 2.13.

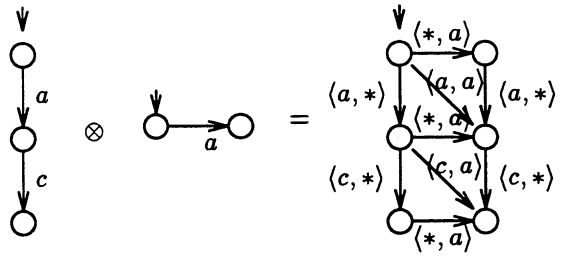


Figure 2.14: $G_1 \otimes G_2$

Soient $G_1 = \langle Q_1, i_1, E_1, I_1, \rightarrow_1, l_1 \rangle, G_2 = \langle Q_2, i_2, E_2, I_2, \rightarrow_2, l_2 \rangle \in \mathbf{A}$, et $* \notin E_1 \cup E_2 \cup Act$. La *composition parallèle* de G_1 et G_2 est le STA $G_1 \otimes G_2$ étiqueté sur $(Act \times \{*\}) \cup (\{*\} \times Act) \cup (Act \times Act)$ et défini par

$$G_1 \otimes G_2 \stackrel{\text{def}}{=} \langle Q_1 \times Q_2, (i_1, i_2), E_3, I_3, \rightarrow_3, l_3 \rangle$$

$$E_3 \stackrel{\text{def}}{=} (E_1 \times \{*\}) \cup (\{*\} \times E_2) \cup (E_1 \times E_2)$$

$$(a_1, a_2) I_3 (b_1, b_2) \stackrel{\text{def}}{\Leftrightarrow} (a_1 = * \vee b_1 = * \vee a_1 I_1 b_1) \wedge (a_2 = * \vee b_2 = * \vee a_2 I_2 b_2)$$

$$l(a_1, a_2) \stackrel{\text{def}}{=} \begin{cases} (l_1(a_1), *) & \text{si } a_2 = * \\ (*, l_2(a_2)) & \text{si } a_1 = * \\ (l_1(a_1), l_2(a_2)) & \text{sinon} \end{cases}$$

Un exemple de composition parallèle est donné Figure 2.14.

Soient $G_1 = \langle Q_1, i_1, E_1, I_1, \rightarrow_1, l_1 \rangle, G_2 = \langle Q_2, i_2, E_2, I_2, \rightarrow_2, l_2 \rangle \in \mathbf{A}$ tel que $Q_1 \cap Q_2 = \emptyset$, $E_1 \cap E_2 = \emptyset$, et soit $i \notin Q_1 \cup Q_2$. La *somme non déterministe* de G_1 et G_2 est le STA $G_1 + G_2$ défini par

$$G_1 + G_2 \stackrel{\text{def}}{=} \langle Q_1 \cup Q_2 \cup \{i\}, i, E_1 \cup E_2, I_1 \cup I_2, \rightarrow_3, l_1 \cup l_2 \rangle$$

$$\rightarrow_3 \stackrel{\text{def}}{=} \rightarrow_1 \cup \rightarrow_2 \cup \{(i, e, u) : i_k \stackrel{e}{\rightarrow}_k u, k = 1, 2\}$$

Un exemple de somme non-déterministe est donné Figure 2.15.

Le lemme suivant exprime le fait que les combinateurs sont définis sur le comportement des STA.

Lemme 2.10 $=_{déplie}$ *est une congruence pour les combinateurs* $;, \otimes$ *et* $+$.

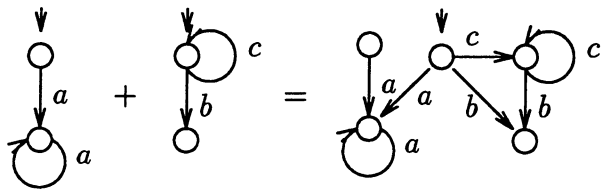


Figure 2.15: $G_1 + G_2$

2.7 Sur le choix de la définition

Notre définition de STA est quasiment celle de [WN92], excepté que nous n'exigeons pas que tout événement apparaisse dans au moins une transition. Nous estimons que cette propriété n'est pas significative dans le cadre de notre travail.

Nous exigeons la stabilité en avant. Notons que celle-ci nous facilite l'étude des liens qu'il y a entre les STA et les autres modèles des systèmes réactifs. Cette étude fait l'objet du chapitre suivant.

Les STA qui ne satisfont pas la commutativité sont connus sous le nom d'*Automates de Trace* [Sta89b, Sta89a], ceux qui ne satisfont pas la stabilité en avant ont fait l'objet des travaux de Shields [Shi85b, Shi85a] sous le nom de *"Asynchronous State Machines"* (on les dénommera *STA non-stables*). Dans [Bed87], les deux versions, avec ou sans cette propriété, sont considérées.

Un STA qui ne satisfait ni la commutativité, ni la stabilité en avant, est simplement un ST non-extensionnel.

Chapitre 3

Rappels sur les liens entre les Systèmes de Transitions Asynchrones et d'autres modèles des systèmes réactifs

Dans ce chapitre, nous étudions les liens entre les STA et les autres modèles des systèmes réactifs.

Il existe aujourd'hui une multitude de modèles de systèmes réactifs dans la littérature. Devant une telle situation, exhiber la relation entre le modèle sur lequel on travaille et certains modèles mieux connus est une tâche importante. C'est l'objectif principal de ce chapitre, qui est largement extrait de [WN92]. Notons que ceci nous permet entre autre de mieux saisir les concepts de base de notre modèle, à savoir: état, événement, indépendance.

Les liens entre les STA et d'autres modèles ont déjà été étudiés par Shields [Shi85b, Shi85a] qui établit des relations entre les STA (dans leur version non-stable en avant), les *Langages de Traces de Mazurkiewicz* [Maz77, Maz89] et les *Structures d'Evénements Premières (SE)* [NPW81, Win89a]. Dans sa thèse, Bednarczyk [Bed87] présente une autre façon de relier les STA avec les Langages de Traces de Mazurkiewicz, les SE et les *Réseaux de Petri (RP)* [Pet62, Rei85]. Il utilise des techniques de la théorie des catégories (une introduction à cette théorie peut se trouver dans [AM75, Pie91]). En utilisant aussi la théorie des catégories, Winskel et Nielsen [WN92] situent les STA dans un cadre où tous les modèles qu'on vient de citer sont représentés, ainsi que les *Langages de Traces de Hoare* [Hoa85], les *Arbres de Synchronisation* [Mil80], les *Systèmes de Transition* [Kel76] et les *Systèmes de Transition avec relation d'Indépendance (STI)*.

L'utilisation de la théorie des catégories permet de donner des définitions propres des opérations de composition parallèle et de somme non-déterministe à tous ces modèles, comme produit et somme dans leurs catégories respectives. Elle permet aussi, en utilisant les notions de réflex-

ions et co-réflexions, de donner des traductions entre ces formalismes, de façon compositionnelle par rapport à ces combinateurs. Suivant la même technique, d'ailleurs très agréable, [SNW93] établissent les relations entre plusieurs modèles et les STI.

Dans ce chapitre, nous rappelons (sans utiliser de constructions catégoriques) quelques traductions de [WN92]. Mais on ne s'intéresse pas ici aux résultats de compositionalité: on rappellera les résultats selon lesquels ces traductions préservent le comportement, voire même la structure du système, dans le cas où ceci a un sens.

Nous montrerons les relations entre les STA et les RP, les SE, les ST et les STI. Il est facile de justifier notre choix: les SE, les RP et les ST sont parmi les modèles les plus connus. Notre intérêt pour les STI est fondé sur leur étroite ressemblance avec les STA.

Dans les trois premières sections de ce chapitre, nous présentons les relations entre les STA et les autres modèles: dans la Section 3.1 nous relions les STA et les RP; dans la Section 3.2 les STA et les SE et dans la Section 3.3 les STA, les ST et les STI. Nous terminons ce chapitre par quelques remarques dans la Section 3.4.

3.1 STA et RP

Les Réseaux de Petri sont les modèles les plus anciens et les plus connus de la sémantique des systèmes réactifs. Il en existe plusieurs variantes, nous considérons ici celle étudiée dans [WN92], qui a des liens clairs avec les STA.

Un RP peut être considéré comme un système de transition où les transitions ne dépendent pas d'un état global. Une occurrence d'un événement affecte seulement les conditions dans son voisinage. L'indépendance des événements devient donc une notion dérivée: deux événements sont indépendants quand leurs voisinages sont disjoints.

Définition 3.1 *Un* Réseaux de Petri *(étiqueté sur Act) est une structure* $\mathcal{R} = \langle B, M_0, E, {}^\bullet(\_), (\_)^\bullet, l\rangle$ *où:*

- B *est un ensemble de* conditions, *avec un* marquage initial $M_0 \subseteq B$, *non vide,*
- E *est un ensemble d'événements,*
- ${}^\bullet(\_) : E \mapsto \mathcal{P}(B)$ *est la fonction de* pré-condition *qui satisfait* $\forall e \in E.\ {}^\bullet e \neq \emptyset$,
- $(\_)^\bullet : E \mapsto \mathcal{P}(B)$ *est la fonction de* post-condition *qui satisfait* $\forall e \in E.\ e^\bullet \neq \emptyset$,
- $l : E \mapsto Act$ *est la fonction d'étiquetage.*

Nous notons $\boldsymbol{R}$ *la classe des RP étiquetés sur Act.*

Le marquage initial consiste en un un sous-ensemble de conditions qui sont satisfaites au début de chaque exécution. Un *marquage* est un sous-ensemble de conditions, qui formalisent notre notion d'état comme l'ensemble des condition qui sont satisfaites à un instant donné. Le marquage change avec l'occurrence des événements. Les *transitions* $M \xrightarrow{e} M'$ expriment un changement d'état lorsque l'événement e a lieu.

Définition 3.2 *Soit* $\mathcal{R} = \langle B, M_0, E, {}^\bullet(\_), (\_)^\bullet, l\rangle \in \boldsymbol{R}$, $M, M' \subseteq B$ *et* $e \in E$. *Une* transition *dans* $\mathcal{R}$ *est définie par* $M \xrightarrow{e} M' \overset{def}{\Leftrightarrow} {}^\bullet e \subseteq M$, $e^\bullet \subseteq M'$ *et* $M \setminus {}^\bullet e = M' \setminus e^\bullet$.

Les événements a, b *sont* indépendants *(noté* aIb*) ssi* ${}^\bullet a^\bullet \cap {}^\bullet b^\bullet = \emptyset$ *(où* ${}^\bullet e^\bullet \overset{def}{=} {}^\bullet e \cup e^\bullet$*).*

De là, la construction d'un STA à partir d'un RP est très simple:

Définition 3.3 *On définit* $ra : \boldsymbol{R} \mapsto \boldsymbol{A}$ *comme la fonction qui associe à chaque réseau* $\mathcal{R} = \langle B, M_0, E, {}^\bullet(\_), (\_)^\bullet, l\rangle \in \boldsymbol{R}$ *le STA* $ra(N) = \langle \mathcal{P}(B), M_0, E, I, \rightarrow, l\rangle$

La construction dans l'autre sens n'est pas aussi directe. On doit préserver les événements, et "décentraliser" les états, de telle façon que la relation d'indépendance reste codée dans le voisinage. La technique utilisée a été introduite dans [ER90], et utilisée dans l'étude des liens entre les RP élémentaires et les ST élémentaires [ER90, NRT92], une autre variante des RP et les STA [WN92], et les Réseaux de Trace et les Automates de Trace [BD92, BD93].

On définit les *conditions* dans un STA comme un couple d'ensembles d'états et de transitions, qui satisfont certaines propriétés. Cette définition est équivalent a celle de [WN92], et aussi a celle de *région* de [MN92].

Définition 3.4 *Soit* $G = \langle Q, i, E, I, \rightarrow, l\rangle \in \boldsymbol{A}$. *Une* condition *dans* G *est un couple* $b \stackrel{def}{=} \langle b_Q, b_\rightarrow\rangle \in \mathcal{P}(Q) \times \mathcal{P}(\rightarrow)$ *tel que* $b_Q \neq \emptyset$ *et*

$$
\begin{aligned}
s \xrightarrow{e} u \in b_\rightarrow \quad &\Rightarrow \quad s \in b_Q \wedge u \in b_Q \\
s \xrightarrow{e} u \notin b_\rightarrow \wedge s \in b_Q \wedge s_1 \xrightarrow{e} u_1 \quad &\Rightarrow \quad s_1 \xrightarrow{e} u_1 \notin b_\rightarrow \wedge s_1 \in b_Q \\
s \xrightarrow{e} u \notin b_\rightarrow \wedge u \in b_Q \wedge s_1 \xrightarrow{e} u_1 \quad &\Rightarrow \quad s_1 \xrightarrow{e} u_1 \notin b_\rightarrow \wedge u_1 \in b_Q \\
s \xrightarrow{a} u, s_1 \xrightarrow{b} u_1 \notin b_\rightarrow \wedge (s \in b_Q \vee s \in b_Q) \wedge (s_1 \in b_Q \vee u_1 \in b_Q) \quad &\Rightarrow \quad \neg(aIb)
\end{aligned}
$$

Soit B *l'ensemble des conditions de* G. *Pour chaque* $e \in E$ *on définit* $e^\bullet \stackrel{def}{=} \{b \in B : \exists s \xrightarrow{e} u \notin b_\rightarrow, u \in b_Q\}$, ${}^\bullet e \stackrel{def}{=} \{b \in B : \exists s \xrightarrow{e} u \notin b_\rightarrow, s \in b_Q\}$, *et pour chaque* $s \in Q$, $M(s) \stackrel{def}{=} \{b \in B : s \in b_Q\}$

On peut imaginer que chaque condition d'un STA consiste à peindre les états et les transitions de deux couleurs, mettons noir pour les objets qui sont dans la condition et blanc pour les autres. La première propriété exprime le fait que toute flèche noire se trouve entre états noirs. La suivante exprime que si une flèche blanche part d'un état noir, toutes les flèches étiquetées par le même événement agissent de la même façon. La troisième propriété dit la même chose pour les flèches blanches qui arrivent à un état noir. Enfin, la quatrième exprime qu'il ne peut y avoir de diamant avec toutes les flèches blanches et tous les états noirs.

Définition 3.5 *Soit* $G = \langle Q, i, E, I, \rightarrow, l\rangle \in \boldsymbol{A}$. *On définit* $ar : \boldsymbol{A} \mapsto \boldsymbol{R}$ *comme* $ar(G) \stackrel{def}{=} \langle B, M(i), E, {}^\bullet(\_), (\_)^\bullet, l\rangle$, *où* B *est l'ensemble des conditions de* G *et* ${}^\bullet(\_), (\_)^\bullet : E \mapsto B$ *sont définies selon 3.4*

La figure 3.1 montre un exemple de traduction par *ar*. Les conditions de G sont représentées par les petits carrés pointillés, où seulement les objets appartenant à la condition sont présents. La représentation du réseau est classique. Les cercles représentent les conditions, les rectangles les événements, et les relations de pré- et post-conditions sont données par les flèches. Les jetons dans les places représentent le marquage initial.

Le lemme suivant établit que *ar* préserve l'indépendance des événements et les transitions.

Lemme 3.6 *Soit le STA* $G = \langle Q, i, E, I, \rightarrow, l\rangle$. *Si* aIb *alors* ${}^\bullet a^\bullet \cap {}^\bullet b^\bullet = \emptyset$ *dans* $ar(G)$. *Si* $s \xrightarrow{e} u$ *alors* $M(s) \xrightarrow{e} M(u)$ *dans* $ar(G)$.

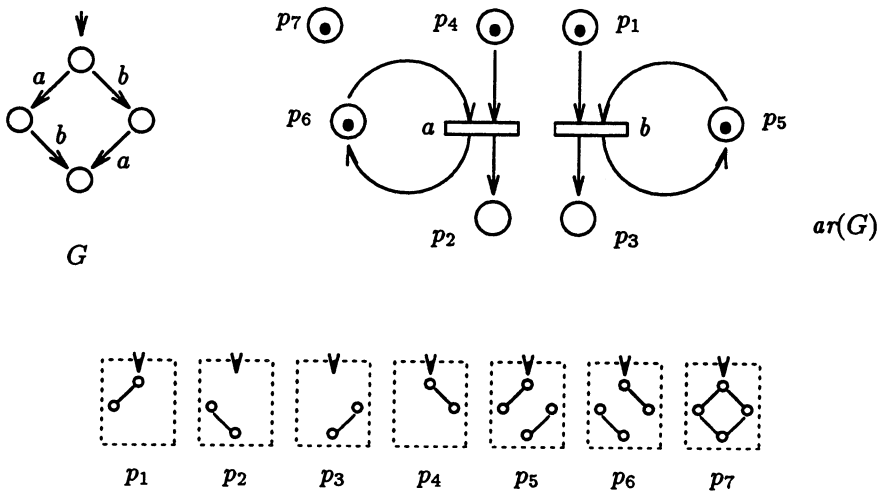


Figure 3.1: Exemple de G et $ar(G)$. (aIb)

Malheureusement, la réciproque de ce lemme n'est pas vraie dans le cas général: le réseaux peuvent avoir "plus de transitions" que les STA de départ. Par exemple, dans la figure 3.2, $ar(G)$ peut effectuer une première transition étiquetée par b, qui ne correspond à aucune transition de G. L'*élémentarité* des STA est une condition suffisante pour éviter cette situation. Un STA est dit élémentaire s'il satisfait certaines conditions d'atteignabilité, et si pour chaque couple d'états différents, il existe une condition qui contient exactement un des deux états. Il en est de même pour les transitions.

Définition 3.7 *Soit* $G = \langle Q, i, E, I, \rightarrow, l \rangle \in \boldsymbol{A}$. G *est* élémentaire *ssi:*

- $\forall e \in E.\ \exists s, u \in Q.\ s \xrightarrow{e} u$
- $\forall s \in Q.\ \exists \rho \in Calc(G).\ s = cible(\rho)$
- $\forall s, u \in Q, s \neq u$ *il existe une condition* b *telle que* $s \in b$ *et* $u \notin b$.
- $\forall s \in Q, \forall e \in E.\ s \not\xrightarrow{e}$ *il existe une condition* b *telle que* $b \in {}^\bullet e$ *et* $s \notin b$.

Pour la famille des STA élémentaires, ar ne rajoute pas de nouvelles transitions.

Lemme 3.8 *Soit* G *un STA élémentaire*

- aIb *dans* G *ssi* ${}^\bullet a^\bullet \cap {}^\bullet b^\bullet = \emptyset$ *dans* $ar(G)$.
- $M(s) \xrightarrow{e} M(u)$ *dans* $ar(G)$ *ssi* $s \xrightarrow{e} u$ *dans* G.
- *Si* $M(i) \rightarrow \ldots \rightarrow M$ *alors* $\exists s \in Q.\ M = M(s)$.

3.2 STA et SE

L'idée de base dans les structures d'événements est de représenter les systèmes par un ensemble d'événements et la description explicite des *dépendances causales* et des *conflits* entre ces événements. Le comportement du système est alors induit par ces dépendances causales, représentées par un ordre partiel entre les événements, et le parallélisme entre événements est modélisé par

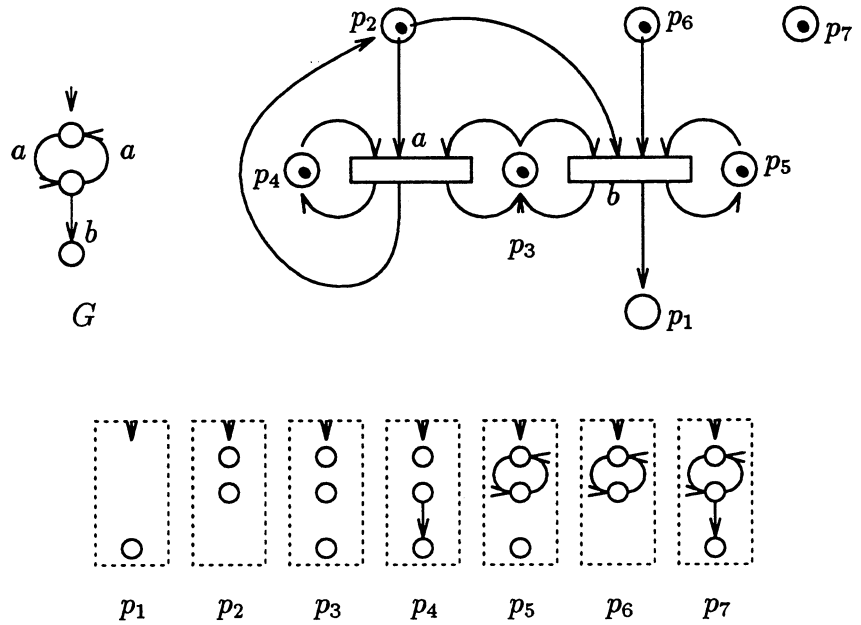


Figure 3.2: $ar(G)$ a trop de transitions quand G n'est pas élémentaire

l'absence d'ordre. De plus, on sait modéliser le non-déterminisme par une relation entre les événements, appelée relation de *conflit*: deux événements en conflit ne peuvent apparaître dans une même exécution. L'état du système est ici une notion dérivée, qui correspond à l'ensemble des événements qui ont eu lieu.

Définition 3.9 *Une* structure d'événements première étiquetée (SE) *est une structure* $\mathcal{F} = \langle F, \leq, \#, l \rangle$ *composée*

- *d'un ensemble* F *d'*événements,
- *d'un ordre partiel* $\leq \subseteq F \times F$ *appelé relation de* causalité,
- *d'une relation binaire symétrique, irréflexive* $\# \subseteq F \times F$ *appelée relation de* conflit *telle que* $\leq \cap \# = \emptyset$, *pour chaque* e, $\{e_1 : e_1 \leq e\}$ *et fini (hypothèse de causes finies) et* $e\#e_1 \leq e_2$ *implique* $e\#e_2$ *(hérédité de conflit), et*
- *d'une fonction d'étiquetage* $l : F \mapsto Act$.

On note $\boldsymbol{E}$ *la classe des structures d'événements premières étiquetées.*

Dans la représentation graphique des SE, nous écrirons les noms d' événements ainsi que leurs étiquettes ($a{:}\alpha$, où $a \in F$ et $\alpha = l(a)$). Seuls les conflits immédiats (et non ceux obtenus par hérédité de conflit) seront indiqués. Enfin, la relation $\leq$ est représentée par des arcs orientés, en omettant les causes dérivées par transitivité. La figure 3.3 représente une SE dans laquelle les événements a et b sont concurrents. Ces deux événements sont en conflit avec l'événement c, et par hérédité de conflit, avec l'événement d dont c est la cause.

Les notions d'état et d'indépendance sont ici des notions dérivées. Un état de calcul d'une SE est un ensemble d'événements x, appelé *configuration*, qui ont eu lieu. Si un événement

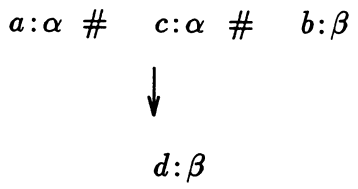


Figure 3.3: Un exemple de structure d'événements

a eu lieu, alors toutes ses causes ont également eu lieu. De plus, deux événements en conflit ne peuvent appartenir au même état. L'indépendance d'événements, appelé *concurrence*, est définie comme le complément de l'union de la causalité et du conflit.

Définition 3.10 *Étant donnée une SE $\mathcal{F}$, une* configuration *de $\mathcal{F}$ est un ensemble $x \subseteq F$ tel que $e \in x$ et $e_1 \leq e$ implique $e_1 \in x$ (fermé a gauche) et $e, e_1 \in x$ implique $\neg(e \# e_1)$ (sans conflit).*

Nous notons $Conf(\mathcal{F})$ l'ensemble de toutes les configurations de $\mathcal{F}$.

Nous disons que deux événements a et b sont concurrents, *noté a co b, ssi $\neg(a \leq b \vee b \leq a \vee a \# b)$.*

Les événements apparaissent comme des "sauts" atomiques entre configurations. On verra que ces sauts correspondent aux transitions dans un STA.

Définition 3.11 *Soit $\mathcal{F} = \langle F, \leq, \#, l \rangle \in \boldsymbol{E}$. Une* transition *dans $\mathcal{F}$ est un triplet $(s, e, u) \in Conf(\mathcal{F}) \times F \times Conf(\mathcal{F})$, notée $s \xrightarrow{e} u$, telle que $s \xrightarrow{e} u \overset{def}{\Leftrightarrow} e \notin s$ et $u = s \cup \{e\}$.*

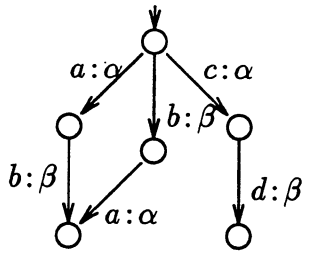


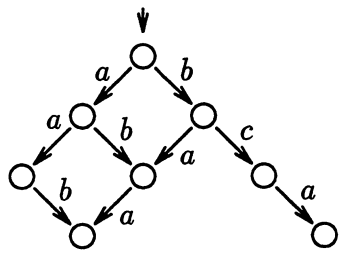
Figure 3.4: Le STA correspondant à la ES de la Figure 3.3 (aIb)

Les configurations, la relation d'indépendance *co* et les transitions, nous permettent de définir un STA:

Définition 3.12 *A chaque $\mathcal{F} \in \boldsymbol{E}$ on peut associer un STA $ea(\mathcal{F}) \overset{def}{=} \langle Conf(\mathcal{F}), \emptyset, E, co, \rightarrow, l \rangle$.*

La construction d'une SE à partir d'un STA est un peu plus compliquée car les événements dans les SE ne peuvent apparaître qu'une fois dans une exécution. La solution est tout à fait classique: il faut rajouter de l'information concernant le contexte à chaque occurrence d'un événement dans l'exécution. Cette information doit nous permettre, d'une part, de distinguer différentes apparitions d'un événement, et d'autre part, nous permettre d'identifier des occurrences où le contexte diffère (par exemple, par l'apparition d'un événement indépendant). Cette

abstraction doit identifier les trois occurrences de b dans le STA de la figure 3.5, et aussi montrer que les cinq transitions étiquetées par a représentent seulement trois occurrences différentes de cet événement, dans trois contextes différents, en identifiant, par exemple les occurrences de a dans le calcul $\langle a, b\rangle$ et $\langle b, a\rangle$.



G

Figure 3.5: Un exemple de STA, avec aIb

C'est notre définition de *SE-événement* qui accomplit cette tâche. Intuitivement, un SE-événement $[\rho a]$ représente l'occurrence de l'événement a dans dans la portion "essentielle" de la configuration C_ρ. Formellement, un SE-événement est une classe d'équivalence sur $Calc(G)$.

Définition 3.13 *Soit $G \in \mathbf{A}$, on définit le* SE-événement *représenté par $\rho a \in Calc(G)$ comme la classe d'équivalence de ρa dans la relation $\sim \subseteq Calc(G) \times Calc(G)$ ($[\rho a]$), où $\sim$ est définie comme la plus petite relation d'équivalence satisfaisant*

$$\begin{array}{rll} aIb & \textit{implique} & \rho a \sim \rho ba \\ C_\rho = C_\sigma & \textit{implique} & \rho a \sim \sigma a \end{array}$$

On définit aussi les événements d'un calcul *et d'un STA comme* $ev(\rho) \stackrel{def}{=} \{[\sigma] : \lambda < \sigma \leq \rho\}$ *et* $ev(G) \stackrel{def}{=} \bigcup_{\rho \in Calc(G)} ev(\rho)$.

Tous les éléments d'une classe $[\rho a]$ finissent par le même événement a, et représentent l'occurrence de a dans le contexte ρ. L'ensemble des calculs ρ est fermé par rapport à la permutation d'événements indépendants, et l'inclusion (à droite) d'événements indépendants de a.

Il y a dans $[\rho a]$ des éléments minimaux par rapport à cette inclusion. Ces éléments sont les exécutions les plus brèves que nous construit le contexte, et sont des permutations les unes des autres. On utilisera les éléments minimaux pour doter l'ensemble de SE-événements d'une relation d'ordre partiel et d'une relation de conflit, pour obtenir une SE.

Définition 3.14 *Un ordre partiel dans $ev(G)$ est définie par*

$$m_1 \leq m_2 \stackrel{def}{\Leftrightarrow} \exists \sigma \textit{ minimal pour } < \textit{ dans } m_2.\ m_1 \in ev(\sigma)$$

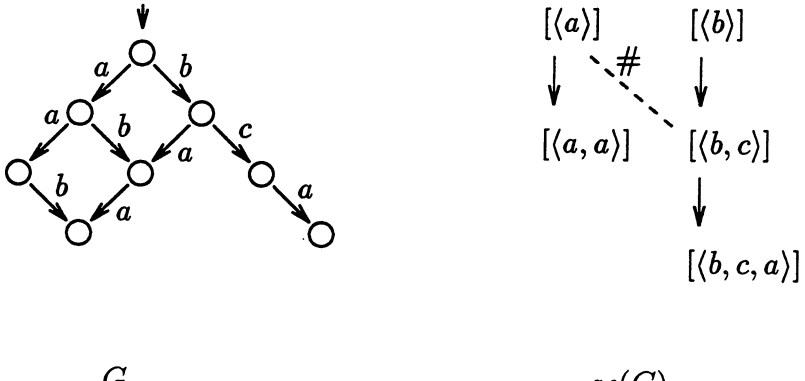
La relation de conflit dans $ev(G)$ *($\# \subseteq ev(G) \times ev(G)$) est définie par*

$$m_1 \# m_2 \stackrel{def}{=} \exists m_1', m_2'.\ m_1' \leq m_1 \wedge m_2' \leq m_2 \wedge m_1' \#_0 m_2'$$

où $m_1' \#_0 m_2' \stackrel{def}{=} \exists \sigma, a, b.\ \sigma a \in m_1' \wedge \sigma b \in m_2' \wedge a \neq b \wedge \neg(aIb)$

Définition 3.15 *A chaque $G \in \mathbf{A}$, on peut associer une SE $ae(G) \stackrel{def}{=} \langle ev(G), \leq, \#, l_1 \rangle$, où $\leq$ et $\#$ sont définies en 3.14 et $l_1([\sigma a]_\sim) \stackrel{def}{=} l(a)$*

La figure 3.6 montre un exemple de transformation *ae*. Les SE-événements sont $[\langle a \rangle] = \{\langle a \rangle, \langle b, a \rangle\}$, $[\langle a, a \rangle] = \{\langle a, a \rangle, \langle b, a, a \rangle, \langle a, b, a \rangle\}$, $[\langle b \rangle] = \{\langle b \rangle, \langle a, b \rangle, \langle a, a, b \rangle\}$, $[\langle b, c \rangle] = \{\langle b, c \rangle\}$ et $[\langle b, c, a \rangle] = \{\langle b, c, a \rangle\}$.



G $ae(G)$

Figure 3.6: $ae(G)$, avec aIb.

L'énoncé de l'adéquation de la transformation *ae* repose sur la préservation de l'espace des configurations.

Lemme 3.16 *La fonction h définie par $h(C_\rho) \stackrel{def}{=} ev(\rho)$ est un isomorphisme*

$$h : \langle Conf(G), \subseteq \rangle \mapsto \langle Conf(ae(G)), \subseteq \rangle$$

3.3 STA, STI et ST

Les STI peuvent s'obtenir a partir des STA en oubliant l'identité des événements, mais en gardant la relation d'indépendance engendrée sur les transitions. Si on oublie cette relation d'indépendance, on obtient les ST.

Définition 3.17 *Un* Système de Transition *(ST) étiqueté sur Act est une structure $T = \langle Q, i, \rightarrow \rangle$ où Q est un ensemble d'états, avec un état initial i et $\rightarrow \subseteq Q \times Act \times Q$ est une relation de transition.*

Les STI sont des ST enrichis par une relation d'indépendance sur les flèches qui satisfait les mêmes propriétés que la relation d'indépendance sur les événements d'un STA.

Définition 3.18 *Un* Système de Transition avec Indépendance *(STI) étiqueté sur Act est une structure $T_I = \langle Q, i, I, \rightarrow \rangle$ où $\langle Q, i, \rightarrow \rangle$ est un ST, et $I \subseteq \rightarrow \times \rightarrow$ est une relation irréflexive et symétrique d'*indépendance *qui satisfait:*

- $s \stackrel{\alpha}{\rightarrow} s_1 \smile s \stackrel{\alpha}{\rightarrow} s_2$ *implique* $s_1 = s_2$
- $s \stackrel{\alpha}{\rightarrow} s_1 I s \stackrel{\beta}{\rightarrow} s_2$ *implique* $\exists u, s \stackrel{\alpha}{\rightarrow} s_1 I s_1 \stackrel{\beta}{\rightarrow} u$ *et* $s \stackrel{\beta}{\rightarrow} s_2 I s_2 \stackrel{\alpha}{\rightarrow} u$

- $s \xrightarrow{\alpha} s_1 I s_1 \xrightarrow{\beta} u$ *implique* $\exists s_2, s \xrightarrow{\beta} s_2 I s_2 \xrightarrow{\alpha} u$
- $s \xrightarrow{\alpha} s_1 \smile s_2 \xrightarrow{\alpha} u I v \xrightarrow{\beta} v_1$ *implique* $s \xrightarrow{\alpha} s_1 I v \xrightarrow{\beta} v_1$

où $\smile$ *est la plus petite relation d'équivalence sur* $\rightarrow$ *qui contient* $\prec$ *avec*

$$s \xrightarrow{\alpha} s_1 \prec s_2 \xrightarrow{\alpha} u \stackrel{def}{\Leftrightarrow} s \xrightarrow{\alpha} s_1 I s \xrightarrow{\beta} s_2 \wedge s \xrightarrow{\alpha} s_1 I s_1 \xrightarrow{\beta} u \wedge s \xrightarrow{\beta} s_2 I s_2 \xrightarrow{\alpha} u$$

Les événements ne sont pas des objets de base dans les STI, mais peuvent être construits comme les classes d'équivalence induites par la relation $\smile$. La dernière condition dit que la relation d'indépendance peut être définie directement sur les événements. Les trois premières expriment que cette relation satisfait les mêmes propriétés que celles définies sur les STA.

Les ST et les STI peuvent être considérés trivialement comme des STA.

Définition 3.19 *A chaque* $STT = \langle Q, i, \rightarrow \rangle$ *on peut associer un STA* $ta(\langle Q, i, \rightarrow \rangle) \stackrel{def}{=} \langle Q, i, \rightarrow, \emptyset, \rightarrow_1, l \rangle$ *où* $\rightarrow_1 \stackrel{def}{=} \{(s,t,u) : t = s \xrightarrow{\alpha} u\}$ *et* $l(s, s \xrightarrow{\alpha} u, u) \stackrel{def}{=} \alpha$.

A chaque $STIT_I = \langle Q, i, \rightarrow, I_{tsi} \rangle$ *on peut associer un STA* $tia(\langle Q, i, \rightarrow, I_{tsi} \rangle) \stackrel{def}{=} \langle Q, i, E, I, \rightarrow_2, l \rangle$ *où* $E \stackrel{def}{=} \{[s \xrightarrow{\alpha} u]_\smile\}$, $[t_1]_\smile I [t_2]_\smile \stackrel{def}{\Leftrightarrow} t_1 I_{tsi} t_2$, $\rightarrow_2 \stackrel{def}{=} \{(s, [t]_\smile, u) : t = s \xrightarrow{\alpha} u\}$ *et* $l([s \xrightarrow{\alpha} u]_\smile) \stackrel{def}{=} \alpha$.

Les STA obtenus par *tia* à partir d'un STI sont exactement ceux qui satisfont:

- $s \xrightarrow{a:\alpha} u$ et $s \xrightarrow{e:\alpha} u$ implique $a = e$.
- $s \xrightarrow{a} u$ et $s' \xrightarrow{a} u'$ impliquent qu'il existe une séquence d'états $s = s_0 \ldots s_n = s'$ tel que $\forall j = 0 \ldots n-1$ il existe b_j, avec $b_j I e$, $s_j \xrightarrow{b_j} s_{j+1}$ ou $s_{j+1} \xrightarrow{b_j} s_j$.

La traduction dans l'autre sens est aussi simple pour les ST.

Définition 3.20 *A chaque STA* G *on peut associer un ST* $at(\langle Q, i, E, I, \rightarrow, l \rangle) \stackrel{def}{=} \langle Q, i, \rightarrow_3 \rangle$, *où* $\rightarrow_3 \stackrel{def}{=} \{(s, \alpha, u) : s \xrightarrow{e} u \wedge l(e) = \alpha\}$

Toutes ces traductions préservent la structure du graphe étiqueté sur Act, et dans le cas où cela a un sens, la relation d'indépendance entre les transitions voisines.

Toutes nos définitions de graphes étiquetés sont extensionelles, c.-à-d. qu'entre deux états, on ne peut avoir deux transitions avec la même étiquette. Par conséquent, par *at*, différentes transitions $s \xrightarrow{a:\alpha} u$ et $s \xrightarrow{b:\alpha} u$ donnent une unique transition $s \xrightarrow{\alpha}_3 u$.

Cette identification rend la construction des STI à partir des STA légèrement plus compliquée. Soit le STA G de la Figure 3.7. Il n'est pas possible de construire un STI avec le graphe sous-jacent T (dans la même figure) en ayant les mêmes exécutions. Selon qu'on déclare les α-transitions indépendantes ou non des β-transitions, on ne pourra pas reproduire l'exécution $\langle c, b \rangle$, où les événements sont reliés par l'ordre partiel, ou bien on aura l'exécution $\langle \beta, \alpha \rangle$, avec les deux transitions non indépendantes, ce qui ne correspond à aucune comportement du G.

Pour éviter ce problème, on dépliera d'abord les STA avant d'oublier l'identité des événements. Cette traduction préserve trivialement l'espace des configurations modulo isomorphisme.

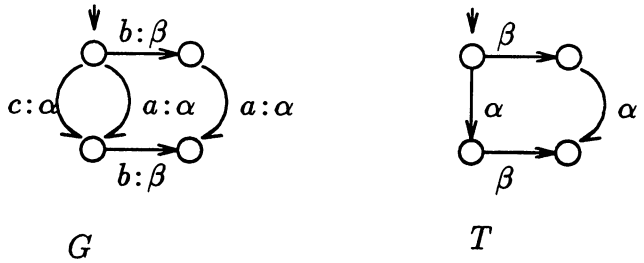


Figure 3.7: Choisir $ati(G) = T$ est une mauvaise idée. (aIb)

Définition 3.21 *Soit le STA* $G = \langle Q, i, E, I, \rightarrow, l\rangle$*, et son dépliage* $déplie(G) = \langle Conf(G), \emptyset, E_d, I_d, \rightarrow_d, l_d\rangle$*. Le STI associé à* G *est défini par* $ati(G) \stackrel{def}{=} \langle Conf(G), \emptyset, I_4, \rightarrow_4\rangle$*, où* $x_1 \stackrel{\alpha}{\rightarrow}_4 x_2 \stackrel{def}{\Leftrightarrow} x_1 \stackrel{a:\alpha}{\rightarrow}_d x_2$ *et* $x_1 \stackrel{\alpha}{\rightarrow}_4 x_2 I_4 x_3 \stackrel{\beta}{\rightarrow}_4 x_4 \stackrel{def}{\Leftrightarrow} x_1 \stackrel{a:\alpha}{\rightarrow}_d x_2 \wedge x_3 \stackrel{b:\beta}{\rightarrow}_d x_4 \wedge aIb$

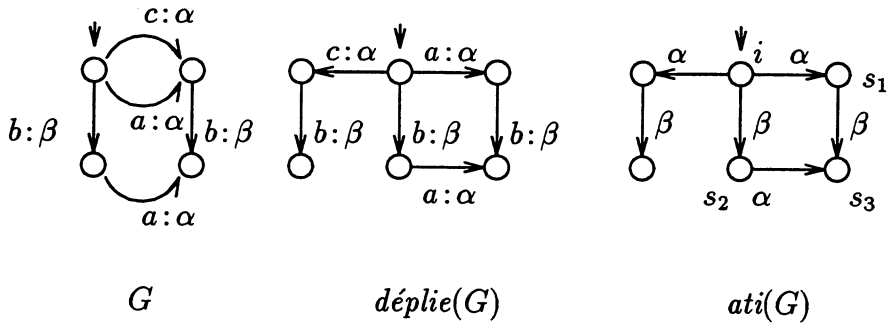


Figure 3.8: Un exemple de $ati(G)$

La Figure 3.8 montre un exemple de transformation ati $(aIb, i \stackrel{\alpha}{\rightarrow} s_1 I i \stackrel{\beta}{\rightarrow} s_2, i \stackrel{\alpha}{\rightarrow} s_1 I s_1 \stackrel{\beta}{\rightarrow} s_3$ $s_2 \stackrel{\alpha}{\rightarrow} s_3 I i \stackrel{\beta}{\rightarrow} s_2, s_2 \stackrel{\alpha}{\rightarrow} s_3 I s_1 \stackrel{\beta}{\rightarrow} s_3)$.

3.4 Quelques commentaires

Les transformations présentées dans ce chapitre nous permettent de voir certains modèles du parallélisme comme cas particuliers des STA. Dans cette thèse, on dira qu'un STA est un RP quand il est l'image d'un RP par notre traduction, et de même pour les autres modèles.

La classe des STA élémentaires (introduite dans la Section 3.1) est assez large pour contenir $ra(\mathbf{R}_s)$, où $\mathbf{R}_s$ est la classe des Réseaux de Petri *saufs*. Un réseau est dit sauf ssi pour tout marquage atteignable M, et pour tout événement e, ${}^\bullet e \subseteq M$ implique $e^\bullet \cap (M \setminus {}^\bullet e) = \emptyset$.

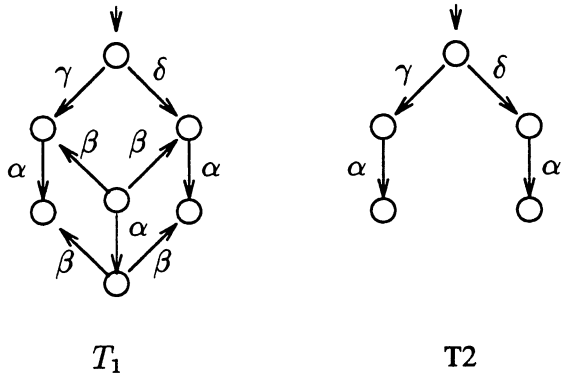
La classe des STA élémentaires est aussi suffisamment large pour pouvoir contenir la classe $ea(\mathbf{E})$. D'autres SE plus générales que les Premières peuvent être considérées comme cas particuliers des STA. Il existe une transformation simple [Sas93] des SE Stables [Win89a] dans des STI, qui peut être reprise pour leur transformation dans des STA.

Mais il est difficile d'imaginer une transformations des SE non-stables dans des STA.

Les STI simplifient agréablement les STA au sens où certains détails peut-être trop syntaxiques, sans support intuitif en termes de comportement dans les STA, sont enlevés. Dans les STI, la notion d'événement est définie "localement" dans le graphe sous-jacent, à partir de l'indépendance des transitions.

Mais cette approche a aussi ses point faibles: Deux transitions éloignées peuvent être déclarées indépendantes, même quand les événements qu'elles représentent ne forment aucun diamant.

Un problème plus important est que la notion d'événement comme classe d'équivalence induite par $\smile$ est sensible aux parties non atteignables. Par exemple, la Figure 3.9 montre deux STI avec la même partie atteignable. Et pourtant si on suppose que les β-transitions sont indépendantes des α-transitions, alors seulement dans T_1 les α-transitions appartiennent au même événement.



T_1 T2

Figure 3.9: Dans les STI, la relation $\sim$ dépend de la partie non atteignable

Chapitre 4

Raffinement d'actions

Dans ce chapitre, nous proposons une définition de *raffinement d'actions* dans les STA.

Le raffinement d'actions est une opération ayant pour effet de remplacer des transitions, que l'on considère élémentaires à un certain niveau d'abstraction, par des STA. Cette notion reflète le développement de programmes par raffinements successifs.

Notre construction repose sur une combinaison adéquate des techniques utilisées pour le raffinement d'actions dans les ST [GW89] et celles pour les SE [GG89c], de telle façon que les résultats obtenus pour ces deux modèles sont généralisés. Plus précisement, notre opération de raffinement coïncide avec celle de [GW89] (resp. [GG89c]) quand elle est appliquée à un STA représentant un ST (resp. une SE); l'image par la fonction de raffinement est alors un ST (resp. une SE).

Les opérations de raffinement ainsi définies sont des endo-foncteurs dans une large sous-catégorie des STA.

Outre la généralisation des travaux sur les ST et les ES, notre travail peut être utilisé pour d'autres modèles de systèmes réactifs. On voit trivialement que nous avons ainsi une définition de raffinement pour les STI. Nous pensons que notre définition s'applique aux Automates de Traces, et aux STA non-stables en avant.

La première section de ce chapitre introduit le raffinement d'actions dans les STA (Section 4.1). La section suivante est consacrée à l'étude des exécutions dans les STA raffinés (Section 4.2), ce qui nous permet de montrer que notre raffinement coïncide avec celui de [GG89c] sur les SE. Le chapitre finit par la caractérisation des raffinements en termes de catégories (Section 4.3).

4.1 Le raffinement d'actions dans les STA

Dans cette section, nous définissons le raffinement d'actions sur les STA. Nous présentons d'abord notre démarche d'une façon informelle.

Dans [GW89] les auteurs définissent le raffinement sur les ST. Cet opérateur remplace chaque transition $s \xrightarrow{\alpha} u$ par une copie de $r(\alpha)$, le ST qui raffine l'action α. A l'issue de ce remplacement, l'état initial de cette copie $r(\alpha)$ est identifié à s, et tous les état terminaux (dans le cas où ils existent) sont identifiés à u. Le point important est la description de l'ensemble des nouveaux états et des transitions entre ces états.

Dans notre cas chaque flèche étiquetée par $e \in E$ dans le graphe sous jacent à G est remplacée par une copie séparée du STA $r(l(e))$.

Pour décrire les événements de $r(G)$ nous suivons la technique de [GG89c] où $E_{r(G)}$ est l'ensemble de couples de la forme (e, e_1) avec $e \in E$ et $e_1 \in r(l(e))$. Suivant cette même technique, deux événements sont indépendants s'ils sont obtenus par raffinement de deux événements indépendants de G ou s'ils sont deux événements indépendants dans une même copie de $r(\alpha)$.

Cette définition d'indépendance dans $r(G)$ engendre quelques problèmes subtils dans les STA quand un même événement apparaît dans deux arêtes adjacentes d'un même chemin. Dans la Figure 4.1, nous tentons de raffiner G par la fonction r. La structure T (obtenue en remplaçant chaque occurrence de e par une copie de $r(\alpha)$) correspond assez bien à notre intuition, puisque l'exécution de la première copie de $r(\alpha)$ doit être complète avant de commencer l'exécution de la seconde. Malheureusement T n'est pas un STA. Des diamants doivent être rajoutés aux côtés car le calcul $s \xrightarrow{(e,b)}_T \xrightarrow{(e,a)}_T$, ne satisfait pas la condition de commutativité, étant donné que $(e, a)I(e, b)$ (même si ces deux événements proviennent de deux occurrences différentes de e dans G). Si on rajoute les diamants manquants, comme on le montre par G_1, le résultat est bien un STA, mais on perd la séquentialité entre les deux occurrences de e.

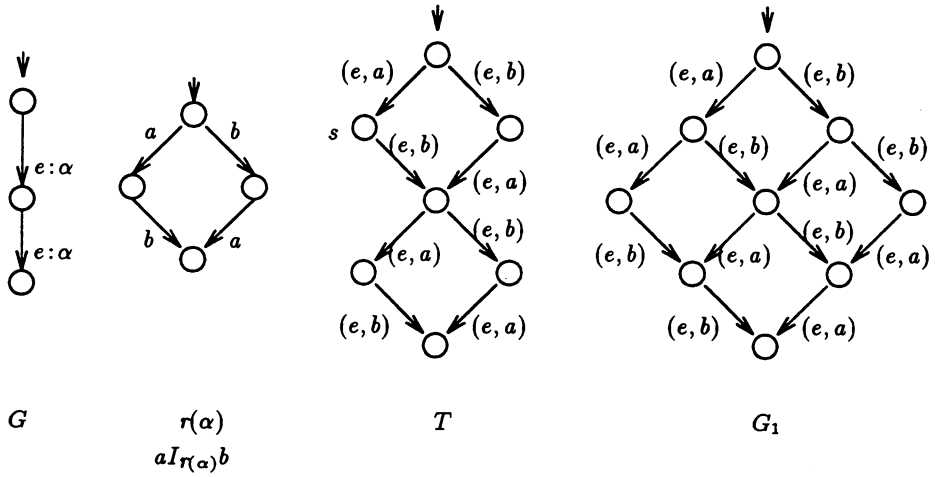


Figure 4.1: Le problème du raffinement avec contact (aIb)

Nous avons alors été amenés, dans notre définition de raffinement à ne considérer que les STA *sans contact*, i.e. ceux qui vérifient que pour tous $s_1, s_2, s_3 \in Q$, $a, b \in E$, $s_1 \xrightarrow{a} s_2 \xrightarrow{b} s_3$

implique $a \neq b$.

Notons que toutes les SE sont sans contact, ainsi que les ST qui n'ont pas de cycles de longueur un. En fait, dans les ST la relation d'indépendance est toujours vide et ce genre de problème n'apparaît pas.

Il est à remarquer toutefois que la propriété d'être sans contact n'est pas une restriction importante. En effet, pour chaque STA il existe un STA sans contact équivalent dans un sens très fort (cette notion d'equivalence, appelée dI-bisimulation, est presentée dans le Chapitre 5).

La deuxième restriction syntaxique consiste à ne considérer dans $r(Act)$ que des STA *enracinés* , ceux dans lesquels, il n'existe pas de transitions qui arrivent à l'état initial (c-à-d. $\forall s \xrightarrow{a} u.\ u \neq i$).

En effet, nous montrons à l'aide de la figure 4.2 que les points de choix du G ne sont pas préservés par r quand $r(\alpha)$ n'est pas enraciné. Dans G_1 l'événement b peut être choisi à différents niveaux d'un calcul, ce qui n'est pas le cas dans G. Cette restriction à des STA enraciné n'est pas pour autant un inconvénient car il est trivial de construire pour tout STA, un STA enraciné équivalent modulo dépliage.

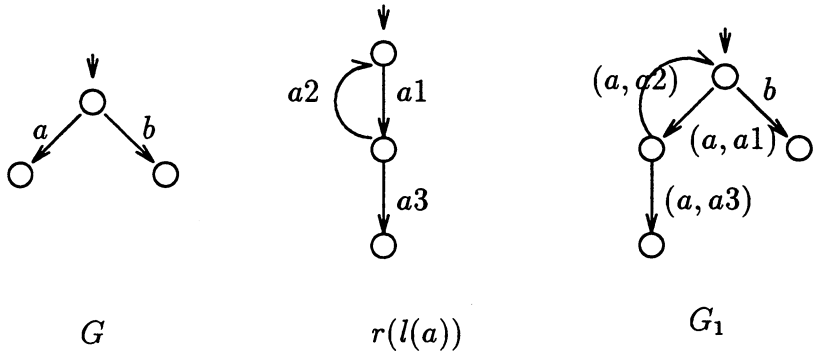


Figure 4.2: Le problème du raffinement non enraciné

Une troisième restriction, la seule véritablement significative, est celle qui interdit le *raffinement d'oubli*, où $r(\alpha)$ peut avoir comme seul calcul la séquence vide (c-à-d. $i_{r(\alpha)} \not\rightarrow$).

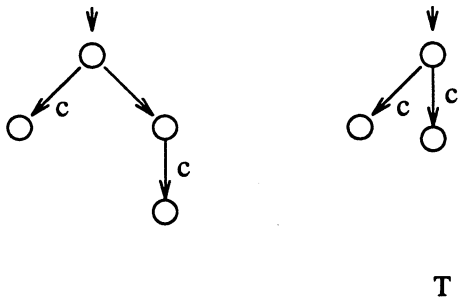


Figure 4.3: Le probleme du raffinement d'oubli ($r(l_G(a)) = \emptyset$)

Comme montre la Figure 4.3, le déterminisme peut être perdu avec des raffinement d'oubli.

Il existe bien sûr des façons de traiter le raffinement d'oubli. Par exemple, en le représentant comme un blocage: le résultat d'un raffinement r avec $i_{r(\alpha)} \not\rightarrow$ est alors d'éliminer simplement

toutes les transitions étiquetées par des événements $a:\alpha$.

Tous les résultats de ce chapitre peuvent être reproduits avec cette alternative, sauf un: le raffinement ainsi défini n'est pas une extension de celui classique sur les ST. C'est pour cela que l'on choisit d'interdire le raffinement d'oubli.

Nous définissons maintenant formellement les fonctions de raffinement que nous nous proposons de considérer.

Définition 4.1 *Une* fonction de raffinement *est une fonction* $r : Act \mapsto \boldsymbol{A}$ *t.q. pour tout* $\alpha \in Act$, $r(\alpha)$ *est sans contact, enraciné, et a au moins un calcul non-vide.*

Nous allons maintenant définir l'opération de raffinement sur les STA sans contact. Dans la suite, nous écrirons $r(G)$ avec l'hypothèse implicite que G est sans contact.

La tâche la plus difficile est la définition de l'ensemble des états de $r(G)$.

Dans [GW89], les nouveaux états introduits dans le ST par le raffinement d'actions sont obtenus de la façon suivante: chaque transition de la forme $s \xrightarrow{\alpha} u$ dans le ST de départ doit être remplacée par une copie du ST $r(\alpha)$. Différentes copies pour différents transitions sont obtenues simplement en marquant chaque copie par la transition qu'elle raffine. Pour chaque état s_1 de $r(\alpha)$, la copie qui raffine la transition $s \xrightarrow{\alpha} u$ a un état $(s \xrightarrow{\alpha} u, s_1)$.

Dans notre cas, les transitions ont la forme $s \xrightarrow{e_1} u$. Le déterminisme des STA nous permet de faire abstraction de l'état u. Nous pouvons identifier simplement les différentes copies d'un état s_1 de $r(l(e_1))$ comme $(s,(e_1,s_1))$, au lieu de $(s \xrightarrow{e_1} u, s_1)$.

Mais cette technique inspirée de celle des ST ne nous suffit pas pour bien gérer la relation d'indépendance dans $r(G)$. Quand deux transitions indépendantes (c-à-d. étiquetées par des événements a et b indépendantes) partent d'un état s de G, les événements dans les copies de $r(l(a))$ et $r(l(b))$ sont indépendants, ils vont donc générer des diamants, et des nouveaux états. Ces diamants et ces états ne correspondent pas au raffinement d'une seule des deux transitions, mais raffinement des deux du fait de leur indépendance. Ici, la notion de ST-état (Définition 2.5) va nous être utile.

Nous allons simplement généraliser la notation $(s,(e_1,s_1))$, en considérant des ST-états au lieu des transitions. Un état de $r(G)$ aura la forme $(s,\{\ldots(e_j,s_j)\ldots\})$, où $(s,\{\ldots e_j \ldots\}) \in Diam(G)$ et $s_j \in Q_{r(l(e_j))}$.

Intuitivement, le ST-état $(s,\{\ldots e_j \ldots\})$ exprime le fait que les événements $\{\ldots e_j \ldots\}$ peuvent être exécutés en parallèle à partir de l'état s. Quand l'exécution de ces événements est détaillée par la fonction de raffinement, en les remplaçant chacun par un STA, on peut identifier les états atteints pendant l'exécution parallèle de tous ces événements en sachant à quel niveau on se trouve selon chaque "coordonnée" e_j. D'où la notation $(s,\{\ldots(e_j,s_j)\ldots\})$.

La définition suivante nous permet d'associer à chaque ST-état avec n événements indépendants, l'ensemble des états atteints pendant l'exécution parallèle de ces événements. Cette dernière forme dans notre intuition un cube à n dimensions. Dans chaque état du cube, chaque événement e_j apparaît associé à un état de $r(l(e_j))$. Les états initiaux et les états finaux de $r(l(e_j))$ ne sont pas considérés comme des états du cube, car ils représentent soit une exécution de l'événement e_j non encore engagée, ou une exécution déjà finie.

Définition 4.2 *Soient $G = \langle Q, i, E, I, \rightarrow, l\rangle$ et r une fonction de raffinement. Pour chaque ST-état $(s, \{\ldots e_j \ldots\}) \in Diam(G)$, on définit le* r-cube *(ou* cube *pour simplifier) de s par:*

$$cube(s, \{\ldots e_j \ldots\}) \stackrel{def}{=} \{(s, \{\ldots (e_j, s_j) \ldots\}) : \forall j.\ s_j \in S_{r(l(e_j))}, s_j \neq i_{r(l(e_j))}, s_j \rightarrow_{r(l(e_j))}\}$$

Pour faciliter la lecture, on écrira souvent (s, Π) au lieu de $(s, \{\ldots (e_j, s_j) \ldots\})$, et quand $(e, a) \notin \Pi$, $(s, \Pi(e, a))$ au lieu de $(s, \Pi \cup \{(e, a)\})$.

Nous pouvons maintenant définir notre opération de raffinement d'actions sur des STA.

Définition 4.3 *Soit $G = \langle Q, i, E, I, \rightarrow, l\rangle \in \boldsymbol{A}$, sans contact, et une fonction de raffinement r. Le* raffinement de G par r *est $r(G) = \langle S_r, i_r, E_r, I_r, \rightarrow_r, l_r\rangle$, où*

- $E_r \stackrel{def}{=} \bigcup_{e \in E}(\{e\} \times E_{r(l(e))})$
- $(e, a) I_r (e', a') \stackrel{def}{\Leftrightarrow} eIe' \vee (e = e' \wedge a I_{r(l(e))} a')$
- $S_r \stackrel{def}{=} \bigcup_{(s,P) \in D(G)} cube(s, P)$ *est l'union de tous les composants des cubes,*
- $i_r \stackrel{def}{=} (i, \emptyset)$
- $\rightarrow_r \subseteq S_r \times E_r \times S_r$ *est définie selon quatre cas*

$$\begin{array}{rcll}
(s_1, \Pi) & \stackrel{(e,a)}{\rightarrow} & (s_1, \Pi(e, s_e)) & si\ \ i_{r(l(e))} \stackrel{a}{\rightarrow}_{r(l(e))} s_e \rightarrow_{r(l(e))} \\
(s_1, \Pi(e, s_e)) & \stackrel{(e,a)}{\rightarrow} & (s_1, \Pi(e, s'_e)) & si\ \ s_e \stackrel{a}{\rightarrow}_{r(l(e))} s'_e \rightarrow_{r(l(e))} \\
(s_1, \Pi(e, s_e)) & \stackrel{(e,a)}{\rightarrow} & (s_2, \Pi) & si\ \ s_e \stackrel{a}{\rightarrow}_{r(l(e))} \not\rightarrow_{r(l(e))},\ \ s_1 \stackrel{e}{\rightarrow} s_2 \\
(s_1, \Pi) & \stackrel{(e,a)}{\rightarrow} & (s_2, \Pi) & si\ \ i_{r(l(e))} \stackrel{a}{\rightarrow}_{r(l(e))} s_e \not\rightarrow_{r(l(e))},\ \ s_1 \stackrel{e}{\rightarrow} s_2 \\
 & & & \quad et\ pour\ chaque\ (e', s') \in \Pi,\ e'Ie
\end{array}$$

  Les trois premiers cas montrent respectivement comment: s'engager, avancer et finir le raffinement d'un événement e. Le dernier est un cas particulier, ou l'état s_e auquel on arrive au moment de s'engager est déjà un état terminal.

- $l_r((e, a)) \stackrel{def}{=} l_{r(l(e))}(a)$

La Figure 4.4 montre un exemple de raffinement.

Lemme 4.4 *Soient $G = \langle Q, i, E, I, \rightarrow, l\rangle \in \boldsymbol{A}$, sans contact, et une fonction de raffinement r, alors $r(G) \in \boldsymbol{A}$.*

Preuve. Il faut vérifier le déterminisme, la stabilité en avant et la commutativité de $\rightarrow_r$. Pour chaque propriété, il faut analyser les différents cas de la définition de $\rightarrow_r$, et utiliser comme hypothèses le fait que G et $r(Act)$ sont de STA. □

Remarque 2 *Notre démarche fait que notre définition de raffinement coïncide trivialement avec celle de [GW89] quand G et $r(Act)$ sont des ST.*

Le lemme suivant caractérise l'ensemble des ST-états du $r(G)$.

Lemme 4.5 *Soit $G = \langle Q, i, E, I, \rightarrow, l\rangle \in \boldsymbol{A}$, sans contact, et une fonction de raffinement r. L'ensemble des ST-états de $r(G)$, $Diam(r(G))$ est l'ensemble de tous les couples*

$$(\bar{s}, \bar{P}) = ((s, \{\ldots (a_j, s_j) \ldots\}), \{\ldots (a_k, c_k) \ldots\} \cup \{\ldots (b_l, d_l) \ldots\})$$

où $\{\ldots a_j \ldots\} \cap \{\ldots b_l \ldots\} = \emptyset$, tels que

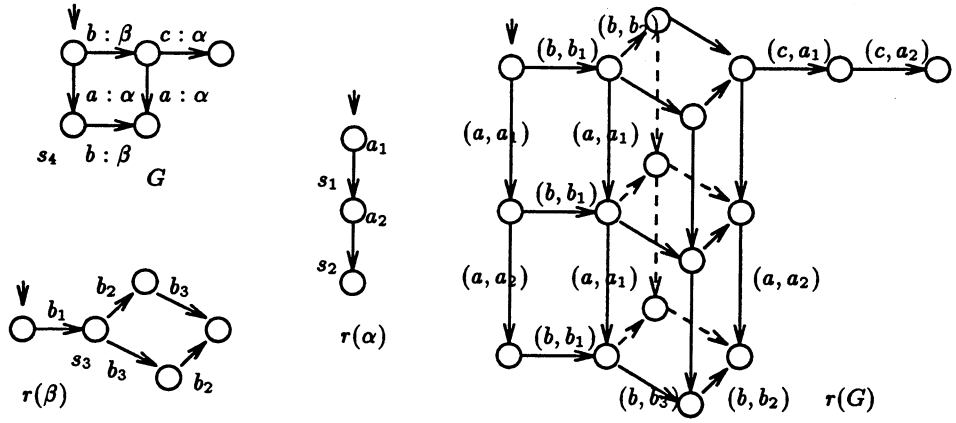


Figure 4.4: An example of refinement $(aIb, b_2 I_{r(\beta)} b_3)$

- $\bar{s} = (s, \{\ldots a_j \ldots b_l \ldots\}) \in Diam(G)$.
- $(s, \{\ldots (a_j, s_j) \ldots\}) \in Q_{r(G)}$.
- $\forall a_j.\ (s_j, P_j) \in Diam(r(l(a_j)))$ *où* $P_j \stackrel{def}{=} \{c : (a_j, c) \in \bar{P}\}$.
- $\forall b_l.\ (i_{r(l(b_l))}, P_l) \in Diam(r(l(b_l)))$ *où* $P_l \stackrel{def}{=} \{d : (b_l, d) \in \bar{P}\}$.

on note

$$(s, \{\ldots (a_j, (s_j, P_j)) \ldots (b_l, (i_{r(l(b_l))}, P_l)) \ldots\}) \stackrel{def}{=} (\bar{s}, \bar{P})$$

Soient $u_j \stackrel{def}{=} cible(s_j, P_j)$ *et* $u_l \stackrel{def}{=} cible(i_{r(l(b_l))}, P_l)$.
La cible de $(\bar{s}, \bar{P})$ *est* $cible(\bar{s}, \bar{P}) = (u, \{\ldots (a_j, u_j) \ldots (b_l, u_l) \ldots\})$ *où*

- *à droite apparaissent tous les états cibles qui ne sont pas des états bloqués, (i.e.* $u_j \rightarrow_{r(l(a_j))}$, $u_l \rightarrow_{r(l(b_l))}$*)*
- $s \stackrel{a_{j_1}}{\rightarrow} \cdots \stackrel{a_{j_m} b_{l_1}}{\rightarrow} \cdots \stackrel{b_{l_o}}{\rightarrow} u$, *où* $\{a_{j_1} \ldots a_{j_m}, b_{l_1} \ldots b_{l_o}\} \subseteq \{\ldots a_j \ldots b_l \ldots\}$ *est le sous-ensemble des événements associés aux ST-états dont les états cibles sont des états bloqués.*

La taille de $(\bar{s}, \bar{P})$ *est* $|(\bar{s}, \bar{P})| = \Sigma_{a_j} |(s_j, P_j)| + \Sigma_{b_l} |(i_{r(l(b_l))}, P_l)|$

Preuve. Les seuls points non triviaux sont $(s_j, P_j) \in Diam(r(l(a_j)))$ et $(i_{r(l(b_l))}, P_l) \in Diam(r(l(b_l)))$, pour lesquels l'hypothèse de G sans contact est nécessaire.

On peut facilement construire une preuve (pas très elegante) à partir de la caractérisation des calculs de $r(G)$ (voir section 4.2, page 42).

□

4.2 Raffinement des exécutions

Le lemme suivant nous montre comment à partir d'un calcul du STA raffiné $\sigma \in Calc(r(G))$, on peut retrouver un calcul plus abstrait de G, pour lequel, il est en un certain sens, le raffinement (on le note $r^{-1}(\sigma) \in Calc(G)$). On peut retrouver aussi, pour chaque occurrence d'un événement e dans $r^{-1}(\sigma)$, un calcul de $r(l(e))$ qui le raffine.

Lemme 4.6 *Chaque calcul $\sigma \in Calc(r(G))$ peut être transformé, par permutation d'événements indépendants, en un autre calcul, concaténation de séquences $\sigma_1\sigma_2 \ldots \sigma_n$, où chaque σ_j a la forme $\langle (e_j, a_{j_1}) \ldots (e_j, a_{j_{m_j}}) \rangle$, avec $\rho = \langle e_1, e_2 \ldots e_n \rangle \in Calc(G)$ et $\sigma'_j = \langle a_{j_1}, \ldots a_{j_{m_j}} \rangle \in Calc(r(l(e_j)))$.*

En plus, si j n'est pas maximal dans ρ alors σ'_j est un calcul complet, $\sigma'_j \in CalcC(r(l(\rho(j))))$.

L'ordre des couples (e_j, a_{j_1}) dans σ nous donne une (unique) séquence $\langle e_1, e_2 \ldots e_n \rangle \in Calc(G)$, notée $r^{-1}(\sigma)$.

Preuve. La preuve se fait par induction sur $\mid \sigma \mid$. Le cas $\mid \sigma \mid = 0$ est trivial.

Considérons $\sigma' \in Calc(r(G))$ de la forme $\sigma(e, a)$ et supposons que σ puisse être transformé par permutation d'événements indépendants en $\sigma_1\sigma_2 \ldots \sigma_n$ où $\sigma_j = \langle (e_j, a_{j_1}) \ldots (e_j, a_{j_{m_j}}) \rangle$ et en posant $\rho \stackrel{\text{def}}{=} \langle e_1, e_2 \ldots e_n \rangle$

- $\rho \in Calc(G)$
- $\langle a_{j_1}, a_{j_2}, \ldots a_{j_{m_j}} \rangle \in Calc(r(l(\rho(j))))$
- Si $\rho(j)$ n'est pas maximal dans ρ, alors $\langle j_1, j_2 \ldots, j_{m_j} \rangle \in CalcC(r(l(\rho(j))))$.

Soit $k \stackrel{\text{def}}{=} Max\{j : \neg(\rho(j) I e)\}$.

Si k n'existe pas, alors $cible(\sigma) = (s, \Pi)$, ou $\neg\exists(e, s_e) \in \Pi$, $\langle a \rangle \in Comp(r(l(e)))$, les éléments non-maximaux dans σ' sont les mêmes que dans σ, donc la séquence $\sigma_1 \ldots \sigma_n \langle (e, a) \rangle$ satisfait les conditions cherchées.

Si k existe, alors on distingue deux cas selon que $\rho(k) = e$ ou non.

- Si $\rho(k) = e_k = e$. Puisque $r(G)$ est un STA et que e est indépendant de tous les $\rho(l)$ $(k < l \leq n)$, on peut permuter $\sigma(e, a)$ et obtenir la séquence $\sigma_1 \ldots \sigma_k \langle (e, a) \rangle \sigma_{k+1} \ldots \sigma_n$ qui est aussi un calcul de $r(G)$. Soit $cible(\sigma_1 \ldots \sigma_k) = (s, \Pi)$.
  La transition pour arriver à (s, Π) a été introduite en utilisant les cas un ou deux de la définition de $\rightarrow_r$. Dans un autre cas, on aurait $\stackrel{e}{\rightarrow} s \stackrel{e}{\rightarrow}$, qui est interdit par l'hypothèse de STA sans contact. On sait qu'il existe un couple $(e, s_k) \in \Pi$, donc $s_k \stackrel{a}{\rightarrow}_{r(l(e))}$, et $\langle a_{k_1} \ldots a_{k_{m_k}}, a \rangle \in Calc(r(l(e)))$.
  Notons que $r^{-1}(\sigma)$ et $r^{-1}(\sigma(e, a))$ sont égaux et ont donc les mêmes éléments non-maximaux.
- Sinon, $\rho(k) = e_k \neq e$. Soit $cible(\sigma_1 \ldots \sigma_n) = (s, \Pi)$. Comme $\neg(e I e_k)$, e n'apparaît pas dans Π. Ainsi, la transition étiquetée par (e, a) ne peut apparaître qu'en utilisant les cas un ou quatre de la définition de $\rightarrow_r$.
  La séquence $\sigma_1 \ldots \sigma_n \langle (e, a) \rangle$ est un bon candidat.
  Il reste à vérifier que pour tout j maximal dans ρ mais non maximal dans ρe, le calcul σ_j est complet. En effet, comme $\neg(\rho(j) I e)$, $\rho(j)$ n'apparaît pas dans Π, le calcul σ_j a abouti à un état bloqué, il est donc complet.

□

Il est bien sûr possible d'associer à chaque calcul $\sigma \in Calc(G)$ son raffinement $r(\sigma) \subseteq Calc(r(G))$. Le prochain lemme montre que pour tout $\rho \in r(\sigma)$, $r^{-1}(\rho) = \sigma$.

Définition 4.7 *Soient r une fonction de raffinement, $G \in \boldsymbol{A}$, sans contact, et $\rho = \langle e_1, e_2, \ldots e_n \rangle \in Calc(G)$.*

Le raffinement de ρ par r *est le plus petit ensemble* $r(\rho)$ *de séquences d'événements de* $E_{r(G)}$, *fermé par permutation d'événements indépendants selon* $I_{r(G)}$ *qui contient toutes les séquences de la forme* $\sigma_1\sigma_2\ldots\sigma_n$ *telles que pour chaque* j, $\sigma_j = e_j \times \sigma'_j$ *(où* $a \times \langle a_1 \ldots a_n\rangle \stackrel{def}{=} \langle (a,a_1)\ldots(a,a_n)\rangle$*), avec* $\sigma'_j \in Calc(r(l(\rho(j))))$ *si* j *est maximal in* ρ, *et* $\sigma'_j \in CalcC(r(l(\rho(j))))$ *dans les autres cas.*

Le lemme suivant exprime la compositionalité du raffinement d'actions sur les calculs (c.-à-d. les calculs de $r(G)$ sont les raffinements des calculs de G).

Il sert d'abord à établir les liens entre le raffinement des STA et celui des SE. Il servira ensuite à prouver la congruence relativement au raffinement de certaines équivalences présentées dans le chapitre 5.

Lemme 4.8 $Calc(r(G)) = r(Calc(G)) \stackrel{def}{=} \bigcup_{\rho\in Calc(G)} r(\rho)$

Preuve. Le sens $\subseteq$ est immédiat en utilisant le Lemme 4.6.

Pour le sens $\supseteq$, soit $\rho \in Calc(G)$, il faut montrer que $r(\rho) \subseteq Calc(r(G))$. Puisque par définition $r(\rho)$ et $Calc(r(G))$ sont clos par permutation d'événements indépendants, il suffit de considérer les séquences $\sigma_1\ldots\sigma_n \in r(\rho)$ où σ_j est de la forme $\rho(j)\times\sigma'_j$ avec $\sigma'_j \in Calc(r(l(\rho(j))))$. La preuve est alors facile et elle se fait par induction sur la longueur de ces séquences. □

Maintenant qu'on est sur que les raffinements d'un calcul sont des calculs du raffinement, on peut définir aisément le raffinement *d'une configuration.*

Définition 4.9 *Soient* r *une fonction de raffinement,* $G \in \boldsymbol{A}$, *sans contact, et* $\rho \in Calc(G)$.

Le raffinement de la configuration $Conf(\rho)$ par r *est l'ensemble* $r(Conf(\rho)) \subseteq Conf(r(G))$ *défini comme* $r(Conf(\rho)) \stackrel{def}{=} \{Conf(\sigma) : \sigma \in r(\rho)\}$.

La compositionalité du raffinement sur les configurations est un corollaire direct du Lemme 4.8.

Corollaire 4.10 $Conf(r(G)) = r(Conf(G)) \stackrel{def}{=} \bigcup_{x\in Conf(G)} r(x)$

Il est facile de voir que dans le cas où le STA G et tous les $r(\alpha)$ sont complètement déterminés par l'ensemble des configurations (cas de STA déplié), ce dernier lemme peut être utilisé comme définition du raffinement, puisque $r(Conf(G))$ détermine complètement $r(G)$.

C'est le cas des STA qui sont des SE ($G, r(\alpha) \in ea(SE)$). En plus, dans les SE, on peut abstraire l'ordre partiel dans chaque configuration et les considérer simplement comme ensembles. Un même événement ne peut apparaître qu'une fois dans une exécution, et l'ordre partiel est complètement codé dans l'ensemble des configurations.

Dans ce cas particulier, notre définition du raffinement des configurations (Définition 4.9 et le lemme de compositionalité du raffinement sur les configurations (Lemme 4.10) coïncident avec ceux dans [GG89c] (proposition 1.7), qui caractérisent le raffinement dans les SE. Ceci nous donne le lemme suivant.

Lemme 4.11 *Notre définition du raffinement coïncide avec celle de [GG89c] pour les STA qui sont des SE.*

En fait le même raisonnement sert a prouver un résultat plus général. Notre définition de raffinement coïncide avec celle de [GG89c] pour des SE plus générales: les Structures de Configurations qui sont aussi des STA, parmi lesquelles on trouve les SE stables.

4.3 Le raffinement comme construction catégorique

Nous montrons dans cette section que le raffinement d'actions est compatible avec les morphismes. Notons d'abord que les morphismes préservent le contact, et reflètent donc l'absence des contacts.

Le lemme suivant établit le fait que les morphismes sont "préservés" par le raffinement. En termes de catégories, les fonctions de raffinements sont des endo-foncteurs dans $\mathbf{A}_{sc}$, la "full"sous-catégorie des STA sans contact.

Lemme 4.12 *Soient G_1, $G_2 \in \boldsymbol{A}$ sans contact, le morphisme $h : G_1 \mapsto G_2$ et la fonction de raffinement r. Il existe un morphisme $r(h) = (r(h)_S, r(h)_E) : r(G_1) \mapsto r(G_2)$, où*

$$r(h)_S(s, \{\ldots(e_j, s_j)\ldots\}) \stackrel{def}{=} (h_S(s), \{\ldots(h_E(e_j), s_j)\ldots\})$$
$$r(h)_E(e, a) \stackrel{def}{=} (h_E(e), a)$$

Preuve. Il est clair que $r(h)_S : S_{r(G_1)} \mapsto S_{r(G_2)}$ et $r(h)_E : E_{r(G_1)} \mapsto E_{r(G_2)}$.
On vérifie facilement que $r(h)_S$ préserve l'état initial et que $r(h)_E$ préserve l'étiquetage.

En utilisant la notation introduite au le Lemme 4.5, on a

$$r(h)_D(s, \{\ldots(a_j, (s_j, P_j))\ldots(b_l, (i_{r(l(b_l))}, P_l))\ldots\}) =$$
$$(h_S(s), \{\ldots(h_E(a_j), (s_j, P_j))\ldots(h_E(b_l), (i_{r(l(b_l))}, P_l))\ldots\})$$

En utilisant cette representation, le Lemme 4.5 et les propriétés de h_D, on vérifie aisément que $r(h)_D : Diam(r(G_1)) \mapsto Diam(r(G_2))$, qu'il est surjectif, qu'il préserve les cibles et la taille des ST-états.

Il reste à montrer que $r(h)_D$ reflète la croissance des ST-états.
Soient $r(h)_D(s_1, P_1) = (s_2, P_2)$ et $(s_2, P_2 \cup \widetilde{P_2}) \in Diam(r(G_2))$, où

- $(s_1, P_1) = (s, \{\ldots(a_j, (s_j, P_j))\ldots(b_l, (i_{r(l(b_l))}, P_l))\ldots\})$,
- $(s_2, P_2) = (h_S(s), \{\ldots(h_E(a_j), (s_j, P_j))\ldots(h_E(b_l), (i_{r(l(b_l))}, P_l))\ldots\})$, et
- $(s_2, P_2 \cup \widetilde{P_2}) =$
  $(h_S(s), \{\ldots(h_E(a_j), (s_j, P_j \cup \widetilde{P_j}))\ldots(h_E(b_l), (i_{r(l(b_l))}, P_l \cup \widetilde{P_l}))\ldots$
  $(\widetilde{b_m^2}, (i_{r(l(\widetilde{b_m^2}))}, \widetilde{P_m}))\ldots\})$.

En utilisant que $h_D : Diam(G_1) \mapsto Diam(G_2)$ reflète la croissance des ST-états, que $h_D(s, \{\ldots a_j \ldots b_l \ldots\}) = (h_S(s), \{\ldots h_E(a_j)\ldots h_E(b_l)\ldots\})$ et que $(h_S(s), \{\ldots h_E(a_j)\ldots h_E(b_l)\ldots \widetilde{b_m^2}\ldots\}) \in Diam(r(G_2))$, on sait qu'il existe $(s, \{\ldots a_j \ldots b_l \ldots \widetilde{b_m^1}\ldots\}) \in Diam(G_1)$
t.q. $h_D(s, \{\ldots a_j \ldots b_l \ldots \widetilde{b_m^1}\ldots\}) = (h_S(s), \{\ldots h_E(a_j)\ldots h_E(b_l)\ldots \widetilde{b_m^2}\ldots\})$.

Le ST-état $(s, \{\ldots(a_j, (s_j, P_j \cup \widetilde{P_j}))\ldots(b_l, (i_{r(l(b_l))}, P_l \cup \widetilde{P_l}))\ldots(\widetilde{b_m^1}, (i_{r(l(\widetilde{b_m^1}))}, \widetilde{P_m}))\ldots\})$
est alors un bon candidat.

□

Corollaire 4.13 *Chaque fonction de raffinement r peut être étendue à un endo-foncteur dans la full sous-catégorie des STA sans contact $r : A_{sc} \mapsto A_{sc}$.*

Finalement, nous montrons que lorsque deux fonctions de raffinement r_1 et r_2 peuvent être reliées par des morphismes ($\forall \alpha \in Act.\ \exists h^\alpha : r_1(\alpha) \mapsto r_2(\alpha)$) pour chaque STA G on peut trouver un morphisme $\eta_G : r_1(G) \mapsto r_2(G)$.

En termes de catégories, $\forall \alpha \in Act.\ \exists h^\alpha : r_1(\alpha) \mapsto r_2(\alpha)$ suffit pour assurer l'existence d'une transformation naturelle $\eta : r_1 \dot{\mapsto} r_2$.

Lemme 4.14 *Soient $G \in \mathbf{A}$ sans contact, et deux fonctions de raffinement r_1, r_2 t.q. pour chaque $\alpha \in Act$ il existe $h^\alpha : r_1(\alpha) \mapsto r_2(\alpha)$. Alors, il existe $\eta_G \stackrel{def}{=} (\eta_{GS}, \eta_{GE}) : r_1(G) \mapsto r_2(G)$, où*

$$\begin{aligned} \eta_{GS}(s, \{\ldots(e_i, s_i)\ldots\}) &\stackrel{def}{=} (s, \{\ldots(e_i, h_S^{l(e_i)}(s_i))\ldots\}) \\ \eta_{GE}(e, a) &\stackrel{def}{=} (e, h_E^{l(e)}(a)) \end{aligned}$$

Preuve. Il est clair que $\eta_{GS} : S_{r_1(G)} \mapsto S_{r_2(G)}$ et $\eta_{GE} : E_{r_1(G)} \mapsto E_{r_2(G)}$.

On vérifie facilement que η_{GS} préserve l'état initial et η_{GE} préserve l'étiquetage.

En utilisant la notation introduite dans le Lemme 4.5, et en écrivant $h_E^\alpha(\{e_1 \ldots e_m\})$ au lieu de $\{h_E^\alpha(e_1) \ldots h_E^\alpha(e_m)\}$, on a

$$\begin{aligned} &\eta_{GD}(s, \{\ldots(a_j, (s_j, P_j))\ldots(b_l, (i_{r_1(l(b_l))}, P_l))\ldots\}) \;= \\ &(s, \{\ldots(a_j, (h_S^{l(a_j)}(s_j), h_E^{l(a_j)}(P_j)))\ldots(b_l, (h_S^{l(b_l)}(i_{r_1(l(b_l))}), h_E^{l(b_l)}(P_l)))\ldots\}) \end{aligned}$$

En utilisant cette représentation, le Lemme 4.5 et les propriétés de h_D^α, on vérifie aisément que $\eta_{GD} : Diam(r_1(G)) \mapsto Diam(r_2(G))$, qu'il est surjectif, et qu'il préserve les cibles et la taille des ST-états.

Il reste à montrer que η_{GD} reflète la croissance des ST-états.

Soient $\eta_{GD}(s_1, P_1) = (s_2, P_2)$ et $(s_2, P_2 \cup \widetilde{P_2}) \in Diam(r_2(G))$, où

- $(s_1, P_1) = (s, \{\ldots(a_j, (s_j, P_j))\ldots(b_l, (i_{r_1(l(b_l))}, P_l))\ldots\})$,
- $(s_2, P_2) =$
  $(s, \{\ldots(a_j, (h_S^{l(a_j)}(s_j), h_E^{l(a_j)}(P_j)))\ldots(b_l, (h_S^{l(b_l)}(i_{r_1(l(b_l))}), h_E^{l(b_l)}(P_l)))\ldots\})$, et
- $(s_2, P_2 \cup \widetilde{P_2}) =$
  $(s, \{\ldots(a_j, (h_S^{l(a_j)}(s_j), h_E^{l(a_j)}(P_j) \cup \widetilde{P_j^2}))\ldots(b_l, i_{r_2(l(b_l))}, h_E^{l(b_l)}(P_l) \cup \widetilde{P_l^2}))$
  $\ldots(\widetilde{b_m}, (h_S^{l(\widetilde{b_m})}(i_{r_2(l(\widetilde{b_m}))}), \widetilde{P_m^2}))\}$.

Puisque h_D^α reflète la croissance des ST-états, on sait que

- pour chaque a_j, comme $h_D^{l(a_j)}(s_j, P_j) = (h_S^{l(a_j)}(s_j), h_E^{l(a_j)}(P_j))$
  et $(h_S^{l(a_j)}(s_j), h_E^{l(a_j)}(P_j) \cup \widetilde{P_j^2}) \in Diam(r_2(l(a_j)))$, il existe $\widetilde{P_j^1}$ tel que
  $h_D^{l(a_j)}(s_j, P_j \cup \widetilde{P_j^1}) = (h_S^{l(a_j)}(s_j), h_E^{l(a_j)}(P_j) \cup \widetilde{P_j^2})$.
- pour chaque b_l, comme $h_D^{l(b_l)}(i_{r_1(l(b_l))}, P_l) = (h_S^{l(b_l)}(i_{r_1(l(b_l))}), h_E^{l(b_l)}(P_l))$
  et $(h_S^{l(b_l)}(i_{r_1(l(b_l))}), h_E^{l(b_l)}(P_l) \cup \widetilde{P_l^2}) \in Diam(r_2(l(b_l)))$, il existe $\widetilde{P_l^1}$ tel que
  $h_D^{l(b_l)}(i_{r_1(l(b_l))}, P_l \cup \widetilde{P_l^1}) = (h_S^{l(b_l)}(i_{r_1(l(b_l))}), h_E^{l(b_l)}(P_l) \cup \widetilde{P_l^2})$.

- pour chaque $\widetilde{b_m}$, comme $h_D^{l(\widetilde{b_m})}(i_{r_1(l(\widetilde{b_m}))}, \emptyset) = (i_{r_2(l(\widetilde{b_m}))}, \emptyset)$ et $(i_{r_2(l(\widetilde{b_m}))}, \widetilde{P_m^2}) \in Diam(r_2(l(\widetilde{b_m})))$, il existe $\widetilde{P_m^1}$ tel que $h_D^{l(\widetilde{b_m})}(i_{r_1(l(\widetilde{b_m}))}, \widetilde{P_m^1}) = (h_S^{l(\widetilde{b_m})}(i_{r_1(l(\widetilde{b_m}))}), \widetilde{P_m^2})$.

  Le ST-état $(s, \{\ldots(a_j, (s_j, P_j \cup \widetilde{P_j^1}))\ldots(b_l, (i_{r_1(l(b_l))}, P_l \cup \widetilde{P_l^1}))\ldots(\widetilde{b_m}, (i_{r_1(l(\widetilde{b_m}))}, \widetilde{P_m^1}))\ldots\})$ est alors un bon candidat. □

Corollaire 4.15 *Soient r_1, r_2 des fonctions de raffinements tels que $\forall \alpha \in Act.\ \exists h^\alpha : r_1(\alpha) \mapsto r_2(\alpha)$. Il existe entre les foncteurs r_1 et r_2 de la full-sub-catégorie des STA sans contact une transformation naturelle $\eta : r_1 \dot{\mapsto} r_2$.*

Chapitre 5

Bisimulations

Dans ce chapitre on s'intéresse aux équivalences sémantiques de bisimulations sur les STA et leur compatibilité avec l'opération de raffinement d'actions.

Nous abordons la question en proposant une adaptation aux STA de l'équivalence d'ordre mixte (*mo-équivalence*) de Degano, De Nicola et Montanari [DDM89]. La mo-équivalence a été introduite sur les SE, et elle est bien connue dans la littérature pour sa compatibilité avec le raffinement d'actions. En plus, il existe de multiples caractérisations de la mo-équivalence sur les SE, sur les RP et sur les Arbres Causaux [DD89].

La mo-équivalence est définie comme une relation entre les calculs, qui préserve, en plus de la structure arborescente, l'ordre total dans le calcul et l'ordre partiel reflétant la causalité.

Même si nous montrons que la mo-équivalence est compatible avec le raffinement dans les STA, deux points font que cette équivalence n'est pas complètement satisfaisante. Premièrement elle est définie sur les calculs. Cette propriété est gênante déjà dans les SE, où la configuration et non le calcul est la notion naturelle d'exécution. Dans les STA, la question est autre. Même si les calculs sont notre notion d'exécution de base, ils sont dérivés des états et transitions, qui sont primitifs. Deuxièmement la mo-équivalence préserve l'ordre partiel, qui est bien une notion de base dans les SE. Mais dans les STA l'ordre partiel est une notion dérivée de l'indépendance.

La mo-équivalence est donc définie sur des notions non-primitives dans les STA. Notre effort dans ce chapitre est guidé vers la recherche des équivalences définies sur les notions primitives: indépendance, état, transition.

Encore une dernière remarque. Bien qu'il soit vrai que la mo-équivalence est une relation très fine dans les SE, qui sont des comportements, la situation est tout a fait différente dans les STA, qui sont des systèmes. On peut donc trouver dans les STA des équivalences intéressantes plus fines que la mo-équivalence.

A notre connaissance, la seule proposition dans la littérature d'une bisimulation en termes d'indépendance est celle de Mukund et Nielsen [MN92] (ici appelée *cI-bisimulation*) préservant l'indépendance le long des calculs. Nous montrons que la cI-bisimulation est compatible avec le raffinement, qu'elle est strictement plus fine que la mo-équivalence dans les STA, et que le deux équivalences coïncident sur les SE.

Nous nous intéressons ensuite à des bisimulations préservant l'indépendance définies directement sur les STA, et non sur ces calculs. Nous introduisons deux nouvelles équivalences appelées gI-bisimulation et dI-bisimulation.

La *gI-bisimulation* (pour "globaly Indépendance preserving bisimulation") est définie à partir d'une relation sur les états, comme dans la bisimulation forte, et une autre sur les événements ($\sim \subseteq E_1 \times E_2$) qui préserve les étiquettes et l'indépendance. Une transition étiquetée e_1 dans un STA doit être imitée dans l'autre par un transition étiquetée e_2 avec $e_1 \sim e_2$.

Malheureusement la gI-bisimulation n'est pas une congruence pour le raffinement: la préservation "globale" de l'indépendance est trop fine et pas assez subtile pour être préservée par le raffinement.

Nous proposons alors la *dI-bisimulation*, définie de façon standard à partir de nos morphismes, qui est compatible avec le raffinement. La dI-bisimulation est moins fine que la gI-bisimulation, et préserve l'indépendance des événements d'une façon plus subtile.

La dI-bisimulation ne préserve pas les pomsets. Elle n'implique donc pas la mo-équivalence.

Dans les STA les quatre équivalences sont différentes. Si on se restreint aux SE, la mo-équivalence et la cI-bisimulation coïncident. A notre connaissance, les deux nouvelles ne correspondent sur les SE à aucune autre équivalence dans la littérature. Sur les ST, toutes les équivalences étudiées dans ce chapitre coïncident avec la bisimulation forte sur le ST sous-jacent.

Ce chapitre est organisé en trois sections: La première contient les définitions des bisimulations (Section 5.1). La seconde est consacré aux liens entre les équivalences (Section 5.2), et la dernière à l'étude des propriétés de congruence (Section 5.3).

5.1 Bisimulations sur les STA

Dans cette section nous donnons notre extension de la mo-équivalence aux STA, nous rappelons la cI-bisimulation, et nous introduisons la gI-bisimulation et la dI-bisimulation. Nous rappelons aussi la bisimulation classique, appelée "bisimulation d'entrelacement" dans ce contexte, afin de compléter le cadre des équivalences de bisimulations sur les STA.

Définition 5.1 *Soient $G_1, G_2 \in \boldsymbol{A}$. Une relation $R \subseteq Calc(G_1) \times Calc(G_2)$ est une* équivalence d'ordre mixte (mo-équivalence) entre G_1 et G_2 *si $\lambda R \lambda$, et pour tous $\sigma_1\ R\ \sigma_2$,*

- $\forall j,k \in \{1 \ldots |\sigma_1|\}$, $j \leq_{\sigma_1} k \Leftrightarrow j \leq_{\sigma_2} k$ *et* $l_1(\sigma_1(j)) = l_2(\sigma_2(j))$.
- *s'il existe* $\rho_1 = \sigma_1 e_1 \in Calc(G_1)$ *alors il existe* $\rho_2 = \sigma_2 e_2 \in Calc(G_2)$ *avec* $\rho_1 R \rho_2$.
- *vice versa, s'il existe* $\rho_2 = \sigma_2 e_2 \in Calc(G_2)$ *alors il existe* $\rho_1 = \sigma_1 e_1 \in Calc(G_1)$ *avec* $\rho_1 R \rho_2$.

On note $R : G_1 \approx_{mo} G_2$, *lorsque* R *est une mo-équivalence entre* G_1 *et* G_2, *et* $G_1 \approx_{mo} G_2$ *si il existe* R *telle que* $R : G_1 \approx_{mo} G_2$.

Notons qu'une définition équivalente de la relation $\approx_{mo}$ peut être donnée en demandant que co_σ soit préservé au lieu de $\leq_\sigma$.

La bisimulation sur les STA de [MN92], appelée ici *cI-bisimulation* est obtenue en renforçant les conditions demandées pour $\approx_{mo}$: On demande de préserver I, au lieu de co_σ.

Définition 5.2 *Soient* $G_1, G_2 \in \boldsymbol{A}$. *Une relation* $R \subseteq Calc(G_1) \times Calc(G_2)$ *est une* cI-bisimulation entre G_1 et G_2 *si* $\lambda R \lambda$, *et pour tout* $\sigma_1\ R\ \sigma_2$,

- $\forall j, k \in \{1 \ldots |\sigma_1|\}$, $\sigma_1(j) I_1 \sigma_1(k) \Leftrightarrow \sigma_2(j) I_2 \sigma_2(k)$ *et* $l_1(\sigma_1(j)) = l_2(\sigma_2(j))$.
- *s'il existe* $\rho_1 = \sigma_1 e_1 \in Calc(G_1)$ *alors il existe* $\rho_2 = \sigma_2 e_2 \in Calc(G_2)$ *avec* $\rho_1 R \rho_2$.
- *vice versa, s'il existe* $\rho_2 = \sigma_2 e_2 \in Calc(G_2)$ *alors il existe* $\rho_1 = \sigma_1 e_1 \in Calc(G_1)$ *avec* $\rho_1 R \rho_2$.

On note $R : G_1 \approx_{cI} G_2$, *lorsque* R *est une cI-bisimulation entre* G_1 *et* G_2, *et* $G_1 \approx_{cI} G_2$ *si il existe* R *telle que* $R : G_1 \approx_{cI} G_2$.

$\approx_{cI}$ respecte l'indépendance le long des calculs (d'où l'origine du c dans cI-bisimulation). Une condition plus forte est que cette indépendance soit préservée globalement. Elle permet la définition de la *globally Independence preserving bisimulation (gI-bisimulation).*

Définition 5.3 *Soient* $G_1, G_2 \in \boldsymbol{A}$. *Une* gI-bisimulation entre G_1 et G_2 *est un couple de relations* $(R, \sim)$, $R \subseteq Q_1 \times Q_2$ *et* $\sim \subseteq E_1 \times E_2$ *tel que*

- $\sim$ *préserve l'étiquetage et la relation d'indépendance (i.e.* $e_1 \sim e_2$ *implique* $l_1(e_1) = l_2(e_2)$ *et* $a_1 \sim a_2$ *et* $b_1 \sim b_2$ *implique* $a_1 I_1 b_1 \Leftrightarrow a_2 I_2 b_2$).
- R *préserve l'état initial* $(i_1 R i_2)$ *et pour tout* $s_1 R s_2$,
  - *si* $s_1 \xrightarrow{e_1}_1 u_1$, *il existe* $s_2 \xrightarrow{e_2}_2 u_2$ *avec* $e_1 \sim e_2$ *et* $u_1 R u_2$.
  - *vice versa, si* $s_2 \xrightarrow{e_2}_2 u_2$, *il existe* $s_1 \xrightarrow{e_1}_1 u_1$ *avec* $e_1 \sim e_2$ *et* $u_1 R u_2$.

On note $(R, \sim) : G_1 \approx_{gI} G_2$, *lorsque* $(R, \sim)$ *est une gI-bisimulation entre* G_1 *et* G_2, *et* $G_1 \approx_{gI} G_2$ *si il existe* $(R, \sim)$ *telle que* $(R, \sim) : G_1 \approx_{gI} G_2$.

Maintenant, on introduit l'équivalence engendrée par les morphismes des STA.

Définition 5.4 *Etant donnés* $G_1, G_2 \in \boldsymbol{A}$, G_1 *et* G_2 *sont* dI-bisimilaires *(noté* $G_1 \approx_{dI} G_2$*) ssi il existe un* $G \in \boldsymbol{A}$ *et deux morphismes* $j^1 : G \mapsto G_1, j^2 : G \mapsto G_2$.

Lemme 5.5 $\approx_{mo}$, $\approx_{cI}$, $\approx_{gI}$ *et* $\approx_{dI}$ *sont des relations d'équivalence sur les STA.*

Preuve. Le seul point difficile est la transitivité de $\approx_{dI}$: il faut utiliser le Lemme 2.8. □

Remarque 3 *Notons que l'union des mo-équivalences est une mo-équivalence, et l'union des cI-bisimulation est une cI-bisimulation.*

Par contre, l'union des gI-bisimulations n'est pas forcement une gI-bisimulation, même dans les SE, comme le montre l'exemple dans la Figure 5.1. Le couple $(Id|_Q, Id|_E)$ *est une auto gI-bisimulation dans G, et aussi* $(Z, \sim)$, *où* $Z = \{(i,i), (s_1,s_2), (s_2,s_1), (u,u)\}$, $\sim = \{(a,b), (b,a)\}$. *Mais* $(Id|_E \cup \sim)$ *ne préserve pas l'indépendance, donc l'union de ces deux gI-bisimulations n'est pas une gI-bisimulation.*

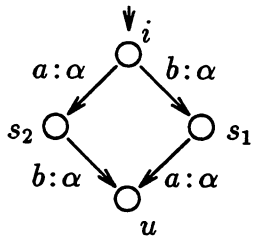


Figure 5.1: L'union des gI-bisimulations n'est pas forcement une gI-bisimulation (aIb).

Afin de permettre les comparaisons, nous rappelons ici la définition de la bisimulation classique sur les ST, appelée dans notre contexte *bisimulation d'entrelacement.*

Définition 5.6 *Etant donnés $G_1, G_2 \in \mathbf{A}$, une* bisimulation d'entrelacement entre G_1 et G_2 *est une relation $R \subseteq Q_1 \times Q_2$ qui préserve l'état initial ($i_1 R i_2$) et telle que pour tous $s_1 R s_2$,*

- *si $s_1 \xrightarrow{e_1}_1 u_1$, il existe $s_2 \xrightarrow{e_2}_2 u_2$ avec $l_1(e_1) = l_2(e_2)$ et $u_1 R u_2$.*
- *vice versa, si $s_2 \xrightarrow{e_2}_2 u_2$, il existe $s_1 \xrightarrow{e_1}_1 u_1$ avec $l_1(e_1) = l_2(e_2)$ et $u_1 R u_2$.*

On note $R : G_1 \approx_i G_2$, lorsque R est une bisimulation d'entrelacement entre G_1 et G_2, et $G_1 \approx_i G_2$ si il existe R telle que $R : G_1 \approx_i G_2$.

5.2 Classification des bisimulations

Les liens entre les bisimulations sont établis dans le théorème suivant, et résumés dans la Figure 5.2.

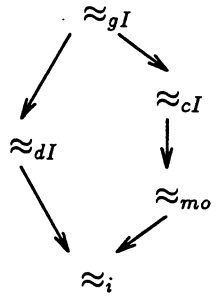


Figure 5.2: Hiérarchie des équivalences sémantiques sur les STA

Théorème 5.7 *Pour tous $G_1, G_2 \in \mathbf{A}$,*

1	$(R, \sim) : G_1 \approx_{gI} G_2$ *(R et R^{-1} complètes)*	*implique*	$G_1 \approx_{dI} G_2$
2	$G_1 \approx_{dI} G_2$	*implique*	$G_1 \approx_i G_2$
3	$G_1 \approx_{gI} G_2$	*implique*	$G_1 \approx_{cI} G_2$
4	$R : G_1 \approx_{cI} G_2$	*implique*	$R : G_1 \approx_{mo} G_2$
5	$R : G_1 \approx_{cI} G_2$	*implique*	$\bar{R} : G_1 \approx_i G_2$

où $\bar{R} \stackrel{def}{=} \{(cible(\sigma), cible(\rho)) : \sigma R \rho\}$.

Preuve. **1.** Il suffit de trouver $G_3 \in \mathbf{A}$, $j^1 : G_3 \mapsto G_1$ et $j^2 : G_3 \mapsto G_2$. Le bon candidat est $G_3 \stackrel{\text{def}}{=} \langle R, (i_1, i_2), \sim, I_3, \rightarrow_3, l_3 \rangle$ où $(a_1, a_2) I_3 (b_1, b_2) \stackrel{\text{def}}{\Leftrightarrow} a_1 I_1 b_1 (\Leftrightarrow a_2 I_2 b_2)$, $(s_1, s_2) \stackrel{(e_1,e_2)}{\rightarrow}_3 (u_1, u_2) \stackrel{\text{def}}{\Leftrightarrow} s_1 \stackrel{e_1}{\rightarrow}_1 u_1 \wedge s_2 \stackrel{e_2}{\rightarrow}_2 u_2$ et $l_3((e_1, e_2)) \stackrel{\text{def}}{=} l_1(e_1)(= l_2(e_2))$. Les morphismes j^1 et j^2 sont les projections gauche et droite dans les états et les événements.

2. Conséquence directe de notre définition de $\approx_{dI}$ et la caractérisation de la bisimulation d'entrelacement par des morphismes (voir [AC88]).

3. Soit $(R, \sim) : G_1 \approx_{gI} G_2$ et $\bar{R} \subseteq Calc(G_1) \times Calc(G_2)$ la plus petite relation qui satisfait $\lambda \bar{R} \lambda$ et pour chaque $\rho \bar{R} \sigma$, $a \sim b$, $cible(\rho a) R cible(\rho b)$ implique $\rho a \bar{R} \rho b$. Reste à prouver que $\bar{G} : G_1 \approx_{cI} G_2$.

4 est facile, et **5** est bien connu. □

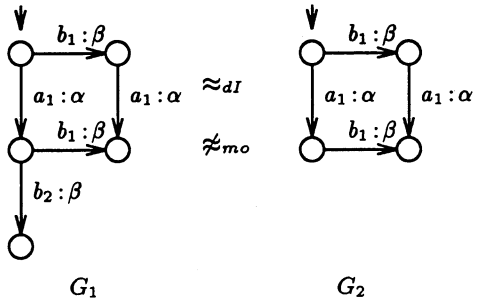


Figure 5.3: La dI-bisimulation n'implique pas la mo-equivalence ($a_1 I b_1$).

Nous montrons qu'aucune flèche ne peut être ajoutée a la Figure 5.2 (hormis celles obtenues par transitivité).

- La Figure 5.3 nous montre que $\approx_{dI}$ n'implique pas $\approx_{mo}$, même dans le cas des SE non auto-concurrentes[1] (pour voir que $G_1 \approx_{dI} G_2$ il suffit d'observer qu'il existe un morphisme $j : G_1 \mapsto G_2$ avec $j_E(b_2) = b_1$).

- L'exemple classique de la "loi d'absorption" (Figure 5.4) nous montre que $\approx_{cI}$ n'implique pas $\approx_{dI}$.

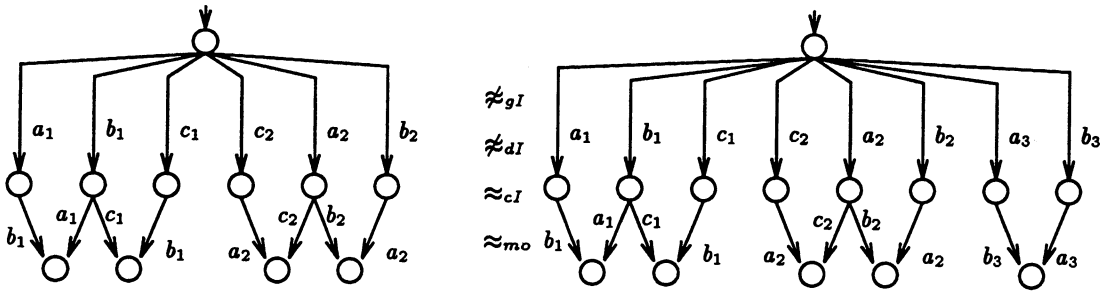
  Pour vérifier $G_3 \not\approx_{dI} G_4$ il suffit de montrer que pour tout $G \in \mathbf{A}$ tel qu'il existe $j^4 : G \mapsto G_4$, il n'existe pas un morphisme $j^3 : G \mapsto G_3$.

  Etant donné que j^4 reflète les ST-états, il doit exister $(i, \{a, b\}) \in Diam(G)$ tel que $j^4_D(i, \{a, b\}) = (i_2, \{a_3, b_3\})$. Puisque j^4 préserve les transitions, il n'y a pas de transitions étiquetées c sortant de s_a or s_b (ou $i \stackrel{a}{\rightarrow} s_a$, $i \stackrel{b}{\rightarrow} s_b$). Alors, on ne peut pas trouver un morphisme $j^3 : G \mapsto G_3$ puisque dans ce cas une c transition doit partir de $j^3(s_a)$ ou $j^3(s_b)$, transition que j^3 ne peut pas refléter.

- L'exemple de la Figure 5.4 nous permet aussi de voir que $\approx_{cI}$ n'implique pas $\approx_{gI}$. Pour vérifier $G_3 \not\approx_{gI} G_4$ il suffit d'observer que pour simuler dans G_3 les transitions étiquetées a_3 et b_3 de G_2 on doit prendre a_1 et b_2. Alors il faut considérer $a_1 \sim a_3$, $b_2 \sim a_3$ qui ne préserve pas l'indépendance: $a_3 I_2 b_3, \neg a_1 I_1 b_2$.

- L'exemple de la Figure 5.5 nous montre que $\approx_{mo}$ n'implique pas $\approx_{cI}$ dans le cas général.

[1] Une SE est auto-concurrente s'il existe un couple d'événements $e_1 \, coe_2$, avec le même label



$G_3 = ((\alpha + \gamma)|\beta) + (\alpha|(\gamma + \beta))$, $G_4 = G_1 + (\alpha|\beta)$.
$(a_i I b_i, c_1 I b_1, c_2 I a_2, l(a_i) = \alpha, l(b_i) = \beta, l(c_i) = \gamma)$

Figure 5.4: La loi d'aborption ($G_3 = G_4$) est valide pour $\approx_{mo}$ et $\approx_{cI}$, mais non valide pour $\approx_{gI}$ et $\approx_{dI}$.

- Finalement, la figure 5.6 nous montre que $\approx_i$ est strictement plus fine que toutes les autres équivalences considérées ci-dessous.

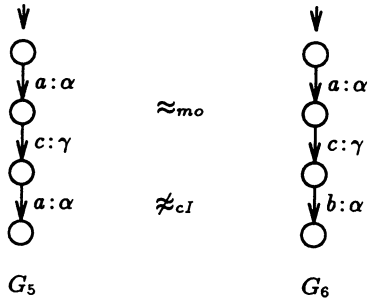


Figure 5.5: La mo-équivalence n'implique pas la cI-bisimulation (aIb)

En considérant les équivalences sur les SE, on voit que nos contre-exemples sont encore valides sauf celui de la Figure 5.5. Mise à part la flèche que relie $\approx_{cI}$ et $\approx_{mo}$, les autres implications continuent à être strictes sur le SE. Le lemme suivant nous montre que $\approx_{mo}$ et $\approx_{cI}$ coïncident sur une large famille des STA

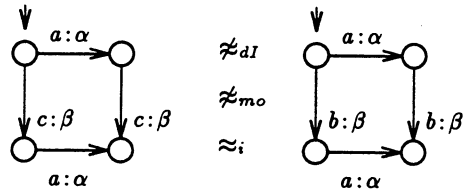


Figure 5.6: La bisimulation d'entrelacement est strictement plus fine que les autres. (aIb).

Lemme 5.8 *Pour tous $G_1, G_2 \in \boldsymbol{A}$ tel que pour chaque $\sigma \in Calc(G_n)$, $j,k = 1..$ $\mid \sigma \mid$, $\sigma(i) I_n \sigma(j) \Rightarrow ico_\sigma j$, $R : G_1 \approx_{mo} G_2$ implique $R : G_1 \approx_{cI} G_2$.*

La preuve du lemme est triviale, et son corollaire plus intéressant est

Corollaire 5.9 *Pour tous $G_1, G_2 \in \boldsymbol{A}$ qui sont des SE, $R : G_1 \approx_{mo} G_2$ ssi $R : G_1 \approx_{cI} G_2$.*

Finalement, le lemme suivant nous montre que toute la hiérarchie s'effondre sur la bisimulation d'entrelacement dans le cas des ST.

Lemme 5.10 *Pour tous les STA qui sont des ST $\approx_{gI} \Leftrightarrow \approx_{cI} \Leftrightarrow \approx_{mo} \Leftrightarrow \approx_i$.*
Si tous les états de $G_1, G_2 \in \boldsymbol{A}$ sont atteignables, $G_1 \approx_{dI} G_2 \Leftrightarrow G_1 \approx_i G_2$.

5.3 Bisimulations et raffinement

La notion formelle de *compatibilité* d'une équivalence avec le raffinement est la propriété de *congruence*: une équivalence est une congruence si ses classes d'équivalences sont préservées par le raffinement.

Définition 5.11 *Une relation d'équivalence $\approx$ sur $\boldsymbol{A}$ est une* congruence par rapport au raffinement d'actions *ssi pour chaque $G_1, G_2 \in \boldsymbol{A}$ sans contact, $G_1 \approx G_2$ et chaque couple de fonctions de raffinement r_1, r_2 telles que $\forall \alpha \in Act.\ r_1(\alpha) \approx r_2(\alpha)$, $r(G_1) \approx r(G_2)$*

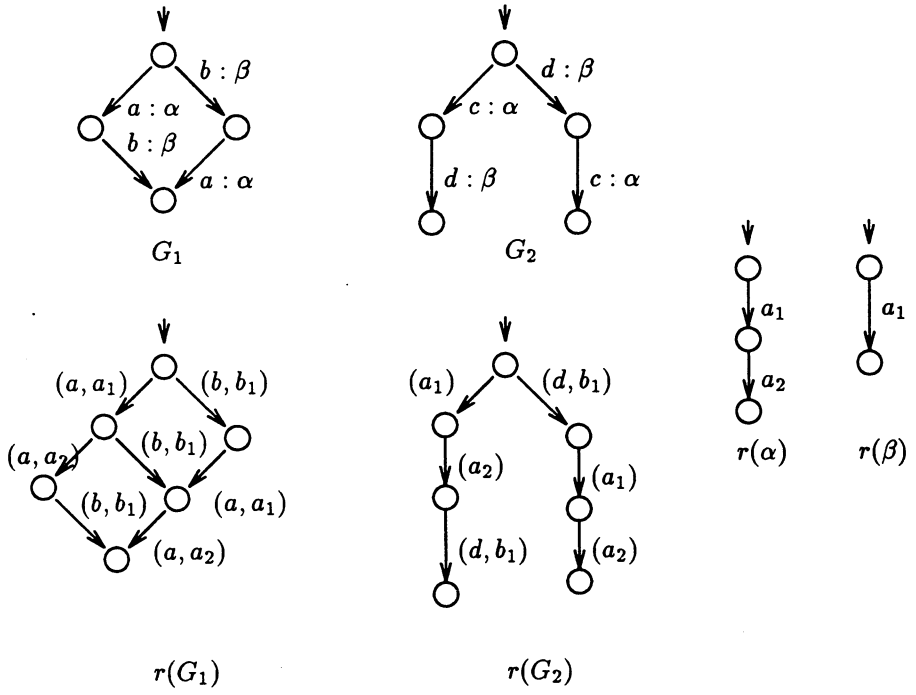


Figure 5.7: La bisimulation d'entrelacement n'est pas une congruence: $G_1 \approx_i G_2$ et $r(G_1) \not\approx_i r(G_2)$ (aIb)

La bisimulation d'entrelacement n'est pas une congruence par rapport au raffinement d'actions. L'exemple classique apparaît dans la Figure 5.7.

On peut prouver aussi facilement à l'aide de l'exemple de la Figure 5.8 que $\approx_{gI}$ n'est pas une congruence pour le raffinement. Dans cette figure, $G_1 \approx_{gI} G_2$ mais $r(G_1) \not\approx_{gI} r(G_2)$.

Pour obtenir une gI-bisimulation entre $r(G_1)$ et $r(G_2)$ et simuler (a,d) par (c,d), et (b,e) par (c,e). Mais $(a,d) \sim (c,d), (b,e) \sim (c,e)$ ne préserve pas l'indépendance: $(c,d) I_{r(G_2)} (c,e)$ et $\neg(a,d) I_{r(G_1)} (b,d)$.

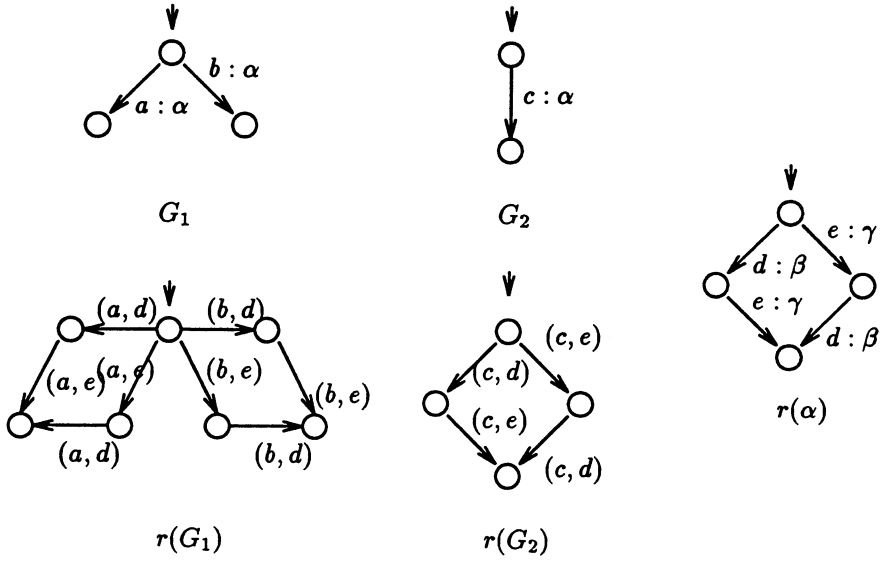


Figure 5.8: La gI-bisimulation n'est pas une congruence: $G_1 \approx_{gI} G_2$ et $r(G_1) \not\approx_{gI} r(G_2)$ (eId)

Les autres équivalences étudiées dans ce chapitre sont des congruences par rapport au raffinement d'actions.

Théorème 5.12 *$\approx_{mo}$, $\approx_{cI}$ et $\approx_{dI}$ sont des congruences pour le raffinement d'actions.*

Preuve. • $\approx_{mo}, \approx_{cI}$.

La preuve utilise une définition auxiliaire: Pour (e,a) dans un calcul $\sigma(e,a) \in Calc(r(G))$ quelle occurrence de e dans $r^{-1}(\sigma)$ est en train d'être raffinée. Soit $\sigma(e,a) \in Calc(r(G))$, on définit $pos(\sigma(e,a))$ comme la position dans $r^{-1}(\sigma(e,a))$ de la dernière occurrence de e.

Nous présentons la preuve pour le cas $r_1 = r_2$, qui nous permet de montrer la démarche, d'ailleurs classique. La version générale est aussi classique, et peut se prouver sans grande difficulté.

Soient $G_1, G_2 \in \mathbf{A}$, r une fonction de raffinement et $R : Calc(G_1) \times Calc(G_2)$ donnés. On pose $\bar{R} \subseteq Calc(r(G_1)) \times Calc(r(G_2))$ comme

$$\sigma = \langle \ldots (e_j, a_j) \ldots \rangle \; \bar{R} \; \sigma' = \langle \ldots (e'_j, a'_j) \ldots \rangle \quad \overset{\text{def}}{\Leftrightarrow}$$
$$\begin{cases} r^{-1}(\sigma) R r^{-1}(\sigma') & \\ a_i = a'_i & \text{pour tout } i \in 1 \ldots n \\ pos(\sigma(1) \ldots \sigma(i)) = pos(\sigma'(1) \ldots \sigma'(i)) & \text{pout tout } i \in 1 \ldots n \end{cases}$$

On vérifie facilement que

- si $R : G_1 \approx_{mo} G_2$ alors $\bar{R} : r(G_1) \approx_{mo} r(G_2)$,
- si $R : G_1 \approx_{cI} G_2$ alors $\bar{R} : r(G_1) \approx_{cI} r(G_2)$.

• $\approx_{dI}$.

Voir Figure 5.9.

D'après la définition de la dI-bisimulation, $\forall \alpha \in Act.\ r_1(\alpha) \approx_{dI} r_2(\alpha)$ implique qu'il existe r_3 telle que $\forall \alpha \in Act.\ \exists j^1_\alpha : r_3(\alpha) \mapsto r_1(\alpha) \wedge \exists j^2_\alpha : r_3(\alpha) \mapsto r_2(\alpha)$. Les couples (r_3, r_1) et (r_3, r_2) vérifient les conditions du Lemme 4.14.

Prenons $G_3 \in \mathbf{A}$ sans contact, $j^1 : G_3 \mapsto G_1$, $j^2 : G_3 \mapsto G_2$.

D'après le Lemme 4.12 il existe des morphismes $r_3(j^1) : r_3(G_3) \mapsto r_3(G_1)$ et $r_3(j^2) : r_3(G_2) \mapsto r_3(G_2)$.

D'après le Lemme 4.14 il existe des morphismes $\eta_{G_1}^{r_3,r_1} : r_3(G_1) \mapsto r_1(G_1)$, et $\eta_{G_2}^{r_3,r_2} : r_3(G_2) \mapsto r_2(G_2)$.

Donc il suffit de prendre $r_3(G_3)$ et les morphismes $r_3(j^1) \circ \eta_{G_1}^{r_3,r_1}$ et $r_3(j^2) \circ \eta_{G_2}^{r_3,r_2}$ pour prouver que $r_1(G_1) \approx_{dI} r_2(G_2)$. □

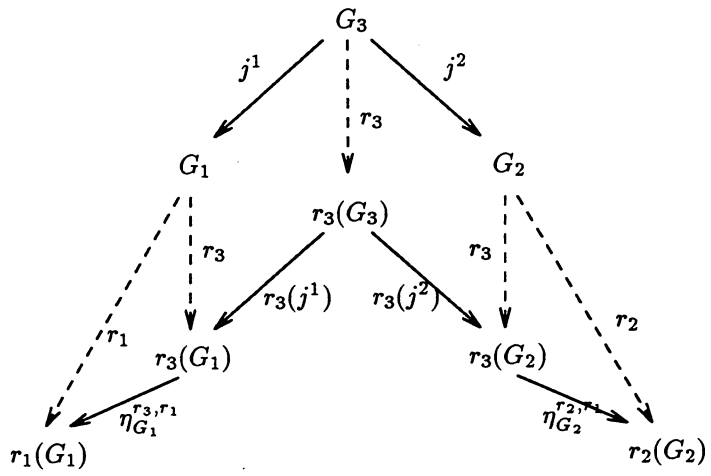


Figure 5.9: La preuve du théorème 5.12

Chapitre 6

Sémantique Opérationnelle Structurée

Dans ce chapitre on s'intéresse à l'utilisation de la *Sémantique Opérationnelle Structurée* [Plo81] pour donner des sémantiques aux *Langages de Description de Processus* dans les STA. Les résultats de cette étude ont un intérêt pratique. Ils permettent principalement d'obtenir:

- une méthode générale pour construire une sémantique des Langages de Description de Processus dans les STA.
- une caractérisation d'une large famille de combinateurs pour lesquels les bisimulations introduites dans les Chapitre 5 sont des congruences.

Une retombée théorique de ce résultat est la preuve que le raffinement d'actions, tel qu'il apparaît dans la Définition 4.3, ne peut être représenté par des combinateurs de cette famille.

Comme applications particulières, on obtient une sémantique d'une variante du langage CCS [Mil80, Mil89] dans les STA, et les preuves de congruence.

Les *Langages de Description des Processus (LDP)* [Mil80, Mil89, Hoa85, BW90] sont des langages de spécification souvent très simples, permettant d'étudier les problèmes de base liés à la sémantique des systèmes réactifs. En général, ils ne s'intéressent pas aux thèmes "orthogonaux" aux aspects réactifs, comme par exemple les types de données.

Plotkin [Plo81] a proposé une méthode générale pour définir la sémantique opérationnelle des langages de programmation dans les ST. Cette méthode, connue sous le nom de *SOS* (pour "Structural Operational Semantics"), consiste à définir des règles structurelles sur les termes du langage. Les règles forment un système de déduction naturelle qui permet de prouver les transitions d'états d'un ST, où les états sont les termes du LDP.

De nos jours, la méthode SOS est très largement utilisée, et on compte de nombreux langages munis d'une sémantique SOS, e.g. pour CCS voir [Mil80, Mil89].

Badouel et Darondeau [BD92] proposent une méthode générale pour extraire à partir des preuves de transitions, l'information de transitions indépendantes. Ils considèrent des spécifications SOS d'un format particulier (appelé *SOS de base*), des quelles ils engendrent des Automates de Traces [Sta89b, Sta89a]. A notre connaissance, la technique pour extraire une information à partir des preuves de transitions est due à [BC88], où une sémantique de non-entrelacement pour un sous-ensemble de CCS est donnée. Depuis, cette approche s'est répandue dans la littérature (voir par exemple [CGM90, Fer90, Yan93]).

Certaines sémantiques des LDP sur les STA basées sur des techniques dénotationnelles ou sur des variantes de la technique de Plotkin sont proposées dans la littérature. Dans [Bed87] et dans [WN92], des LDP assez généraux sont munis d'une sémantique sur des modèles très proches des STA (dans le premier cas, sur les *STA schématiques*, dans l'autre, sur les STI). En utilisant ces deux techniques, Boudol et Castellani [BC90] présentent plusieurs sémantiques équivalentes pour CCS, dont une sur les STA. Dans tous ces cas, la sémantique opérationnelle est obtenue en suivant les techniques de Plotkin, par codage de l'information d'indépendance dans la preuve des transitions.

En suivant une approche légèrement différente, [MN92] présentent la sémantique d'une version restreinte de CCS sur les STA, en définissant l'indépendance à partir de la notion de localité [BC91, BCHK92, Ace91, MY92].

Un des travaux les plus intéressants sur les SOS se trouve dans [GV88] où les auteurs démontrent que la bisimulation classique sur les ST est une congruence pour tous les combinateurs décrits au moyen de règles SOS respectant un certain format syntaxique. Ce résultat établit une nouvelle technique de preuve: pour prouver la congruence pour un combinateur, il suffit de montrer qu'il peut être décrit au moyen de règles SOS respectant ce format.

Ici nous montrons comment engendrer des STA à partir de spécifications SOS de base. Puis, nous établissons des variantes du théorème de [GV88] en démontrant que la mo-équivalence, la cI-bisimulation et la gI-bisimulation sont des congruences pour tous les combinateurs d'une large famille de LDP, décrits par des spécifications SOS de base.

Le définition de congruence pour la gI-bisimulation n'est pas standard, puisqu'on doit tenir compte du fait que l'union des gI-bisimulations n'est pas, dans le cas général, une gI-bisimulation.

Quelques efforts ont été fait dans le but de donner une caractérisation de la classe des STA représentables par des SOS-spécifications. Malheureusement, nous ne sommes pas parvenu à des résultats concrets.

Ce chapitre est organisé en trois sections. Dans la Section 6.1 nous donnons la définition des SOS de base, et nous montrons comment en dériver des STA. Dans la Section 6.2 nous établissons nos résultats de congruence. Nous terminons ce chapitre dans la Section 6.3 par une application au langage CCS.

6.1 Des STA dérivés des Spécifications SOS

Dans cette section, nous rappelons la technique SOS, dans un cadre simplifié de formats de règles, appelé *SOS de base* [BD92] (ou encore SOS dans la suite). En suivant l'approche de [BD92], nous montrons comment construire des STA à partir de spécifications SOS de base.

Une spécification SOS est donnée par un langage de termes et un ensemble de règles. Plotkin montre comment associer un ST à cette spécification, où les états sont les termes du langage, et les transitions sont celles que l'on peut prouver à partir du système de règles. Pour obtenir des STA, il faut aussi construire un ensemble d'événements, et une relation d'indépendance.

Suivant [BD92], les événements s'obtiennent par abstraction sur les preuves. Le point technique le plus délicat de cette construction est le choix de cette abstraction. Le choix proposé dans [BD92] consiste à abstraire de chaque règle r de la forme

$$\frac{u_{j_1} \stackrel{\alpha_{j_1}}{\rightarrow} v_{j_1} \quad \ldots \quad u_{j_k} \stackrel{\alpha_{j_k}}{\rightarrow} v_{j_k}}{f(u_1 \ldots u_n) \stackrel{\alpha}{\rightarrow} g(v_1 \ldots v_n)}$$

un opérateur d'arité n sur les preuves, noté $\rho_r = \langle\langle \alpha_1, \ldots \alpha_n, \alpha\rangle, g\rangle$, avec $\alpha_j = *$ quand $j \notin \{j_1 \ldots j_k\}$.

Les *événements* sont les termes clos construit sur ces opérateurs.

Les spécifications SOS de base peuvent être représentées par des *SOS-automates*, ou graphes étiquetés ayant comme états les combinateurs du LDP et comme étiquettes les opérateurs sur les preuves.

La *relation d'indépendance* entre les événements (ou preuves) est définie (d'une façon presque inductive) à partir d'une relation semblable sur les étiquettes du SOS-automate (opérateurs sur les preuves).

Dans la suite, nous supposons donnés un ensemble $V = \{u, v, u_1, v_1, \ldots\}$ de *variables* et un *alphabet d'actions* $Act = \{\alpha, \beta, \alpha_1, \beta_1 \ldots\}$.

Définition 6.1 *Soit une signature (à une sorte)* $\Sigma = (F, ari)$ *où* F *est un ensemble de* noms de fonctions *disjoint de* V, *et* $ari : F \rightarrow N$ *est l'arité des noms de fonctions de* F.

- *Un terme* t *sur la signature* Σ, *ou encore un* Σ-terme, *est une fonction partielle* $t : (N_+)^* \mapsto (F \cup V)$. *Le domaine de* t, *noté* $Dom(t)$, *est un ensemble de chemins (codés comme séquences d'entiers positifs), fermé par préfixe, contenant le chemin vide* λ, *et tel que* $\forall s \in Dom(t)$, $ari(t(s)) = k$ *implique* $s.j \in Dom(t)$ *ssi* $j \in \{1, \ldots, k\}$. $t[s]$ *dénote le sous-arbre de* t *à la position* s.
- T_Σ *dénote l'ensemble des termes* clos *(i.e. ne contenant pas de variables) sur la signature* Σ.

Définition 6.2 *Une* Spécification Opérationnelle Structurée de base (SOS-spécification) *est une structure* $P = (\Sigma, R)$ *où* $\Sigma = (F, ari)$ *est une signature (les éléments de* F *sont parfois appelés des* combinateurs*) et* R *est un ensemble de règles de la forme*

$$\frac{u_{j_1} \stackrel{\alpha_{j_1}}{\rightarrow} v_{j_1} \quad \ldots \quad u_{j_k} \stackrel{\alpha_{j_k}}{\rightarrow} v_{j_k}}{f(u_1 \ldots u_n) \stackrel{\alpha}{\rightarrow} g(v_1 \ldots v_n)}$$

$1 \leq j_1 < j_2 < \ldots < j_k \leq n$, *où* $\{u_1, \ldots u_n, v_{j_1}, \ldots v_{j_k}\}$ *sont des variables distinctes,* $\alpha, \alpha_j \in Act$, $f, g \in F$ *avec* $ari(f) = ari(g) = n$. *Les expressions* $u_{j_l} \xrightarrow{\alpha_{j_l}} v_{j_l}$ *s'appellent les* prémisses *de la règle, et* $f(u_1 \ldots u_n) \xrightarrow{\alpha} g(v_1 \ldots v_n)$ *sa* conclusion.

Les règles sans prémisses sont appelées des axiomes.

Définition 6.3 *Soit* $P = (\Sigma, R)$ *une SOS-spécification. A chaque* $r \in R$ *de la forme*

$$\frac{u_{j_1} \xrightarrow{\alpha_{j_1}} v_{j_1} \quad \ldots \quad u_{j_k} \xrightarrow{\alpha_{j_k}} v_{j_k}}{f(u_1 \ldots u_n) \xrightarrow{\alpha} g(v_1 \ldots v_n)}$$

on associe un opérateur n-aire sur les preuves $\rho = \langle(\alpha_1 \ldots \alpha_n, \alpha), g\rangle$ *où* $\alpha_j = *$ *ssi* $j \notin \{j_1 \ldots j_k\}$. *On pose* $act_j(\rho) \stackrel{def}{=} \alpha_j$, $act(\rho) \stackrel{def}{=} \alpha$, $op(\rho) \stackrel{def}{=} g$ *et* $ari(\rho) = n$.

En rajoutant la constante $*$ *à l'ensemble des opérateurs (avec* $ari(*) = 0$ *et* $act(*) = *$*) on obtient la signature* R_*. *Une* preuve schématique *de transition dans* P *est un* R_**-terme* e *qui satisfait la condition de* cohérence *suivante:*

$$\forall s \in Dom(e).\ (s.j \in Dom(e) \Rightarrow act_j(e(s)) = act(e(s.j))$$

On note E_P *l'ensemble des preuves schématiques de* P. $D(e) \stackrel{def}{=} \{s \in Dom(e) : e(s) \neq *\}$, *appelé* la partie active *du domaine de* e.

Dans les SOS-automates, une règle r ayant comme conclusion $f(u_1 \ldots u_n) \xrightarrow{\alpha} g(v_1 \ldots v_n)$ et comme opérateur associé ρ_r est représentée par une transition $f \xrightarrow{\rho_r}_\gamma g$.

Définition 6.4 *A toute SOS-spécification* $P = ((F, ari), R)$ *on associe un* SOS-automate, *noté* γ_P, *et défini comme le graphe orienté et étiqueté* $\langle F, \rightarrow_{\gamma_P}, R_* \setminus \{*\}\rangle$ *où* $f \xrightarrow{\rho}_{\gamma_P} g$ *ssi la conclusion de la règle* $r \in R$ *a la forme* $f(\ldots) \xrightarrow{\alpha} g(\ldots)$.

Le SOS-automate γ_P contient toute l'information d'une SOS-spécification P. On peut maintenant définir l'automate de preuves Γ_P associé à P, et à partir duquel on construira un STA

Définition 6.5 *Soit* $P = (\Sigma, R)$ *une SOS-spécification, l'*automate de preuves Γ_P *est le graphe orienté étiqueté* $\langle T_\Sigma, \rightarrow_{\Gamma_P}, E_P\rangle$, *où*

$$t \xrightarrow{e}_{\Gamma_P} u \stackrel{def}{\Leftrightarrow} Dom(e) \subseteq Dom(t) = Dom(u) \quad et \quad \begin{cases} t(s) \xrightarrow{e(s)}_{\gamma_P} u(s) & si\ s \in D(e) \\ t(s) = u(s) & sinon \end{cases}$$

Pour définir le STA, il ne manque plus que la définition de la relation d'indépendance. Nous procédons en deux étapes, d'abord en définissant une relation d'indépendance $\diamond_{\gamma_P}$ sur le SOS-automate, puis, en l'étendant aux événements pour obtenir une relation $\diamond_{\Gamma_P}$ dans l'automate de preuves Γ_P, stable en avant et commutative, mais non forcement irréflexive.

Définition 6.6 *Soit le SOS-automate* $\gamma_P = (F, \rightarrow_{\gamma_P}, R_*)$. *Nous définissons* $\diamond_{\gamma_P} \subseteq (R_* \setminus \{*\}) \times (R_* \setminus \{*\})$ *comme*

$$\rho_1 \diamond_{\gamma_P} \rho_2 \stackrel{def}{\Leftrightarrow} \begin{cases} \exists f \in F.\ f \xrightarrow{\rho_1}_{\gamma_P} \xrightarrow{\rho_2}_{\gamma_P} & \\ \forall f, g \in F.\ (f \xrightarrow{\rho_1}_{\gamma_P} \xrightarrow{\rho_2}_{\gamma_P} g) \Leftrightarrow (f \xrightarrow{\rho_2}_{\gamma_P} \xrightarrow{\rho_1}_{\gamma_P} g) & et \end{cases}$$

C'est précisément ici que notre démarche s'éloigne de celle de [BD92], pour lesquels la relation d'indépendance, notée $\|$, est définie sur les SOS-automates par $\rho_1 \| \rho_2$ ssi $op(\rho_1) = op(\rho_2)$, $op(\rho_1) \xrightarrow{\rho_1}_{\gamma_P}$ et $op(\rho_1) \xrightarrow{\rho_2}_{\gamma_P}$.

On montre aisément que $\diamond_{\gamma_P} \subseteq \|$, et que en général, cette inclusion est stricte.

Lemme 6.7 *Soit le SOS-automate γ_P.*

$$\rho_1 \diamond_{\gamma_P} \rho_2 \quad ssi \quad op(\rho_1) = op(\rho_2) \;\; et \;\; \begin{cases} op(\rho_1) \xrightarrow{\rho_1}_{\gamma_P} & et \\ op(\rho_1) \xrightarrow{\rho_2}_{\gamma_P} & et \\ \forall f \in F.\ (f \xrightarrow{\rho_1}_{\gamma_P} \Leftrightarrow f \xrightarrow{\rho_2}_{\gamma_P}) \end{cases}$$

Définition 6.8 *Soit P une SOS-spécification. La relation $\diamond_{\Gamma_P} \subseteq E_P \times E_P$ est définie par $a \diamond_{\Gamma_P} b \stackrel{def}{\Leftrightarrow} \forall s \in D(a) \cap D(b).\ a(s) \diamond_{\gamma_P} b(s)$.*

Lemme 6.9 *Soit une SOS-spécification P. La relation $\diamond_{\Gamma_P}$ est stable en avant et commutative dans Γ_P.*

Preuve. Commutativité: Etant donnés $a \diamond_{\Gamma_P} b$ et $t \xrightarrow{a} t_1 \xrightarrow{b} u$, nous définissons le terme t_2 par $Dom(t_2) = Dom(t)$ et

$$t_2(s) \stackrel{\text{def}}{=} \begin{cases} t(s) & \text{si } s \notin D(b) \\ u(s) & \text{si } s \in D(b) \end{cases}$$

On vérifie facilement que $t \xrightarrow{b} t_2 \xrightarrow{a} u$.

Stabilité en avant: Etant donnés $a \diamond_{\Gamma_P} b$, $t \xrightarrow{a} t_1$ et $t \xrightarrow{b} t_2$, nous définissons le terme u par $Dom(u) = Dom(t)$ et

$$t_2(s) \stackrel{\text{def}}{=} \begin{cases} t(s) & \text{si } s \notin D(a) \cup D(b) \\ t_1(s) & \text{si } s \in D(a) \\ t_2(s) & \text{si } s \in D(a) \end{cases}$$

($t_1(s) = t_2(s)$ pour tout $s \in D(a) \cap D(b)$). On vérifie facilement que $t_1 \xrightarrow{b} u$ et $t_2 \xrightarrow{a} u$. □

Ce dernier lemme nous permet de justifier la définition suivante:

Définition 6.10 *Etant donnés une SOS-spécification $P = (\Sigma, R)$, et $p \in T_\Sigma$, on peut définir le STA $G_{P,p}$, d'état initial p, par*

$$G_{P,p} \stackrel{def}{=} \langle T_\Sigma, p, E_P, I_P, \rightarrow_{\Gamma_P}, l \rangle$$

où $I_P = \diamond_{\Gamma_P} \setminus Id|_{E_P \times E_P}$ et $l(e) = act(e(\lambda))$.

Lorsqu'il n'y a pas d'ambiguïté, nous notons p au lieu de $G_{P,p}$.

6.2 Les bisimulations comme congruences

Une condition suffisante pour que les bisimulations étudiées dans le chapitre 5 soient des congruences pour tous les combinateurs d'une SOS-spécification (σ, R), est que chaque axiome $r \in R$ satisfasse $\neg\rho_r \diamond_{\gamma_P} \rho_r$.

L'exemple suivant nous montre que cette condition ne peut pas être affaiblie trivialement.

Exemple 1 *Soit la signature* $\Sigma = (\{\surd, \uparrow, \downarrow, |\}, ari)$ *avec* $ari(\surd) = ari(\uparrow) = ari(\downarrow) = 0$, $ari(|) = 2$ *et* R *l'ensemble de règles*

$$\frac{}{\surd \xrightarrow{\alpha} \surd} \quad \frac{}{\uparrow \xrightarrow{\alpha} \downarrow} \quad \frac{}{\downarrow \xrightarrow{\alpha} \uparrow} \quad \frac{p \xrightarrow{\alpha} p_1}{p \mid q \xrightarrow{\alpha} p_1 \mid q} \quad \frac{q \xrightarrow{\alpha} q_1}{p \mid q \xrightarrow{\alpha} p \mid q_1} \quad \frac{p \xrightarrow{\alpha} p_1 \quad q \xrightarrow{\alpha} q_1}{p \mid q \xrightarrow{\alpha} p_1 \mid q_1}$$

traduites par les opérateurs respectifs $\rho_\surd = \langle\langle\alpha\rangle, \surd\rangle$, $\rho_\downarrow = \langle\langle\alpha\rangle, \downarrow\rangle$, $\rho_\uparrow = \langle\langle\alpha\rangle, \uparrow\rangle$, $\rho_g = \langle\langle\alpha, *, \alpha\rangle, |\rangle$. $\rho_d = \langle\langle *, \alpha, \alpha\rangle, |\rangle$. $\rho_c = \langle\langle\alpha, \alpha, \alpha\rangle, |\rangle$.

Notons que la première règle est un axiome, et que son opérateur associée satisfait $\rho_\surd \diamond_{\gamma_P} \rho_\surd$. *Nous montrons que la mo-équivalence, la cI-bisimulation et la gI-bisimulation ne sont pas des congruences pour* $|$.

Il est facile de prouver que pour $\approx \in \{\approx_{mo}, \approx_{cI}, \approx_{gI}\}$, *on a* $\surd \approx \uparrow$. *Cependant il n'y a pas de bisimulation reliant* $(\surd \mid \surd) \mid \surd$ *et* $(\uparrow|\uparrow) \mid \uparrow$: *en effet, le premier peut s'engager dans deux événements indépendants* $(\rho_c(\rho_g\rho_\surd)\rho_\surd$ *et* $\rho_c(\rho_d\rho_\surd)\rho_\surd)$, *et ce n'est pas le cas pour le second.*

Le lemme suivant établit plusieurs versions équivalentes de notre conditions suffisante. Il nous dit que la condition suffisante est que I_P soit construit d'une façon purement inductive a partir de $\diamond_{\gamma_P}$.

Lemme 6.11 *Soit* (Σ, R) *une SOS-spécification. Les conditions suivantes sont équivalentes:*

- 1. *Pour tout axiome* $r \in R$, $\neg(\rho_r \diamond_{\gamma_P} \rho_r)$.
- 2. $\diamond_{\Gamma_P}$ *est irréflexive* $(\diamond_{\Gamma_P} \cap Id = \emptyset)$.
- 3. I_P *est construit d'une façon purement inductive à partir du* $\diamond_{\gamma_P}$: $aI_Pb \Leftrightarrow \forall s \in D(a) \cap D(b).\ a(s) \diamond_{\gamma_P} b(s)$[1]

Preuve. Il est trivial de prouver $2 \Leftrightarrow 3$.

$2 \Rightarrow 1$. Soit un axiome $r \in R$, et l'événement $e = \rho_r * \cdots *$ (avec $ari(\rho_r)$ étoiles).

Si $\diamond_{\Gamma_P}$ est irréflexive, $\neg(eI_Pe)$. Alors pour le seul élément de $D(e) = \{\lambda\}$, $\neg e(\lambda) \diamond_{\gamma_P} e(\lambda)$. Mais $e(\lambda) = \rho_r$ donc $\neg\rho_r \diamond_{\gamma_P} \rho_r$.

$1 \Rightarrow 2$. Dans chaque événement e il existe au moins une occurrence de ρ_r pour un axiome $r \in R$ dans une position $s \in D(e)$. Comme $\neg\rho_r \diamond_{\gamma_P} \rho_r$, $\neg e \diamond_{\Gamma_P} e$. □

6.2.1 La mo-équivalence et la cI-bisimulation comme congruences

Pour la mo-équivalence et la cI-bisimulation, la preuve de congruence est fondée sur la construction des bisimulations entre $f(p_1 \ldots p_n)$ et $f(q_1 \ldots q_n)$ à partir de celles qui relient p_j et q_j pour chaque $j \in [1..n]$.

Un premier lemme nous dit comment extraire un calcul σ_j de p_j à partir d'un calcul σ de $f(\ldots p_j \ldots)$. C'est une construction en deux temps: d'abord on considère les composants de chaque élément de σ à la position j, $\langle\sigma(1)[j], \sigma(2)[j] \ldots \sigma(|\sigma|)[j]\rangle$, et ensuite on élimine les $*$ de cette séquence pour obtenir σ_j. On utilise une fonction π_j pour retenir l'information précisant quelles places occupe chaque élément du σ_j dans σ $(\sigma_j(k) = \sigma(\pi_j(k))[j])$.

Le résultat de cette construction sera noté $\sigma|_{j,\pi_j} \stackrel{\text{def}}{=} \sigma_j$.

[1] *Dans le cas général on a l'implication seulement dans le sens* $\Leftarrow$, *c'est le cas, par exemple, du couple* $(\rho_\surd, \rho_\surd)$ *de l'Exemple 1*

Lemme 6.12 *Soit $\sigma \in Calc(f(\ldots p_j \ldots))$. Pour tout $j = 1, \ldots, ari(f)$ il existe un calcul $\sigma_j \in Calc(p_j)$ et une fonction monotone et injective $\pi_j : [1, \ldots, |\sigma_j|] \mapsto [1, \ldots, |\sigma|]$ tel que pour tout $l = 1, \ldots, |\sigma_j|$, $\sigma(\pi_j(l))[j] = \sigma_j(l)$ et pour $k \notin \pi_j([1..|\sigma_j|])$, $\sigma(k)[j] = *$.*

Nous notons $\sigma|_{j,\pi_j} \stackrel{def}{=} \sigma_j$.

Théorème 6.13 *Soit $P = (\Sigma, R)$ une SOS-spécification dans les conditions du Lemme 6.11. La mo-équivalence et la cI-bisimulation sont des congruences pour tous les combinateurs de Σ.*

C.-à-d. pour tout $\approx \in \{\approx_{mo}, \approx_{cI}\}$, $f \in F$ avec $ari(f) = n$, si $p_i, q_i \in T_\Sigma$ $(i = 1, \ldots, n)$ sont tels que $\forall i = 1, \ldots, n.\ p_i \approx q_i$ alors $f(\ldots p_j \ldots) \approx f(\ldots q_j \ldots)$

Preuve. Soit $\approx \in \{\approx_{mo}, \approx_{cI}\}$, $P = ((F, ari, R))$ une SOS-spécification telle que $\diamond_{\Gamma_P}$ est irréflexive, $f \in F$ avec $ari(f) = n$, $p_j, q_j \in T_\Sigma$ $(j \in [1..n])$, $Z_j : p_j \approx q_j$.

On pose $Z \subseteq Calc(f(\ldots p_j \ldots)) \times Calc(f(\ldots q_j \ldots))$ comme

$$\sigma Z \rho \stackrel{\text{def}}{\Leftrightarrow} |\sigma| = |\rho| \quad et \quad \begin{cases} \forall k = 1, \ldots, |\sigma|\ .\ \sigma(k)[\epsilon] = \rho(k)[\epsilon] & \text{et} \\ \forall j = 1, \ldots, n.\ \exists \pi_j.\ \rho|_{j,\pi_j} Z_j \sigma|_{j,\pi_j} \end{cases}$$

On vérifie facilement que $Z : f(\ldots p_j \ldots) \approx f(\ldots q_j \ldots)$. □

6.2.2 La gI-bisimulation comme congruence et l'inexprimabilite du raffinement d'actions

Nous nous intéressons ici aux gI-bisimulations définies entre deux STA $G_{P,p}$ et $G_{P,q}$ pour une spécification donnée P, c-à-d. aux gI-bisimulations qui relient des STA qui diffèrent seulement par leurs états initiaux.

Lemme 6.14 *Soient $G = \langle Q, p, E, I, \rightarrow, l\rangle$, $G' = \langle Q, q, E, I, \rightarrow, l\rangle$ deux STA et $\sim \subseteq E \times E$ une relation quelconque. Si $\forall a \sim b.\ \forall e \in E.\ aIe \Leftrightarrow bIe$ alors $\forall a \sim b, a' \sim b'.\ aIa' \Leftrightarrow bIb'$.*

Théorème 6.15 *Soit $P = (\Sigma, R)$ une SOS-spécification dans les conditions du Lemme 6.11. La gI-bisimulation est une congruence pour tous les combinateurs de Σ.*

En notant simplement $t(\in T_\Sigma)$ au lieu de $G_{P,t}$, suffit de montrer que pour tout $f \in F$ avec $ari(f) = n$, s'il existe $(Z, \sim)$ tel que pour tout $p_i, q_i \in T_\Sigma$ $(i = 1 \ldots n)$ $(Z, \sim) : p_i \approx_{gI} q_i$ alors il existe $(Z_1, \sim_1)$ tel que pour tout i, $(Z_1, \sim_1) : p_i \approx_{gI} q_i$ et $(Z_1, \sim_1) : f(p_1 \ldots p_n) \approx_{gI} f(q_1 \ldots q_n)$

Preuve. On définit Z_1 comme la plus petite relation telle que $Z \subseteq Z_1 \subseteq T_\Sigma \times T_\Sigma$ et qui satisfait pour tout $f \in F$ avec $ari(f) = n$, $\forall i.\ p_i Z_1 q_i$ implique $f(p_1 \ldots p_n) Z_1 f(q_1 \ldots q_n)$.

On définit $\sim_1$ comme la plus petite relation $\sim \subseteq \sim_1 \subseteq E_P \times E_P$ qui satisfait : pour toute opérateur n-aire ρ, $\forall i.\ a_i \sim_1 b_i$ implique $\rho a_1 \ldots a_n \sim_1 \rho b_1 \ldots b_n$ [2].

Maintenant, il faut prouver que $(Z_1, \sim_1)$ est une gI-bisimulation. La construction est très similaire à celle du [GV88]. La seule question non-triviale qui reste est de vérifier que $\sim_1$ préserve l'indépendance.

Après le Lemme 6.14 et l'hypothèse de nous trouver dans les conditions du Lemme 6.11, il suffit de prouver que $\forall a, b, e \in E_p$, $a \sim_1 b$ et $\forall s \in D(a) \cap D(e).\ a(s) \diamond_{\gamma_P} e(s)$ implique

[2] où $\rho e_1 \ldots e_n$ dénote le schéma de preuve de racine ρ et de sous-arbres $e_1 \ldots e_n$.

$\forall s \in D(b) \cap D(e).\ b(s) \diamond_{\gamma_P} e(s)$. Nous allons le prouver par induction dans la construction de $\sim_1$.

Le cas de base ($a \sim b$) est trivial.

Soient $a \sim_1 b, a = \rho a_1 \ldots a_n, b = \rho b_1 \ldots b_n$. Notre hypothèse d'induction dit que pour tout $e \in E_p$, pour tous les couples (a_j, b_j), $\forall s \in D(a_j) \cap D(e).\ a_j(s) \diamond_{\gamma_P} e(s)$ implique $\forall s \in D(b_j) \cap D(e).\ b_j(s) \diamond_{\gamma_P} e(s)$.

Il suffit alors simplement de vérifier pour a et b que pour tout e, $a(\lambda) \diamond_{\gamma_P} e(\lambda)$ implique $b(\lambda) \diamond_{\gamma_P} e(\lambda)$, ce qui est trivial, puisque $a(\lambda) = b(\lambda)$. □

La retombée la plus intéressante de ce résultat est que le raffinement d'actions, pour lequel la gI-bisimulation n'est pas une congruence, n'est pas exprimable dans une large famille des spécifications SOS de base.

Corollaire 6.16 *Soit $\mathcal{L}$ un LDP tel que sa sémantique puisse être donnée à l'aide d'une SOS-spécification de base dans les conditions du Lemme 6.11. $\mathcal{L}$ ne possède pas de combinateur qui représente le raffinement d'actions tel qu'il apparaît dans la Définition 4.3.*

6.3 Un exemple : le langage CCS

Nous illustrons ici la technique présentée dans la Section 6.1, en donnant une sémantique au langage CCS [Mil80, Mil89] dans les STA. Cette sémantique est exactement celle de [BD92] sur les Automates de Traces, puisque les automates définis pour CCS sont en fait des STA.

Le point le plus délicat de cette construction est le traitement des combinateurs dynamiques de CCS (le préfixage et la somme non-déterministe) qui ne sont pas directement représentables dans le format SOS de base.

Soit $\Delta = \{\alpha \ldots\}$ un alphabet d'actions et $\bar{\Delta} = \{\bar{\alpha} \ldots\}$ l'alphabet d'actions complémentaires, avec $\Delta = \bar{\bar{\Delta}}$, $\Lambda \equiv \Delta \cup \bar{\Delta} = \{\lambda \ldots\}$, et $\tau \notin \Lambda$. Soit $Act = \Lambda \cup \{\tau\} = \{\mu \ldots\}$ l'alphabet d'actions de CCS.

Nous considérons pour le langage CCS la signature Σ_{CCS} qui a comme combinateurs *nil* d'arité 0, $\mu.$, id, $\backslash_\alpha$, $[\Phi]$ d'arité 1 et $\mid +$, π_1, π_2 d'arité 2. Φ est une fonction $\Phi : Act \mapsto Act$ qui satisfait $\Phi(\bar{\lambda}) = \overline{\Phi(\lambda)}$, $\Phi(\mu) = \tau$ ssi $\mu = \tau$.

Soit $CCS = (\Sigma_{CCS}, R_{CCS})$ notre SOS-spécification de CCS, où R_{CCS} est l'ensemble de règles apparaissant dans la Figure 6.1. Les schémas de règles correspondant sont

$$\bullet^\mu = \langle\langle *, \mu\rangle, id\rangle \qquad id^\mu = \langle\langle \mu, \mu\rangle, id\rangle$$
$$\pi_1^\mu = \langle\langle \mu, *, \mu\rangle, \pi_1\rangle \qquad \pi_2^\mu = \langle\langle *, \mu, \mu\rangle, \pi_2\rangle$$
$$|_1^\mu = \langle\langle \mu, *, \mu\rangle, |\rangle \qquad |_2^\mu = \langle\langle *, \mu, \mu\rangle, |\rangle \qquad |_3^\lambda = \langle\langle \lambda, \bar{\lambda}, \tau\rangle, |\rangle$$
$$\backslash_\alpha^\mu = \langle\langle \mu, \mu\rangle, \backslash_\alpha\rangle \qquad \Phi^\mu = \langle\langle \mu, \Phi(\mu)\rangle, [\Phi]\rangle$$

Le SOS-automate correspondant à CCS est dessiné en Figure 6.2. La relation $\diamond_{\gamma_{CCS}} \subseteq R_{CCS} \times R_{CCS}$

$$\rho_1 \diamond_{\gamma_{CCS}} \rho_2 \quad ssi \quad \rho_1 = \rho_2 \neq \bullet^\mu \ \ ou \ \ \rho_1, \rho_2 \in \{|_1^\mu, |_2^\mu, |_3^\alpha\}$$

peut facilement s'extraire de l'automate. Elle coïncide avec la relation $\|_{CCS}$ de [BD92].

Lemme 6.17 *$\approx_{mo}$, $\approx_{cI}$ et $\approx_{gI}$ sont des congruences pour tous les combinateurs de CCS.*

$$\frac{}{\mu.p \xrightarrow{\mu} id(p)} \qquad \frac{p \xrightarrow{\mu} q}{id(p) \xrightarrow{\mu} id(q)}$$

$$\frac{p \xrightarrow{\mu} q}{p + p' \xrightarrow{\mu} \pi_1(q, p')} \qquad \frac{p' \xrightarrow{\mu} q'}{p + p' \xrightarrow{\mu} \pi_2(p, q')}$$

$$\frac{p \xrightarrow{\mu} q}{\pi_1(p, p') \xrightarrow{\mu} \pi_1(q, p')} \qquad \frac{p' \xrightarrow{\mu} q'}{\pi_2(p, p') \xrightarrow{\mu} \pi_2(p, q')}$$

$$\frac{p_1 \xrightarrow{\mu} q_1}{p_1|p_2 \xrightarrow{\mu} q_1|p_2} \qquad \frac{p_2 \xrightarrow{\mu} q_2}{p_1|p_2 \xrightarrow{\mu} p_1|q_2} \qquad \frac{p_1 \xrightarrow{\alpha} q_1 \quad p_2 \xrightarrow{\bar{\alpha}} q_2}{p_1|p_2 \xrightarrow{\tau} q_1|q_2}$$

$$\frac{p \xrightarrow{\mu} q}{p\backslash_\alpha \xrightarrow{\mu} q\backslash_\alpha} \ (\mu \notin \{\alpha, \bar{\alpha}\}) \qquad \frac{p \xrightarrow{\mu} q}{p[\Phi] \xrightarrow{\Phi(\mu)} q[\Phi]}$$

Figure 6.1: R_{CCS}

Preuve. Les seuls axiomes du R_{CCS} ont comme opérateurs $\bullet^\mu$, et puisque $\neg(\bullet^\mu \diamond_{\gamma_{CCS}} \bullet^\mu)$ on est alors dans les conditions des Théorèmes 6.13 et 6.15. □

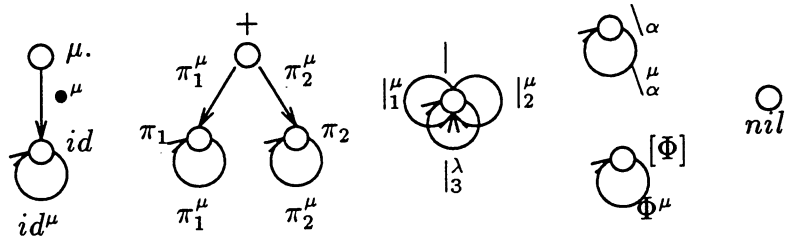


Figure 6.2: Le SOS automate du CCS

Cette construction pour CCS nous montre que la relation d'indépendance I_P de la Définition 6.10 est parfois insatisfaisante. Par exemple, pour $p \stackrel{\text{def}}{=} \alpha.0 \,|\, \beta.0$, il n'existe pas de gI-bisimulation qui relie p et $p + 0$.

Ceci est dû au fait que les deux événements $|_1^\alpha \bullet^\alpha$ et $|_2^\beta \bullet^\beta$ dans lesquels p peut être engagé sont indépendants. Il en est de même pour les deux événements $\pi_1^\alpha \, |_1^\alpha \bullet^\alpha$ et $\pi_1^\beta \, |_2^\beta \bullet^\beta$, dans lesquels $p + 0$ peut être engagé. Mais il n'existe pas de relation $\sim$ qui préserve l'indépendance en reliant $\pi_1^\alpha \, |_1^\alpha \bullet^\alpha \sim |_1^\alpha \bullet^\alpha$, puisque $|_1^\alpha \bullet^\alpha \; I_P \; |_2^\beta \bullet^\beta \;\not I_P \; \pi_1 \, |_1^\alpha \bullet^\alpha$.

Pour cette raison, les STA que nous dérivons du langage CCS ne sont pas gI-bisimilaires à ceux de [MN92].

On peut relâcher la définition 6.10, par exemple, en donnant non pas une relation I_P, mais une famille de I_P^j, et une sémantique possible pour chaque choix d'un I_P^j. Il est facile de voir que pour chaque I_P^j irréflexive qui satisfait

$$I_P \subseteq I_P^j \subseteq I_P \cup \not\frown$$

où $a \smile b \stackrel{\text{def}}{\Leftrightarrow} \exists s.\ s \xrightarrow{a} \xrightarrow{b} \vee s \xrightarrow{b} \xrightarrow{a} \vee (s \xrightarrow{a} \wedge s \xrightarrow{b})$ on obtient bien un STA.

Pour CCS, le plus petit I^*_{CCS} qui satisfait

$$aI^*_{CCS}b \quad \textit{implique} \quad \forall e \in E.\ aI^*_{CCS}\pi_1(b,e),\ \ aI^*_{CCS}\pi_2(e,b),\ \ aI^*_{CCS}id(b)$$

nous donne un STA gI-bisimilaire à celui de [MN92].

Chapitre 7

Conclusion et perspectives

Dans ce travail nous avons contribué à établir sur les Systèmes de Transitions Asynchrones des constructions intéressantes pour la sémantique de systèmes réactifs: le raffinement d'actions, des bisimulations compatibles avec le raffinement d'actions, et la technique de Sémantique Opérationnelle Structurée.

Nous avons proposé au Chapitre 4 une définition du raffinement d'actions, qui généralise des travaux précédents sur les ST et les SE.

Dans le Chapitre 5 nous avons montré des équivalences compatibles avec le raffinement: la mo-équivalence [DDM89] que nous avons adaptée aux STA, la cI-bisimulation [MN92], et une nouvelle équivalence, la dI-bisimulation qui est définie sur l'automate et préserve l'indépendance.

Finalement, au Chapitre 6 nous adaptons la technique de [BD92, BD93] pour employer des STA comme modèles pour des termes dont la sémantique est définie selon la méthode de Plotkin. Nos contributions sont d'une part l'énoncé d'une condition suffisante sur les spécifications pour que la mo-équivalence et la cI-bisimulation soient des congruences, et d'autre part la preuve que le raffinement d'actions ne peut pas être exprimé dans des spécifications qui satisfont cette condition.

Dans la suite de ce chapitre nous dégageons quelques lignes d'actions pour le futur.

L'approche "géométrique" de notre définition de raffinement a beaucoup guidé notre intuition, et elle nous semble assez solide aujourd'hui pour essayer d'adapter notre définition à d'autres modèles. Cela semble très facile pour d'autre ST avec information sur le parallélisme et nous nous proposons de l'adapter à différentes variantes des "automates géométriques" [Pra91, Dro90, Gun91, Gun92]).

Nous avons analysé la possibilité d'élargir notre définition du raffinement avec la notion de

"raffinement de l'information sur la causalite", à la façon de [JPZ91, DGR92].

Mais le raffinement de la relation de causalité dans les SE, comme il est proposé dans ces papiers, peut changer d'une façon inattendue la structure des choix, comme le montre la Figure 7.1.

Dans $\mathcal{E}$ on doit choisir entre e_2 et e_3 après avoir exécuter e_1. Imaginons qu'on raffine l'événement e_3 de $\mathcal{E}$ par un couple d'événements indépendants e_3' et e_3'' parmi lesquels seulement e_3' hérite les liens de causalité avec e_1. Dans $\mathcal{F}$ on peut commencer une exécution par e_3'', et ainsi exclure l'exécution de e_2 *avant même* d'exécuter e_1.

Figure 7.1: Le problème du raffinement de la causalité dans les SE.

Le "raffinement de l'indépendance" dans les STA conduit à des changements encore plus dramatiques: on peut même faire disparaître les point de choix, comme le montre la Figure 7.1. L'événement b, avec $\neg(aIb)$ est raffiné par b_1 a droite, avec aIb_1. Le choix dans l'état initial disparaît.

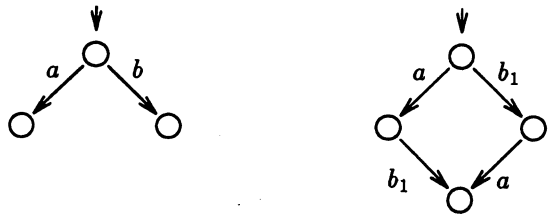


Figure 7.2: Le problème du raffinement de l'indépendance $(\neg aIb, aIb_1)$.

Il est possible que cette notion de raffinement de l'information sur la causalité soit compatible avec des équivalences qui ne prennent pas en compte les points de choix, mais elle ne semblent pas d'un grand intérêt dans notre cadre.

Nous nous sommes particulièrement intéressés aux bisimulations définies sur les objets de base des STA (états, transitions, indépendance) au lieu de celles définies sur les exécutions et l'ordre partiel. Ceci est du en partie au fait que nous croyons que cette voie mène vers des résultats qui peuvent être à la base d'outils de vérification automatique.

A notre avis, la considération pour la dI-bisimulation de morphismes qui soient des extensions de ceux qui caractérisent la bisimulation classique dans les ST est un choix judicieux. On s'attend à profiter de l'expérience existant sur ces algorithmes pour la bisimulation classique.

Il reste un problème avant de s'attaquer à l'étude de ces algorithmes, c'est celui de l'inexistence de représentants canoniques dans les classes d'équivalence de la dI-bisimulation, comme on peut le constater à partir de l'exemple de la Figure 2.10.

Outre les aspects algorithmiques, il reste beaucoup de travail à faire sur le terrain des bisimulations.

Le premier objectif serait d'étendre les ST-bisimulations de [GV87, Gla90c, Vog90a], qui sont de congruences optimales pour le raffinement, aux STA. Elles doivent se placer naturellement quelque part entre la mo-équivalence et la bisimulation d'entrelacement.

On est à la recherche aussi d'une bisimulation plus fine mais aussi proche que possible de la dI-bisimulation, compatible avec le raffinement, et qui préserve les pomsets. Cette équivalence serait-elle comparable avec la mo-équivalence?

Et puis bien sûr, on pourrait essayer d'imaginer quel sens donner aux actions invisibles dans les STA et aux bisimulations adéquates.

Après notre discussion dans l'introduction du Chapitre 5 sur la mo-équivalence, une autre voie reste à explorer. Il s'agirait de chercher des bisimulations causales qui soient des congruences pour le raffinement, mais définies sur les configurations, et non sur les calculs. Nous donnons une réponse incomplète à cette question dans la Section 7.1.

Même si on ne suit pas l'approche extrême du [Gog91], l'utilisation des éléments de la Théorie des Catégories dans notre travail a été pour nous extrêmement utile.

Dans un futur très proche, nous comptons utiliser cette approche pour montrer les liens entre les raffinements comme foncteurs et des bisimulations adéquates sur différents modèles arborescents.

Sur les SOS la première question ouverte est de trouver une condition suffisante pour que la dI-bisimulation soit une congruence. Pour cela il serait utile de trouver une définition "à la Milner" de la dI-bisimulation.

La question qui semble plus intéressante, et est, à l'origine, liée à l'étude des SOS, mais dépasse largement ce cadre, est celui de l'étude de la *localité dans les STA et dans les SOS.* Cette question est discutée à la Section 7.2.

Finalement, nous croyons que la famille des spécifications dans lesquels le raffinement d'actions est inexprimable est bien plus vaste que celui pour lequel nous l'avons prouvé. Nous croyons en fait que le raffinement est inexprimable en utilisant le format *sans copie* [Pin91, Pin93], et nous cherchons aujourd'hui une façon de le prouver.

Les deux sections qui suivent présentent des résultats préliminaires sur une nouvelle bisimulation d'ordre partiel (Section 7.1) et sur la localité (Section 7.2).

7.1 Sur une nouvelle bisimulation d'ordre partiel

La question à laquelle nous nous intéressons ici est: *Est-ce qu'il existe une bisimulation d'ordre partiel compatible avec le raffinement qui ne soit pas définie sur les calculs?*

On n'a pas de réponse complète. Mais on veut essayer ici d'attirer l'attention du lecteur sur la *pomset bisimulation avant et arrière sur les configurations* qui est à notre avis une bonne candidate.

Les bisimulations d'entrelacement avant et arrière ont été introduites par De Nicola, Montanari et Vaandrager [DNMV90] comme une variante de la bisimulation d'entrelacement, ou la

propriété de transfert est demandée comme d'habitude, *en avant* mais aussi *en arrière*, suivant la notion d'exécution adéquate. Dans les ST, cette notion est sans doute le calcul.

Deux notions différentes de bisimulations avant et arrière on été proposées dans la littérature.

Premièrement Cherief [Che92a] propose une bisimulation avant arrière sur les SE, en considérant les transitions étiquetées par des pomsets, et les calculs comme exécutions. Cette équivalence, coïncide avec la mo-équivalence et peut être caractérisée par une variante simple de la logique de Hennessy et Milner (HML) [HM85], avec pomsets comme modalités de futur et de passé. Une étude complète de cette équivalence a été réalisée par Cherief [Che92c, Che92b] et Pinchinat [Pin93].

En utilisant des configurations comme exécutions, une bisimulation avant-arrière pour SE non auto-concurrents a été introduite par Goltz, Kuiper et Penczek [GKP92]. L'absence des événements auto-concurrents leur permet de limiter la propriété de transfert aux événements (au lieu des pomsets) sans perte d'expressivité. Ils ont montré que la logique POL de Sinachopoulos [Sin90] est adéquate pour cette équivalence, et qu'elle coïncide avec la *hereditary strong history preserving bisimulation* de Bednarczyk [Bed91], qui est une congruence pour le raffinement d'actions.

Nous proposons d'étudier la *pomset bisimulation avant arrière sur les configurations*, qui peut être vue comme une généralisation de celle de [GKP92] aux STA (avec auto-concurrence).

Etant données deux configurations x et y, on note $x \xrightarrow{p} y$ quand il existe un chemin $\sigma\pi$ t.q. $x = C_\sigma$, $y = C_{\sigma\pi}$ et p est un pomset sur Act obtenue en remplaçant chaque événement de C_π par son étiquette ($L_p(j) = l_G(L_\pi(j))$).

Définition 7.1 *Deux STA G_1 et G_2 sont* pomset bisimilaires avant arrière sur les configuration, *noté $G_1 \approx_{bf} G_2$ ssi il existe une relation $R \subseteq Conf(G_1) \times Conf(G_2)$ tel que les configurations vides sont reliées par R et pour tout $x_1 R x_2$,*

- *(avant)*
  - *pour chaque $y_1 \in Conf(G_1)$, $x_1 \xrightarrow{p}_1 y_1$, il existe $y_2 \in Conf(G_2)$ t.q. $x_2 \xrightarrow{p}_2 y_2$,*
  - *vice versa, pour chaque $y_2 \in Conf(G_2)$, $x_2 \xrightarrow{p}_2 y_2$, il existe $y_1 \in Conf(G_1)$ t.q. $x_1 \xrightarrow{p}_1 y_1$;*
- *(arrière dans la configuration)*
  - *pour chaque $z_1 \in Conf(G_1)$, $z_1 \xrightarrow{q}_1 x_1$, il existe $z_2 \in Conf(G_2)$ t.q. $z_2 \xrightarrow{q}_2 x_2$,*
  - *et vice versa, pour chaque $z_2 \in Conf(G_2)$, $z_2 \xrightarrow{q}_2 x_2$, il existe $z_1 \in Conf(G_1)$ t.q. $z_1 \xrightarrow{q}_1 x_1$,*

Il est facile de montrer que $\approx_{bf}$ est strictement plus fine que la pomset bisimulation de Boudol et Castellani [BC87] (noté $\approx_{pomset}$) et aussi que la *NMS partial order equivalence* (noté $\approx_{po}$) de Degano, De Nicola et Montanari [DDM87] (qui est aussi connue comme *weak history preserving bisimulation* [Dev88, OGG88, GG90c]). Il est aussi facile de montrer que $\approx_{dI}$, qui ne préserve pas les pomsets même dans le cas non auto-concurrent, n'implique pas $\approx_{bf}$, ni $\approx_{pomset}$, ni $\approx_{po}$.

Il est seulement un peu plus difficile de prouver que $\approx_{gI}$ implique $\approx_{bf}$. La Figure 5.5 (où $G_5 \approx_{bf} G_6$) montre que cette implication est stricte. En plus, la loi d'absorption qui n'est pas

valide pour $\approx_{bf}$ ($G_3 \not\approx_{bf} G_4$ dans la Figure 5.4) montre que $\approx_{bf}$ et $\approx_{cI}$ sont incomparables, et que $\approx_{mo}$ n'implique pas $\approx_{bf}$, même dans les SE non auto-concurrents.

Nos résultats sur le placement de la $\approx_{bf}$ parmi les équivalences sont résumés dans la Figure 7.3. Les flèches pointillées sont des conjectures.

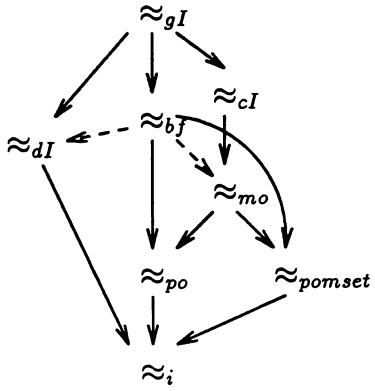


Figure 7.3: La *pomset bisimulation avant arrière sur les configurations* parmi les autres équivalences

Bien sûr, il est facile de caractériser $\approx_{bf}$ par une variante de la logique HML avec des pomsets sur Act comme modalités de futur et de passé, interprétées sur les configurations.

Nous conjecturons que la $\approx_{bf}$ est compatible avec le raffinement d'actions. Ce résultat est déjà connu pour un cas restreint, celui des SE sans auto-concurrence [GKP92, Bed91]. Nous conjecturons aussi que $\approx_{bf}$ implique $\approx_{mo}$, mais la preuve de ces deux résultats a été particulièrement élusive.

7.2 Localité dans les STA et dans les SOS

Le rapprochement avec les travaux de [MN92] a la fin du Chapitre 6 doit nous mener à mieux comprendre la notion de *localité* [BC91, BCHK92, Ace91, MY92], définie d'une façon syntaxique sur CCS, et une autre définit sur les STA, non reliée au langage, que l'on peut formaliser comme une propriété sur les événements:

$$\begin{aligned} \textit{localité} &: E \mapsto \mathcal{P}(E) \\ \textit{localité}(a) &\stackrel{\text{def}}{=} \{e \in E : \forall b \in E.\ aIb \Leftrightarrow eIb\} \end{aligned}$$

Un autre notion de localité sans référence au langage, a été proposée par Yankelevich dans [Yan93] dans une variante des RP.

Dans [MN92] les auteurs utilisent la localité de base dans CCS pour obtenir l'indépendance. La définition que nous venons d'énoncer suit le chemin inverse.

Notre définition est compatible avec celle des travaux de [MN92]. Soit (Loc_{MN}) la définition de localité pour les STA considérée dans [MN92]. Pour chaque $a, b \in E_{MN}$ $\textit{localité}(a) = \textit{localité}(b)$ ssi $Loc_{MN}(a) = Loc_{MN}(b)$.

La preuve est facile: si $Loc_{MN}(a) = Loc_{MN}(b)$, par définition de l'indépendance, $\forall e \in E_{MN}.\ aI_{MN}e \Leftrightarrow bI_{MN}b$, donc $localité(a) = localité(b)$.

Dans l'autre sens il suffit de vérifier que E_{MN} est suffisamment riche pour nous permettre de trouver, pour chaque couple d'événements a, b tels que $Loc_{MN}(a) \neq Loc_{MN}(b)$, un autre événement e qui est indépendant d'un seul des deux précédents. Donc $localité(a) \neq localité(b)$.

Il y a ici deux perspectives ouvertes. Premièrement, explorer notre proposition de localité "sémantique" dans les STA. Deuxièmement, essayer de donner une construction générale pour la localité dans les spécifications au format SOS de base.

Bibliographie

[AC88] A. Arnold and P. Crubille. A linear algorithm to solve fixed-point equations on transition systems. *Information Processing Letters*, 29(2):57–66, September 1988.

[Ace91] L. Aceto. A static view of localities. Research Report 1483, INRIA, July 1991.

[AD89] A. Arnold and A. Dicky. An algebraic characterization of transition systems equivalences. *Information and Computation*, 82(2):198–229, August 1989.

[AM75] Arbib and Manes. *Arrows, Structures and Functors: The Categorical Imperative.* Academic Press, 1975.

[BC87] G. Boudol and I. Castellani. On the semantics of concurrency: Partial orders and transition systems. In *Proc. CAAP'87, Pisa, LNCS 249*, pages 123–137. Springer-Verlag, March 1987.

[BC88] G. Boudol and I. Castellani. A non-interleaving semantics for CCS based on proved transitions. *Fundamenta Informaticae*, XI:433–452, 1988. Available as INRIA Research Report 919.

[BC90] G. Boudol and I. Castellani. Three equivalent semantics for CSS. In *Semantics of Systems of Concurrent Processes, La Roche Posay, France, LNCS 469*, pages 96–141. Springer-Verlag, April 1990.

[BC91] G. Boudol and I. Castellani. Flow models of distributed computations: three equivalent semantics for CCS. Research Report 1484, INRIA, July 1991.

[BCHK92] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. In *Proc. CONCUR'92, Stony Brook, NY, LNCS 630*, pages 108–122. Springer-Verlag, August 1992.

[BD92] E. Badouel and Ph. Darondeau. Structural operational specifications and trace automata. In *Proc. CONCUR'92, Stony Brook, NY, LNCS 630*, pages 302–316. Springer-Verlag, August 1992.

[BD93] E. Badouel and Ph. Darondeau. Trace nets and process automata. To appear, 1993.

[BDE] E. Best, R. Devillers, and J. Esparza. General refinement and recursion operators for the Petri box calculus. Proc. STACS'92.

[BDKP91] E. Best, R. Devillers, A. Kiehn, and L. Pomello. Concurrent bisimulations in Petri nets. *Acta Informatica*, 28:231–264, 1991.

[Bed87] M. A. Bednarczyk. *Categories of Asynchronous Systems.* PhD thesis, Univ. Sussex, October 1987. Available as CS R 1/88.

[Bed91] M. Bednarczyk. Hereditary history preserving bisimulations. Draft, 1991.

[BHR84] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, July 1984.

[BIM88] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced: preliminary report. In *Proc. 15th ACM Symp. Principles of Programming Languages, San Diego, CA*, pages 229–239, January 1988.

[Bou89] G. Boudol. Atomic actions. *EATCS Bull.*, 38:136–144, June 1989.

[BW90] J. C. M. Baeten and W. P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge Univ. Press, 1990.

[CGM90] A. Corradini, G. Gerrari, and U. Montanari. Transition systems with algebraic structure as models of computations. In *Semantics of Systems of Concurrent Processes, La Roche Posay, France, LNCS 469*. Springer-Verlag, April 1990.

[Che92a] F. Cherief. Back and forth bisimulations on prime event structures. In *Proc. PARLE'92, Paris, LNCS 605*, pages 843–858. Springer-Verlag, June 1992.

[Che92b] F. Cherief. *Contributions à la Sémantique du Parallélisme: Bisimulations pour le Raffinement et le Vrai Parallélisme*. Thèse de Doctorat, I.N.P. de Grenoble, France, October 1992.

[Che92c] F. Cherief. Investigations of back and forth bisimulations on prime event structures. *Computers and Artificial Intelligence*, 11(5):481–496, 1992.

[CMP87] L. Castellano, G. De Michelis, and L. Pomello. Concurrency vs interleaving: an instructive example. *EATCS Bull.*, 31:12–15, February 1987.

[CR93] R. Constantini and A. Rensink. Abstraction and refinement in configuration structures. Draft paper, 1993.

[DD89] Ph. Darondeau and P. Degano. Causal trees. In *Proc. 16th ICALP, Stresa, LNCS 372*, pages 234–248. Springer-Verlag, July 1989.

[DD90] Ph. Darondeau and P. Degano. Event structures, causal trees, and refinements. In *Proc. Math. Found. Comp. Sci. Banska Bystrica, CZ, LNCS 452*, pages 239–245. Springer-Verlag, 1990.

[DDM87] P. Degano, R. De Nicola, and U. Montanari. Observational equivalences for concurrency models. In M. Wirsing, editor, *Formal Description of Programming Concepts - III, Proc. of the third IFIP WG 2.1 working conf., Ebberup 1986*, pages 105–129. North-Holland, 1987.

[DDM88] P. Degano, R. De Nicola, and U. Montanari. A distributed operational semantics for CCS based on condition/event systems. *Acta Informatica*, 26:59–91, 1988.

[DDM89] P. Degano, R. De Nicola, and U. Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 438–466. Springer-Verlag, 1989.

[Dev88] R. Devillers. On the definition of a bisimulation notion based on partial words. *Petri Net Newsletter*, (29):16–19, 1988.

[Dev90] R. Devillers. Maximality preserving bisimulation. Tech. Report LIT-214, Lab. Informatique Théorique, Université Libre de Bruxelles, March 1990.

[DG92] P. Degano and R. Gorrieri. An operational definition of action refinement. Tech. Report 28/92, Univ. Pisa, Dept. of Computer Science, 1992.

[DGR92] P. Degano, R. Gorrieri, and G. Rosolini. A categorical view of process refinement. To appear Proc. REX'92 School, LNCS, 1992.

[DNMV90] R. De Nicola, U. Montanari, and F. Vaandrager. Back and forth bisimulations. In *Proc. CONCUR'90, Amsterdam, LNCS 458*, pages 152–165. Springer-Verlag, August 1990.

[Dro90] M. Droste. Concurrency, automata and domains. In *Proc. 17th ICALP, Warwick, LNCS 443*, pages 195–208. Springer-Verlag, July 1990.

[ER90] A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. part I: Basic notions and the representation problem. *Acta Informatica*, 27:315–342, 1990.

[Fer90] G. Ferrari. *Unifying Models of Concurrency.* PhD thesis, Univ. Pisa, 1990. Available as report TD-4/90.

[GG89a] R. J. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. In *Proc. Math. Found. Comp. Sci. Porabka-Kozubnik, LNCS 379*, pages 237–248. Springer-Verlag, August 1989.

[GG89b] R. J. van Glabbeek and U. Goltz. Partial order semantics for refinement of actions -neither necessary nor always sufficient but appropriate when used with care. *EATCS Bull.*, 38:154–163, June 1989.

[GG89c] R. J. van Glabbeek and U. Goltz. Refinement of actions in causality based models. In *Stepwise Refinement of Distributed Systems. Models, Formalisms, Correctness, Mook, LNCS 430*, pages 267–300. Springer-Verlag, May 1989.

[GG90a] R. J. van Glabbeek and U. Goltz. A deadlock-sensitive congruence for action refinement. Research Report TUM-I9044, Institut für Informatik, TUM, Munich, November 1990.

[GG90b] R. J. van Glabbeek and U. Goltz. Equivalences and refinement. Research Report TUM-I9024, Institut für Informatik, TUM, Munich, July 1990.

[GG90c] R. J. van Glabbeek and U. Goltz. Equivalences and refinement. In *Semantics of Systems of Concurrent Processes, La Roche Posay, France, LNCS 469*, pages 309–333. Springer-Verlag, April 1990.

[GGR93] U. Goltz, R. Gorrieri, and A. Rensink. On syntactic and semantic action refinement. Hildesheimer Informatik-Berichte 17/92, Universität Hildesheim, Institut für Informatik, 1993.

[GKP92] U. Goltz, R. Kuiper, and W. Penczek. Propositional temporal logics and equivalences. In *Proc. CONCUR'92, Stony Brook, NY, LNCS 630*, pages 222–236. Springer-Verlag, August 1992.

[Gla90a] R. J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions.* PhD thesis, Free Univ. of Amsterdam, May 1990.

[Gla90b] R. J. van Glabbeek. The refinement theorem for ST-bisimulation semantics. In *Proc. IFIP Working Conf. on Programming Concepts and Methods, Sea of Galilee, Israel*, 1990.

[Gla90c] R. J. van Glabbeek. The refinement theorem for ST-bisimulation semantics. Research Report CS-R9002, CWI, January 1990.

[GMM90] R. Gorrieri, S. Marchetti, and U. Montanari. A^2CCS: A simple extension of CCS for handling atomic actions. *Theoretical Computer Science*, 72:203–223, 1990.

[Gog91] J. A. Goguen. A categorical manifesto. *Math. Struct. in Comp. Science*, 1:46–67, 1991.

[Gor91] R. Gorrieri. *Refinement, Atomicity and Transactions for Process Description Languages.* PhD thesis, Univ. Pisa, March 1991.

[Gun91] J. Gunawardena. Geometric logic, causality and event structures. In *Proc. CONCUR'91, Amsterdam, LNCS 527*, pages 266–280. Springer-Verlag, August 1991.

[Gun92] J. Gunawardena. Causal automata. *Theoretical Computer Science*, 101:265–288, 1992.

[GV87] R. J. van Glabbeek and F. Vaandrager. Petri net models for algebraic theories of concurrency. In *Proc. PARLE'87, vol. II: Parallel Languages, Eindhoven, LNCS 259*, pages 224–242. Springer-Verlag, June 1987.

[GV88] J. F. Groote and F. W. Vaandrager. Structured operational semantics and bisimulation as a congruence. Research Report CS-R8845, CWI, November 1988.

[GV89] J. F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence (extended abstract). In *Proc. 16th ICALP, Stresa, LNCS 372*, pages 423–438. Springer-Verlag, July 1989.

[GW89] R. J. van Glabbeek and W. P. Weijland. Refinement in branching time semantics. In *Proc. AMAST Conf., Iowa City*, pages 197–201, 1989. Available as CWI Report CS-R8922.

[Hen91] M. Hennessy. A proof system for weak ST-bisimulation over a finite process algebra. Report 6/91, Univ. Sussex, Brighton, GB, June 1991.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, January 1985.

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, August 1978.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice Hall Int., 1985.

[JPZ91] W. Janssen, M. Poel, and J. Zwiers. Action systems and action refinement in the development of parallel systems. In *Proc. CONCUR'91, Amsterdam, LNCS 527*, pages 298–315. Springer-Verlag, August 1991.

[Kel76] R. M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, July 1976.

[Lam86] L. Lamport. On interprocess communication. *Distributed Computing*, 1:77–101, 1986.

[Lee90] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science, vol. B.* Elsevier Science Publishers, 1990.

[Maz77] A. Mazurkiewicz. Concurrent program schemes and their interpretation. Research Report DAIMI PB-7, Comp. Sci. Dept., Aarhus Univ., 1977.

[Maz89] A. Mazurkiewicz. Basic notions of trace theory. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 285–363. Springer-Verlag, 1989.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall Int., 1989.

[MN92] M. Mukund and M. Nielsen. CCS, locations and asynchronous transition systems. In *Proc. 12th Conf. Found. of Software Technology and Theor. Comp. Sci. New Delhi, India, LNCS*, pages 328–341. Springer-Verlag, December 1992. Also in Resarch Report DAIMI PB-395, Aarhus Univ.

[MY92] U. Montanari and D. Yankelevich. A parametric approach to localities. In *Proc. 19th ICALP, Vienna, LNCS 623.* Springer-Verlag, July 1992.

[NPW81] M. Nielsen, G. D. Plotkin, and G. Winskel. Petri nets, event structures and domains, Part I. *Theoretical Computer Science*, 13(1):85–108, 1981.

[NRT92] M. Nielsen, G. Rozenberg, and P. S. Thiagarajan. Elementary transition systems. *Theoretical Computer Science*, 96(1):3–33, 1992.

[OGG88] E.-R. Olderog, U. Goltz, and R. J. van Glabbeek. Combining compositionality and concurrency, summary of the GMD-workshop, königswinter. Tech. Report GMD-320, St Augustin, 1988.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Proc. 5th GI Conf. on Th. Comp. Sci., LNCS 104*, pages 167–183. Springer-Verlag, March 1981.

[Pet62] C. A. Petri. *Kommunikation mit Automaten.* PhD thesis, Univ. Bonn, 1962. Schriften des Instituts für Instrumentelle Mathematik.

[Pie91] B. C. Pierce. *Basic Category Theory for Computer Scientists.* MIT Press, 1991.

[Pin91] S. Pinchinat. Ordinal processes in comparative concurrency semantics. In *Proc. 5th Workshop on Computer Science Logic, Bern, LNCS 626*, pages 293–305. Springer-Verlag, October 1991.

[Pin93] S. Pinchinat. *Des Bisimulations pour la Sémantique des Systèmes Réactifs.* Thèse de Doctorat, I.N.P. de Grenoble, France, January 1993.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Lect. Notes, Aarhus University, Aarhus, DK, 1981.

[Pnu85] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th ICALP, Nafplion, LNCS 194*, pages 15–32. Springer-Verlag, July 1985.

[Pra86] V. R. Pratt. Modeling concurrency with partial orders. *Int. J. Parallel Programming*, 15(1):33–71, 1986.

[Pra91] V. R. Pratt. Modelling concurrency with geometry. In *Proc. 18th ACM Symp. Principles of Programming Languages, Orlando*, pages 311–322, January 1991.

[Rei85] W. Reisig. *Petri Nets. An Introduction*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.

[RT88] A. Rabinovich and B. A. Trakhtenbrot. Behavior structures and nets. *Fundamenta Informaticae*, 11(4):357–404, 1988.

[Sas93] V. Sassone. Personal communication, 1993.

[Shi85a] M. W. Shields. Concurrent machines. *The Computer Journal*, 28(5):449–465, 1985.

[Shi85b] M. W. Shields. Deterministic asynchronous automata. In *Formal Methods in Programming*. North-Holland, 1985.

[Sim85] R. De Simone. Higher-level synchronising devices in MEIJE-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[Sin90] A. Sinachopoulos. Partial order logics for elementary net systems: State- and event-approaches. In *Proc. CONCUR'90, Amsterdam, LNCS 458*, pages 442–455. Springer-Verlag, August 1990.

[SNW93] V. Sassone, M. Nielsen, and G. Winskel. A hierarchy of models for concurrency. To appear as Technical Report DAIMI, Computer Science Departament, Aarhus University, 1993.

[Sta89a] E. W. Stark. Compositional relational semantics for indeterminate dataflow networks. In *Proc. Category Theory and Computer Science, LNCS 389*, pages 52–74. Springer-Verlag, 1989.

[Sta89b] E. W. Stark. Concurrent transition systems. *Theoretical Computer Science*, 64:221–269, 1989.

[Vog90a] W. Vogler. Bisimulation and action refinement. SFB-Bericht 342/10/90, Institut für Informatik, TUM, Munich, May 1990.

[Vog90b] W. Vogler. Failures semantics of Petri nets and the refinement of places and transitions. Tech. Report TUM-I9003, Institut für Informatik, TUM, Munich, January 1990.

[Vog91] W. Vogler. Bisimulation and action refinement. In *Proc. STACS'91, Hamburg, LNCS 480*, pages 309–321. Springer-Verlag, February 1991.

[Win89a] G. Winskel. An introduction to event structures. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 364–397. Springer-Verlag, 1989.

[Win89b] G. Winskel. A note on model checking the modal ν-calculus. In *Proc. 16th ICALP, Stresa, LNCS 372*, pages 761–772. Springer-Verlag, July 1989.

[WN92] G. Winskel and M. Nielsen. Models for concurrency. Research Report DAIMI PB-492, Comp. Sci. Dept., Aarhus Univ., November 1992. To appear in the Handbook of Logic in Computer Science, ed. S. Abramsky, D. M. Gabbay and T. S. E. Maibaum.

[Yan93] D. N. Yankelevich. *Parametric Views of Process Description Languages.* PhD thesis, Univ. Pisa, 1993.

AUTORISATION de SOUTENANCE

VU les dispositions de l'Arrêté du 30 mars 1992 relatif aux Etudes doctorales

VU les rapports de présentation de

. Monsieur DARONDEAU Philippe , Directeur de Recherche

. Monsieur GORRIERI Roberto , Professeur

Monsieur ECHAGUE Juan Vicente

est autorisé(e) à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, spécialité " Informatique "

Fait à Grenoble, le 7 juin 1993

/ BV.

Maurice RENAUD
Président
de l'Institut National Polytechnique
de Grenoble

INPG