Contributions à la sémantique du parallélisme : bisimulations pour le raffinement et le vrai parallélisme

Ferroudja Cherief

THESE

Présentée par

Ferroudja CHERIEF

Pour obtenir le grade de DOCTEUR

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

(Arrêté ministériel du 30 mars 1992)

Spécialité : INFORMATIQUE

==========

Contributions à la sémantique du Parallélisme : Bisimulations Pour le Raffinement et le Vrai Parallélisme

==========

Date de soutenance : 8 Octobre 1992

Composition du jury :

J. Sifakis	Président
Ph. Darondeau U. Goltz	Rapporteurs
Ph. Jorrand E. Pelz Ph. Schnoebelen	Examinateurs

Thèse préparée au sein du Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle (LIFIA)

Résumé : Dans ce travail, nous nous intéressons essentiellement aux trois points de vue importants adoptés dans la sémantique du parallélisme: l'étude d'équivalences comportementales compatibles avec le raffinement d'actions, les modèles dits de "vrai parallélisme", et les liens qui existent entre les logiques modales et les équivalences comportementales.

Parmi les équivalences de bisimulation qui traitent les situations où certaines actions sont invisibles ou non observables de l'extérieur, la τ-bisimulation ou équivalence observationnelle de Milner et la η-bisimulation ne sont pas compatibles avec le raffinement d'actions. La bisimulation de branchement et la Δ-bisimulation sont compatibles avec le raffinement d'actions. Ceci nous a suggéré la recherche d'équivalences compatibles avec le raffinement et contenues dans la τ-bisimulation ou la η-bisimulation

Aussi, nous avons défini la notion de bisimulation avant arrière sur les structures d'événements premières qui constituent un modèle de base pour le vrai parallèlisme. Nous avons généralisé cette nouvelle définition à la step, partial word et pomset bisimulation avant arrière. Nous avons comparé ces nouvelles équivalences aux autres équivalences de bisimulation sur les structures d'événements premières. Il s'est avéré essentiellement que la pomset et la partial word bisimulations avant arrière coïncident avec la history preserving bisimulation. Partant de ce résultat, nous avons proposé une logique l_{bf}, adéquate à la history preserving bisimulation. Nous avons aussi montré que la loqique lp de [DNF 90] caractérise elle aussi la history preserving bisimulation.

Mots Clés: Sémantique du Parallélisme, Bisimulation, Actions invisibles, Logique modale, Logique temporelle, Systèmes de transitions, Structures d'événements premières.

Abstract : In this work, we are interested in the three main viewpoints used for the specification of parallel programs: the study of behavioural equivalences that are well behaved against refinement of actions, the so called truly concurrent models, and the link between modal logics and behavioural equivalences. Among all the equivalences that deal with silent actions, τ-bisimulation or Milner's observational equivalence and η-bisimulation are not well behaved w.r.t. refinement of actions while branching and Δ-bisimulations are. This led us looking for the coarsest aquivalences that are well behaved w.r.t refinement and contained in τ-bisimulation or η-bisimulation.

We also propose a definition of back and forth bisimulation on prime event structures. We show that our proposal can be adapted in a uniform way to yield step, partial word, and pomset back and forth bisimulations. It turns out that partial word and pomset back and forth bisimulations both coïncide with history preserving bisimulation.Using this new result, we propose a logic l_{bf}, that narurally characterizes history preserving bisimulation. We also prove that the l_p logic of [DNF 90] also characterizes history preserving bisimulation.

Key Words: Semantics of Concurrency, Bisimulation, Silent actions, Modal logic, Temporal logic, Transition systems, Prime event structures.

Remerciements

Je tiens à remercier les membres du Jury:

Joseph SIFAKIS, Directeur de Recherche au CNRS, de m'avoir fait l'honneur de présider mon jury de thèse.

Phillipe DARONDEAU Chargé de recherche au CNRS (IRISA), d'avoir bien voulu examiner ce travail en tant que rapporteur, et pour toutes ses remarques très constructives

Ursula GOLZ Professeur à l'université de Hildesheim d'avoir accepté de juger ce travail. Je lui témoigne toute ma reconnaissance.

Elisabeth PELZ, Maître de conférences à l'Université de Paris Sud de m'avoir fait l'honneur d'être membre du jury.

Je voudrais également remercier tout particulièrement Philippe JORRAND, Directeur de recherche au CNRS, qui a dirigé cette thèse et qui n'a cessé de me prodiguer ses encouragements et orientations qui m'ont été d'un très grand soutien. Je lui suis très reconnaissante de la grande disponibilité, de l'intérêt et de la patience dont il n'a cessé de faire preuve tout au long de ce travail.

Je tiens à remercier très sincèrement Philippe SCHNOEBELEN chargé de recherche au CNRS (IMAG), qui à travers sa grande attention pour ce travail, n'a ménagé ni son temps, ni ses conseils judicieux pour me faire bénéficier de son expérience. Je lui suis très reconnaissante et j'ai eu grand plaisir à travailler avec lui.

Un grand merci à Sophie PINCHINAT et François LAROUSSINIE avec lesquels j'ai eu beaucoup de plaisir à travailler.

Je tiens également à remercier Zineb HABBAS, Juan ECHAGÜE, Ciryl AUTANT, Wilfried PFISTER pour tous les échanges de vues et discussions que nous avons eues.

Même si je ne peux être exhaustive, je ne pourrai terminer sans exprimer ma reconnaissance à Jacques MOSSIERE , Directeur de L'ENSIMAG. Sans lui il aurait été très difficile de terminer ce travail.

Enfin je profite de cette occasion pour exprimer mon amitié à tous les membres de l'équipe parallélisme du LIFIA.

Sommaire

Introduction

La conception de systèmes parallèles ou réactifs nécessite la réunion de trois éléments :

- un langage de description de systèmes,
- un langage (logique) pour l'étude des propriétés des systèmes,
- des modèles qui sont utilisés comme domaines sémantiques par la logique et le langage de description.

Les systèmes parallèles sont fréquemment décrits à l'aide de langages basés sur les algèbres de processus tels CCS [Mil 80], CSP [Hoa 85], Meije [AB 84] et ACP [BK 84]. Les algèbres de processus décrivent le comportement attendu d'un programme à un certain niveau d'abstraction. Chaque opérateur d'une algèbre représentera une "façon de composer" des systèmes. Cette façon de composer est complètement spécifiée par des règles qui indiquent quels peuvent être les comportements d'un système en fonction des comportements de ses composants. C'est Plotkin [Plo 81] qui a introduit ce style de définition de sémantique dite "opérationnelle" d'un langage.

Suivant la voie ouverte par Milner, la sémantique d'un langage parallèle est déterminée par la donnée de sa sémantique opérationnelle qui modélise le comportement des programmes et d'une relation d'équivalence (entre comportements de programmes) dont le choix est fonction des critères d'abstraction que l'on souhaite prendre en compte. Cette relation d'équivalence doit être une congruence pour tous les opérateurs de combinaison de programmes afin de garantir que deux programmes équivalents (ayant des comportements équivalents) restent indistinguables quel que soit l'environnement dans lequel on les place.

Un processus est une classe d'équivalence de comportements. C'est un objet d'un modèle, (c.à.d. un sens possible pour un programme). Deux programmes sont équivalents si et seulement s'ils définissent le même processus.

Dans la littérature, on rencontre deux grands types d'équivalences sémantiques : les équivalences dites de temps "linéaire" et les équivalences dites de temps "arborescent". Dans le premier cas, le processus défini par un programme est déterminé par ses séquences d'exécutions. L'équivalence de traces en est un exemple classique [Hoa 80]. Dans le second cas, deux programmes équivalents ont même structure de branchement. Le moment du choix non déterministe est respecté. L'équivalence de bisimulation, première équivalence comportementale proposée par Milner [Mil 80] est une équivalence de temps arborescent. Cette équivalence distinguera par exemple le processus a(b+c) du processus ab + ac. Alors que les deux processus ont même ensemble de traces (Ø, a, ab, ac), ils n'ont pas la même structure de branchement : quand le processus a(b+c) a effectué l'action a, il a la possibilité de choisir entre b et c. Ce comportement n'est pas possible dans le processus ab+ac. L'équivalence de bisimulation est plus forte que l'équivalence de traces : elle distingue plus de processus. Il existe plusieurs équivalences entre les traces et la bisimulation. Une étude comparative de ces équivalences se trouve dans [vG 90a], [DN 87].

Suivant l'approche de Milner, dans le chapitre 1 de cette thèse, nous donnons une sémantique à un langage CCS_0, issu de CCS. Considérant les systèmes de transitions comme domaine sémantique, nous construisons une algèbre de systèmes de transitions avec comme équivalence entre systèmes, la bisimulation forte.

Les premières équivalences sémantiques proposées dans la littérature étaient basées sur l'hypothèse d'atomicité des actions que peuvent effectuer les systèmes. Or, il est largement reconnu qu'il est important de pouvoir décrire des systèmes complexes à différents niveaux d'abstraction : une action à un niveau abstrait peut représenter un processus complexe à un niveau plus concret. Le remplacement (raffinement) d'une action par un comportement arbitraire formalise cette notion de non-atomicité des actions. Ainsi, on s'intéresse de plus en plus aux équivalences sémantiques compatibles avec le raffinement d'actions.

Nous avons étudié dans le modèle de graphes ou systèmes de transitions le comportement des équivalences de bisimulation vis à vis de l'opération de raffinement. La bisimulation forte décrite dans le chapitre 1 est une congruence pour le raffinemnet d'actions. Notons que dans cette équivalence, toutes les actions sont considérées visibles ou observables de l'extérieur.

L'un des critères d'abstraction (ou d'observation), le plus célèbre, est celui proposé par Milner pour rendre invisible à l'extérieur le comportement interne d'un processus. La τ-bisimulation ou équivalence observationnelle basée sur ce critère d'abstraction n'est pas compatible avec le raffinement d'actions. Récemment, Van Glabbeek et Weijland [vGW 89] ont proposé la bisimulation de branchement, qui elle aussi traite les situations où certaines actions sont invisibles mais de plus est une équivalence compatible avec le raffinement. Nous renvoyons au chapitre 2 de cette thèse pour les motivations de cette nouvelle équivalence. Dans ce même chapitre, nous avons étudié les diverses équivalences de bisimulation qui traitent l'action invisible τ et nous avons cherché quelle était la plus grande (i.e.fully abstract) équivalence compatible avec le raffinement et contenue dans la τ-bisimulation (ou η-bisimulation). Nous montrons que la Δ-bisimulation est la plus grande congruence pour le raffinement contenue dans la τ-bisimulation et que la quasi-branching bisimulation, une nouvelle équivalence, est la plus grande congruence pour le raffinement contenue dans la η-bisimulation [CS 91], [CS 93]. Il est clair que le choix du modèle de graphes dans ce travail est essentiellement motivé par les résultats établis dans [vGW 89].

L'opération de raffinement ne semble pas être naturelle dans les systèmes de transitions. Par exemple, cette opération n'est pas distributive par rapport à la composition parallèle. Les modèles de graphes comme tous les premiers modèles proposés pour CCS sont intrinséquement séquentiels. Dans ces modèles le parallélisme est réduit au non-déterminisme (les processus a|b et a.b+b.a sont équivalents). Sur ces modèles on parle de sémantiques basées sur l'entrelacement (interleaving en anglais) qui consiste à entremêler les séquences d'actions des processus mis en parallèle. Cette insuffisance des systèmes de transitions a ouvert le grand débat autour de la notion de vrai parallélisme.
Une approche très largement adoptée fut l'étude des sémantiques basées sur des modèles opérationnels incluant la notion d'actions concurrentes. Parmi ces modèles dits de "vrai parallèlisme", les réseaux de Pétri [Pet 64] sont les plus connus. Ces modèles sont basés sur la notion de causalité entre les actions que peut effectuer un système [Pet 64], [Maz 87], [Win 80], [Win 89], [Pom 85], [NPW 81]. Les structures d'événements constituent un modèle de base pour le vrai parallèlisme pour les systèmes où les événements (ou actions) effectués sont les éléments prépondérants.

Dans le chapitre 3, sur les structures d'événements premières, nous étudions les équivalences de bisimulation ainsi que leurs comportements vis à vis du raffinement d'actions. Nous présentons également la notion des ST-sémantiques, mieux adaptées au raffinement.

En définissant la notion de bisimulation vers l'avant et vers l'arrière, De Nicola, Montanari et Vaandrager ont proposé une nouvelle façon de comparer les processus. Dans ces équivalences, les processus se simulent non seulement en faisant les mêmes pas en avant (transitions classiques) mais aussi en faisant des retours en arrière le long de leurs histoires respectives.

Les systèmes parallèles ou réactifs [Pnu 86] sont en général des programmes qui ne terminent pas. Ces systèmes interagissent avec leur environnement au cours de leurs calculs. Ce comportement interactif est plus important que la notion d'état final (s'il existe). Pour étudier les propriétés de tels systèmes, la logique de Hoare [Hoa 69], [Apt 81] ne suffit pas car elle considère un programme comme une relation entre un état initial et un état final. Pour les systèmes parallèles ou réactifs il faut un formalisme capable d'exprimer les propriétés des états rencontrés lors d'une exécution d'un système. La logique temporelle constitue à ce jour, le formalisme le plus adéquat pour la spécification et la vérification de systèmes parallèles. Comme pour les équivalences comportementales, on retrouve la controverse entre les points de vue "linéaire" et "arborescent" : les logiques de temps linéaire expriment des propriétés sur des séquences d'exécution, les logiques de temps arborescent expriment des propriétés d'états dans un graphe.

Dans notre étude, nous nous intéressons plus particulièrement aux liens qui existent entre les logiques temporelles et les équivalences comportementales. Dans le premier chapitre, nous avons consacré la dernière section au rappel du résultat célèbre d'adéquation qui relie la logique HML (Hennessy Milner Logic) à l'équivalence de bisimulation : dans le cadre des systèmes de transitions à branchement fini, deux systèmes satisfont les mêmes formules de la logique HML si et seulement s'ils sont bisimilaires [HM 85].

Dans la première partie du chapitre 4, nous adaptons la notion de bisimulation avant arrière aux structures d'événements premières [Che 92]. De façon uniforme, nous généralisons notre définition à la step, partial word , et pomset bisimulation avant arrière. Nous comparons ces nouvelles équivalences aux bisimulations classiques ainsi qu'aux ST-bisimulations sur les structures d'événements premières. Il s'est avéré que la partial word et la pomset bisimulation avant arrière coïncident avec la "history preserving bisimulation" (équivalence compatible avec le raffinement d'actions). Nous avons étudié le comportement des nouvelles équivalences vis à vis de l'opération de raffinement.

Dans la deuxième partie de ce dernier chapitre, nous proposons deux caractérisations logiques pour la history preserving bisimulation [CLP 92].

Chapitre 1

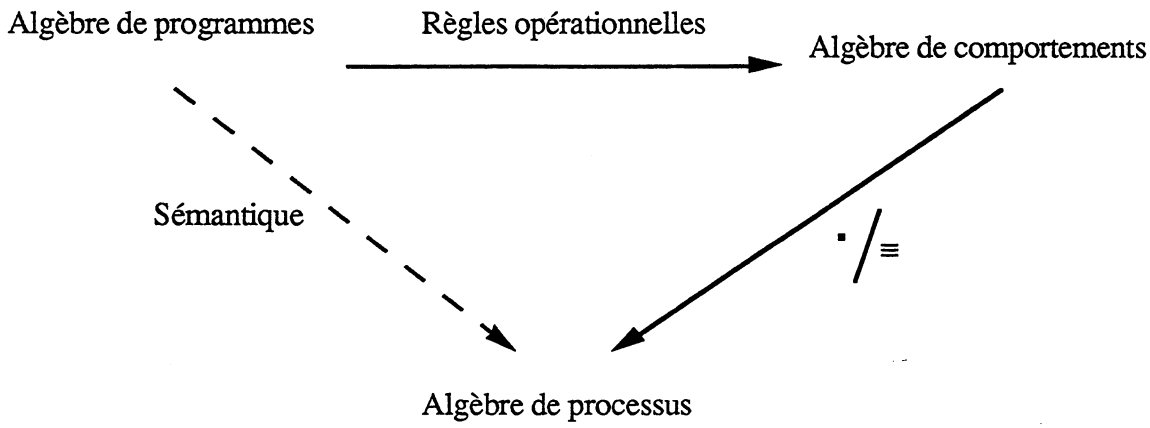
CCS0: L'Algèbre de processus de base

1.1 Introduction

Les systèmes parallèles sont composés d'agents indépendants, communicant au cours de leurs calculs. Aussi, il est reconnu depuis longtemps que la notion mathématique de fonction ne permet pas de donner une sémantique satisfaisante pour les systèmes parallèles et communicants. L'approche la plus simple consiste à étendre les notions bien connues de la théorie des automates. Ainsi Petri [Pet 84] a enrichi cette théorie en ajoutant explicitement le concept de parallélisme, ce qui a donné naissance aux réseaux de Petri.

Une autre approche proposée par Milner [Mil 80] dans son calcul CCS, consiste à introduire des constructeurs permettant de combiner des programmes parallèles. Parmi ces constructeurs figure évidemment la composition parallèle. Il a également introduit un combinateur permettant de rendre invisible à l'extérieur, le comportement interne d'un programme. Il existe de nombreux travaux semblables, en particulier TCSP de Brookes, Hoare et Roscoe [BHR 84].

Le comportement d'un programme se calcule à l'aide de règles opérationnelles et les équivalences "comportementales" de programmes sont obtenues de façon indirecte en définissant les équivalences sur ces comportements, i.e. sur les systèmes de transitions engendrés ou leurs dépliages en arbres. Nous schématisons ceci comme suit:



Un processus est donc une classe d'équivalence de comportements, c'est un objet d'un modèle (c.a.d. un sens possible pour un programme). Deux programmes sont équivalents si et seulement si ils définissent le même processus.
Les relations d'équivalences sur les comportements doivent être des congruences par rapport aux opérateurs de combinaison de programmes afin de garantir que deux programmes équivalents restent équivalents quand on les place dans des environnements arbitraires.

Dans les équivalences sémantiques, on distingue en général:

• les équivalences dites de temps "linéaire" des équivalences de temps "arborescent". Dans le cas linéaire un programme est défini par ses séquences d'exécutions possibles. L'équivalence de traces [Hoa 80] en est un exemple classique. Dans le cas arborescent, deux programmes équivalents ont même structure de branchement, i.e. le moment du choix non déterministe est respecté. C'est le cas de la bisimulation, première équivalence comportementale proposée par Milner [Mil 80].

• Les équivalences sémantiques qui distinguent deux programmes qui ne diffèrent que par leurs comportements internes de celles qui les confondent.

• Les équivalences qui sont basées sur l'atomicité des actions (i.e. les actions ne peuvent pas être remplacées par un comportement arbitraire) de celles qui ne le sont pas.

Par ailleurs, le comportement des systèmes parallèles n'a pas pour objectif d'obtenir un résultat final, mais plutôt le maintien de l'interaction avec l'environnement. Une interaction pouvant contenir éventuellement un état final [Pnu 77], [Pnu 86]. Ainsi il est important de pouvoir parler

des états rencontrés lors d'un calcul. La logique temporelle qui permet d'inclure le temps dans les raisonnements formels constitue un formalisme capable d'exprimer des propriétés des états atteints lors d'une exécution d'un système parallèle.
Comme pour les équivalences comportementales, il existe une controverse entre logique temporelle de temps linéaire et logique temporelle de temps arborescent. Les logiques de temps linéaire expriment des propriétés sur des séquences d'exécution. Les logiques de temps arborescent expriment des propriétés d'états (ou de chemins)dans un graphe. Dans ce cas la structure de branchement est préservée pour chaque état. Nous renvoyons le lecteur intéressé par le débat autour de cette controverse à [BMP 83], [CE 81], [QS 83], [Lam 80], [EH 83].
Dans le même esprit que les travaux de Hoare et Floyd [Hoa 69], [Flo 67], les logiques temporelles peuvent être utilisées pour donner une sémantique à un langage parallèle à l'aide d'équivalence induite par la logique. Deux programmes équivalents satisfont exactement les mêmes formules de la logique. Il existe de nombreux travaux reliant ces équivalences induites par une logique modale aux équivalences comportementales. Le résultat classique étant celui de Hennessy et Milner [HM 85] qui lie la bisimulation à la logique HML.

Dans ce chapitre, nous définissons un langage CCS_0 qui servira de cadre de travail pour la suite. Pour donner une sémantique à notre langage de base, nous associons aux programmes CCS_0 des graphes (ou systèmes de transitions). Ce modèle de graphes est plus fin que la notion mathématique de fonction, il nous permet d'avoir des informations sur les états internes d'un programme. Il repose sur l'idée de pas élémentaires d'un programme (il reste à préciser cette notion). Nous définissons une algèbre de graphes ayant comme signature par l'ensemble des opérateurs de combinaison de termes CCS_0, et toute relation de congruence sur cette algèbre donne alors une sémantique pour CCS_0. Ainsi nous donnons une sémantique à l'aide d'une équivalence arborescente: l'équivalence de bisimulation. Nous étudions le comportement de cette équivalence vis à vis des opérateurs CCS_0 ainsi que vis à vis de l'opération de remplacement d'une action (raffinement) par un comportement arbitraire. Cette opération de raffinement formalise la notion de non atomicité des actions. Dans la deuxième partie du chapitre, nous présentons une caractérisation logique de la bisimulation.

1.2 Le langage CCS_0

Nous introduisons ici l'algèbre de processus CCS_0 issue de CCS de Milner [Mil 80]. Pour décrire la syntaxe de CCS_0, nous nous donnerons les ensembles A et $\overline{A}$ d'actions où A contient a, b, c, ... et $\overline{A}$ et $\overline{a}$, $\overline{b}$, $\overline{c}$, ... et soit $Act = A \cup \overline{A}$.
L'ensemble $P=\{p,q, ...\}$ des processus est donné par la grammaire suivante:

$$P \ni p,q ::= Nil \mid \alpha.p \mid p+q \mid p \| q \mid p \setminus B \mid p\,[\varphi] \quad pour \ \alpha \in Act \cup \{\tau\} \ et \ B \subseteq Act$$

Notons que τ, l'action interne ou invisible résulte de la synchronisation entre processus.
- *Nil* est le processus inactif,
- $\alpha.p$ est le processus p préfixé par une action α quelconque,
- $p+q$ est la somme non déterministe des processus p et q,
- $p \| q$ est la composition parallèle (avec communication) des processus p et q,
- $P \setminus B$ (restriction) où B étant un sous ensemble de Act, est le processus p où les actions qui sont dans B ne sont plus possibles. La restriction permet de localiser (de rendre internes) certaines communications.
- $P[\varphi]$ (renommage) où φ est une application de Act dans Act est le processus p où les actions ont été renommées par φ. Le renommage permet d'établir les communications souhaitées.

Nous noterons plus simplement le processus *a.Nil* et *ab* le processus *a.b.Nil*.
Un processus est en fait un terme de l' algèbre T_Σ avec $\Sigma = \{Nil, +, \|, \alpha, \beta, \ldots\}$.

Dans ce chapitre, nous interprétons les programmes CCS_0 en termes de graphes (ou machines non-déterministes). Ainsi, dans la section qui suit nous définissons une algèbre de graphes ayant comme signature l'ensemble des opérateurs de combinaison de termes CCS_0, et toute relation de congruence sur cette algèbre de graphes donne alors une sémantique pour CCS_0.

1.3 Les modèles de graphes

Dans cette section nous définissons une sémantique pour CCS_0 en termes de graphes de transitions.

1.3.1 Les graphes de transitions

Definition 1.1 (graphe étiqueté ou système de transitions)
Un graphe étiqueté est un quadruplet (Q, Act, $\rightarrow$, q_0) où

- Q est un ensemble $\{r,s, q_1,q_2,\ldots\}$ d'états ou noeuds,
- Act est un ensemble $\{a,b,\ldots\}$ d'actions,
- $\rightarrow \subseteq Q \times Act \times Q$ est la relation de transition,
- $q_0 \in Q$ est l'état initial ou racine du graphe.

L'ensemble Act d'actions sera laissé implicite quand cela n'introduit pas d'ambiguïtés.

Une transition (r, a, s) $\in \rightarrow$ est notée r $\xrightarrow{a}$ s. Elle dénote la possibilité de passer de l'état r à l'état s quand l'action a est effectuée. s est un successeur de r (ou un a-successeur). Comme il est usuel, Act* désigne l'ensemble des mots sur Act (suite finie d'actions). Nous noterons u.v la concaténation de v et de u. On peut alors étendre aux suites d'actions la relation de transition "$\rightarrow$" dans un système donné. Plus précisément, la relation $\rightarrow$ se généralise à $\Rightarrow \subseteq Q \times Act^* \times Q$. Par suite, si $w = a_1.a_2...a_n$ est une séquence d'actions, et si $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \rightarrow ... \xrightarrow{a_n} q_n$ est une suite de transitions, on notera $q_0 \overset{w}{\Rightarrow} q_n$. En particulier $q \overset{\lambda}{\Rightarrow} q$ où λ est la séquence vide. Un état q_n est dit accessible ou atteignable à partir de l'état q_0 s'il existe une suite finie de transitions telle que: $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \rightarrow ... \xrightarrow{a_n} q_n$. Un graphe est dit *connecté* quand tous ses états sont atteignables à partir de la racine. Notons qu'il est toujours possible d'élaguer un graphe en ne conservant que ses états atteignables. Si Q_r dénote l'ensemble des états atteignables à partir de la racine, le graphe élagué est défini comme suit: pour $g = (Q, \rightarrow, q_0)$, Elague(g) = $(Q_r, \rightarrow \cap (Q_r \times Act \times Q_r), q_0)$.

Remarque : Elague(Elague(g)) = Elague(g).

L'identité des noeuds ne nous intéresse pas. C'est la structure des graphes qui est importante. Ainsi deux graphes qui ne diffèrent que par les noms de leurs états sont considérés identiques.

Définition 1.2 (Isomorphisme de graphes)
Deux graphes g et h sont isomorphes, noté $g \equiv h$, s'il existe une bijection f R entre les noeuds de g et h telle que:
i) Les racines de g et h sont reliées par R
ii) Si r R s et r' R s', alors r $\xrightarrow{a}$ r' est une transition de g ssi s $\xrightarrow{a}$ s' est une transition de h.

Comme les états non atteignables ne jouent aucun rôle et que seule la structure d'un graphe nous intéresse, on définit:

Définition 1.3 (Equivalence structurelle)
Deux graphes g et h sont structurellement équivalents, noté $g \backsimeq h$ si Elague(g) $\equiv$ Elague(h).

La remarque ci-dessus permet d'établir que Elague(g) $\backsimeq$ g.

Dans la suite, deux graphes structurellement équivalents seront considérés identiques. Il sera alors toujours possible de considérer que deux graphes ont des ensembles d'états disjoints. Les définitions ne dépendront pas du représentant choisi.

Dans les représentations graphiques, nous indiquons un état initial par une flèche incidente et les arêtes sont étiquetées par des actions.

Exemple la figure 1.1 est un exemple de graphe étiqueté g sur un ensemble d'actions Act= {a, b, c} . g a trois états dont un est initial. g est de la forme (Q, Act, $\rightarrow$, q_1) avec Q={q_1, q_2, q_3, q_4 }, $\rightarrow$ = {(q_1, c, q_2), (q_2, b, q_3), (q_2, a, q_4) , (q_4, a, q_1)}.

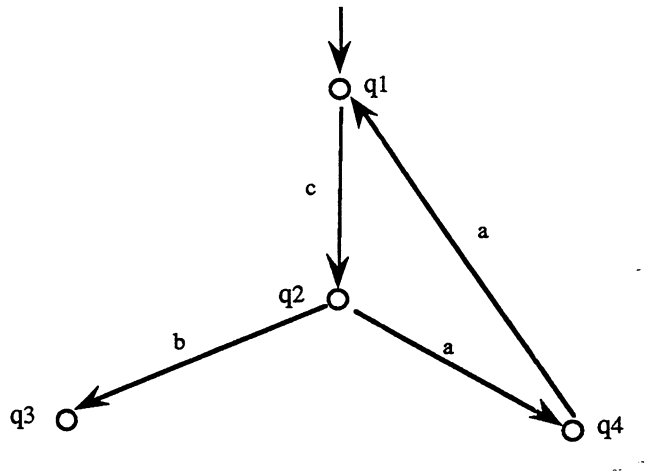


Figure 1.1: Un graphe étiqueté

Definition 1.4 (graphe de transitions)
Un graphe de transitions est un graphe *étiqueté* et *connecté*.

1.3.2 Les opérations sur les graphes

Un graphe est dit *à racine initiale* si il n'a aucune arête ayant comme extrémité finale la racine. Il est *non trivial* s'il contient au moins une arête. Nous notons *G* l'ensemble des graphes de transitions à racine initiale et non triviaux.

Definition 1.5 (Préfixage par une action)
Pour tout graphe g = (Q, Act, $\rightarrow$, q_1) de *G* et action a de Act, le graphe a.g est défini par a.g = (Q $\cup$ {q_0}, Act, $\rightarrow \cup$ {(q_0, a, q_1)} , q_0) .

a.g est obtenu à partir de *g* en ajoutant un nouvel état q_0, qui sera la racine de *a.g* ainsi qu'une a-transition (transition étiquetée par a) de q_0 vers la racine q_1 de *g*.

On suppose dorénavant que tous les graphes sont étiquetés sur le même alphabet Act, omis par la suite.

Exemple La figure 1.2 donne l'exemple du processus a.c + a préfixé par l'action a .

Definition 1.6 (Somme non déterministe)
Pour tous graphes $g_1 = (Q_1, Act, \rightarrow_1 , q_1)$ et $g_2 = (Q_2, Act, \rightarrow_2 , q_2)$ de *G* $g_1 + g_2$ est défini par
$g_1 + g_2 = (Q_1 \cup Q_2 \cup \{q_0\}, \rightarrow_1 \cup \rightarrow_2 \cup \rightarrow_0 , q_0)$

où $\rightarrow_0 \subseteq \{q_0\} x \ Act \ x (Q_1 \cup Q_2)$ est définie par :
$q_0 \xrightarrow{a}_0 q$ ssi $q_1 \xrightarrow{a}_1 q$ ou $q_2 \xrightarrow{a}_2 q$.

$g_1 + g_2$ est obtenu en conservant tous les états de g_1 et g_2 et en remplaçant q_1 et q_2 par q_0 qui sera la racine de $g_1 + g_2$. Cette racine aura comme transitions possibles, les successeurs de q_1 et q_2 réunis.

Si on ne considère que la structure des graphes, l'opération de choix non déterministe (+) est associative, commutative et admet un élément neutre.

Proposition 1.1
Pour tous graphes g_1, g_2 et g_3

$$g_1 + g_2 \simeq g_2 + g_1$$
$$g_1 + (g_2 + g_3) \simeq (g_1 + g_2) + g_3$$
$$g_1 + NIL \simeq g_1$$

où NIL est le graphe qui contient un seul état et une relation de transition vide.

Exemple La figure 1.3 est un exemple de somme non déterministe de $g_1 = ac + b$ et de $g_2 = a$.

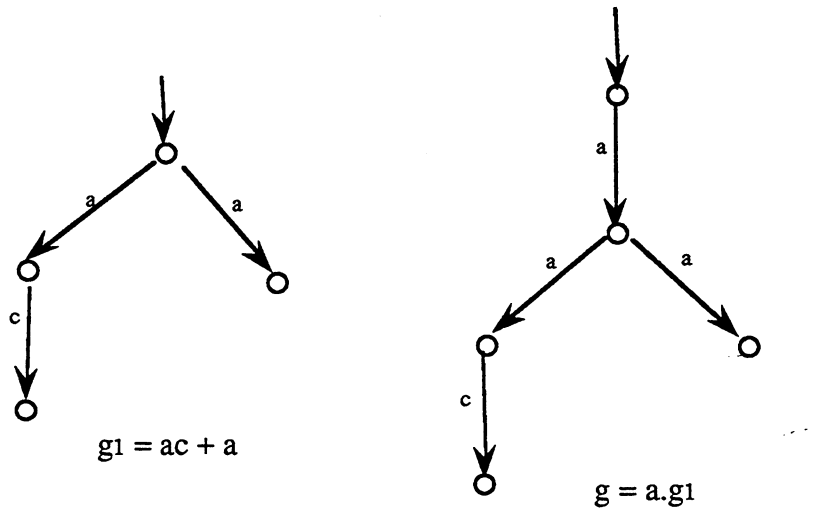


Figure 1.2 : Préfixage

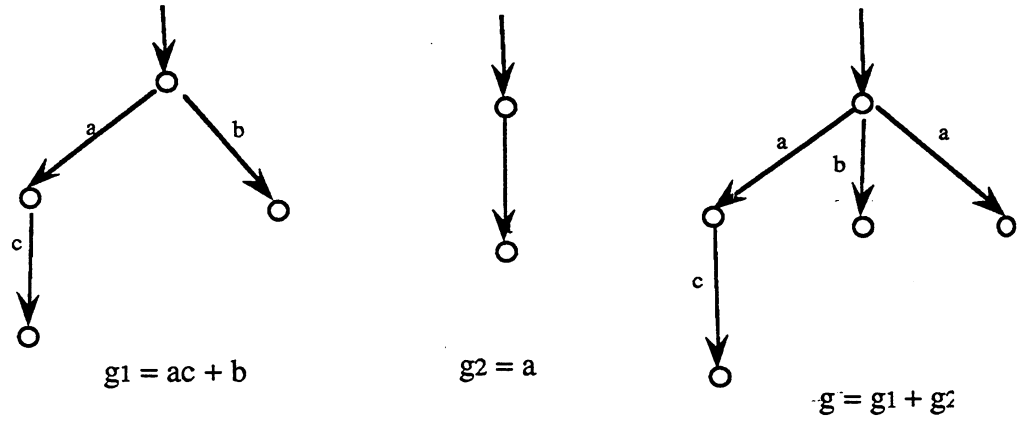


Figure 1.3 : Somme non déterministe

Definition 1.7 (composition parallèle)

Pour tous graphes $g_1 = (Q_1, \rightarrow_1, q_1)$ et $g_2 = (Q_2, \rightarrow_2, q_2)$ de G, $g_1 \| g_2$ est défini par:

$g_1 \| g_2 = (Q_1 \times Q_2, \rightarrow_1 \| \rightarrow_2, q_1 \times q_2)$ avec:

- $(r,s) \xrightarrow{\tau} (r',s') \in \rightarrow_1 \| \rightarrow_2$ si $\exists\, a \in Act$, $r \xrightarrow{a} r' \in \rightarrow_1$ et $s \xrightarrow{\bar{a}} s' \in \rightarrow_2$
- $(r,s) \xrightarrow{\alpha} (r',s) \in \rightarrow_1 \| \rightarrow_2$ si $r \xrightarrow{\alpha} r' \in \rightarrow_1$ pour $\alpha \in Act \cup \{\tau\}$
- $(r,s) \xrightarrow{\alpha} (r,s') \in \rightarrow_1 \| \rightarrow_2$ si $s \xrightarrow{\alpha} s' \in \rightarrow_2$ pour $\alpha \in Act \cup \{\tau\}$

Un état de g_1 en parallèle avec g_2 est une paire d'un état de g_1 et d'un état de g_2, une transition de $g_1 \| g_2$ est soit une transition de g_1, soit une transition de g_2, soit encore l'occurrence simultanée d'une a-transition de g_1 et d'une $\bar{a}$-transition de g_2 ou l'inverse.

On notera que l'opération de mise en parallèle est commutative, associative et admet un élément neutre modulo l'équivalence structurelle.

Proposition 1.2

Pour tous graphes g_1, g_2 et g_3

$$g_1 \| g_2 \simeq g_2 \| g_1$$
$$g_1 \| (g_2 \| g_3) \simeq (g_1 \| g_2) \| g_3$$
$$g_1 \| \mathrm{NIL} \simeq g_1$$

Exemple La figure 1.4 donne un exemple de composition parallèle de a.b et $\bar{a}$.c

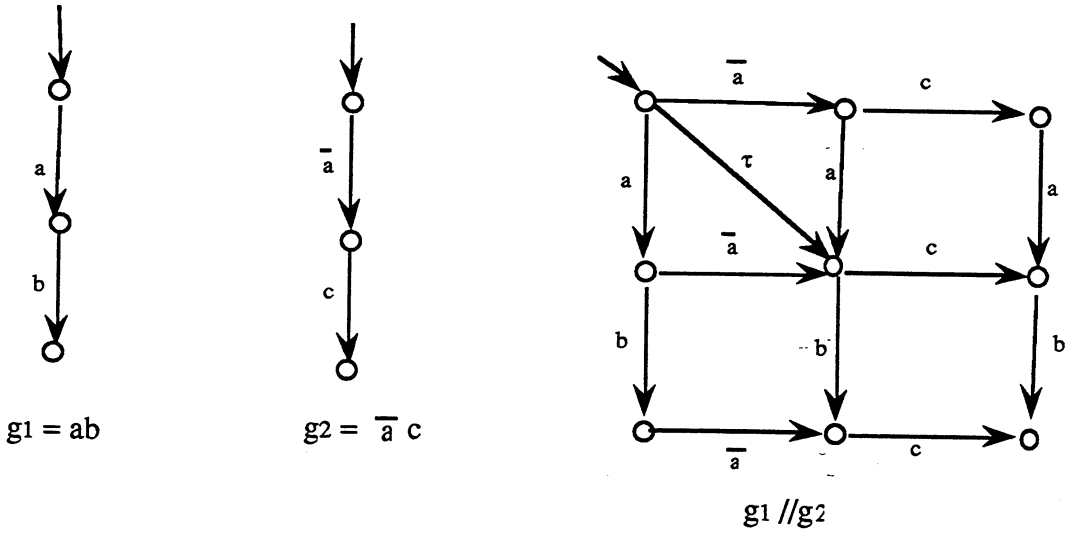


Figure 1.4 : Composition parallèle

Definition 1.8 (Restriction)

Pour tout graphe $g = (Q, \rightarrow, q)$ de G et toute partie $B \in \mathcal{P}(\mathrm{Act})$, le graphe g\B est défini par: $g\backslash B = (Q, \rightarrow_{\backslash B}, q)$ où $(q, a, q') \in \rightarrow_{\backslash B}$ ssi $(q, a, q') \in \rightarrow$ et $a, \bar{a} \notin B$.

Le graphe g\B est le graphe g auquel on retire toutes les transitions étiquetées par une action a si a ou $\bar{a}$ est dans B. On notera g\a quand $B = \{a\}$.

Exemple La figure 1.5 contient l'exemple des graphes $g_1 = a.c.b + a.b$ et $g = g_1\backslash c$.

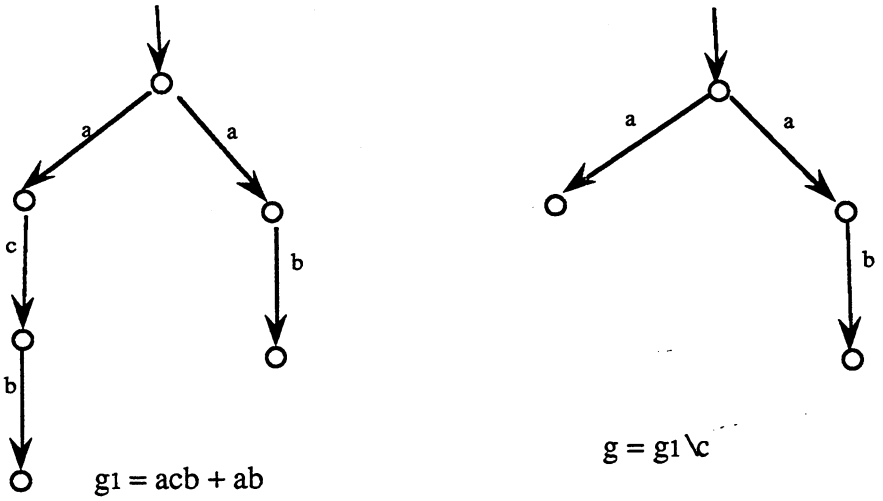


Figure 1.5 : Restriction

Definition 1.9 (Renommage)

Pour tout graphe $g = (Q, \rightarrow, q)$ de G et toute application φ: Act $\rightarrow$ Act, le graphe $g[\varphi]$ est défini par: $g[\varphi] = (Q, \rightarrow_{[\varphi]}, q)$ où $(q, e', q') \in \rightarrow_{[\varphi]}$ ssi il existe $q \xrightarrow{e} q'$ telle que $\varphi(e)=e'$.

Le graphe $g[\varphi]$ a la même structure d'états que g, mais les actions étiquetant les transitions ont été remplacées par leurs images par φ. On notera $g[a_1/b_1, a_2/b_2,..., a_n/b_n]$ pour $g[\varphi]$ où φ est l'application qui renomme a_i en b_i pour $i \in [1..n]$ et laisse le reste de Act inchangé.

1.4 Equivalences de bisimulations

Dans cette section, nous introduisons d'abord l'équivalence de bisimulation forte, puis la bisimulation modulo un critère d'abstraction.

Toutes les équivalences de bisimulation seront formellement définies sur l'ensemble des graphes de transitions, noté *G*. Dans toute la suite, pour tous graphes g et h de *G*, nous appellerons une relation symétrique entre les noeuds de g et h, une relation $R \subseteq$ noeuds(g)xnoeuds (h) $\cup$ noeuds(h) x noeuds (g) telle que r *R* s si et seulement si s *R* r. Notons que pour toute relation binaire *R*, on emploiera indifféremment les trois notations équivalentes : r*R*s, *R*(r, s) et (r,s)$\in$ *R*.

1.4.1 Bisimulation forte

La notion de bisimulation forte a été introduite par [Par 81], utilisée par Milner dans [Mil 83]. Cette même notion a été formulée différemment par Milner dans [Mil 80].

Dans une sémantique de bisimulation, l'état d'un programme à un instant donné est complètement déterminé par la donnée de l'ensemble de ses reconfigurations, c.à.d. l'ensemble des actions effectuées à cet instant et ce que l'on peut connaitre des états atteints.

On dit que deux programmes se trouvent dans des états équivalents si et seulement si toute séquence d'actions effectuées par l'un est reproduite par l'autre et que tous les états correspondants atteints lors de ces exécutions sont à leur tour équivalents (la structure de branchement de chaque état est respectée). L'équivalence de bisimulation formalise cette notion d'équivalence de comportements de programmes.

Dans toutes les définitions d'équivalences, nous considérons des graphes de transitions avec des ensembles d'états disjoints.

Definition 1.10 (bisimulation forte)
Deux graphes g et h de G sont fortement bisimilaires s'il existe une relation symétrique $R \subseteq$ noeuds(g) x noeuds(h) $\cup$ noeuds(h)x noeuds(g) (appelée bisimulation forte) telle que:

i) les racines de g et h sont reliées par R

ii) pour toute transition r $\xrightarrow{a}$ r' de g ou h et état s de g ou h tel que r R s, il existe une transition s $\xrightarrow{a}$ s' dans g ou h telle que r' R s'

Cette propriété de transfert est schématisée par la figure 1.6.

On notera g $\leftrightarrow$ h quand g et h sont deux graphes fortement bisimilaires.

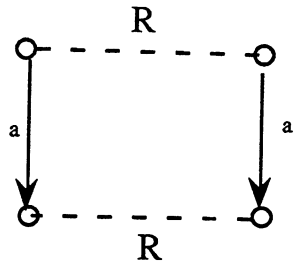
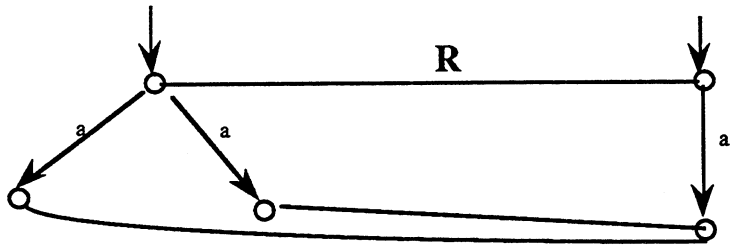


Figure 1.6

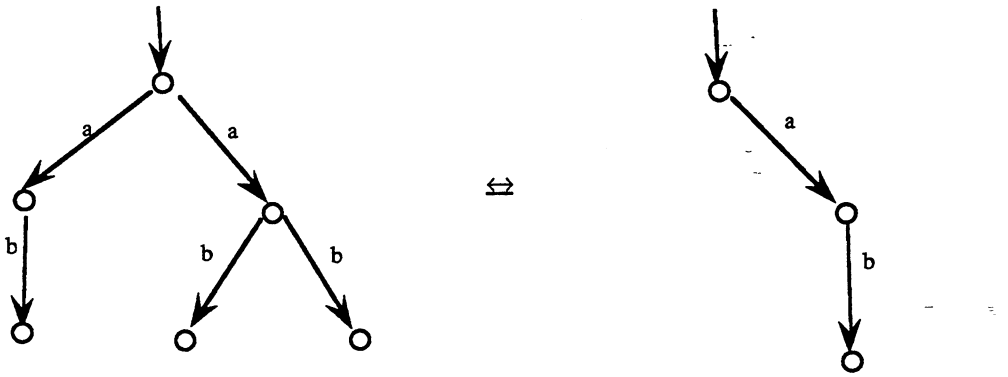
La bisimulation est une relation d'équivalence sur *G*.
Il est facile de vérifier que l'union arbitraire de relations de bisimulation forte est une bisimulation forte. Aussi, si deux graphes g et h sont bisimilaires, l'équivalence $\leftrightarrows$ est la plus grande bisimulation entre g et h

Exemple les deux graphes de la figure 1.7 sont bisimilaires par la relation **R** représentée sur la figure. La figure 1.8 contient un autre exemple de deux graphes bisimilaires.



$g_1 = a + a \leftrightarrows g_2 = a$

Figure 1.7



$g_1 = ab+a(b+b)$ $\leftrightarrows$ $g_2 = ab$

Figure 1.8

Les graphes g_1 et g_2 de la figure 1.9 ne sont pas bisimilaires: en effet supposons que *R*: $g_1 \leftrightarrows g_2$. Les points (i) et (ii) de la définition 1.10 exigent $q_0 \, R \, q'_0$. On a $q_0 \xrightarrow{a} q_1$ dans g_1. Il faut donc qu'un a-successeur (état atteignable par une a-transition) de q'_0 soit relié à q_1 par *R* . Mais

ni q'_1 ni q'_2 ne peuvent simuler les possibilités de q_1. Par exemple q'_1 n'a pas de transition possible par c.

1.4.2 C-bisimulations

La bisimulation forte est rarement utilisée telle quelle dans la pratique. Par exemple, dans le cadre de la vérification, elle ne permet pas de rendre compte de la différence d'abstraction qui existe entre une spécification et une implémentation d'un programme.
Une approche adoptée fut l'établissement de critère d'abstraction ou d'observation.

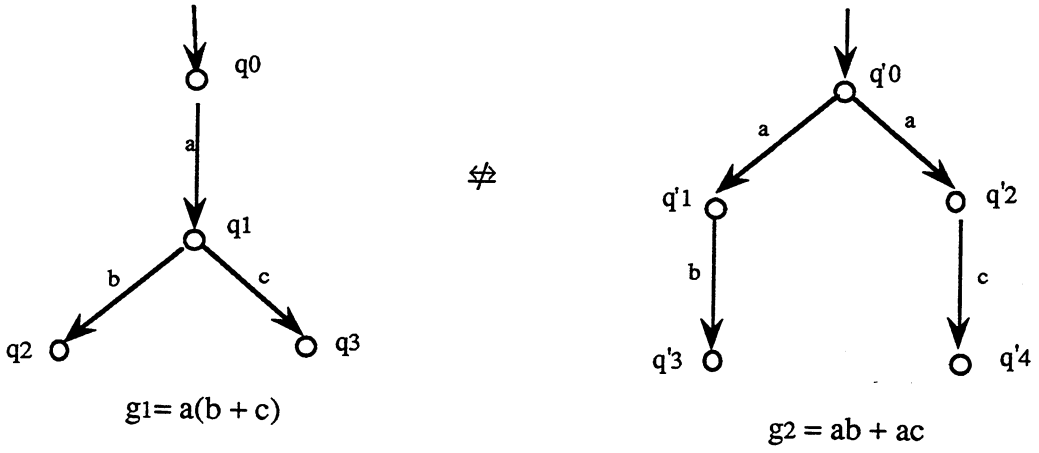


Figure 1.9

L'abstraction permet de faire varier la notion de pas de calcul obtenant ainsi diverses équivalences de bisimulations. On ne considère plus les actions atomiques ou "concrètes" que peut effectuer un système mais ses actions abstraites [Bou 85] ou actions observables, constituées d'un ensemble de séquences d'actions concrètes. Un critère d'abstraction ou d'observation est une partition partielle sur l'ensemble des séquences d'actions concrètes du système c.à.d. un ensemble $C = \{s_i \mid i \in N\}$ "d'actions abstraites" $s_i \subseteq Act^*$ deux à deux disjointes.

Definition 1.11 (Critère d'observation)
Soit $S = (Q, Act, \rightarrow, q_0)$ un système de transitions, un critère d'observation C de S est une congruence du monoïde Act^*, i.e. $u \; C \; u'$ et $v \; C \; v' \Rightarrow u.v \; C \; u'.v'$
Une classe d'équivalence est un ensemble de séquences d'actions considérées comme ayant le même effet visible de l'extérieur. Ainsi $[u]_C$ est appelée un observable ou une action abstraite.

Deux processus sont dits C-bisimilaires s'il existe une relation entre leurs états respectifs compatible avec les transitions observées selon un critère C, i.e., chacun peut simuler l'autre suivant le critère d'observation C en atteignant des états C-bisimilaires.

Definition 1.12 (C-bisimulation)
Deux graphes g et h de *G* sont C-bisimilaires avec C un critère d'observation sur Act, s'il existe une relation symétrique *R* entre les noeuds de g et h (appelée C-bisimulation) telle que:
i) les racines de g et h sont reliées par *R*
ii) pour toute transition $r \stackrel{u}{\Rightarrow} r'$ de g ou h ($u \in Act^*$), et état s de h tel que r *R* s, il existe une transition $s \stackrel{v}{\Rightarrow} s'$ dans g ou h tel que u *C* v et r' *R* s'.
Cette propriété de transfert est schématisée par la figure 1.10

On notera $g \backsimeq_C h$ quand g et h sont deux graphes C-bisimilaires.

La C-bisimulation est une relation d'équivalence. L'union de C-bisimulation est une C-bisimulation.

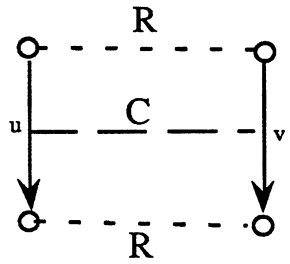


Figure 1.10

Notons que la bisimulation forte est une C-bisimulation avec un critère d'observation dit "fort" car il s'agit de l'égalité syntaxique entre les actions .

Notons que l'isomorphisme de graphes est une bisimulation bijective. Ainsi il est clair que deux graphes structurellement équivalents sont bisimilaires.

$$g \backsimeq h \Rightarrow g \leftrightarrows h$$

1.4.3 Bisimulation et équivalence forte

La première équivalence comportementale a été définie dans [HM 80] et [Mil 80] sous le nom d'équivalence forte ou congruence forte dans [Mil 80]. Cette équivalence est définie comme

l'intersection d'une suite décroissante de relations. Ces relations sont définies sur des expressions CCS qui sont aussi des états d'un système de transitions. Sur le domaine de graphes elle est définie comme suit :

Définition 1.13 (équivalence forte)
Pour tous états r, s d'un graphe de transitions, a une action de Act :

- $r \sim_0 s$ toujours vraie
- $r \sim_{k+1} s$ si et seulement si :
  - pour toute transition $r \xrightarrow{a} r'$, il existe $s \xrightarrow{a} s'$ tel que $r' \sim_k s'$.
  - et réciproquement, pour toute transition $s \xrightarrow{a} s'$, il existe $r \xrightarrow{a} r'$ tel que $r' \sim_k s'$.

On définit $\sim$ par :

- $r \sim s$ si et seulement si pour tout $k \in N$: $r \sim_k s$.

Intuitivement, deux états r et s sont distingués s'ils sont distingués à un niveau n fini.

Deux graphes $g_1 = (Q_1, \rightarrow_1, q_1)$ et $g_2 = (Q_2, \rightarrow_2, q_2)$ de G, sont fortement équivalents si $q_1 \sim q_2$.
On notera $g_1 \sim g_2$ quand g_1 et g_2 sont deux graphes fortement équivalents.

Définition 1.14
Soit $g = (Q, \rightarrow, q_0)$ un graphe de transitions:

i) $q \in Q$ est à branchement fini si l'ensemble $\{q' \in Q \mid q \xrightarrow{a} q'\}$ est fini pour tout $a \in Act$.
ii) g est à branchement fini si tous ses états atteignables sont à branchement fini.

Dans le cadre du branchement fini, la bisimulation forte distingue exactement ce que l'équivalence forte distingue [Mil 88], [BBK 87].

1.4.4 Propriétés de la bisimulation forte

L'équivalence de bisimulation entre graphes de transitions hérite des propriétes des équivalences plus fortes: la proposition 1.1 permet d'établir que l'opération de choix non déterministe est commutative et associative modulo la bisimulation forte. NIL est un élément neutre.

Pour tous graphes g_1, g_2 et g_3 de G

$$g_1 + g_2 \leftrightarrows g_2 + g_1$$
$$g_1 + (g_2 + g_3) \leftrightarrows (g_1 + g_2) + g_3$$
$$g_1 + NIL \leftrightarrows g_1$$

Nous avons la loi d'idempotence qui est aussi vérifiée : pour tout graphe g de G,

$$g + g \leftrightarrows g$$

La proposition 1.2 permet d'établir que l'opération de mise en parallèle est commutative, associative et admet un élément neutre pour l'équivalence de bisimulation.
Pour tous graphes g_1, g_2 et g_3 de G

$$g_1 \parallel g_2 \leftrightarrows g_2 \parallel g_1$$
$$g_1 \parallel (g_2 \parallel g_3) \leftrightarrows (g_1 \parallel g_2) \parallel g_3$$
$$g_1 \parallel NIL \leftrightarrows g_1$$

1.4.5 Définition par point fixe de la bisimulation

Soient $g_i = (Q_i, \rightarrow_i, I_i)$, i = 1, 2 deux graphes de processus. Soit $(\mathcal{P}(Q_1 x Q_2), \subseteq)$ le treillis complet des relations sur $Q_1 x Q_2$ où $\mathcal{P}(Q_1 x Q_2)$ est l'ensemble des relations binaires sur $Q_1 x Q_2$.
On définit une fonction $\mathcal{B} : \mathcal{P}(Q_1 x Q_2) \rightarrow \mathcal{P}(Q_1 x Q_2)$ par :
Si $R \subseteq Q_1 x Q_2$ alors $(q_1, q_2) \in \mathcal{B}(R)$ si et seulement si pour toute action a de Act :

- si $q_1 \xrightarrow{a} q'_1$ alors il existe $q_2 \xrightarrow{a} q'_2$ avec $(q'_1, q'_2) \in R$
- si $q_2 \xrightarrow{a} q'_2$ alors il existe $q_1 \xrightarrow{a} q'_1$ avec $(q'_1, q'_2) \in R$

Une relation R sur $Q_1 x Q_2$ est une bisimulation forte si et seulement si $R \subseteq \mathcal{B}(R)$

Proposition 1.3 [Mil 89]
L'équivalence de bisimulation $\leftrightarrows$ est le plus grand point fixe de $\mathcal{B}$: $\leftrightarrows = \mathcal{B}(\leftrightarrows)$

1.5 Propriètés de congruences de la bisimulation

Une des propriétés importantes de la bisimulation est qu'elle se comporte bien vis-à-vis des opérations de combinaison de graphes i.e. l'équivalence de bisimulation est une congruence

pour les opérations de préfixage par une action, de somme non déterministe de composition parallèle, de restriction et de renommage.

Proposition 1.3
Pour tous graphes g, g_1, g_2 de G , e de Act, B sous ensemble de Act et φ une application de Act dans Act :

$$g_1 \leftrightarrows g_2 \Rightarrow \; e.g_1 \leftrightarrows e.g_2$$
$$g_1 + g \leftrightarrows g_2 + g$$
$$g_1 \| g \leftrightarrows g_2 \| g$$
$$g_1 \setminus B \leftrightarrows g_2 \setminus B$$
$$g_1 [\varphi] \leftrightarrows g_2 [\varphi]$$

1.6 Bisimulations et raffinements d'actions

Dans les équivalences de bisimulation définies ci-dessus, les actions que peuvent effectuer les processus sont considérées "atomiques" ou indivisibles. Par ailleurs, dans la conception descendante de systèmes distribués, il est important de modéliser les processus à différents niveaux d'abstraction: les actions à un niveau abstrait peuvent représenter des processus complexes à un niveau plus concret. Cette méthodologie n'est donc pas compatible avec la non divisibilité des actions. A cet effet, Pratt[Pra 86], Lamport[Lam 86], et beaucoup d'autres ont plaidé pour l'utilisation d'équivalences sémantiques qui ne soient pas basées sur l'atomicité des actions. De telles équivalences permettent aussi une preuve de programmes parallèles à différents niveaux d'abstraction.

Par ailleurs, Castellano, De Michelis et Pomello[CDMP 87] ont souligné que le concept d'atomicité d'actions pouvait être formalisé par la notion de raffinement d'actions. Une équivalence sémantique se comporte bien vis à vis de l'opération de raffinement, si deux processus équivalents restent équivalents quand on remplace toutes les occurrences d'une action par un processus arbitraire raf(a). Raf(a) peut être le processus $a_1.a_2$ par exemple. Raf(a) est un graphe de G. Le raffinement est une application de Act dans G.

Definition1.15 (graphe raffiné)
Soit g = (Q, $\rightarrow$, q_0) un graphe de G et raf :Act $\rightarrow$ G, un application de l'ensemble des actions dans l'ensemble des graphes de transitions. Nous appelons feuille ou noeud terminal d'un

graphe un noeud qui n'a pas de successeur. Un noeud interne est un noeud qui n'est ni une racine ni une feuille. Le graphe raffiné raf(g) est construit comme suit:

Pour toute transition $r \xrightarrow{a} r'$ ($a \in Act$) de g

i) identifier r avec la racine d'une copie raf(a) du graphe raf(a).

ii) identifier r' avec tous les noeuds terminaux de raf(a).

iii) supprimer la transition $r \xrightarrow{a} r'$.

Notons que nous n'aurons jamais à identifier r et r', puisque dans *G* nous ne considérons que des graphes non triviaux. Il est également intéressant de souligner que tous les noeuds ou états du graphe g sont des noeuds de raf(g) et que l'action interne τ ne peut être remplacée par un graphe.

Exemple la figure 1.11 contient l'exemple du graphe ab+ac où l'action a est remplacée par le processus $a_1.a_2$

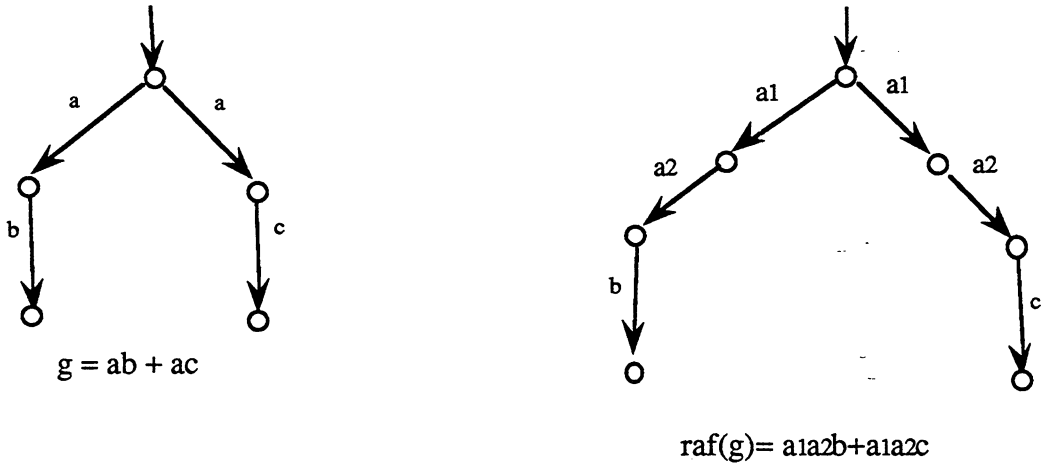


Figure 1.11 : raffinement de graphe

Definition 1.16 (compatibilité avec le raffinement)
Une relation d'équivalence $\approx$ est dite compatible avec le raffinement d'actions si pour toute action a de Act et applications raf, raf': Act $\rightarrow$ *G*, nous avons :

i) $g \approx h \Rightarrow raf(g) \approx raf(h)$,

ii) $raf(a) \approx raf'(a) \Rightarrow raf(g) \approx raf'(g)$

Une relation d'équivalence est donc compatible avec le raffinement d'actions, si cette équivalence est une congruence pour toute opération de raffinement raf.

Pour étudier le comportement des équivalences de bisimulation par rapport au raffinement d'actions, nous définissons une relation entre les noeuds des graphes raffinés.

Definition 1.17 (relation entre graphes raffinés)
Soit raf : Act $\rightarrow G$, une application de l'ensemble des actions dans l'ensemble des graphes de transitions G. Soit R: g $\leftrightarrows$ h une relation de bisimulation entre les graphes g et h. La relation raf(R) est la plus petite relation entre les noeuds de raf(g) et raf(h) telle que:
i) $R \subseteq$ raf(R)
ii) si r $\xrightarrow{a}$ r' est une transition de g et s $\xrightarrow{a}$ s' une transition de h (a $\in$ Act) telles que R(r,s) et R(r',s') et que ces deux transitions sont remplacées respectivement par des copies $\underline{\text{raf(a)}}$ et $\overline{\text{raf(a)}}$ de raf(a), alors les noeuds de $\underline{\text{raf(a)}}$ et $\overline{\text{raf(a)}}$ sont reliés par raf(R) si et seulement s'ils sont des copies du même noeud de raf(a).

Remarques: - Sur les noeuds de g et h, la relation raf(R) est égale à R.

- Si deux noeuds r et s sont reliés par raf(R) dans les graphes raffinés, r est un noeud de g si et seulement si s est un noeud de h.

La bisimulation forte est compatible avec le raffinement d'actions:

Proposition 1.5
Pour tous graphes g et h de G , a de Act et raf,raf' :Act $\rightarrow G$ deux applications de l'ensemble des actions visibles dans l'ensemble des graphes de transitions

$$\text{i) } g \leftrightarrows h \Rightarrow \text{raf(g)} \leftrightarrows \text{raf(h)}$$
$$\text{ii) raf(a)} \leftrightarrows \text{raf'(a)} \Rightarrow \text{raf(g)} \leftrightarrows \text{raf'(g)}$$

où raf(g), raf'(g) et raf(h) sont les graphes raffinés.

Preuve
Nous montrons uniquement le point i).
Soit R: g $\leftrightarrows$ h, montrons raf(R) : raf(g)$\leftrightarrows$ raf(h)
Nous prouvons cette implication en vérifiant les points (i) et (ii) de la définition 1.10

i) Il est clair que les racines de raf(g) et raf(h) sont reliées pa raf(R)

ii) Supposons raf(R)(r,s) et $r \xrightarrow{a} r'$ une transition de raf(g) (le cas où $r \xrightarrow{a} r'$ est une transition de raf(h) est similaire). Deux cas sont alors possibles:

1) Les noeuds r et s proviennent de g et h

. Si r' est aussi un noeud de g, nous avons nécéssairement s' dans h tel que $s \xrightarrow{a} s'$ et R(r',s') donc raf(R)(r',s').

.Sinon dans g nous avons nécessairement une transition $r \xrightarrow{b} r^*$ et $r \xrightarrow{a} r'$ est une copie d'une transition partant de la racine de raf(b). D'après R: $g \leftrightarrows h$ il existe s^* dans h tel que $s \xrightarrow{b} s^*$ et $R(r^*,s^*)$. Par construction de raf(R), il existe s' dans raf(h) tel que r' et s' soient des copies du même noeud de raf(b) et donc $s \xrightarrow{a} s'$ et raf(R)(r', s').

2) Les noeuds r et s proviennent des copies de raf(b) et $\overline{\text{raf(b)}}$ d'un graphe raf(b). r et s ne sont ni des copies de la racine, ni des copies de feuilles de raf(b). Il s'en suit nécessairement que $r \xrightarrow{a} r'$ est une transition de raf(b). raf(R)(r,s) implique que r et s sont des copies du même noeud de raf(b). Donc il existe $s \xrightarrow{a} s'$ dans $\overline{\text{raf(b)}}$ tel que r' et s' soient des copies du même noeud de raf(b). Donc par construction de raf(R) nous avons raf(R)(r',s'). □

1.7 Bisimulation et logique temporelle

Dans cette section, nous étudions les liens qui existent entre la logique HML et l'équivalence de bisimulation.
La définition de la logique HML suppose donné un ensemble Act = {a,b,...} d'actions.

Définition 1.18 (Syntaxe de HML)
Les formules de la logique HML sont données par la grammaire suivante :

$\varphi, \varphi' ::= T \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \langle a \rangle \varphi$ où a est une action de Act.

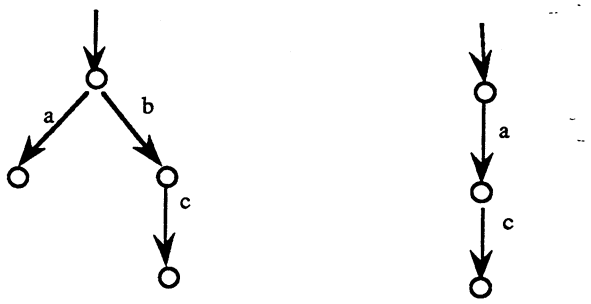
Nous interprétons les formules de HML sur les modèles de graphes.
Etant donné un graphe $g = (Q, \rightarrow, q_0)$ de G, on définit une relation sémantique $\models_g \subseteq Q \times HML$. On écrit $q \models_g \varphi$ quand $(q, \varphi) \in \models_g$. On dit que l'état q satisfait la formule φ dans

le modèle g ou que φ est vraie dans l'état q. Le nom du modèle sera omis quand cela n'introduit pas d'ambiguités.
On dira que le graphe g satisfait une formule φ, noté $g \models \varphi$ si son état initial la satisfait ($q_0 \models_g \varphi$).
Dans les exemples nous utiliserons les abréviations habituelles : on écrit F pour ¬T, $\varphi \vee \varphi'$ pour $\neg(\neg\varphi \wedge \neg\varphi')$, $[a]\varphi$ pour $\neg\langle a\rangle\neg\varphi$

Définition 1.19 (Sémantique de HML)
Soit $g = (Q, \rightarrow, q_0)$ un graphe de transitions, la relation $\models \subseteq Q \times HML$ est définie inductivement sur la structure de φ comme suit :

- $q \models T$ toujours
- $q \models \neg\varphi$ ssi $q \not\models \varphi$
- $q \models \varphi \wedge \varphi'$ ssi $q \models \varphi$ et $q \models \varphi'$
- $q \models \langle a\rangle \varphi$ ssi il existe $q \xrightarrow{a} q'$ telle que $q' \models \varphi$

Exemple : $p \models \langle a\rangle([b]F \vee [c]F)$ signifie qu'il est possible qu'après avoir effectué l'action a, le processus se trouve dans un état où il ne peut pas effectuer l'action b ou l'action c.
Dans l'exemple de la figure 1.12 $g_1 \models \langle a\rangle [c]F$ alors que $g_2 \not\models \langle a\rangle [c]F$.



$g_1 = a + bc$ $g_2 = ac$

Figure 1.12

La logique HML définit une équivalence ($\sim_{HML}$) sur les graphes : deux graphes sont $\sim_{HML}$ équivalents s'ils satisfont exactement les mêmes formules HML.
Nous écrirons $HML(g) = \{\varphi \in HML \mid g \models \varphi\}$, l'ensemble des formules HML satisfaites par g.

Définition 1.20 (Equivalence HML)
Pour tous graphes g_1, g_2 de *G* :

$$g_1 \sim_{HML} g_2 \text{ si et seulement si } HML(g_1) = HML(g_2)$$

Dans le cas des graphes à branchement fini, l'équivalence $\sim_{HML}$ distingue exactement ce que la bisimulation distingue.

Proposition 1.6 [HM 85]
Pour tous graphes $g_1 = (Q_1, \rightarrow_1, I_1)$, $g_2 = (Q_2, \rightarrow_2, I_2)$ à branchement fini :

$$g_1 \leftrightarrows g_2 \Leftrightarrow g_1 \sim_{HML} g_2$$

Exemple : Les graphes de la figure 1.9 ne sont pas bisimilaires donc pas $\sim_{HML}$ équivalents. Seul le graphe $g_1 \models \langle a \rangle (\langle b \rangle T \wedge \langle c \rangle T)$.

1.8 Conclusions

Dans ce chapitre, nous avons présenté un langage de base CCS_0. Nous avons donné sa sémantique à l'aide de l'équivalence de bisimulation forte qui préserve fortement la structure arborescente des graphes. Nous avons vu que cette équivalence se comporte bien vis à vis de tous les opérateurs considérés. De plus, c'est une équivalence compatible avec le remplacement d'actions par un comportement arbitraire. Nous avons également rappelé le résultat connu qui relie la bisimulation à la logique modale HML. Aussi il est bien connu que la bisimulation forte preserve l'ensemble des propriétés que l'on peut exprimer dans les logiques temporelles telles que CTL* [CES 83], LTAC [QS 83]. C'est aussi une équivalence importante d'un point de vue algorithmique : il existe un algorithme efficace de calcul du quotient d'un graphe de processus pour cette équivalence. Cet algorithme est utilisé en pratique par la plupart des méthodes de vérifications pour d'autres équivalences de bisimulations [Mou 92], [PT 87], [Fer 90].

Dans ce chapitre, toutes les actions considérées sont observables ou visibles de l'extérieur. Il existe des bisimulations qui traitent les situations où certaines actions sont invisibles. Nous étudions ces bisimulations dans le chapitre 2.

Chapitre 2

τ-bisimulations et full abstraction pour le raffinement d'actions

2.1 Introduction

Dans les langages comme CCS [Mil 80], [Mil 89], CSP[Hoa 78], [Hoa 80]et ACP[BK 85], les opérateurs de combinaison de processus offrent la possibilité de décrire des processus en les construisant de façon hiérarchique à partir de processus plus simples. Pour vérifier le comportement de tels processus, seules leurs possibilités de communication avec l'extérieur nous intéressent. Le comportement interne d'un processus est formalisé par la notion d'abstraction. La notion d'abstraction est une des caractéristiques essentielles des algèbres de processus puisqu'elle permet de cacher des actions, donc de les rendre invisibles ou non observables.

Dans ACP_τ de Bergstra et Klop [BK 85], l'abstraction est une opération qui renomme une action en une action silencieuse appelée τ. Dans CCS de Milner, les transitions par des actions silencieuses résultent de la synchronisation entre processus.

Il est intéressant de noter que dans l'équivalence de bisimulation forte définie dans le premier chapitre, l'action invisible τ n'a fait l'objet d'aucun traitement particulier. Pour cette équivalence toutes les actions sont considérées observables de l'extérieur.

Dans ce chapitre, nous nous proposons d'étudier les équivalences de bisimulation spécialement adaptées aux situations où certaines actions sont invisibles. Il s'agit principalement de l'équivalence observationnelle de Milner [Mil 80], aussi connue sous le nom de τ-bisimulation. Nous présentons également trois autres variantes de cette équivalence, en l'occurrence la bisimulation de branchement (branching bisimulation), la Δ-bisimulation et la η-bisimulation. Toutes ces équivalences sont plus fortes que l'équivalence observationnelle (i.e. elles identifient moins de processus) . Nous étudions les propriétés de congruence de ces équivalences par rapport aux opérateurs classiques de combinaison de graphes et au raffinement d'actions. Il s'avère que la bisimulation de branchement et la Δ-bisimulation sont compatibles avec le raffinement alors que la τ-bisimulation et la η-bisimulation ne le sont pas. Cette dernière propriété nous a conduit à chercher la plus grande (i.e. fully abstract) équivalence compatible avec le raffinement et contenue dans la τ-bisimulation (respectivement dans la η-bisimulation). Nous montrons que la Δ-bisimulation est la plus grande congruence pour le raffinement contenue dans la τ-bisimulation et que la quasi-branching bisimulation, une nouvelle équivalence, est la plus grande congruence pour le raffinement contenue dans la η-bisimulation [CS 91], [CS 93]. Nous terminons le chapitre en présentant d'autres caractérisations de la bisimulation de branchement [DNMV 90]. Ces caractérisations sont basées sur la notion de bisimulation avant arrière.

2.2 τ-bisimulations

2.2.1 τ-bisimulation ou équivalence observationnelle

Dans l'équivalence observationnelle de Milner, on ne peut distinguer par la seule observation un système effectuant une ou plusieurs actions internes d'un système ne faisant aucune action. Par suite, toute transition par une action visible d'un processus est reproduite par l'autre processus modulo un nombre arbitraire de τ-transitions avant et après l'action observable. Par exemple si un processus effectue la séquence de transitions $r \xrightarrow{a} r_1 \xrightarrow{\tau} r_2 \xrightarrow{b} r'$ où a et b sont des actions de Act, l'observateur verra l'occurrence de a, puis au bout d'un certain temps, celle de b. Pour l'observateur tout se passe comme s'il s'agissait de le séquence $r \xrightarrow{a} \xrightarrow{b} r'$. Nous écrivons $r \Rightarrow s$ s'il existe un chemin $r \xrightarrow{\tau} r_1 \xrightarrow{\tau} r_2 \ldots \xrightarrow{\tau} s$. $\Rightarrow$ est la fermeture réflexive transitive de $\xrightarrow{\tau}$. Pour $a \in Act \cup \{\tau\}$, nous écrivons $r \stackrel{a}{\Rightarrow} r'$ s'il existe r_1, r_2 tel que $r \Rightarrow r_1 \xrightarrow{a} r_2 \Rightarrow r'$.

Dans toutes les définitions qui suivent les relations d'équivalence seront définies sur l'ensemble G des graphes de transitions non triviaux et à racine initiale. De plus, nous supposons que les graphes de G ont des ensembles d'états disjoints.

Définition 2.1 (τ-bisimulation ou équivalence observationnelle)[wei 89]
Deux graphes g et h de G sont τ-bisimilaires s'il existe une relation symétrique $R \subseteq$ noeuds(g) x noeuds(h) $\cup$ noeuds(h)x noeuds(g) (appelée une τ-bisimulation) telle que:
i) les racines de g et h sont reliées par R
ii) pour toute transition $r \xrightarrow{a} r'$ de g ou h ($a \in Act \cup \{\tau\}$), et état s de g ou h tels que rRs,
- soit $a = \tau$ et $r' R\ s$
- soit il existe un chemin dans g ou h de la forme $s \Rightarrow s_1 \xrightarrow{a} s_2 \Rightarrow s'$ tel que $r' R\ s'$

La "propriété de transfert" peut être schématisée par la figure 2.1.

On notera $g \leftrightarrows_\tau h$ quand g et h sont deux graphes τ-bisimilaires.

La τ-bisimulation est une relation d'équivalence et l'union arbitraire de τ-bisimulations est une τ–bisimulation. Ainsi si $g \leftrightarrows_\tau h$, il existe une plus grande τ-bisimulation entre g et h.

La τ-bisimulation peut aussi être définie comme une C-bisimulation avec un critère d'observation dit faible: c'est la congruence sur $(Act \cup \{\tau\})^*$ engendrée par l'équation $\tau = \lambda$ où λ dénote la séquence vide : deux séquences équivalentes ont évidemment le même contenu visible.

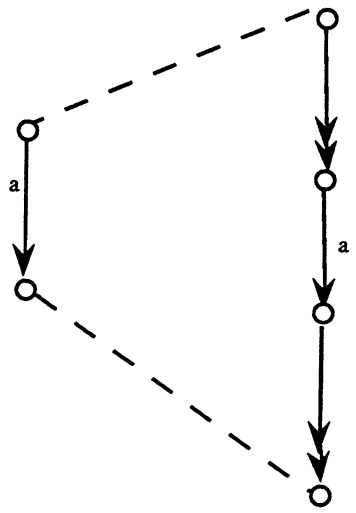


Figure 2.1: τ-bisimulation

Exemple Les deux graphes de la figure 2.2 sont τ-bisimilaires par la relation **R** représentée. La figure 2.3 donne un autre exemple de deux graphes τ-bisimilaires.

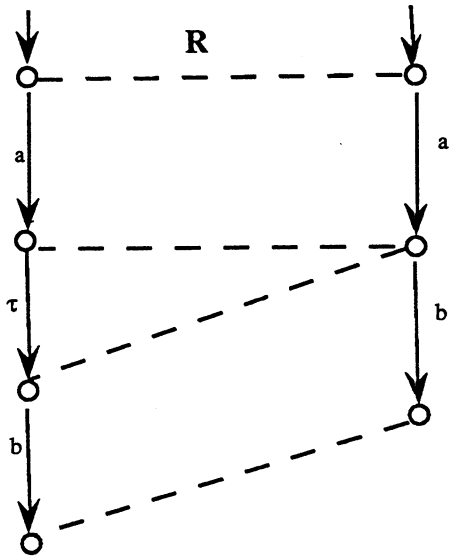
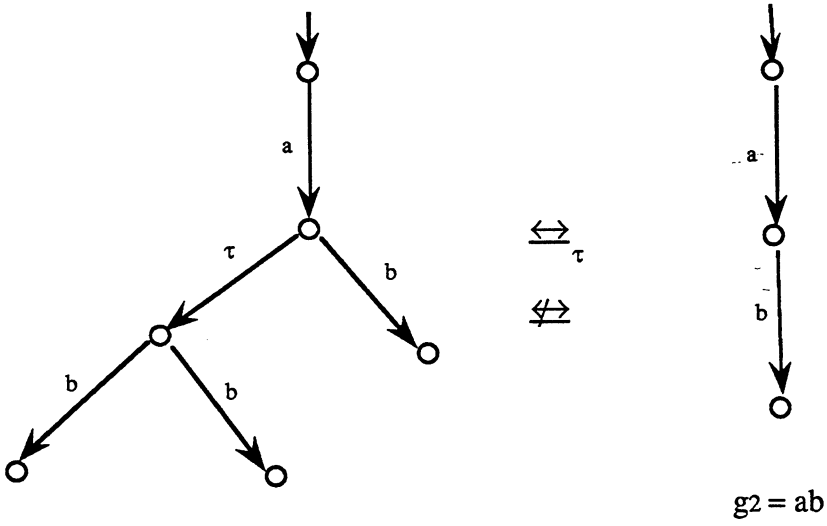


Figure 2.2



$g_1 = a(\tau(b+b)+b)$

$g_2 = ab$

Figure 2.3

Les graphes de la figure 2.4 ne sont pas τ-bisimilaires: supposons R:$g_1 \leftrightarrows_\tau g_2$. Les points (i) et (ii) de la définition 2.1 exigent $q_0 R q'_0$. Si la transition $q_0 \xrightarrow{a} q_1$ de g_1 est imitée par la transition $q'_0 \xrightarrow{a} q'_1$ de g_2 alors $q_1 R q'_1$. $q'_1 \xrightarrow{\tau} q'_2$ est une transition possible de g_2. Mais q'_2 ne peut être relié par R à q_1: puisque q'_2 n'a pas de transition possible par c. $q_0 \xrightarrow{a} q_1$ ne peut pas non plus être imité par $q'_0 \stackrel{a}{\Rightarrow} q'_2$ car q'_2 ne peut être relié à q_1 par R : q'_2 n'a pas de successeur par c.
Ainsi les conditions de la définition 2.1 ne peuvent pas être vérifiées.

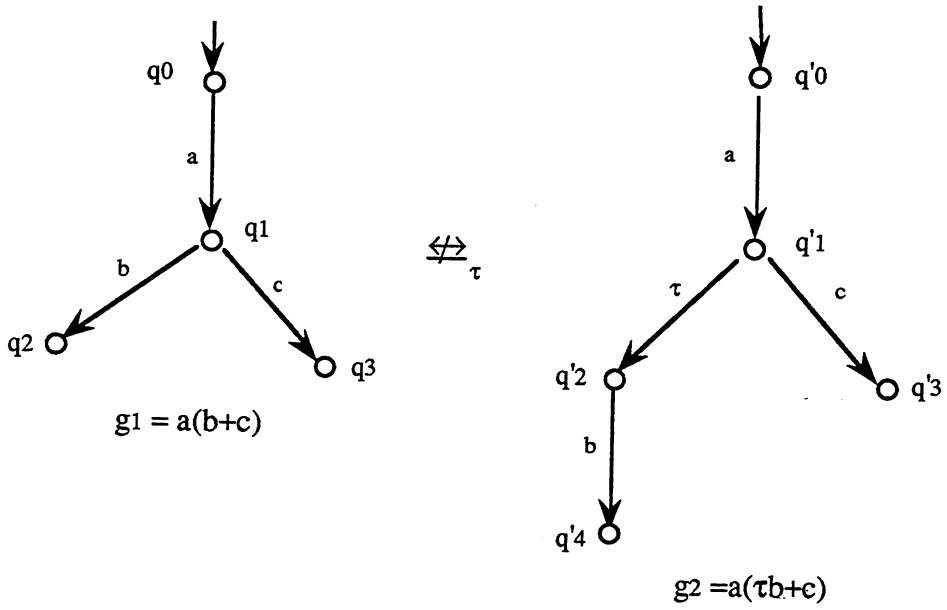


Figure 2.4

Il est clair que la bisimulation forte ($\leftrightarrows$) est plus fine que la τ-bisimulation: la bisimulation forte permet de distinguer plus de processus. En effet, toute bisimulation forte est une τ-bisimulation. Nous avons donc la proposition suivante.

Proposition 2.1
Pour tous graphes g et h de G

$$g \leftrightarrows h \Rightarrow g \leftrightarrows_\tau h$$

Cette implication est stricte. Par exemple les graphes g_1 et g_2 de la figure 2.3 sont τ-bisimilaires mais pas bisimilaires.

2.2.2 Bisimulation de branchement, Δ-bisimulation, η-bisimulation

L'équivalence observationnelle de Milner ne préserve pas la structure de branchement des systèmes. En effet, elle ne respecte pas la notion intuitive d'égalité de comportements sur laquelle repose la bisimulation. Nous avons vu au premier chapitre que toute séquence d'actions effectuées par un système doit être reproduite par l'autre de sorte que les états atteints lors de ces exécutions aient les mêmes possibilités d'évolution. Il est important de noter que dans la définition de l'équivalence observationnelle, un pas par une action visible d'un processus est reproduit par l'autre modulo un nombre arbitraire de τ-transitions avant et après l'action visible. Il est évident que dans ce cas, le statut ou les potentialités des états intermédiaires ont été complètement ignorés. Pour expliciter ceci, nous reproduisons dans la figure 2.5 l'exemple donné dans [Wei 89]. Notons d'abord que les graphes g_1, g_2, g_3 sont τ-bisimilaires.

Les arêtes étiquetées par b ajoutées dans les graphes g_2 et g_3 introduisent des nouvelles possibilités de calcul. A partir de son état initial g_2 peut effectuer la séquence d'actions visibles : $r_0 \xrightarrow{a} r_1 \xrightarrow{b} r_3$. g_1 imite ce comportement en effectuant la séquence $q_0 \xrightarrow{a} q_1 \xrightarrow{\tau} q_2 \xrightarrow{b} q_3$ contenant évidemment les mêmes actions visibles. Clairement les états intermédiaires des deux séquences n'ont pas les mêmes possibilités d'évolution : en effet à partir de l'état q_2, il est possible d'exécuter l'action d_2. Ceci n'est pas possible dans la séquence effectuée par g_2. Le moment du choix non déterministe n'est pas respecté. Notons que quand g_2 a choisi d'effectuer b à partir de l'état r_1 il a choisi en même temps de ne pas effectuer d_1 et d_2 alors que nous venons de voir que ce comportement est imité dans g_1 par un chemin contenant un état (q_2) où l'action d_2 est possible alors que d_1 ne l'est plus.

De façon similaire, quand g_3 effectue la séquence $s_0 \xrightarrow{a} s_1 \xrightarrow{\tau} s_2 \xrightarrow{b} s_4$, g_1 l'imite en effectuant la séquence $q_0 \xrightarrow{a} q_1 \xrightarrow{\tau} q_2 \xrightarrow{b} q_3 \xrightarrow{\tau} q_4$. Cette dernière séquence contient un état (q_3) à partir duquel il possible d'effectuer l'action d_3. Ce comportement n'est pas possible dans g_3. Dans g_3 le choix d'effectuer $s_2 \xrightarrow{b} s_4$ exclut en même temps la possibilité d'effectuer l'action d_3. Ce comportement est imité dans g_1 par un chemin où le choix d'effectuer ou pas l'action d_3 a lieu quand l'action b a effectivement eu lieu.

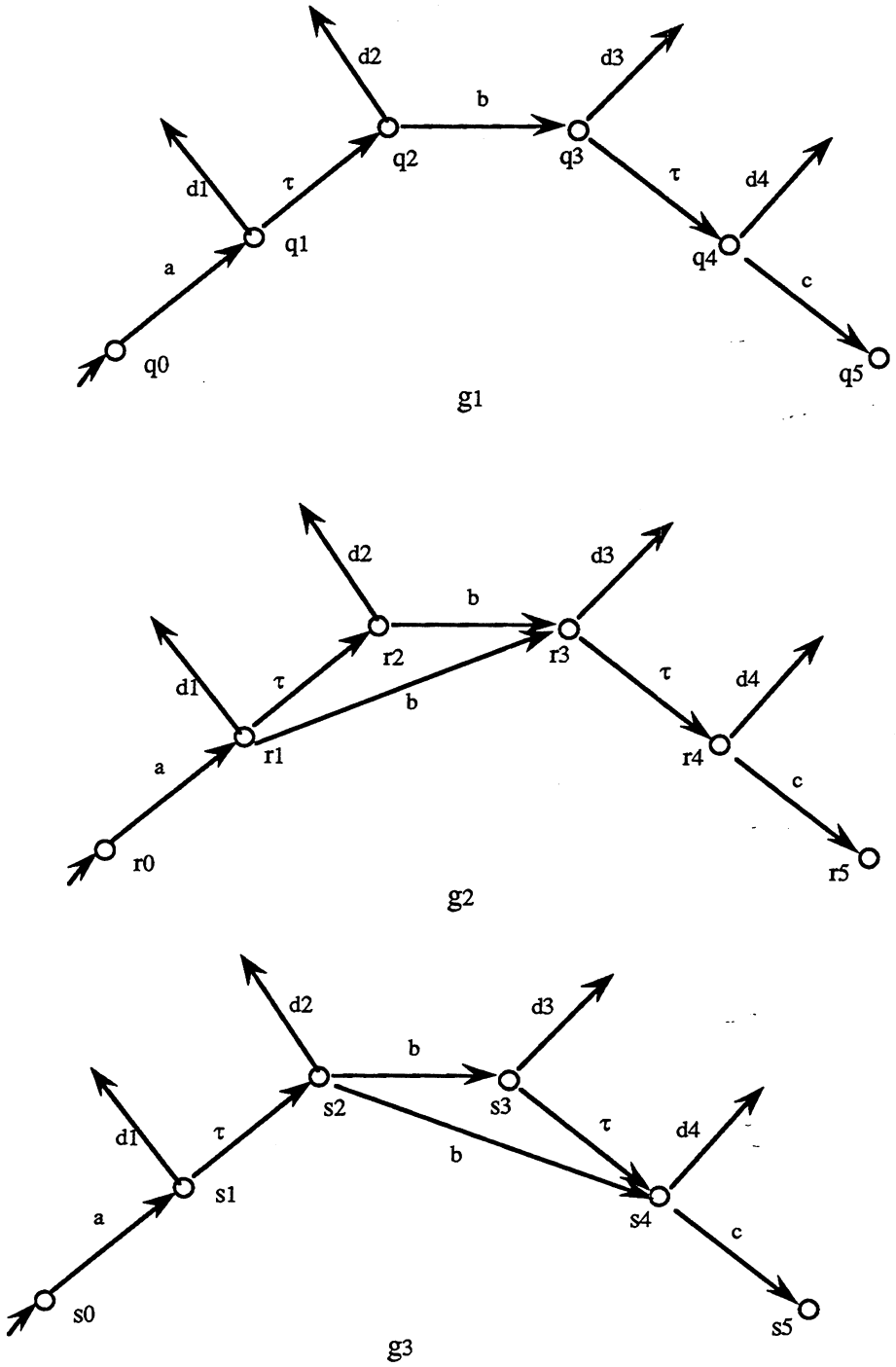


Figure 2.5

Pour remédier à cet inconvénient, Van Glabbeek et Weijland [vGW 89] ont proposé la bisimulation de branchement, une nouvelle équivalence qui préserve totalement la structure arborescente d'un processus. Cette équivalence avait déjà été définie dans [BCG 88] (sous le nom de "stuttering equivalence") sur les structures de Kripke.

Dans cette section, nous définissons donc la bisimulation de branchement ainsi que deux autres variantes de l'équivalence observationnelle: la Δ-bisimulation [Wei 89], [Wal 88] et la η-bisimulation [BvG 87].

Définition 2.2 (bisimulation de branchement)
Deux graphes g et h de G sont "branching bisimilaires" s'il existe une relation symétrique $R \subseteq$ noeuds(g) x noeuds(h) $\cup$ noeuds(h)x noeuds(g) (appelée une bisimulation de branchement) telle que:
i) les racines de g et h sont reliées par R
ii) pour toute transition $r \xrightarrow{a} r'$ de g ou h ($a \in Act \cup \{\tau\}$), et état s de g ou h tels que $r R s$,
- soit $a = \tau$ et $r' R s$
- soit il existe un chemin dans g ou h de la forme $s \Rightarrow s_1 \xrightarrow{a} s_2 \Rightarrow s'$ tel que $rR\ s_1$, $r'Rs_2$ et $r'R\ s'$.

On notera $g \leftrightarrows_b h$ quand g et h sont deux graphes branching bisimilaires.

Remarque 1: la deuxième possibilité du point ii) est équivalente à: il existe dans g ou h un chemin de la forme $s \Rightarrow s_1 \xrightarrow{a} s'$ avec $r\ R\ s_1$ et $r'\ R\ s'$. C'est cette définition équivalente que nous utiliserons dans les preuves.

La bisimulation de branchement est une relation d'équivalence sur G. L'union arbitraire de bisimulations de branchement est une bisimulation de branchement. Ainsi si $g \leftrightarrows_b h$, il existe une plus grande bisimulation de branchement entre g et h.

La propriété de transfert de la bisimulation de branchement peut être schématisée par la figure 2.6.

Exemple La figure 2.7 contient l'exemple des graphes $g_1 = a(b+a)$ et $g_2 = a(\tau(b+a)+b)$ qui sont branching bisimilaires.

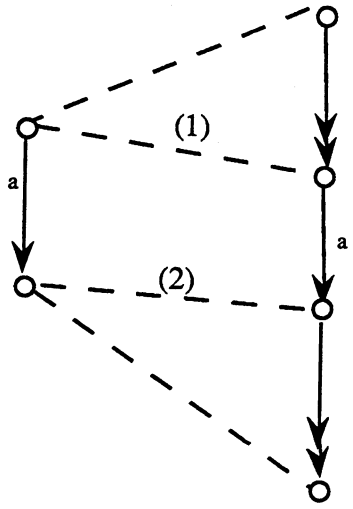


Figure 2.6: bisimulation de branchement

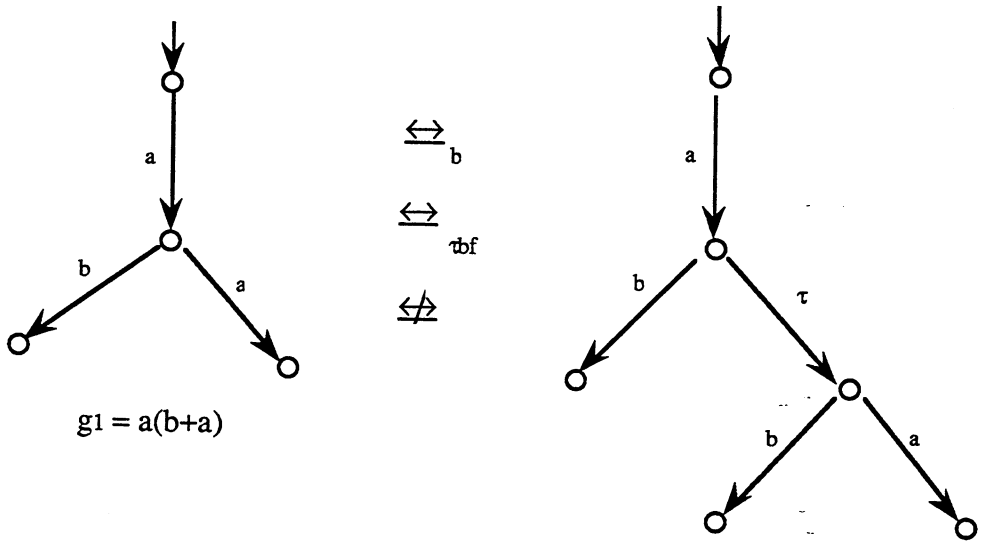


Figure 2.7

La figure 2.6 est la figure 2.1 avec en plus les liens marqués (1) et (2). La bisimulation de branchement exige une condition supplémentaire sur les états intermédiaires s_1 et s_2 ($rR\ s_1$ et $r'R\ s_2$). En d'autres termes, la branching bisimulation préserve totalement la structure arborescente des processus: les deux extrémités d'un chemin par des τ-transitions sont reliées au même noeud. C'est le cas de $s \Rightarrow s_1$ où s et s_1 sont reliés par R à r et de $s_2 \Rightarrow s'$ où s_2 et s' sont reliés par R à r'. Dans cette bisimulation, seules les τ-transitions qui ne provoquent pas de changements d'état du système sont ignorées: de telles τ-transitions relient des noeuds ayant exactement les mêmes potentialités (ceci se montre avec les traces colorées).

La définition 2.2 peut être renforcée en exigeant que tous les états entre s et s_1 soient reliés par R à r et que ceux entre s_2 et s' soient reliés par R à r', donnant ainsi lieu à une nouvelle définition. Le lemme suivant montre que cette nouvelle définition est équivalente à la définition 2.2.

Lemme 2.1 [vGW 89] (stuttering lemma)
Soit R la plus grande bisimulation de branchement entre deux graphes g et h de G. s'il existe un chemin $r \xrightarrow{\tau} r_1 \xrightarrow{\tau} .... \; r_m \xrightarrow{\tau} r'$ $(m \geq 0)$ dans g et s dans h tel que rR s et $r'R$ s, alors pour tout $i \in [1..m]$, nous avons $r_i R$ s.

Preuve Ce lemme se prouve plus facilement sur une variante de la bisimulation de branchement, appelée semi-bisimulation de branchement.

Définition 2.2.1(semi-bisimulation de branchement)
Une semi-bisimulation de branchement entre deux graphes g et h est une relation symétrique $R \subseteq$ noeuds(g) x noeuds(h) $\cup$ noeuds(h)x noeuds(g) telle que:
i) les racines de g et h sont reliées par R
ii) pour toute transition $r \xrightarrow{a} r'$ de g ou h ($a \in Act \cup \{\tau\}$), et état s de g ou h tels que r R s:
(a) soit $a = \tau$ et il existe $s \Rightarrow s'$ dans g ou h tel que rR s' , $r'R$s'
(b) soit il existe un chemin dans g ou h $s \Rightarrow s_1 \xrightarrow{a} s_2 \Rightarrow s'$ tel que rRs_1, $r'Rs_2$ et $r'R$ s'.

Notons que la différence avec la bisimulation de branchement est dans la définition du point (a). La plus grande semi-bisimulation de branchement satisfait la propriété énoncée dans le lemme 2.1. En effet, soit R la plus grande semi-bisimulation de branchement entre g et h, tel qu'il existe un chemin dans g de la forme $r \xrightarrow{\tau} r_1 \xrightarrow{\tau} .... \; r_m \xrightarrow{\tau} r'$ $(m \geq 0)$ et un état s de h avec r R s et r' R s. On construit $R' = R \cup \{(r_i, s): i \in [1..m]\}$. On vérifie facilement que R'est une semi-bisimulation de branchement en vérifiant les points i) et ii) de la définition 2.2.1. Par construction $R \subseteq R'$. Comme R est la plus grande semi-bisimulation de branchement, $R = R'$. Comme de plus, on montre facilement que la plus grande semi-bisimulation de branchement est égale à la plus grande bisimulation de branchement, le lemme 2.1 est ainsi prouvé □.

Il est intéressant de remarquer qu'il n'existe aucune bisimulation de branchement entre les graphes g_1, g_2 et g_3 de la figure 2.5.

A partir de la figure 2.6, il se dégage deux autres variantes de la τ-bisimulation suivant que l'on exige uniquement le lien (1) ou le lien (2). Dans le premier cas, nous obtenons la η-bisimulation et dans l'autre cas la Δ-bisimulation.

La notion de η-bisimulation a été introduite pour la première fois par Baeten et Van Glabbeek [BvG 87] comme une version plus fine de l'équivalence observationnelle.
Du point de vue de l'observateur, tout se passe comme si le début de l'exécution d'un processus peut provoquer un bourdonnement de la machine sur laquelle s'exécute le processus. Quand le bruit s'arrête, l'observateur verra le début de l'exécution d'une action atomique. C'est le cas pour le processus $\tau.a$ par exemple; le bourdonnement correspond en fait à la durée de la transition interne étiquetée par τ. Par contre lors de l'exécution du processus a, l'action a est observée immédiatement.

La Δ-bisimulation a été introduite dans [Wei 89] et [Wal 88]. Une variante de la Δ-bisimulation existe dans [Mil 81].
La différence entre la Δ-bisimulation et la η-bisimulation est que la première autorise des pas par des actions silencieuses avant l'action visible alors que la seconde les autorise après. Dans le cas de la η-bisimulation, le τ se comporte beaucoup plus comme une action visible. Par contre dans la Δ-bisimulation, l'action visible peut n'être observée qu'au bout d'un certain temps. Pour l'observateur, il s'agit d'un retard dans l'éxécution de l'action (d'où son nom Δ-bisimulation ou delay-bisimulation)

Définition 2.3 (η-bisimulation)
Deux graphes g et h de G sont η-bisimilaires s'il existe une relation symétrique $R \subseteq$ noeuds(g) x noeuds(h) ∪ noeuds(h)x noeuds(g) (appelée une η-bisimulation) telle que:
i) les racines de g et h sont reliées par R

ii) pour toute transition $r \xrightarrow{a} r'$ de g ou h ($a \in Act \cup \{\tau\}$), et état s g ou h tels que $r\,R\,s$,
- soit $a = \tau$ et $r'\,R\,s$
- soit il existe un chemin dans g ou h de la forme $s \Rightarrow s_1 \xrightarrow{a} s_2 \Rightarrow s'$ tel que $r R\, s_1$ et $r' R\, s'$.

On notera $g \leftrightarrows_\eta h$ quand g et h sont deux graphes η-bisimilaires.

La η-bisimulation est une relation d'équivalence sur *G*. L'union arbitraire de η-bisimulations est une η-bisimulation.

Exemple La figure 2.8 donne l'exemple des graphes $g_1 = a(\tau b+c) + ab$ et $g_2 = a(\tau b+c)$ qui sont η-bisimilaires.

Définition 2.4 (Δ- bisimulation)
Deux graphes g et h de *G* sont Δ-bisimilaires s'il existe une relation symétrique $R \subseteq$ noeuds(g) x noeuds(h) $\cup$ noeuds(h)x noeuds(g) (appelée une Δ-bisimulation) telle que:
i) les racines de g et h sont reliées par *R*
ii) pour toute transition $r \xrightarrow{a} r'$ de g ou h ($a \in Act \cup \{\tau\}$), et état s de g ou h tels que r *R* s,
- soit $a = \tau$ et r' *R* s
- soit il existe un chemin dans g ou h de la forme $s \Rightarrow s_1 \xrightarrow{a} s_2 \Rightarrow s'$ tel que r' *R* s_2 et r' *R* s'.

On notera $g \leftrightarrows_\Delta h$ quand g et h sont deux graphes Δ-bisimilaires.

Remarque 2: la deuxième possibilité du point ii) est équivalente à: il existe un chemin de la forme $s \Rightarrow s_1 \xrightarrow{a} s'$ avec r' *R* s'.
La Δ-bisimulation est une relation d'équivalence sur *G*. l'union arbitraire de Δ-bisimulations est une Δ-bisimulation.

Exemple La figure 2.9 donne un exemple de deux graphes qui sont Δ-bisimilaires.

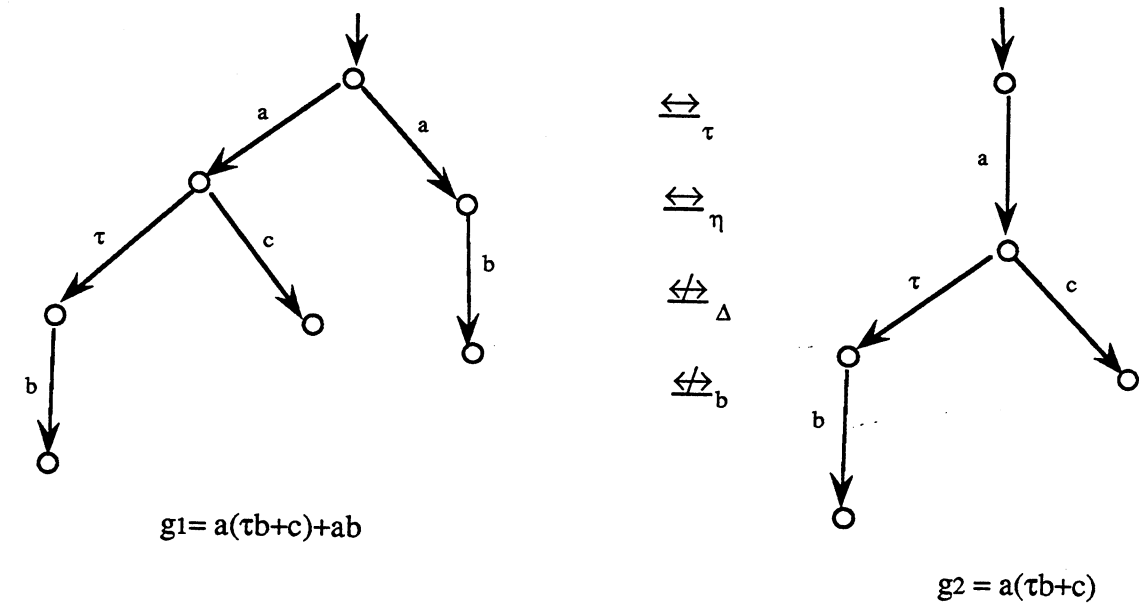


Figure 2.8

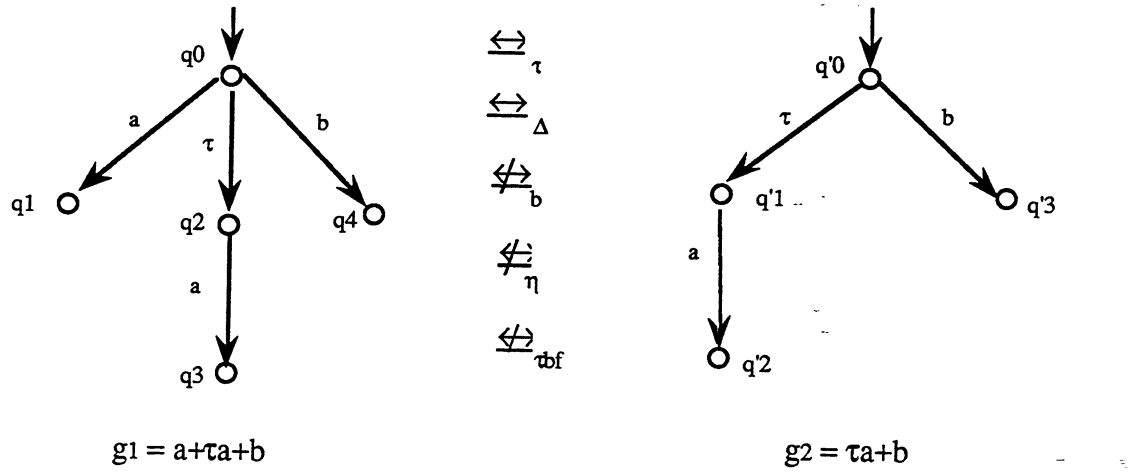


Figure 2.9

La η-bisimulation et la Δ-bisimulation ne sont pas comparables: en effet les graphes de la figure 2.8 qui sont η-bisimilaires ne sont pas Δ-bisimilaires. Ceux de la figure 2.9 qui sont Δ–bisimilaires ne sont pas η-bisimilaires.

En résumé, les définitions 2.1, 2.2, 2.3 et 2.4 permettent d'établir que:

$$g \leftrightarrows_b h \Rightarrow g \leftrightarrows_\eta h \Rightarrow g \leftrightarrows_\tau h$$

$$g \leftrightarrows_b h \Rightarrow g \leftrightarrows_\Delta h \Rightarrow g \leftrightarrows_\tau h$$

$$g \leftrightarrows h \Rightarrow g \leftrightarrows_b h$$

Ces implications sont strictes. En effet, les graphes g_1 et g_3 de la figure 2.5 sont η et τ-bisimilaires mais pas Δ-bisimilaires. Ceux de la figure 2.8 sont η-bisimilaires mais pas branching bisimilaires. Les graphes g_1 et g_2 de la figure 2.5 sont Δ et τ-bisimilaires mais pas η-bisimilaires. Ceux de la figure 2.9 sont Δ-bisimilaires mais pas branching bisimilaires.
Les graphes de la figure 2.7 qui sont branching bisimilaires ne sont pas bisimilaires (fortement).

On dit qu'un graphe est τ-free s'il ne comporte pas de τ-transitions. Notons que toutes les équivalences de bisimulation étudiées coïncident avec la bisimulation forte dans le cas où les deux graphes que l'on compare sont τ–free.
Dans la pratique on peut avoir à comparer un système de transitions τ-free avec un système de transitions quelconque. Par exemple pour vérifier qu'une implémentation est égale à une spécification donnée, on est amené à comparer, pour une relation d'équivalence donnée, un système de transitions modélisant la spécification du comportement attendu qui ne comporte pas de τ-transitions avec un système de transitions modélisant le programme (l'implémentation).
Dans le cas où l'un des deux systèmes à comparer est τ-free, l'équivalence observationnelle (τ-bisimulation) coïncide avec la bisimulation de branchement [vGW 89].

Proposition 2.2 [vGW 89]
Pour tous graphes g et h de *G* tels que g (ou h) soit τ-free :
$g \leftrightarrows_\tau h \Leftrightarrow g \leftrightarrows_b h$

2.3 Propriétés de congruence

Contrairement à la bisimulation forte, la τ-bisimulation ne se comporte pas bien vis à vis de l'opération de choix non déterministe(+): par exemple les graphes τa et a sont τ-bisimilaires alors que τa+b et a+ b ne sont pas τ-bisimilaires.($\tau.a \leftrightarrows_\tau a$ et $\tau.a + b \not\leftrightarrows_\tau a+b$). Milner a résolu le problème en définissant tout simplement une équivalence qui ne confond deux processus que s'ils sont observationnellement équivalents dans tout contexte CCS, obtenant ainsi la congruence observationnelle. Dans [BK 85], les auteurs proposent une caractérisation de la congruence observationnelle à l'aide de relations de bisimulations: cette définition correspond à

la définition 2.1 où au point i) il est exigé que les racines ne soient reliées qu'entre elles (on parle de "rooted" équivalence). Ceci s'applique aux trois variantes de l'équivalence observationnelle comme le montre la propriété suivante:

Pour tous graphes g et h de G , $* \in \{\tau, b, \eta, \Delta\}$
$R: g \leftrightarrows^* h \Rightarrow R: g \leftrightarrows_{r*} h$ si les racines de g et h ne sont reliées qu'entre elles par R.

Il est connu que les équivalences $r*$ pour $* \in \{\tau, b, \eta, \Delta\}$ se comportent bien vis à vis des opérations de préfixage par une action, de choix non déterministe, de mise en parallèle, de restriction et de renommage.

Proposition 2.3 [Wei 89]
Pour tous graphes g, g_1,g_2 de G , e de Act, B sous ensemble de Act, φ une application de Act dans Act et $* \in \{\tau, b, \eta, \Delta\}$

$$g_1 \leftrightarrows_{r*} g_2 \Rightarrow \quad e.g_1 \leftrightarrows_{r*} e.g_2$$
$$g_1 + g \leftrightarrows_{r*} g_2 + g$$
$$g + g_1 \leftrightarrows_{r*} g + g_2$$
$$g_1 \| g \leftrightarrows_{r*} g_2 \| g$$
$$g \| g_1 \leftrightarrows_{r*} g \| g_2$$
$$g_1 \setminus B \leftrightarrows_{r*} g_2 \setminus B$$
$$g_1 [\varphi] \leftrightarrows_{r*} g_2[\varphi]$$

Quand l'ensemble Act des actions contient au moins une action $a \neq \tau$, il a été prouvé dans [vGW 89] que pour $* \in \{\tau, b, \eta, \Delta\}$, $r*$ est la plus grande équivalence contenue dans $*$ et compatible avec l'opération de choix non déterministe. Ainsi $r*$ coincide avec la congruence observationnelle de Milner.

2.4 Raffinement d'actions

Il est bien connu que les langages de description de systèmes parallèles sont en général basés sur la notion d'événement qui correspond aux occurrences d'actions que le système peut effectuer. De plus il est intéressant de pouvoir décrire des systèmes parallèles à différents niveaux d'abstractions. Ainsi, les événements (ou actions) à un niveau abstrait peuvent représenter des processus complexes à un niveau plus concret. Dans ce cas, les actions ne sont plus considérées atomiques. Le concept de non atomicité des actions est formalisé par la notion de raffinement d'actions. Le raffinement consiste à remplacer une action visible d'un processus

par un comportement arbitraire. Cette notion a été définie formellement sur l'ensemble des graphes de transitions dans le chapitre 1.

2.4.1 τ-bisimulation et η-bisimulation et raffinements

La τ et la η-bisimulation ne sont pas compatibles avec le raffinement d'actions.

Contre exemple Rappelons que les graphes g_1 et g_2 de la figure 2.8 sont η-bisimilaires donc τ-bisimilaires. La figure 2.10 illustre le fait que pour raf(a) = $a_1.a_2$, les graphes raf(g_1) et raf(g_2) ne sont même plus τ-bisimilaires (donc pas η-bisimilaires). En effet supposons R:$g_1 \leftrightarrows_\tau g_2$, les points (i) et (ii) de le définition 2.1 exigent $q_0 R\ q'_0$. La transition $q_0 \xrightarrow{a_1} q_2$ de raf(g_1) est imitée par la transition $q'_0 \xrightarrow{a_1} q'_1$ de raf(g_2). Mais $q'_1 \xrightarrow{a_2} q'_2$ de raf(g_2) ne peut pas être imitée par $q_2 \xrightarrow{a_2} q_4$ de raf(g_1) car q_4 ne peut pas être relié par R à q'_2: q_4 n'a pas de transition possible par c.

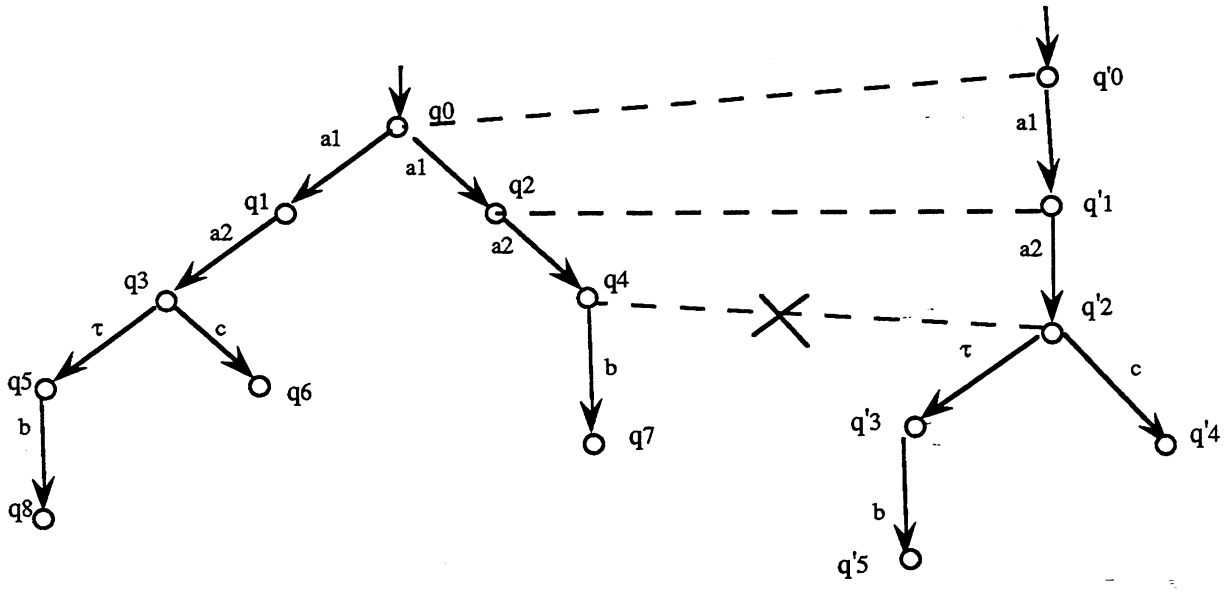


Figure 2.10

2.4.2 bisimulation de branchement et Δ-bisimulation et raffinements

La bisimulation de branchement est une congruence pour le raffinement d'actions

Théorème 2.1 [vGW 89] (théorème de raffinement pour $\leftrightarrows_b$)
Pour tous graphes *g*, *h* de *G* , pour toute action *a* de Act, *raf*, *raf'* deux raffinements

$$i)\ g \leftrightarrows_b h \Rightarrow raf(g) \leftrightarrows_b raf(h)$$
$$ii)\ raf(a) \leftrightarrows_b raf'(a) \Rightarrow raf(g) \leftrightarrows_b raf'(g)$$

Preuve

Nous prouvons le point i)

Nous supposons R: $g \leftrightarrows_b h$, on définit une relation R' entre les graphes raffinés raf(g) et raf(h) comme suit:

r R's si r R s (dans ce cas r et s sont des noeuds de g et h) ou bien si r = $\underline{q}$ et s = $\overline{q}$ (appartenant respectivement à $\underline{raf(a)}$ et $\overline{raf(a)}$) , sont des copies du même noeud interne de raf(a), graphe remplaçant les transitions $r_0 \xrightarrow{a} r_1$ de g et $s_0 \xrightarrow{a} s_1$ de h tel que $r_0 R\ s_0$ et $r_1 R\ s_1$.

Vérifions que R' est bien une bisimulation de branchement:

1)Supposons que les noeuds r et s proviennent de g et h. Nous avons donc r R s et par construction de raf(g), nous avons soit a = τ et $r \xrightarrow{\tau} r'$ est aussi une transition de g ou bien $r \xrightarrow{b} r^*$ est une transition de g et $r \xrightarrow{a} r'$ est une copie d'une transition partant de la racine de raf(b).

- Dans la premier cas, R: $g \leftrightarrows_b h$ implique soit r ' R s ou bien il existe dans h un chemin $s \Rightarrow s_1 \xrightarrow{\tau} s'$ tel que r R s_1 et r' R s'. Par définition du raffinement, ce même chemin existe dans raf(h). Ainsi nous avons r R' s_1 et r' R' s'.
- Dans le deuxième cas, g et h étant branching bisimilaires, il existe dans h un chemin $s \Rightarrow s_1 \xrightarrow{b} s^*$ avec r R s_1 et r* R s*. Par construction de raf(h) (où on remplace b par raf(b)), il existe $s \Rightarrow s_1 \xrightarrow{a} s'$ dans raf(h) avec r R' s_1 et r' R's'.

2) Les noeuds r et s proviennent de copies $\underline{raf(b)}$ et $\overline{raf(b)}$ pour b une action visible de Act. De plus r et s ne sont ni des copies de la racine, ni de feuilles de raf(b). Ainsi $r \xrightarrow{a} r'$ est nécessairement une transition de $\underline{raf(b)}$. r R' s implique que r et s sont des copies du même noeud de raf(b). Donc il existe $s \xrightarrow{a} s'$ dans $\overline{raf(b)}$ avec s' et r' copies du même noeud de raf(b). Clairement, il s'ensuit que r' R' s'.

Notons que si r et s sont respectivement les racines de g et h , nous avons r R' s si et seulement si r R s □

La Δ-bisimulation est aussi compatible avec le raffinement d'actions.

Théorème 2.2 [CS 91] (théorème de raffinement pour $\leftrightarrows_\Delta$)
Pour tous graphes *g*, *h* de *G* , *raf* un raffinement:

$$g \leftrightarrows_\Delta h \Rightarrow raf(g) \leftrightarrows_\Delta raf(h)$$

Preuve
Supposons que *R*: g $\leftrightarrows_\Delta$ h, on définit *R'* entre les graphes raffinés comme suit:
r *R'* s si r *R* s (dans ce cas r et s sont des noeuds de g et h) ou bien si r et s sont des copies d'un même noeud interne à un graphe de raffinement raf(a), inséré à la place des transitions $r_0 \xrightarrow{a} r_1$ de g et $s_0 \xrightarrow{a} s_1$ de h tel que r_1 *R* s_1. Notons que contrairement à la relation *R'* définie dans la preuve du théorème 2.1, seule la relation entre r_1 et s_1 par *R* est exigée. Par conséquent, les racines des graphes insérés ne sont reliées par *R'* que si elles le sont par *R*. La relation *R'* ainsi définie est une Δ-bisimulation entre les graphes raf(g) et raf(h). On prouve facilement cette affirmation en suivant la même démarche que dans la preuve du théorème 2.1. □

Notons que ce résultat a aussi été établi par Devillers dans [Dev 90] sur un modèle de vrai parallélisme.

2.5 Full Abstraction

Au vu des résultats établis dans [vGW 89], il est naturel de se demander si la bisimulation de branchement est la plus grande congruence pour le raffinement contenue dans la τ-bisimulation. D'autre part, nous avons vu dans les sections 2 et 4 de ce chapitre que la bisimulation de branchement est contenue dans la Δ-bisimulation ($\leftrightarrows_b \Rightarrow \leftrightarrows_\Delta$) et que la Δ-bisimulation est une congruence pour le raffinement d'actions. Ainsi, une réponse partielle est apportée: la Δ-bisimulation serait meilleure candidate pour le résultat de " full abstraction " recherché. Dès lors le problème revient à chercher si la Δ-bisimulation est la plus grande congruence pour le raffinement contenue dans la τ-bisimulation. Nous avons vu également que la η–bisimulation n'est pas compatible avec le raffinement . Aussi, nous nous proposons de chercher la plus grande congruence pour le raffinement contenue dans la η-bisimulation.

Définition 2.5 (Full Abstraction)
Soient $\approx_{rx}$ et $\approx_x$ deux équivalences sur G, $\approx_{rx}$ est la plus grande équivalence compatible avec le raffinement et contenue dans $\approx_x$ si $\approx_{rx}$ est définie par:

$$g \approx_{rx} h \text{ si et seulement si } \forall raf: Act \rightarrow G\ ,\ raf(g) \approx_x raf(h)$$

(on dit que $\approx_{rx}$ est " fully abstract " par rapport à $\approx_x$ et au raffinement)
Pour prouver nos résultats de full abstraction, nous utilisons le raffinement appelé "split" qui est un cas particulier de raffinement: dans toute transition par une action visible, un nouveau noeud est inséré i.e. une transition $r \xrightarrow{a} r'$ est remplacée par le graphe $r \xrightarrow{a} q \xrightarrow{a} r'$.
C'est Hennessy [Hen 88] qui a introduit cette idée de split quand il a voulu associer une durée aux actions, donc un début et une fin. Cette idée a été très exploitée par la suite dans la notion des ST-sémantiques [vG 90]. Pour notre part, nous utilisons une version très simple du "split", en ce sens que le début et la fin d'une action seront étiquetées par la même action.

Le résultat de full abstraction repose sur la proposition suivante

Proposition 2.4 [CS 91]
Pour tous graphes g, h de G

$$split(g) \leftrightarrows_\tau split(h) \text{ si et seulement si } g \leftrightarrows_\Delta h$$

La preuve de cette proposition utilise le lemme suivant:

Lemme 2.2 (X-propriété)
Pour tous graphes g et h de G, si R : $g \leftrightarrows_\tau h$ est la plus grande τ-bisimulation entre g et h, alors R a la X-propriété c.à.d:

$\forall r, r'$ *états de g et s, s' états de h: si* $[r \Rightarrow r',\ s \Rightarrow s'$ *tels que* $r\ R\ s'$ *et* $r'\ R\ s\ ]$ *alors* $rR\ s$

Preuve
On définit $R' = R \cup \{(r,s), (s,r) \mid r \Rightarrow r',\ s \Rightarrow s'$ avec $r\ R\ s'$ et $r'R\ s\}$.
On vérifie que R' est une τ-bisimulation. Comme R est la plus grande et que $R \subseteq R'$, on a $R = R'$. Donc R a la X-propriété par construction.
Il est évident que R' est une relation symétrique.
i)Les racines de g et h sont reliées par R' puisqu'elles sont reliées par R.

ii) Supposons r R' s et r $\xrightarrow{a}$ r_1 une transition dans g. Si r R s la propriété de transfert est trivialement vérifiée (puisque R est une τ-bisimulation entre g et h). Dans le cas contraire i.e r non relié à s par R, il existe nécessairement r' et s' appartenant respectivement à g et h tels que r$\Rightarrow$ r', s $\Rightarrow$ s' avec r R s' et r'R s . r R s' et r $\xrightarrow{a}$ r_1, impliquent

- soit a = τ et r_1 R s'. r R s' et r $\Rightarrow$ r' impliquent qu'il existe s" telle s' $\Rightarrow$ s" et r'Rs". r'R s" et r'R s impliquent sR s" et de ce fait sR s' et r_1R s donc r_1R' s.
- soit il existe un chemin dans h de la forme s' $\Rightarrow$ $\xrightarrow{a}$ $\Rightarrow$ s" avec r_1 R s". Donc quand r $\xrightarrow{a}$ r_1, s $\Rightarrow$ s' $\Rightarrow$ $\xrightarrow{a}$ $\Rightarrow$ s" donc s $\Rightarrow$ $\xrightarrow{a}$ $\Rightarrow$ s" avec r_1 R s", donc r_1 R' s". Ce qui correspond exactement à la propriété de transfert de la τ-bisimulation.

□

Preuve (proposition 2.4)

$\Rightarrow$: Supposons que split(g) $\leftrightarrows_\tau$ split(h). Soit R la plus grande τ-bisimulation entre split(g) et split(h). Soit R' la restriction de R aux noeuds de g et h (noeuds existants avant le split). R' est une Δ-bisimulation entre g et h. Nous montrons ceci en vérifiant les points i) et ii) de la définition 2.4.

i) Il est évident que les racines de g et h sont reliées par R'.

ii) Supposons r R' s et r $\xrightarrow{a}$ r' une transition de g (le cas où la transition r $\xrightarrow{a}$ r' est dans h est symétrique). Deux cas sont alors possibles:

- a = τ, r $\xrightarrow{a}$ r' est aussi une transition de split(g). Comme $R' \subseteq R$, split(h) contient un chemin de la forme s $\Rightarrow$ s' tel que r' R's'. Le chemin s $\Rightarrow$ s' de split(h) implique que s' est un noeud qui provient de h. Nous avons donc r' R' s'.

- a ≠ τ, r $\xrightarrow{a}$ r' implique que dans split(g) il y a un chemin de la forme: r$\xrightarrow{a}$ $r_1\xrightarrow{a}$ r'. R:split (g) $\leftrightarrows_\tau$ split (h) implique qu'il existe un chemin s$\Rightarrow s_0\xrightarrow{a}$ $\Rightarrow s_1$ $\Rightarrow$ $\xrightarrow{a}$ s'$\Rightarrow$ s_2 tel que r_1 R s_1 et r' R s_2. La structure de split(h) exige que s_1 soit un nouveau noeud. Ainsi le chemin de split(h) a plus exactement la forme: s$\Rightarrow$ s_0 $\xrightarrow{a}$ $s_1\xrightarrow{a}$ s'$\Rightarrow$ s_2 . Par ailleurs, r_1 Rs_1 et $s_1\xrightarrow{a}$ s' impliquent qu'il existe un chemin $r_1\Rightarrow$ $\xrightarrow{a}$ $\Rightarrow$ r_2 tel que r_2 R s'. La structure de split(g) garantit que ce chemin visite le

noeud r' et plus exactement qu'il est de la forme $r_1 \xrightarrow{a} r' \Rightarrow r_2$. Finalement, nous avons $r' \Rightarrow r_2$ et $s' \Rightarrow s_2$ avec $r' \; R \; s_2$ et $r_2 \; R \; s'$.

Nous avons donc $r' \; R \; s'$ d'après le lemme 2.2, donc $r' \; R' \; s'$ (car r' et s' ne sont pas des nouveaux noeuds). Ainsi si $r \; R' \; s$ et $r \xrightarrow{a} r'$ est une transition de g, il existe dans h un chemin de la forme $s \Rightarrow s_0 \xrightarrow{a} s'$ avec $r' \; R' \; s'$. R' est donc bien une Δ-bisimulation entre g et h.

⇐est triviale. □

Dans ce qui suit, nous définissons une nouvelle équivalence (la quasi-branching bisimulation) qui est la plus grande congruence pour le raffinement contenue dans la η-bisimulation [CS 93].

Définition 2.6 (quasi-branching bisimulation)
Deux graphes g et h de G sont "quasi-branching bisimilaires" s'il existe une relation symétrique $R \subseteq$ noeuds(g) x noeuds(h) ∪ noeuds(h)x noeuds(g) (appelée une quasi-branching bisimulation) telle que:

i) les racines de g et h sont reliées par R

ii) pour toute transition $r \xrightarrow{a} r'$ de g ou h ($a \in Act \cup \{\tau\}$), et état s de g ou h tels que rRs,
- soit $a = \tau$ et il existe un chemin dans g ou h de la forme $s \Rightarrow s'$ et $r' R \; s'$
- soit il existe un chemin dans g ou h de la forme $s \Rightarrow s_1 \xrightarrow{a} s'$ tels que $rR \; s_1$ et $r' R \; s'$.

On notera $g \leftrightarrows_{qb} h$ quand g et h sont deux graphes quasi-branching bisimilaires.

Notons que la différence avec la bisimulation de branchement est qu'on autorise d'imiter une τ-transition par une suite de τ-transitions à condition d'atteindre des états équivalents.

Exemple La figure 2.11 contient deux graphes quasi-branching bisimilaires.

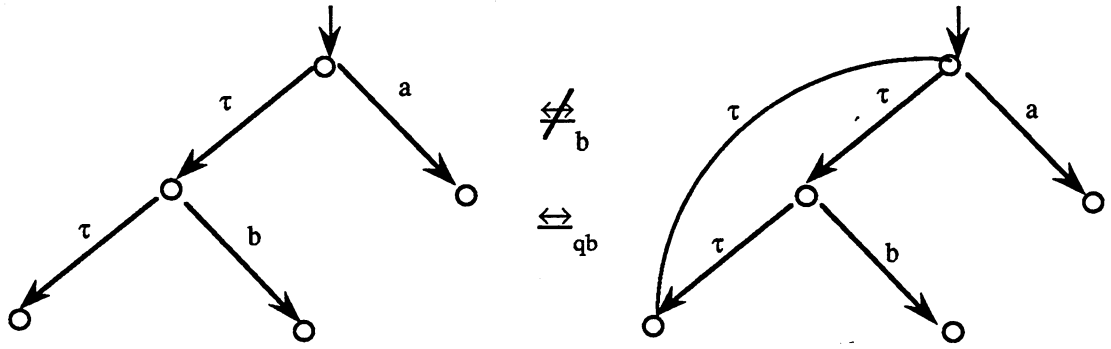


Figure 2.11

$g \leftrightarrows_b h \Rightarrow g \leftrightarrows_{qb} h$. Cette implication est stricte: les graphes de la figure 2.11 ne sont pas branching bisimilaires.

$g \leftrightarrows_{qb} h \Rightarrow g \leftrightarrows_\Delta h$ et $g \leftrightarrows_{qb} h \Rightarrow g \leftrightarrows_\eta h$. Ces deux implications sont strictes: les graphes de la figure 2.12 sont η-bisimilaires et Δ-bisimilaires mais pas qb-bisimilaires.

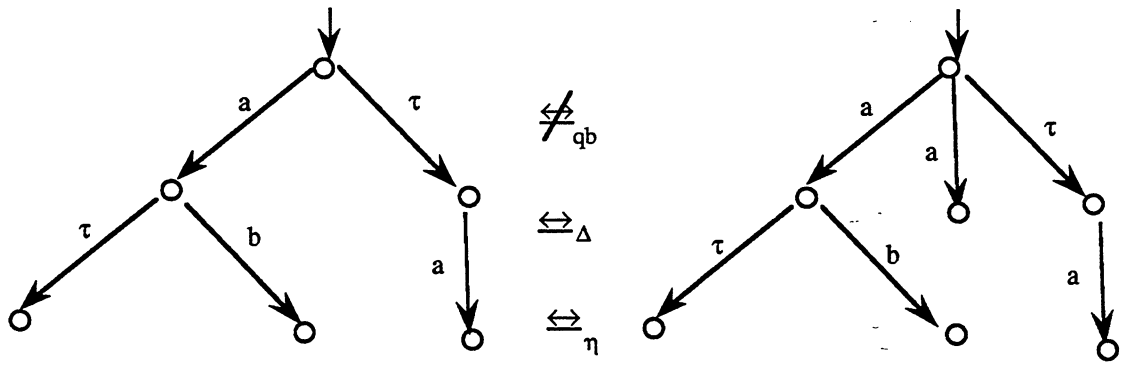


Figure 2.12

Une axiomatisation correcte et complète de la quasi-branching bisimulation existe dans [CS 93].

La quasi-branching bisimilation est compatible avec le raffinement d'actions.

Théorème 2.3 [CS 93] (théorème de raffinement pour $\leftrightarrows_{qb}$)
Pour tous graphes *g*, *h* de *G* , *raf* un raffinement:

$$g \leftrightarrows_{qb} h \Rightarrow raf(g) \leftrightarrows_{qb} raf(h)$$

La preuve de ce théorème est similaire à celle du théorème 2.2. Notons également que le lemme 2.1(stuttering lemma) se prouve facilement pour la quasi-branching bisimulation.

Proposition 2.5 [CS 93]
Pour tous graphes g, h de G

$$split(g) \leftrightarrows_{\eta} split(h) \quad si\ et\ seulement\ si \quad g \leftrightarrows_{qb} h$$

La preuve est similaire à celle de la proposition 2.4.

Théorème 2.4 (Full Abstraction)

(1) $\leftrightarrows_{\Delta}$ *est la plus grande congruence pour le raffinement contenue dans* $\leftrightarrows_{\tau}$
(2) $\leftrightarrows_{qb}$ *est la plus grande congruence pour le raffinement contenue dans* $\leftrightarrows_{\eta}$

Ce théorème est un corollaire de la proposition 2.4, 2.5 et des théorèmes 2.1 et 2.3.
Ceci prouve que le résultat de [CS 91] est erroné puisque c'est la quasi-branching bisimulation et non la branching bisimulation qui est la plus grande congruence pour le raffinement contenue dans la η-bisimulation.

2.6 Autres caractérisations de la bisimulation de branchement

Il existe dans la littérature d'autres caractérisations de la bisimulation de branchement. De Nicola, Montanari et Vaandrager [DNMV 90] ont d'abord introduit la notion de bisimulation avant arrière et ont ensuite montré que la bisimulation de branchement coïncide avec les versions avant arrière des trois équivalences (τ, η, Δ) qui traitent l'action invisible τ. Dans l'équivalence de bisimulation avant arrière, les processus se simulent non seulement en faisant des pas en avant (transitions classiques) mais aussi en faisant des retours en arrière le long de leurs *histoires* respectives.

Définition 2.7
Soit g = (Q, Act, →), un graphe ou système de transitions.

- On appelle *chemin partant de* s_0, une séquence de transitions :

$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \ldots s_{n-1} \xrightarrow{a_n} s_n$.

• Une *histoire ou calcul* partant d'un état s est une paire (s, π), où π est un chemin partant de s. Nous écrirons $\mathcal{H}(s)$ pour l'ensemble des histoires partant de s et $\mathcal{H}(g)$ pour l'ensemble de tous les calculs de g. Si $\sigma = (s, \pi)$ est un calcul, nous noterons $\pi = path(\sigma)$ et $last(\sigma)$ dénotera le dernier état du chemin π.(si $\pi = \lambda$ alors $last(\sigma) = s$)

• Si σ et σ' sont deux calculs, σ.σ' est la concaténation ou la juxtaposition de σ et σ'. Si $\sigma = (s,\pi)$, $\sigma' = (s', \pi')$, σ.σ' est défini seulement si $s' = last(\sigma)$.

•On écrit $\sigma \xrightarrow{a} \sigma'$ s'il existe un calcul $\theta = (s, s \xrightarrow{a} s')$ tel que $\sigma' = \sigma.\theta$

Dans la section suivante, nous définissons la notion de bisimulation avant arrière sur l'ensemble des graphes de transitions *G*.

Definition 2.8 (Bisimulation avant arrière)

Deux graphes $g = (Q_1, Act, \rightarrow_1, q_1)$ et $h = (Q_2, Act, \rightarrow_2, q_2)$ de *G* se bisimulent en avant et arrière (fortement) s'il existe une relation symétrique $R \subseteq \mathcal{H}(g) \times \mathcal{H}(h) \cup \mathcal{H}(h) \times \mathcal{H}(g)$ (appelée bisimulation avant arrière (forte)) telle que:
i) $(q_1, \lambda)\ R\ (q_2, \lambda)$
ii) Pour toute transition $\rho \xrightarrow{a} \rho'$ et σ tel que $\rho\ R\ \sigma$, il existe σ' tel que $\sigma \xrightarrow{a} \sigma'$ et $\rho'\ R\ \sigma'$.
iii) Pour toute transition $\rho' \xrightarrow{a} \rho$ et σ tel que $\rho\ R\ \sigma$, il existe σ' tel que $\sigma' \xrightarrow{a} \sigma$ et $\rho'\ R\ \sigma'$.

On notera $g \leftrightarrows_{bf} h$ quand g et h se bisimulent en avant et en arrière.

Notons que c'est dans le point iii) qu'est exprimé le retour en arrière. Ce retour ne se fait pas de façon arbitraire, il se fait le long des chemins qui ont amené les processus à leurs états respectifs lors de leurs histoires.
Dans la définition 2.8, les transitions ont lieu par des actions visibles. Il est connu que cette nouvelle bisimulation n'augmente pas le pouvoir de distinction de la bisimulation forte.

Proposition 2.6 [DNMV 90]
Pour tous graphes g et h de *G* :

$$g \leftrightarrows h \Leftrightarrow g \leftrightarrows_{bf} h.$$

Preuve

1) $\Rightarrow$: Soit R : $g \leftrightarrows h$ une bisimulation entre g et h. On définit *ct* l'application qui à tout chemin π de g ou h associe sa trace colorée i.e. une séquence obtenue à partir de π où chaque état est remplacé par sa classe d'équivalence modulo la bisimulation. Par suite, $ct((s_0, a_1, s_1),...,(s_{n-1}, a_{n-1}, s_n)) = (s_0/\leftrightarrows, a_1, s_1/\leftrightarrows), ...(s_{n-1}/\leftrightarrows, a_n, s_n/\leftrightarrows)$. On définit R' par :
$R' = \{(\rho, \sigma), (\sigma, \rho) \mid \sigma \in \mathcal{H}(g), \rho \in \mathcal{H}(h)$ avec $ct(path(\sigma)) = ct(path(\rho))\}$ et on vérifie facilement que R' est une bisimulation avant arrière entre g et h.
2) $\Leftarrow$: Supposons R : $g \leftrightarrows_{bf} h$ une bisimulation avant arrière entre g et h, on définit R' par :
$R' = \{(last(\rho), last(\sigma)) \mid \rho\ R\ \sigma\}$. Il est facile de vérifier que R' est une bisimulation forte entre g et h. □

Dans le même état d'esprit, les versions avant arrière de la τ-bisimulation, η-bisimulation et Δ-bisimulation sont définies comme suit:

Nous noterons $\rho \overset{a}{\Rightarrow} \rho'$ s'il existe ρ_1, ρ_2 tel que $\rho \Rightarrow \rho_1 \overset{a}{\rightarrow} \rho_2 \Rightarrow \rho'$ $a \in Act \cup \{\tau\}$.

Définition 2.9 (τ-bisimulation avant arrière)
Deux graphes $g = (Q_1, Act, \rightarrow_1, q_1)$ et $h = (Q_2, Act, \rightarrow_2, q_2)$ de *G* se bisimulent (faiblement) en avant et arrière s'il existe une relation symétrique $R \subseteq \mathcal{H}(g) \times \mathcal{H}(h) \cup \mathcal{H}(h) \times \mathcal{H}(g)$ (appelée une τ-bisimulation avant arrière) telle que:
i) $(q_1, \lambda)\ R\ (q_2, \lambda)$

ii) pour toute transition $\rho \overset{a}{\rightarrow} \rho'$ et σ tel que $\rho\ R\ \sigma$ $(a \in Act \cup \{\tau\})$,
- soit $a = \tau$ et $\rho'\ R\ \sigma$
- soit il existe $\sigma \overset{a}{\Rightarrow} \sigma'$ tel que $\rho' R\ \sigma'$.

iii) pour toute transition $\rho' \overset{a}{\rightarrow} \rho$ et σ tel que $\rho\ R\ \sigma$,

- soit $a = \tau$ et $\rho'\ R\ \sigma$

- soit il existe $\sigma' \overset{a}{\Rightarrow} \sigma$ avec $\rho' R\ \sigma'$.

On notera $g \leftrightarrows_{\tau bf} h$ quand g et h se bisimulent faiblement en avant et en arrière.

Exemple Les graphes g_1 et g_2 de la figure 2.9 sont τ- bisimilaires mais pas τbf-bisimilaires. En effet supposons $R: g_1 \leftrightarrows_{\tau bf} g_2$ la transition $q_0 \overset{a}{\rightarrow} q_1$ de g_1 est simulée en avant par la

transition $q'_0 \stackrel{\tau a}{\Rightarrow} q'_2$ de g_2. A partir de q'_2 quand g_2 effectue l'action a en arrière, il se trouve dans l'état q'_1. g_1 l'imite en effectuant le même pas en arrière, se trouvant ainsi dans l'état q_0. Mais q_0 et q'_1 ne peuvent être reliés par R car il n'est pas possible d'effectuer un b à partir de l'état q'_1.

Les graphes de la figure 2.7 qui sont branching bisimilaires sont τbf bisimilaires.

Définition 2.10 (η-bisimulation avant arrière)
Deux graphes $g = (Q_1, Act, \rightarrow_1, q_1)$ et $h = (Q_2, Act, \rightarrow_2, q_2)$ de G se η-bisimulent en avant et arrière s'il existe une relation symétrique $R \subseteq \mathcal{H}(g) \times \mathcal{H}(h) \cup \mathcal{H}(h) \times \mathcal{H}(g)$ (appelée une η-bisimulation avant arrière) telle que:
i) $(q_1, \lambda) R (q_2, \lambda)$

ii) pour toute transition $\rho \stackrel{a}{\longrightarrow} \rho'$ et σ tel que $\rho R \sigma$ ($a \in Act \cup \{\tau\}$),
- soit $a = \tau$ et $\rho' R \sigma$
- soit il existe un chemin de la forme $\sigma \Rightarrow \sigma_1 \stackrel{a}{\longrightarrow} \sigma_2 \Rightarrow \sigma'$ tel que $\rho R \sigma_1$ et $\rho' R \sigma'$.

iii) Pour toute transition $\rho' \stackrel{a}{\longrightarrow} \rho$ et σ tel que $\rho R \sigma$,
- soit $a = \tau$ et $\rho' R \sigma$
- soit il existe $\sigma', \sigma_1, \sigma_2$ tels que $\sigma' \Rightarrow \sigma_1 \stackrel{a}{\longrightarrow} \sigma_2 \Rightarrow \sigma$ avec $\rho' R \sigma_1$ et $\rho' R \sigma'$.

On notera $g \leftrightarrows_{\eta bf} h$ quand g et h se η-bisimulent en avant et en arrière.

Définition 2.11 (Δ-bisimulation avant arrière)
Deux graphes $g = (Q_1, Act, \rightarrow_1, q_1)$ et $h = (Q_2, Act, \rightarrow_2, q_2)$ de G se Δ-bisimulent en avant et arrière s'il existe une relation symétrique $R \subseteq \mathcal{H}(g) \times \mathcal{H}(h) \cup \mathcal{H}(h) \times \mathcal{H}(g)$ (appelée une Δ-bisimulation avant arrière) telle que:

i) $(q_1, \lambda) R (q_2, \lambda)$

ii) pour toute transition $\rho \stackrel{a}{\longrightarrow} \rho'$ et σ tel que $\rho R \sigma$ ($a \in Act \cup \{\tau\}$), soit $a = \tau$ et $\rho' R \sigma$
soit il existe un chemin de la forme $\sigma \Rightarrow \sigma_1 \stackrel{a}{\longrightarrow} \sigma_2 \Rightarrow \sigma'$ tel que $\rho' R \sigma_2$ et $\rho' R \sigma'$.

iii) pour toute transition $\rho' \stackrel{a}{\longrightarrow} \rho$ et σ tel que $\rho R \sigma$,
- soit $a = \tau$ et $\rho' R \sigma$
- soit il existe $\sigma', \sigma_1, \sigma_2$ tel que $\sigma' \Rightarrow \sigma_1 \stackrel{a}{\longrightarrow} \sigma_2 \Rightarrow \sigma$ avec $\rho R \sigma_2$ et $\rho' R \sigma'$.

On notera $g \leftrightarrows_{\Delta bf} h$ quand g et h se Δ-bisimulent en avant et en arrière.

Il a été montré dans [DNMV 90] que les versions avant arrière de la τ-bisimulation, la η-bisimulation et de la Δ-bisimulation coincident avec la bisimulation de branchement.

Proposition 2.7 [DNNV 90]
Pour tous graphes *g* et *h* de *G*,

$$g \leftrightarrows_{\tau bf} h \Leftrightarrow g \leftrightarrows_{\eta bf} h \Leftrightarrow g \leftrightarrows_{\Delta bf} h \Leftrightarrow g \leftrightarrows_{b} h$$

La preuve de cette proposition repose sur le lemme suivant:

Lemme 2.3 :
Pour tous graphes g et h de *G*, si R : $g \leftrightarrows_{\tau bf} h$ est la plus grande bisimulation avant arrière (faible) alors R vérifie la propriété suivante :
$\forall\ \rho, \rho' \in \mathcal{H}(g), \sigma, \sigma' \in \mathcal{H}(h)$: si $[\rho \Rightarrow \rho', \sigma \Rightarrow \sigma' |\ \rho\ R\ \sigma'$ et $\rho'\ R\ \sigma]$ alors $\rho'\ R\ \sigma'$.

Preuve
Notons que ce lemme énonce une propriété sur les histoires similaire au X-lemma sur les états (Lemme 2.2). Si R est la plus grande τ-bisimulation avant arrière entre g et h, il suffit de construire $R' = R \cup \{ \rho',\sigma'), (\sigma', \rho') \mid \exists\ \rho \in \mathcal{H}(g), \sigma \in \mathcal{H}(h)$ avec $\rho \Rightarrow \rho', \sigma \Rightarrow \sigma'$ et $\rho\ R\ \sigma'$ et $\rho'\ R\ \sigma\}$ et de vérifier que R' est une τbf-bisimulation entre g et h. Il s'ensuit que R' (donc R) a la propriété désirée. □

Il est clair que la bisimulation de branchement coïncide avec sa version avant arrière.

2.7 Caractérisation logique de la bisimulation de branchement

De Nicola et Vaandrager [DNV90] ont proposé trois caratérisations logiques pour la bisimulation de branchement. Nous choisissons de présenter celle qui étend la logique HML parce que d'une part, elle découle de façon très naturelle d'un des résultats de la proposition 2.6: la bisimulation de branchement coïncide avec la version avant arrière de l'équivalence observationnelle, et d'autre part, c'est cette logique que nous généralisons dans le chapitre 4 sur les modèles de vrai parallélisme.
Dans [DNV 90] les auteurs proposent la logique HML_{bf}, une variante de HML munie d'un opérateur du passé qui permet d'analyser le passé d'un calcul donné. L'idée d'introduire les opérateurs du passé existe aussi dans [HS 85] pour l'étude de certaines propriétés (l'équité par exemple) des systèmes.

La définition de la logique HML_{bf} suppose donné un ensemble Act = {a, b, ...} d'actions.

Définition 2.12 (syntaxe de HML_{bf})
Les formules de la logique HML_{bf} sont données par la grammaire suivante :

$\varphi, \varphi' ::= T \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \langle a\rangle \varphi \mid \langle\leftarrow a\rangle \varphi$ où a $a \in$ Act $\cup \{\tau\}$et $\leftarrow$ un opérateur du passé.

La logique HML_{bf} est interprétée sur les histoires ou calculs des graphes de transitions.
$\langle\leftarrow a\rangle \varphi$ par exemple signifie qu'il est possible de faire un retour en arrière par l'action a modulo un nombre arbitraire de τ-transitions avant et après l'action observable a et atteindre une histoire où φ est satisfaite.

Définition 2.13 (Sémantique de HML_{bf})
Soit $g = (Q, \rightarrow, q_0)$ un graphe de processus, $\mathcal{H}(g)$ l'ensemble des histoires de g. La relation $\models \subseteq \mathcal{H}(g) \times HML_{bf}$ est définie inductivement sur la structure de φ comme suit :

- $\sigma \models T$ toujours
- $\sigma \models \neg\varphi$ ssi $\sigma \not\models \varphi$
- $\sigma \models \varphi \wedge \varphi'$ ssi $\sigma \models \varphi$ et $\sigma \models \varphi'$
- $\sigma \models \langle a\rangle \varphi$ ssi il existe $\sigma \stackrel{a}{\Rightarrow} \sigma'$ tel que $\sigma' \models \varphi$
- $\sigma \models \langle\leftarrow a\rangle \varphi$ ssi il existe $\sigma' \stackrel{a}{\Rightarrow} \sigma$ tel que $\sigma' \models \varphi$

Un graphe g satisfait une formule φ, noté $g \models \varphi$ si et seulement si $(q_0, \lambda) \models \varphi$
$\sigma \models \langle a\rangle \varphi$ signifie qu'il est possible modulo un nombre arbitraire de τ d'effectuer l'action a et d'atteindre une histoire où φ est satisfaite.

Exemple Les graphes de la figure 2.8 qui ne sont pas branching bisimilaires sont distingués par exemple par la formule $\varphi \equiv [a][b]\langle\leftarrow b\rangle\langle c\rangle T$. Notons que $g_1 \not\models \varphi$ alors que $g_2 \models \varphi$.
Les graphes de la figure 2.9 sont distingués par la formule $\varphi' \equiv \langle a\rangle[\leftarrow a]\langle b\rangle T$, $g_1 \models \varphi'$ et $g_2 \not\models \varphi'$.

La logique HML_{bf} induit sur les graphes une équivalence notée $\approx_{bf}$: deux graphes sont $\approx_{bf}$ équivalents s'ils satisfont exactement les mêmes formules HML_{bf}. Nous noterons $HML_{bf}(g)$ l'ensemble des formules HML_{bf} satisfaites par le graphe g.

Définition 2.14 (équivalence HML_{bf})
Pour tous graphes g_1, g_2 de G :

$$g_1 \approx_{bf} g_2 \text{ si et seulement si } HML_{bf}(g_1) = HML_{bf}(g_2)$$

Dans le cas des graphes à branchement fini, l'équivalence $\approx_{bf}$ distingue exactement ce que la bisimulation de branchement distingue [DNV 90].

Proposition 2.8
Pour tous graphes $g_1 = (Q_1, \rightarrow_1, q_1)$, $g_2 = (Q_2, \rightarrow_2, q_2)$ à branchement fini :

$$g_1 \leftrightarrows_b g_2 \Leftrightarrow g_1 \approx_{bf} g_2$$

Preuve
Pour tout graphe $g = (Q, \rightarrow, Act \cup \{\tau\})$, on associe un graphe $g_{bf} = (\mathcal{H}(g), \rightarrow_{bf}, Act_{bf})$ où $Act_{bf} = Act \cup \{\tau\} \cup \{\leftarrow a \mid a \in Act \{\tau\}\}$ et $\rightarrow_{bf}$ est définie comme suit :

- $\rho \xrightarrow{a}_{bf} \rho'$ si et seulement si $\rho \stackrel{a}{\Rightarrow} \rho'$ est dans g.

- $\rho \xrightarrow{\leftarrow a}_{bf} \rho'$ si et seulement si $\rho' \stackrel{\leftarrow a}{\Rightarrow} \rho$ est dans g. Il est évident que toute τ-bisimulation avant arrière entre deux graphes g et h est une bisimulation forte entre g_{bf} et h_{bf}. et réciproquement. Or, il est bien connu que la bisimulation forte distingue exactement ce que la logique HML distingue [HM 85]. Donc HML_{bf} caractérise la $\leftrightarrows_{\tau bf}$-bisimulation donc la bisimulation de branchement. □

2.8 Conclusions

Dans ce chapitre, nous avons présenté les équivalences de bisimulation qui traitent les situations où certaines actions sont invisibles. Nous avons étudié leurs comportements vis à vis des opérateurs classiques de combinaison de graphes ainsi que vis à vis de l'opération de raffinement d'actions. Parmi les équivalences présentées, la bisimulation de branchement et la Δ-bisimulation sont compatibles avec le raffinement d'action. De plus, nous avons montré que la quasi-branching bisimulation et la Δ-bisimulation sont les plus grandes congruences par rapport au raffinement contenues respectivement dans la η-bisimulation et la τ-bisimulation. La recherche de congruences (fully abstract) pour le raffinement d'actions est un sujet

d'actualité dans la sémantique du parallèlisme. Des résultats existent sur les réseaux de Petri dans [Vog 90] [JM 92]et sur les structures d'événements dans [Vog 90], [vG 90a] et [vGG 89]. Une autre approche basée sur le raffinement au niveau syntaxique est présentée dans [AH 89] et [AH 90]. Le choix du modèle de graphes dans ce chapitre a été essentiellement motivé par les résultats de [vGW 89] . Dans les chapitres qui suivent, nous nous intéressons aux modèles dits de vrai parallèlisme où l'opération de raffinement semble être plus naturelle.

Chapitre 3

Bisimulations et raffinement sur les structures d'événements premières

3.1 Introduction

La sémantique opérationnelle des langages parallèles est souvent décrite à l'aide de systèmes de transitions étiquetés par des actions [Kel 76]. Par exemple la transition $p \xrightarrow{a} q$ exprime que le processus p accomplit l'action a et ce faisant se reconfigure en un autre processus q.

Cette approche décrit des transitions entre états globaux du système et ne tient pas compte des relations de causalité qui peuvent exister entre les actions effectuées (les actions peuvent provenir de composantes indépendantes du système par exemple). Dans les systèmes de transitions l'opération de composition parallèle n'est pas une opération primitive : exécuter deux actions a et b en parallèle revient à effectuer d'abord a puis b ou bien d'abord b puis a. Dans ce cas, on parle de sémantique basée sur l'entrelacement (interleaving) qui consiste à entremêler les séquences d'actions des processus mis en parallèle. Ainsi dans les modèles de graphes (ou systèmes de transitions étiquetés) on ne distingue pas le non determinisme du vrai

parallélisme. Plus précisément on dit que le parallèlisme est réduit à l'interleaving et au non déterminisme ($a|b = ab + ba$)

Pour rendre compte du parallélisme, il faut pouvoir parler d'action globale d'un système provenant de l'activité de ses composants agissant ensemble. Comme plusieurs sous systèmes peuvent effectuer la même action ensemble, une action globale sera un *multiensemble* d'actions, ou un élément peut avoir plusieurs occurrences. Une telle action globale est en soi atomique sur le plan temporel mais peut être composée d'actions élémentaires simultanées. Ainsi, l'étiquetage des transitions par cette notion d'action globale permet de généraliser les systèmes de transitions évoqués plus haut. Cette variante des systèmes de transitions a été utilisée pour donner une sémantique aux langages comme MEIJE [AB 84] et SCCS [Mil 83], CIRCAL [Mil 85] Cette approche permet de distinguer les agents CCS a.NIL| b.NIL et a.b.NIL + b.a.NIL. Notons cependant que la causalité entre les actions n'est toujours pas prise en compte: par exemple on ne distingue pas le comportement de l'agent a.NIL | b.NIL +a.b.NIL de celui de a.NIL| b.NIL. Par conséquent, les systèmes de transitions peuvent décrire les dépendances temporelles de systèmes mais ne peuvent pas décrire les dépendances de causalité.

Pour remédier à cette insuffisance des systèmes de transitions, des modèles dits de "vrai parallèlisme" sont proposés dans la littérature. Pour notre part nous nous intéressons aux structures d'événements [CFM 83], [MS 81], [NPW 81], [Win 80], et [Win 87]. Plus précisément nous parlerons de structures d'événements premières qui constituent un modèle de base pour le vrai parallélisme. Dans ce modèle, un système parallèle est représenté par un ensemble d'occurrences d'événements munis d'une relation de causalité et d'une relation de conflit qui exprime comment les occurrences de certains événements excluent d'autres. Les structures d'événements premières généralisent les arbres de communication de Milner [Mil 80].

Comme pour les modèles séquentiels, la notion d'observation du système est utilisée pour définir des équivalences comportementales. Ces équivalences permettent de faire abstraction de certains détails inutiles.

Par ailleurs, il est intéressant de pouvoir décrire des systèmes parallèles à différents niveaux d'abstraction. Aussi nous étudions la notion de *raffinement* d'actions qui formalise cette idée de conception descendante de systèmes. Cette opération de raffinement, sujet d'actualité, a été étudiée dans [AH 89], [AH 90], [BDKP 89], [CDMP 87], [vGG 89], [vGG 90], [vGW 89],

[Nie 89] et [Vog 90]. Naturellement, une équivalence sémantique doit être une congruence pour les opérateurs que l'on considère pour construire des programmes parallèles. Il s'avère que dans le cas des sémantiques basées sur l'interleaving, les équivalences de temps linéaire ainsi que les équivalences de temps arborescent (bisimulations) ne sont pas compatibles avec le raffinement d'actions.

Dans [Pra 86], [CDMP 87], les auteurs suggèrent les sémantiques basées sur les ordres partiels (sémantiques où les relations de causalité sont respectées). En effet, les équivalences de temps linéaire basées sur les ordres partiels sont compatibles avec le raffinement d'actions. Par contre, pour les équivalences de temps arborescent, [vGG 89] a montré que seule la "history preserving" bisimulation introduite dans [RT 88], [DNM 89], définie aussi dans [BDKP 91], est compatible avec le raffinement d'actions.

D'autre part, Van Glabbeek [vG 90b] a introduit la notion de ST-sémantique qui n'est pas explicitement basée sur les ordres partiels. L'idée est que l'état d'un système est défini non seulement par les actions qui ont eu lieu pour atteindre cet état mais aussi par celles qui sont en cours (actions actives dont l'exécution a commencé mais n'est pas encore terminée). De plus, il a prouvé que l'interleaving ST-bisimulation est une congruence pour le raffinement d'actions. Dans le même état d'esprit, Vogler dans [Vog 90] a montré que les ST-versions de la pomset, partial word et de la "history preserving" bisimulation sont compatibles avec le raffinement d'actions. De plus il a montré que les ST-versions de l'interleaving, pomset et history preserving bisimulations sont même les plus grandes congruences vis à vis du raffinement contenues respectivement dans l'interleaving bisimulation, la pomset bisimulation et la history preserving bisimulation.

Dans ce chapitre, nous étudions les équivalences de temps arborescent sur les structures d'événements. La section 2 présente les notions de base sur les structures d'événements premières. Dans la section 3, nous étudions les équivalences de bisimulation sur les structures d'événements premières. Dans la section 4, nous étudions le comportement des diverses bisimulations vis à vis de l'opération de raffinement d'actions. La section 5 présente les ST-sémantiques, sémantiques mieux adaptées au raffinement d'actions.

3.2 Les structures d'événements premières

Les structures d'événements constituent un modèle de base pour le vrai parallèlisme. C'est un modèle basé sur les ordres partiels qui expriment les dépendances causales entre les événements que peut effectuer un système.

Dans toute la suite Act = {a,b c,...} désigne un ensemble d'actions et $\mathcal{E}_{prim}$ est le domaine des stuctures d'événements premières étiquetées sur Act.

Définition 3.1 (Structure d'événements première)
Une structure d'événements première (étiquetée sur un alphabet Act) est un quadruplet $\mathcal{E}$ = (E, ≤, #, l) où:

- E est un ensemble d'événements,
- ≤ ⊆ ExE est un ordre partiel (relation de causalité) qui satisfait le principe de causes finies:

  $\forall e \in E : \{ e' \in E \mid e' \leq e \}$ est fini,
- # ⊆ ExE est une relation irreflexive, symétrique (relation de conflit) qui satisfait le principe d'hérédité de conflit:

  $\forall e, e', e'' \in E : e \leq e'$ et $e \# e''$ alors $e' \# e''$,
- l: E → Act est une fonction d'étiquetage.

Act est l'ensemble d'actions que le système modélisé peut effectuer. Un événement de E est l'occurrence d'une action de Act lors d'une exécution. e < e' signifie que e' ne peut avoir lieu que si e a eu lieu (e est une cause de e'). Deux événements e et e' qui sont en conflit (e # e') ne peuvent apparaitre tous les deux dans la même exécution. Deux événements sont indépendants (peuvent avoir lieu en parallèle) s'ils ne sont reliés ni par la relation de causalité ni par la relation de conflit. Si on note *co* la relation d'indépendance, formellement nous avons:

$$e \; co \; e' \text{ si et seulement si } \neg (e < e' \vee e' < e \vee e \# e').$$

Notons que nous écrivons e < e' quand e ≤ e' et e ≠ e'.
On appelle "step" un ensemble d'événements indépendants.
(Ø, Ø, Ø, Ø) est la structure d'événement vide notée 0. Une structure d'événement $\mathcal{E}$ = (E, ≤, #, l) est finie si son ensemble d'événements E est fini. Elle est sans conflit si $\#_{\mathcal{E}} = Ø$.
La notion d'isomorphisme nous permet de faire abstraction des noms des événements.

Définition 3.2 (structures d'événements isomorphes)
Deux structures d'événements $\mathcal{E}$ = (E, ≤, #, l) et $\mathcal{F}$ = (F, ≤', #', l') sont isomorphes si et seulement s'il existe une bijection de E dans F qui respecte les relations de causalité, de conflit et l'étiquetage.

On notera $\mathcal{E} \cong \mathcal{F}$ quand $\mathcal{E}$ et $\mathcal{F}$ sont deux structures d'événements isomorphes.

Définition 3.3 (pomsets)
Les classes d'isomorphismes de structures d'événements sans conflit sont appelées pomsets.

Les pomsets sont des exécutions possibles du système représenté par la structure d'événements.
L'ordre partiel entre les occurrences d'actions représente les dépendances causales dans l'exécution.

L'état d'un système appelé *configuration* est représenté par l'ensemble des événements ayant eu lieu pour atteindre cet état. Cet ensemble est fermé à gauche par rapport à la relation de causalité ($\leq$), toutes les causes d'un événement apparaissant dans une exécution doivent aussi avoir eu lieu. Nous ne considérons que les exécutions finies. Ainsi les configurations sont des ensembles finis.

Definition 3.4 (configuration)
Soit $\mathcal{E} = (E, \leq, \#, l)$ une structure d'événements, une configuration est un ensemble $C \subseteq E$ tel que:

- C est fermé à gauche par rapport à la relation de causalité:

  si $e \in C$ et $e' \leq e$ alors $e' \in C$

- C est sans conflit:

  $\forall e, e' \in C$ alors $\neg (e \# e')$

Nous noterons $\mathcal{C}(\mathcal{E})$ l'ensemble de toutes les configurations de $\mathcal{E}$. Une configuration C est dite *complète* si tous les événements qui ne sont pas dans C sont exclus pour cause du conflit. Formellement:

$$\forall e \in E : e \notin C \Rightarrow \exists e' \in C \text{ tel que } e' \# e$$

Si C est une configuration, $\mathcal{E}$ une structure d'événements, la restriction de $\mathcal{E}$ à C est définie par:

$$\mathcal{E} \lceil C = (C, \leq \cap (C \times C), \# \cap (C \times C), l \lceil C)$$

Par convention, nous désignons par C une configuration ainsi que la structure d'événements sous jacente ($\mathcal{E} \lceil C$).

Dans les représentations graphiques des structures d'événements, nous ne représentons pas les conflits hérités. Seuls les conflits immédiats sont indiqués. La relation de causalité (directe) est schématisée par un trait: e < e' est représenté sur les figures par un trait vertical ou oblique dont les extrémités supérieure et inférieure sont respectivement e et e'. De plus pour expliquer aisément les comportements des systèmes nous écrirons et le nom d'événement et son étiquette par e : a où $e \in E, E \subseteq N$ et a = l(e).

Exemple La structure d'événements qui modélise l'expression CCS a| b + ab est représentée dans la figure 3.1. Ce système peut effectuer soit a et b en parallèle soit a suivi de b. Les exécutions possibles du système sont représentées par les pomsets ∅, a, b, a b et a—b. Ces deux derniers pomsets correspondent à des configurations complètes.

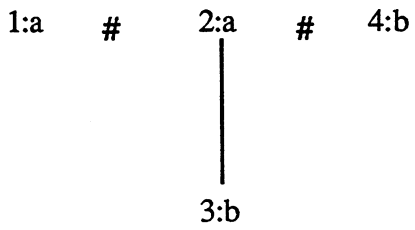


Figure 3.1: a| b +a.b

3.3 Bisimulations et comportements

Pour décrire le comportement d'une structure d'événements on lui associe un système de transitions étiqueté dont le changement d'état permet d'observer une ou plusieurs actions. Deux systèmes de transitions ont même comportement s'ils sont bisimilaires [Mil 83], [Par81] c.à.d. qu'il existe une relation sur les états qui relie deux états seulement si leurs transitions respectives par les mêmes actions ou calculs les font évoluer vers des états encore reliés. Suivant ce qui est observé lors du changement d'état, nous obtenons diverses bisimulations. Si on observe au plus une action, l'exécution de plusieurs actions en parallèle revient à les entremêler obtenant ainsi l'interleaving bisimulation. Le changement d'état peut aussi avoir lieu soit par l'exécution de plusieurs actions indépendantes (un step) soit par plusieurs actions partiellement ordonnées. Dans le premier cas, nous obtenons la step bisimulation et dans le second cas, la pomset ou partial word bisimulation. Les pomsets et les partial words représentent le même objet mathématique à savoir les ordres partiels étiquetés par des actions

[Gra 81]. Nous expliciterons la différence entre les deux dernières bisimulations lors de leurs définitions.

Dans cette section, nous étudions les équivalences de temps arborescent sur les structures d'événements.

Le comportement d'un système parallèle est décrit par un système de transitions dont les états sont des configurations.

Définition 3.4
Soit $\mathcal{E}$ une structure d'événements, $C, C' \in \mathcal{C}(\mathcal{E})$, a $\in$ Act:
i) $C \rightarrow_{\mathcal{E}} C'$ si $C \subseteq C'$
ii) $C \xrightarrow{a} C'$ si a $\in$ Act, $C \rightarrow_{\mathcal{E}} C'$ et $C' - C = \{e\}$ tel que l(e) = a.

Remarque: quand $C \rightarrow_{\mathcal{E}} C'$, la structure d'événements restreinte à l'ensemble d'événements $C' - C$ est finie et sans conflit.

La bisimulation de base sur les structures d'événements est celle où un pas de calcul a lieu en effectuant au plus une action, c'est l'interleaving bisimulation.

Définition 3.5 (interleaving bisimulation)
Deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ sont interleaving bisimilaires s'il existe une relation symétrique $R \subseteq \mathcal{C}(\mathcal{E})\times\mathcal{C}(\mathcal{F}) \cup \mathcal{C}(\mathcal{F})\times\mathcal{C}(\mathcal{E})$ (appelée interleaving bisimulation) telle que:
i) $(\emptyset, \emptyset) \in R$
ii) Pour toute transition $C \xrightarrow{a} C'$ telle que $(C, D) \in R$ il existe D' telle que $D \xrightarrow{a} D'$ avec $(C', D') \in R$

On notera $\mathcal{E} \approx_i \mathcal{F}$ quand $\mathcal{E}$ et $\mathcal{F}$ sont deux structures d'événements interleaving bisimilaires.

Exemple La figure 3.2 donne l'exemple classique des agents CCS P = ab et Q = a.b + b.a qui sont interleaving bisimilaires.

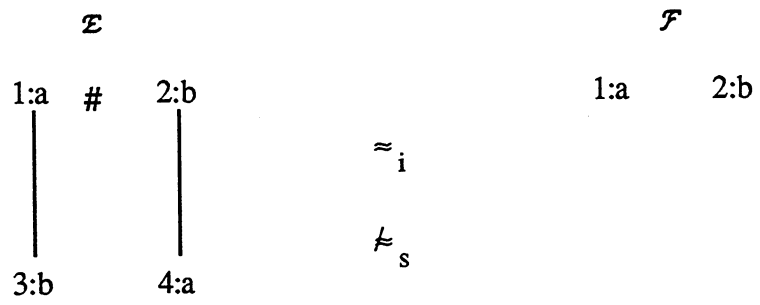


Figure 3.2

La step bisimulation est plus forte que la précédente. Elle permet des transitions par un ensemble d'actions indépendantes ou step. Le mot step prend ses origines des réseaux de Pétri où il dénote un ensemble ou un multiensemble de transitions exécutables simultanément.

Formellement la step sémantique généralise la transition $C \xrightarrow{a} C'$ à $C \xrightarrow{A} C'$ où A est un multiensemble où la même action peut apparaitre plus d'une fois.

Définition 3.6 (step bisimulation)

Deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ sont step bisimilaires s'il existe une relation symétrique $R \subseteq \mathcal{C}(\mathcal{E})\times\mathcal{C}(\mathcal{F}) \cup \mathcal{C}(\mathcal{F})\times\mathcal{C}(\mathcal{E})$ (appelée step bisimulation) telle que:

i) $(\emptyset, \emptyset) \in R$

ii) Pour toute transition $C \xrightarrow{A} C'$ telle que $(C, D) \in R$ il existe D' telle que $D \xrightarrow{A} D'$ avec $(C', D') \in R$

On notera $\mathcal{E} \approx_s \mathcal{F}$ quand $\mathcal{E}$ et $\mathcal{F}$ sont deux structures d'événements step bisimilaires.

Exemple La figure 3.3 donne des exemples de structures d'événéments qui sont step bisimilaires.

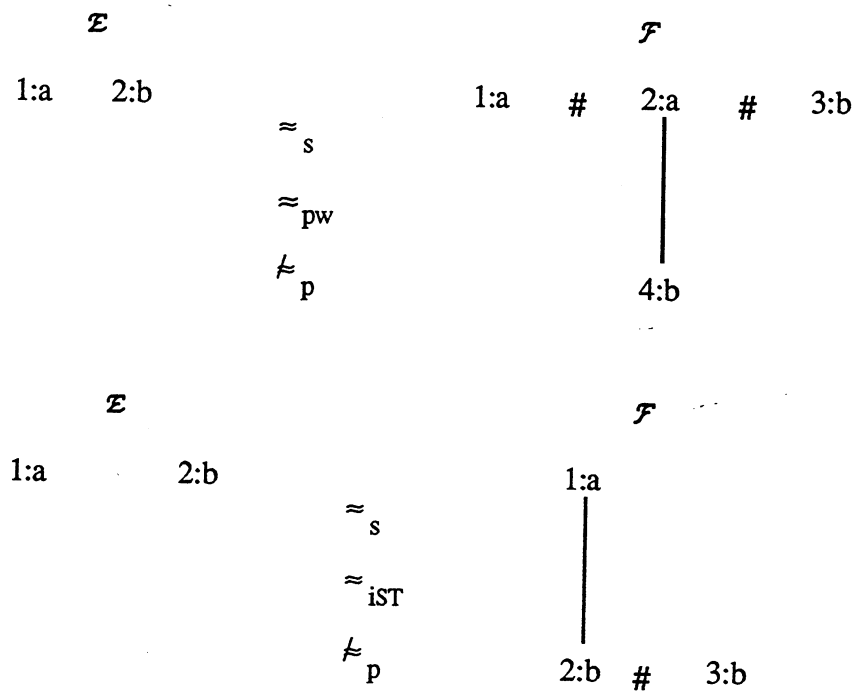


Figure 3.3

Il est bien clair que $\mathcal{E} \approx_s \mathcal{F} \Rightarrow \mathcal{E} \approx_i \mathcal{F}$. Cette implication est stricte: les structures d'événements de le figure 3.2 sont interleaving bisimilaires mais pas step bisimilaires. En effet, seul $\mathcal{F}$ peut éffectuer le step a b.

Boudol et Castelleni [BC 87] ont suggéré une généralisation de l'équivalence de bisimulation en considérant des transitions étiquetées par des pomsets. Ainsi, ils considèrent des transitions $C \xrightarrow{p} C'$ si et seulement si $C \rightarrow_{\mathcal{E}} C'$ où p est un pomset sur Act.

Définition 3.7

$C \xrightarrow{p} C'$ si et seulement si $C \rightarrow_{\mathcal{E}} C'$ et p est la classe d'isomorphisme de C'- C.

Définition 3.8 (pomset bisimulation)
Deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ sont pomset bisimilaires s'il existe une relation symétrique $R \subseteq \mathcal{C}(\mathcal{E})\times\mathcal{C}(\mathcal{F}) \cup \mathcal{C}(\mathcal{F})\times\mathcal{C}(\mathcal{E})$ (appelée pomset bisimulation) telle que:
i) $(\emptyset, \emptyset) \in R$

ii) Pour toute transition $C \xrightarrow{p} C'$ telle que $(C, D) \in R$ il existe D' telle que $D \xrightarrow{p} D'$ avec $(C', D') \in R$

On notera $\mathcal{E} \approx_p \mathcal{F}$ quand $\mathcal{E}$ et $\mathcal{F}$ sont pomset bisimilaires.

Exemple La figure 3.4 donne un exemple de deux systèmes qui sont pomset bisimilaires.

Cette équivalence est plus forte que la step bisimulation : $\mathcal{E} \approx_p \mathcal{F} \Rightarrow \mathcal{E} \approx_s \mathcal{F}$. Cette implication est stricte. Les exemples de la figure 3.3 qui sont step bisimilaires ne sont pas pomset bisimilaires. En effet, dans les deux cas seul $\mathcal{F}$ peut effectuer le pomset a suivi de b (où a est une cause de b).

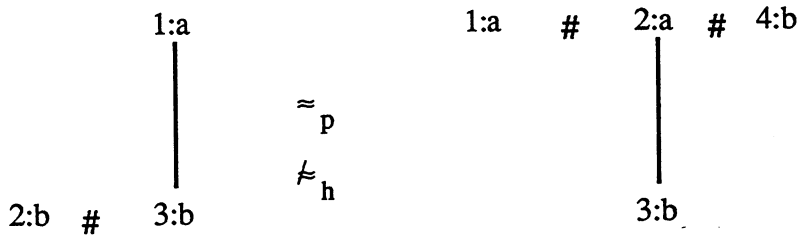


Figure 3.4

Vogler [Vog 90] a introduit une autre équivalence basée sur sur les ordres partiels : la "partial word" bisimulation. Les partial words et les pomsets représentent le même objet mathématique, les ordres partiels étiquetés. La différence essentielle avec la pomset bisimulation est que dans la partial word un pas par un pomset a—b (a cause de b) peut être imité par le pomset a b où a et b sont des actions indépendantes. L'inverse n'est pas possible. Dans ce cas, le parallélisme est plus que l'interleaving mais en même temps contient l'interleaving.

Définition 3.9

Soient C et D deux configurations. On dit que C est moins séquentielle que D s'il existe une bijection $f: C \rightarrow D$ telle que f preserve l'étiquetage et f est un homomorphisme.

Notons que cette définition implique que C et D décrivent les mêmes occurrences d'actions et que toutes les relations de causalité qui existent dans C se retrouvent dans D, mais D peut en contenir d'autres.

Définition 3.10 (partial word bisimulation)
Deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ sont partial word bisimilaires s'il existe une relation symétrique $R \subseteq \mathcal{C}(\mathcal{E})x\mathcal{C}(\mathcal{F}) \cup \mathcal{C}(\mathcal{F})x\mathcal{C}(\mathcal{E})$ (appelée partial word bisimulation) telle que:
i) $(\emptyset, \emptyset) \in R$

ii) Pour toute transition $C \xrightarrow{p} C'$ telle que $(C, D) \in R$ il existe D', q tels que :

- $D \xrightarrow{q} D'$ avec $(C', D') \in R$
- q est moins séquentiel que p, noté $q \sqsubseteq p$.

On notera $\mathcal{E} \approx_{pw} \mathcal{F}$ quand $\mathcal{E}$ et $\mathcal{F}$ sont partial word bisimilaires.

Exemple La figure 3.5 donne des exemples de structures d'événements partial word bisimilaires.

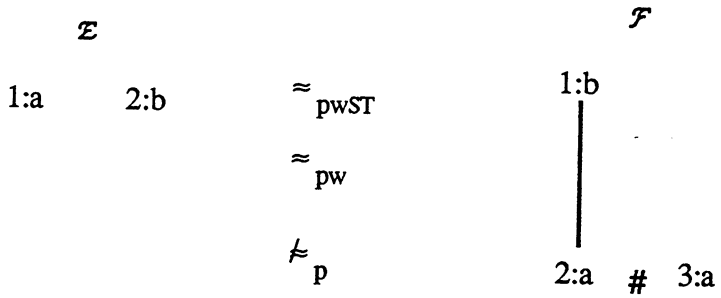


Figure 3.5

Il est facile de remarquer que la partial word bisimulation se situe entre la step bisimulation et la pomset bisimulation. Ainsi $\mathcal{E} \approx_{p} \mathcal{F} \Rightarrow \mathcal{E} \approx_{pw} \mathcal{F} \Rightarrow \mathcal{E} \approx_{s} \mathcal{F}$. Ces implications sont toutes strictes: Les exemples de la figure 3.5 sont partial word bisimilaires mais pas pomset bisimilaires. Seul $\mathcal{F}$ peut effectuer le pomset b—a. La figure 3.6 donne un exemple de deux systèmes step bisimilaires mais pas partial word bisimilaires.

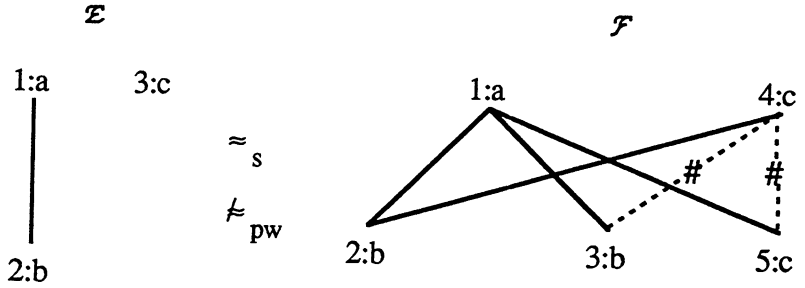


Figure 3.6

Quand $\mathcal{E}$ effectue la transition $\{\emptyset\} \rightarrow_{\mathcal{E}} \{1{:}a, 2{:}b, 3{:}c\}$, $\mathcal{F}$ l'imite en effectuant la transition $\{\emptyset\} \rightarrow_{\mathcal{F}} \{1{:}a, 3{:}b, 5{:}c\}$. Le pomset q = {1:a, 3:b, 5:c} de $\mathcal{F}$ qui a imité le pomset p = {1:a, 2:b, 3:c} de $\mathcal{E}$ n'est pas moins séquentiel que p puisque dans q, 1:a est une cause de 5:c alors que dans p 1:a et 3:c sont deux événements indépendants.

Une autre équivalence de bisimulation basée sur les ordres partiels a été proposé dans [OGvG 88] et [Dev 88], la "weak history preserving" bisimulation définie comme suit sur les structures d'événements:

Définition 3.11 (weak history preserving bisimulation)
Deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ sont weak history preserving bisimilaires s'il existe une relation symétrique $R \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{F}) \cup \mathcal{C}(\mathcal{F}) \times \mathcal{C}(\mathcal{E})$ (appelée weak history preserving bisimulation) telle que:
i) $(\emptyset, \emptyset) \in R$
ii) Si $(C, D) \in R$ alors

- C et D sont isomorphes,
- Pour toute transition $C \rightarrow C'$, il existe D' telle que $D \rightarrow D'$ avec $(C', D') \in R$.

L'isomorphisme entre les configurations reliées par R garantit que les étiquettes des événements de $C' - C$ et de $D' - D$ se correspondent.

On notera $\mathcal{E} \approx_{wh} \mathcal{F}$ quand $\mathcal{E}$ et $\mathcal{F}$ sont weak history preserving bisimilaires.

Exemple La figure 3.7 contient deux structures d'événements qui sont weak history preserving bisimilaires.

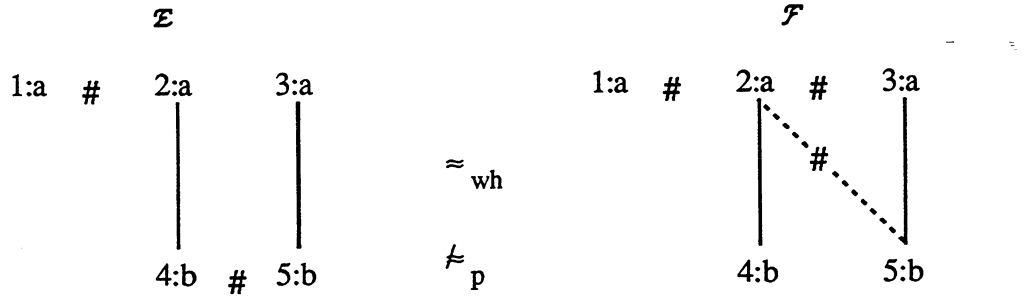


Figure 3.7

Cette équivalence de bisimulation est incomparable avec la pomset bisimulation : les structures d'événements de la figure 3.4 sont pomset bisimilaires mais pas weak history preserving bisimilaires. Quand $\mathcal{F}$ effectue l'événement 2:a, $\mathcal{E}$ l'imite en éffectuant l'événement 1:a. Si maintenant $\mathcal{E}$ effectue l'événement 2:b, $\mathcal{F}$ ne peut effectuer que 3:b pour l'imiter. Ainsi nous avons $\{2:a\} \rightarrow_{\mathcal{F}} \{2:a, 3:b\}$ et $\{1:a\} \rightarrow_{\mathcal{E}} \{1:a, 2:b\}$, il est clair que les configurations $\{2:a, 3:b\}$ de $\mathcal{F}$ et $\{1:a, 2:b\}$ de $\mathcal{E}$ ne sont pas isomorphes.
Les structures d'événements de la figure 3.7 qui sont weak history preserving ne sont pas pomset bisimilaires : dans $\mathcal{E}$, après avoir effectué a, il est toujours possible d'effectuer le pomset a—b (a est une cause de b). Ce comportement n'est pas toujours possible si $\mathcal{F}$ effectue l'événement 2:a, il ne peut en aucun cas effectuer le pomset a—b.

Dans ce qui suit, nous décrivons la bisimulation la plus forte qui est incluse à la fois dans la weak history preserving et la pomset bisimulation : la "history preserving" bisimulation. Elle a été introduite pour la première fois dans [RT 88] sous le nom de "behaviour structure" bisimulation. Elle est aussi définie dans [DDNM 89], [BDKP 91], [vGG 89.

Définition 3.12 (history preserving bisimulation)
Deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ sont history preserving bisimilaires s'il existe une relation symétrique[1] $R \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{F}) \times \mathcal{P}(E_{\mathcal{E}} \times E_{\mathcal{F}}) \cup \mathcal{C}(\mathcal{F}) \times \mathcal{C}(\mathcal{E}) \times \mathcal{P}(E_{\mathcal{F}} \times E_{\mathcal{E}})$ (appelée history preserving bisimulation) telle que:
i)$(\emptyset, \emptyset, \emptyset) \in R$
ii) Si $(C, D, f) \in R$ alors

- f est un isomorphisme entre C et D,
- Pour toute transition $C \rightarrow C'$, il existe D', f' tels que $D \rightarrow D'$ avec $(C', D', f') \in R$ et $f' \lceil C = f$.

On notera $\mathcal{E} \approx_h \mathcal{F}$ quand $\mathcal{E}$ et $\mathcal{F}$ sont history preserving bisimilaires.

Il est clair que la history preserving bisimulation implique la weak history preserving : $\mathcal{E} \approx_h \mathcal{F} \Rightarrow \mathcal{E} \approx_{wh} \mathcal{F}$. Cette implication est stricte, les structures d'événements de la figure 3.7 ne sont pas history preserving bisimilaires. La transition $\{\emptyset\} \rightarrow_{\mathcal{F}} \{2:a\}$ est imitée dans $\mathcal{E}$ par la transition $\{\emptyset\} \rightarrow_{\mathcal{E}} \{2:a\}$ avec l'isomorphisme f: tel f(2)=2. Quand $\mathcal{E}$ effectue $\{2:a\} \rightarrow_{\mathcal{E}} \{2:a, 3:a\}$, $\mathcal{F}$ l'imite par $\{2:a\} \rightarrow_{\mathcal{F}} \{2:a, 3:a\}$ avec l'isomorphisme f': f'(2) = 2, f'(3) = 3 . La

[1] si (C, D, f) $\in R$ alors $(D, C, f^{-1}) \in R$

transition $\{2{:}a, 3{:}b\} \rightarrow_{\mathcal{E}} \{2{:}a, 3{:}a, 5{:}b\}$ est imitée dans $\mathcal{F}$ par $\{2{:}a, 3{:}b\} \rightarrow_{\mathcal{F}} \{2{:}a, 3{:}a, 4{:}b\}$ avec f'' un isomorphisme tel que f''(2) = 3, f''(3) = 2 et f''(4) = 5. Notons que cet isomorphisme ne prolonge pas le précédent. Donc $\mathcal{E}$ et $\mathcal{F}$ ne sont pas history preserving bisimilaires. Il en est de même quand $\mathcal{F}$ effectue 2:a et que $\mathcal{E}$ l'imite en effectuant 3:a.

Une des caractéristiques intéressantes de la history preserving bisimulation est qu'elle préserve les pomsets.

Proposition 3.1 [vG 90a]

$$\mathcal{E} \approx_h \mathcal{F} \Rightarrow \mathcal{E} \approx_p \mathcal{F}$$

Preuve

La preuve est simple, il suffit de montrer qu'une relation R qui est une history preserving bisimulation entre deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ se projette selon ses deux premières composantes en une pomset bisimulation entre $\mathcal{E}$ et $\mathcal{F}$.

Supposons $(C, D, f) \in R$ et $C \xrightarrow{p} C'$. Alors $C \rightarrow C'$, donc il existe D', f' tels que $D \rightarrow D'$ avec $(C', D', f') \in R$ et $f' \lceil C = f$. Comme f' est un isomorphisme entre C' et D' et que $f' \lceil C = f$, nous avons codomaine $(f' \lceil (C' - C)) =$ codomaine (f') - codomaine$(f) = D' - D$. Donc $f' \lceil (C' - C)$ est un isomorphisme entre $C' - C$ et $D' - D$, ce qui implique $D \xrightarrow{p} D'$. La projection de R est donc bien une pomset bisimulation □.

L'implication de la proposition 3.1 est stricte. Les systèmes de la figure 3.4 qui sont pomset bisimilaires ne sont pas weak history preserving donc nécessairement pas history preserving bisimilaires.

Van Glabbeek a aussi montré dans [vG 90a] que dans le cas où il n'y a pas d'autoconcurrence (deux événements indépendants ont la même étiquette), la weak history preserving bisimulation coincide avec la history preserving bisimulation.

Pour terminer cette section, rappelons les implications qui existent entre les équivalences que nous avons décrites:

$$\mathcal{E} \approx_h \mathcal{F} \Rightarrow \mathcal{E} \approx_p \mathcal{F} \Rightarrow \mathcal{E} \approx_{pw} \mathcal{F} \Rightarrow \mathcal{E} \approx_s \mathcal{F} \Rightarrow \mathcal{E} \approx_i \mathcal{F}$$

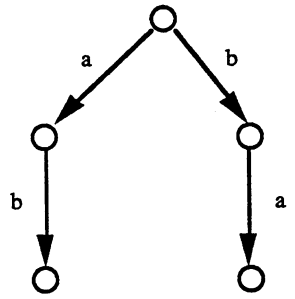
$$\mathcal{E} \approx_h \mathcal{F} \Rightarrow \mathcal{E} \approx_{wh} \mathcal{F} \Rightarrow \mathcal{E} \approx_s \mathcal{F} \Rightarrow \mathcal{E} \approx_i \mathcal{F}$$

3.4 Bisimulations et raffinement d'actions

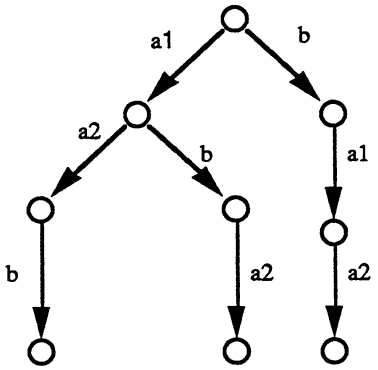
3.4.1 Raffinement d'actions

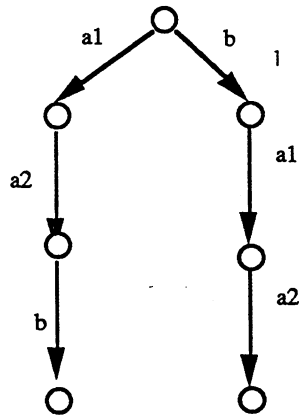
L'opération de raffinement formalise la notion de conception des systèmes parallèles à différents niveaux d'abstraction. Ainsi une action à un niveau abstrait peut représenter un comportement complexe à un niveau plus concret. Il est bien connu que pour définir une opération, on choisit un modèle pour représenter les programmes parallèles et on étudie le comportement des équivalences sémantiques (définies sur ce modèle) par rapport à l'opération en question.
Dans la littérature, on trouve deux grandes catégories de modèles sémantiques: les modèles de graphes (ou modèles séquentiels) décrits dans le chapitre 1 et les modèles de vrai parallélisme dont les structures d'événements, les réseaux de Pétri .

L'opération de raffinement ne semble pas être naturelle sur les graphes : si raf est une fonction de raffinement et P, Q deux processus équivalents raf(P| Q) n'est pas équivalent à raf(P)| raf(Q). Pour cela, il suffit de prendre l'exemple du processus a b équivalent à a.b + b.a qui sont tous deux représentés par le graphe de processus de la figure 3.8. Cependant, si on raffine a en $a_1.a_2$. raf(P) = $(a_1.a_2)$| b et raf(Q) = $a_1.a_2.b + b.a_1.a_2$ dont les graphes respectifs sont donnés dans la figure 3.8. Ces deux graphes n'ont même pas les mêmes séquences d'exécutions: seul le processus raf(P) peut effectuer la séquence $a_1.b.a_2$. Ainsi le raffinement sur les modèles de graphes n'est possible que si l'on impose des contraintes pour éviter ce type de situation. Certains chercheurs utilisent par exemple la notion de raffinement "atomique" [Gor 91] auquel cas la séquence $a_1.b.a_2$ de l'exemple ci dessus n'est plus possible.



P = a|b





raf(a| b) = $(a_1.a_2)$| b

raf(a.b + b.a) = $a_1.a_2.b + b.a_1.a_2$

Figure 3.8

Dans cette section nous étudions l'opération de raffinement sur les structures d'événements premières qui généralisent les graphes d'états. Le raffinement consiste à remplacer une action par une structure d'événements non vide et sans conflit. La contrainte sans conflit pouvant être levée sur des structures d'événements plus générales que les structures d'événements premières [BC 89].

Etant donnée une structure d'événements $\mathcal{E}$, si a est une action à raffiner, la structure d'événements raffinée est obtenue en remplaçant tous les événements étiquetés par a par une copie $\mathcal{E}_e$ de raf(a). Les relations de causalité et de conflit sont héritées de $\mathcal{E}$: tous les événements qui sont des causes de e dans $\mathcal{E}$, deviennent des causes de tous les événements de $\mathcal{E}_e$. Tous les événements qui sont en conflit avec e dans $\mathcal{E}$ sont en conflit avec tous les événements de $\mathcal{E}_e$.

Définition 3.13 (raffinement)

• Une fonction raf: Act $\longrightarrow \mathcal{E}_{prim}$ - {0} est une fonction de raffinement si

$\forall$ a $\in$ Act : raf(a) est finie et est sans conflit. Par convention, nous écrirons raf(e) pour raf($l_{\mathcal{E}}$(e)).

• Si $\mathcal{E} = (E_{\mathcal{E}}, \leq_{\mathcal{E}}, \#_{\mathcal{E}}, l_{\mathcal{E}}) \in \mathcal{E}_{prim}$, raf une fonction de raffinement, raf($\mathcal{E}$) est la structure d'événements raf ($\mathcal{E}$) = ($E_{raf(\mathcal{E})}$, $\leq_{raf(\mathcal{E})}$, $\#_{raf(\mathcal{E})}$, $l_{raf(\mathcal{E})}$) où

- $E_{raf(\mathcal{E})} = \{(e, e') | \ e \in E_{\mathcal{E}}, e' \in E_{raf(e)}\}$
- (d,d') $\leq_{raf(\mathcal{E})}$ (e,e') si et seulement si d$<_{\mathcal{E}}$ e ou bien d=e et d' $<_{raf(d)}$ e'
- (d,d') $\#_{raf(\mathcal{E})}$ (e,e') si et seulement si d$\#_{\mathcal{E}}$ e
- $l_{raf(\mathcal{E})}$ (e, e') = $l_{raf(e)}$ (e')

L'opération de raffinement est une opération bien définie puisqu'on vérifie facilement que:

- Si $\mathcal{E} \in \mathcal{E}_{prim}$, raf($\mathcal{E}$) est aussi une structure d'événements première.
- Si $\mathcal{E} \in \mathcal{E}_{prim}$, raf et raf' deux raffinements tels que raf(a) $\cong$ raf'(a) pour tout a $\in$ Act, alors raf($\mathcal{E}$)$\cong$ raf'($\mathcal{E}$)
- Si $\mathcal{E}$ et $\mathcal{F} \in \mathcal{E}_{prim}$, tel que $\mathcal{E} \cong \mathcal{F}$ alors raf($\mathcal{E}$) $\cong$ raf($\mathcal{F}$).

Le comportement d'une structure d'événement raffinée, raf($\mathcal{E}$) est déterminé par les comportements de $\mathcal{E}$ et des stuctures d'événements qui ont remplacé les actions. Les configurations de raf($\mathcal{E}$) raffinent les configurations de $\mathcal{E}$. Le raffinement d'une configuration C de $\mathcal{E}$ est obtenu en remplaçant tout événement e de C par une configuration non vide C_e de raf(e). Les événements qui sont des prerequis pour d'autres événements de C doivent être remplacés par des configurations complètes.
Formellement, les configurations de raf($\mathcal{E}$) sont caractérisées comme suit:

Proposition 3.2 [vG 90a]
soit $\mathcal{E} \in \mathcal{E}_{prim}$, raf une fonction de raffinement,
i) $\widetilde{C} \subseteq E_{raf(\mathcal{E})}$ est une configuration de raf($\mathcal{E}$) si et seulement si : $\widetilde{C} = \{(e,e') \mid e \in C, e' \in C_e\}$
où C est une configuration de $\mathcal{E}$ et C_e une configuration de raf(e) pour e$\in$ C,
$C_e = E_{raf(e)}$ si e n'est pas maximal dans C par rapport à la relation $<_{\mathcal{E}}$,

ii) Si $\widetilde{C} \rightarrow_{raf(\mathcal{E})} \widetilde{C}'$ alors $pr_1(\widetilde{C}) \rightarrow pr_1(\widetilde{C}')$ où $pr_1(\widetilde{C}) = \{e \mid (e, e') \in \widetilde{C}\}$ (pr_1 est la projection par rapport au premier élément).

3.4.2 Propriétés de congruence

Une équivalence $\approx$ est compatible avec le raffinement d'actions si :
$\forall a \in Act$ et $\forall raf, raf' : Act \rightarrow \mathcal{E}_{prim} - \{0\}$,

- $raf(a) \approx raf'(a) \Rightarrow raf(\mathcal{E}) \approx raf'(\mathcal{E})$
- $\mathcal{E} \approx \mathcal{F} \Rightarrow raf(\mathcal{E}) \approx raf(\mathcal{F})$

L'interleaving bisimulation n'est pas compatible avec le raffinement . Pour s'en convaincre il suffit de raffiner l'action a au pomset a_1— a_2 dans l'exemple de la figure 3.2. $raf(\mathcal{E})$ et $raf(\mathcal{F})$ représentés dans la figure 3.9 ne sont plus interleaving bisimilaires: Ils n'ont même pas les mêmes séquences d'exécution. Seul $raf(\mathcal{F})$ peut effectuer la séquence $a_1.b.a_2$.

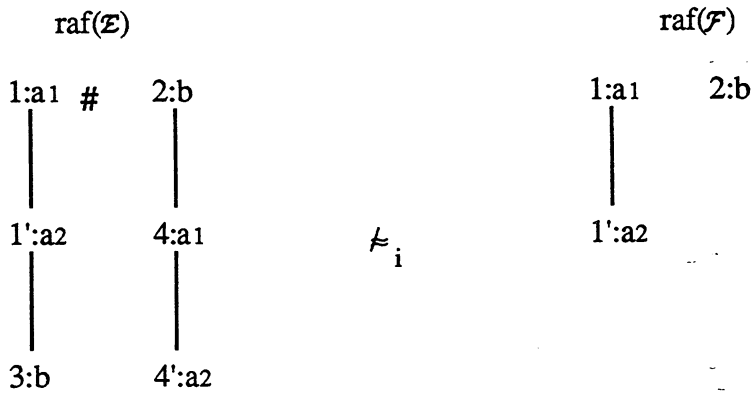


Figure 3.9

La step bisimulation n'est pas une congruence pour le raffinement d'actions. La figure 3.10 représente les structures d'événements du premier exemple de la figure 3.3 où a est raffiné au pomset a_1— a_2. $raf(\mathcal{E})$ et $raf(\mathcal{F})$ ne sont pas step bisimilaires : en effet dans $raf(\mathcal{E})$, après avoir effectué le step $\{1:a_1\}$ il est toujours possible d'effectuer le step $\{2:b\}$. Ce comportement n'est pas possible dans $raf(\mathcal{F})$ quand ce dernier effectue en premier le step $\{2:a_1\}$ ($raf(\mathcal{E})$ et $raf(\mathcal{F})$ ne sont même pas interleaving bisimilaires). Ce même exemple permet d'affirmer que la partial word bisimulation n'est pas compatible avec le raffinement, puisque les deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ qu'on a raffinées sont aussi partial word bisimilaires.

La pomset bisimulation n'est pas compatible avec le raffinement d'actions. Toujours en raffinant l'action a en a_1— a_2, la figure 3.11 contient les structures d'événements raffinées des systèmes de la figure 3.4, Il est facile de vérifier que raf($\mathcal{E}$) et raf($\mathcal{F}$) ne sont pas interleaving bisimilaires, donc pas pomset bisimilaires.

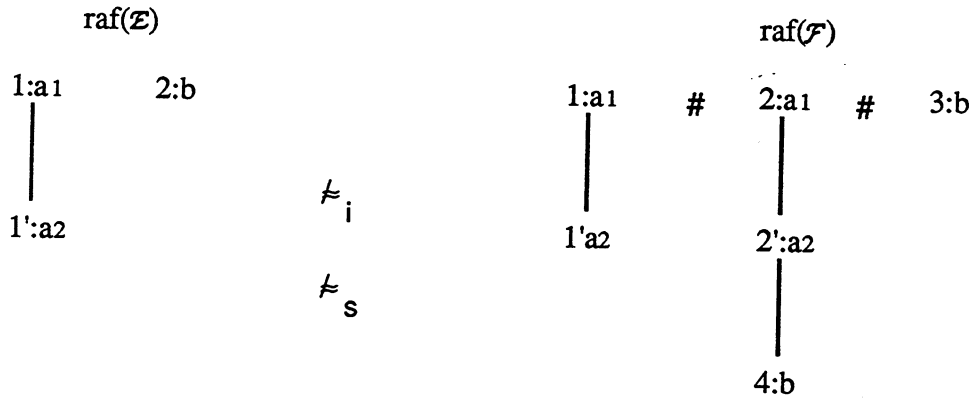


Figure 3.10

De la même façon on vérifie facilement que les structures d'événements de la figure 3.7 qui sont weak history preserving ne sont même plus interleaving bisimilaires quand on raffine l'action a.

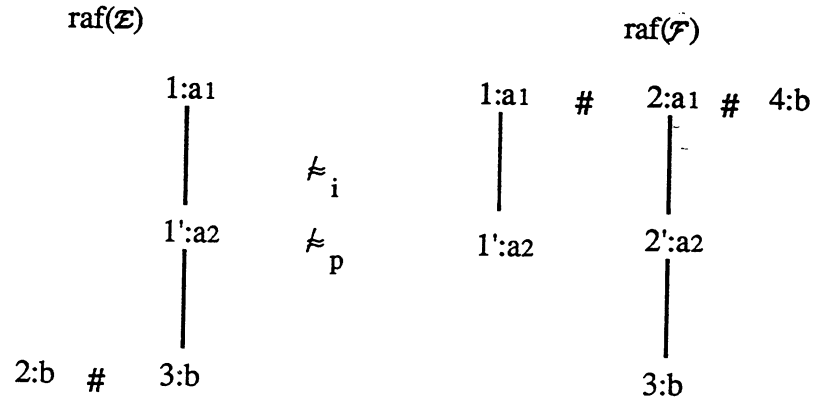


Figure 3.11

La history preserving bisimulation qui respecte à la fois la structure arborescente des systèmes parallèles et les relations de causalité entre les actions est compatible avec le raffinement d'actions.

Proposition 3.3 [vG 90a]

i) $\mathcal{E} \approx_h \mathcal{F} \Rightarrow raf(\mathcal{E}) \approx_h raf(\mathcal{F})$

ii) $\Rightarrow raf(a) \approx_h raf'(a) \Rightarrow raf(\mathcal{E}) \approx_h raf'(\mathcal{E})$

Preuve

Nous prouvons seulement le point i)

Soit $R : \mathcal{E} \approx_h \mathcal{F}$. On définit $\tilde{R} = \{(\tilde{C}, \tilde{D}, \tilde{f}) \in \mathcal{C}(raf(\mathcal{E})) \times \mathcal{C}(raf(\mathcal{F})) \times \mathcal{P}(E_{raf(\mathcal{E})} \times E_{raf(\mathcal{F})}) \mid \exists (C, D, f) \in R$ tel que $\tilde{C}$ et $\tilde{D}$ sont respectivement les raffinements de C et D, $\tilde{f}$ est une bijection de $\tilde{C}$ dans $\tilde{D}$ telle que $\tilde{f}(e, e') = (f(e), e')\}$.

Montrons que $\tilde{R}$ est une history preserving bisimulation entre $raf(\mathcal{E})$ et $raf(\mathcal{F})$.

i) $(\emptyset, \emptyset, \emptyset) \in \tilde{R}$ puisque $(\emptyset, \emptyset, \emptyset) \in R$.

ii) Supposons $(\tilde{C}, \tilde{D}, \tilde{f}) \in \tilde{R}$. Soit $(C, D, f) \in R$ tel que $\tilde{C}$ est le raffinement de C, $\tilde{D}$ est le raffinement de D et $\tilde{f}$ une bijection de $\tilde{C}$ dans $\tilde{D}$ telle que $\tilde{f}(e, e') = (f(e), e')$.

- Il est évident que $\tilde{f} : \tilde{C} \rightarrow \tilde{D}$ est un isomorphisme puisque :

  $(d, d') \leq_{\tilde{C}} (e, e') \Leftrightarrow \tilde{f}(d, d') \leq_{\tilde{D}} \tilde{f}(e, e')$ et $l_{raf(\mathcal{F})}(\tilde{f}(e, e')) = l_{raf(\mathcal{E})}(e, e'$.

- Pour vérifier la propriété de transfert, il faut prouver que quand $\tilde{C} \rightarrow_{raf(\mathcal{E})} \tilde{C}'$, il existe $\tilde{D}'$, $\tilde{f}'$ tels que $\tilde{D} \rightarrow_{raf(\mathcal{F})} \tilde{D}'$, $\tilde{f}' \lceil \tilde{C} = \tilde{f}$ avec $(\tilde{C}', \tilde{D}', \tilde{f}') \in \tilde{R}$.

  Supposons $\tilde{C} \rightarrow_{raf(\mathcal{E})} \tilde{C}'$ c.à.d. $\tilde{C}'$ est une configuration de $raf(\mathcal{E})$ et $\tilde{C} \subseteq \tilde{C}'$. $\tilde{C}' = \{(e, e') \mid e \in C'$ et $e' \in C'_e\}$ où C' est une configuration de $\mathcal{E}$ et $\forall e \in C'$, C'_e est une configuration non vide de raf(e). Soit $C = pr_1(\tilde{C})$ et $C' = pr_1(\tilde{C}')$. D'après la proposition 3. 2, nous avons $C \rightarrow_{\mathcal{E}} C'$. Comme R est une history preserving bisimulation entre $\mathcal{E}$ et $\mathcal{F}$, il existe D', f' tels que $D \rightarrow_{\mathcal{F}} D'$ avec $f' \lceil C = f$ et $(C', D', f') \in R$.

  Soit $\tilde{D}' = \{(f'(e), e') \mid (e, e') \in \tilde{C}'\}$, $\tilde{f}' = \{((e, e'), (f'(e), e')) \mid (e, e') \in \tilde{C}'\}$.

  Par construction $\tilde{D}' = \{(f'(e), e') \mid e \in C', e' \in C'_e\} = \{(e, e') \mid e \in D', e' \in D'_e\}$ où $\forall e \in D'$, $D'_e = C'_d$ où $d = f'^{-1}(e)$. De plus d'après la proposition 3.2, $C_e = E_{raf(e)}$ si e non maximal dans C.

  $D'_e = C'_d$ (où $d = f'^{-1}(e)$) est égale à $E_{raf(d)} = E_{raf(e)}$ si $d = f'^{-1}(e)$ non maximal dans C' donc e non maximal dans D'. $\tilde{D}'$ est donc bien une configuration raffinée de D'. Donc clairement, $(\tilde{C}', \tilde{D}', \tilde{f}') \in \tilde{R}$. □

3.5 ST-sémantiques

Beaucoup de chercheurs ont plaidé pour l'utilisation de l'opération de raffinement d'actions dans le cas où l'on désire concevoir des systèmes parallèles à différents niveaux d'abstraction.

Dans la section précédente, on a vu qu'à l'exception de la history preserving bisimulation (la plus forte) toutes les bisimulations (définies dans ce chapitre) sur les structures d'événements ne sont pas compatibles avec le raffinement. Ce résultat n'est pas surprenant puisque dans toutes ces équivalences, les actions que peuvent effectuer les processus sont considérées atomiques ou instantanées. C'est ainsi que van Glabbeek et Vaandrager ont propoposé pour la première fois dans [vGG 87] la notion de ST-sémantiques comme une variante des "split" sémantiques où le début et la fin de l'occurrence de la même action sont explicitement liés. Les ST-sémantiques se distinguent des autres sémantiques esssentiellement dans la notion d'état. L'état d'un système est défini non seulement par les actions qui ont eu lieu mais aussi par les actions actives : actions en cours non encore terminées.

Définition 3.14 (ST-configuration)
Soit $\mathcal{E} = (E, \leq, \#, l) \in \mathcal{E}_{prim.}$. Une ST-configuration est une paire (C, P) où $C \subseteq E$, $P \subseteq E$ tel que $P \subseteq C$, C est fini, sans conflit avec :

$$e' < e \in C \Rightarrow e' \in P$$

Une ST-configuration représente l'état d'un système parallèle où C (current) contient les événements dont l'exécution a commencé et P (past) contient les événements dont l'exécution est terminée. Il est évident que C - P contient que les éléments maximaux dans C.

Une configuration ordinaire peut être considérée comme une ST-configuration où $P = C$. Nous notons $\delta(\mathcal{E})$ l'ensemble de toutes les ST-configurations d'une stucture d'événements $\mathcal{E}$. Les ST-équivalences sont des relations sur les ST-configurations.

Définition 3.15
$(C, P) \rightarrow_{\mathcal{E}} (C', P')$ si $(C, P), (C', P') \in \delta(\mathcal{E})$, $C \subseteq C'$ et $P \subseteq P'$

Dans ce qui suit, nous étudions les ST-versions des diverses bisimulations sur les structures d'événements.

Définition 3.16 (ST-bisimulations)
Deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ sont α-ST-bisimilaires, pour $\alpha \in$ {interleaving, pomset, partial word, history preserving } s'il existe une relation symétrique [1] $R \subseteq \delta(\mathcal{E}) \times \delta(\mathcal{F}) \times \mathcal{P}(E_{\mathcal{E}} \times E_{\mathcal{F}}) \cup \delta(\mathcal{F}) \times \delta(\mathcal{E}) \times \mathcal{P}(E_{\mathcal{F}} \times E_{\mathcal{E}})$ telle que:

[1] $((C, P), (D, Q), f) \in R$ implique $((D, Q), (C, P), f^{-1}) \in R$

i) $((\emptyset, \emptyset), (\emptyset, \emptyset), \emptyset) \in R$

ii) Si $((C, P), (D, Q), f) \in R$ alors

- $f : C \rightarrow D$ est une bijection qui preserve l'étiquetage et $f(P) = Q$,
- Pour toute transition $(C, P) \rightarrow (C', P')$ il existe D', f' tels que $(D, Q) \rightarrow (D', Q')$ avec $((C', P'), (D', Q'), f') \in R$ et $f' \lceil C = f$.
  - si α = interleaving, pas de condition sur f',
  - si α = partial word, $f'^{-1} : D' - Q \rightarrow C' - P$ est un homomorphisme,
  - si α = pomset, $f'^{-1} : D' - Q \rightarrow C' - P$ est un isomorphisme,
  - si α = history preserving, $f' : C' \rightarrow D'$ est un isomorphisme.

Notons que c'est la bijection f de C dans D telle $f(P) = Q$ qui assure les liens entre les actions actives.

Pour $\alpha \in$ {interleaving, pomset, partial word, history preserving }, on notera respectivement $\mathcal{E} \approx_{iST} \mathcal{F}$, $\mathcal{E} \approx_{pST} \mathcal{F}$, $\mathcal{E} \approx_{pwST} \mathcal{F}$ et $\mathcal{E} \approx_{hST} \mathcal{F}$ quand $\mathcal{E}$ et $\mathcal{F}$ sont α-ST-bisimilaires.

Il est évident que αST est une relation d'équivalence et que $\mathcal{E} \approx_{\alpha ST} \mathcal{F} \Rightarrow \mathcal{E} \approx_{\alpha} \mathcal{F}$ pour $\alpha \in$ {interleaving, pomset, partial word, history preserving}.

La définition 3.16 suggère clairement que :

$$\mathcal{E} \approx_{hST} \mathcal{F} \Rightarrow \mathcal{E} \approx_{pST} \mathcal{F} \Rightarrow \mathcal{E} \approx_{pwST} \mathcal{F} \Rightarrow \mathcal{E} \approx_{iST} \mathcal{F} \qquad (1)$$

L'interleaving ST-bisimulation n'implique pas la pomset bisimulation, les structures d'événements du deuxième exemple de la figure 3.3 sont iST-bisimilaires mais pas pomset bisimilaires.

De plus les implications de *(1)* sont strictes. La figure 3.12 contient deux structures d'événements qui sont iST mais pas partial word bisimilaires dons pas pwST-bisimilaires.

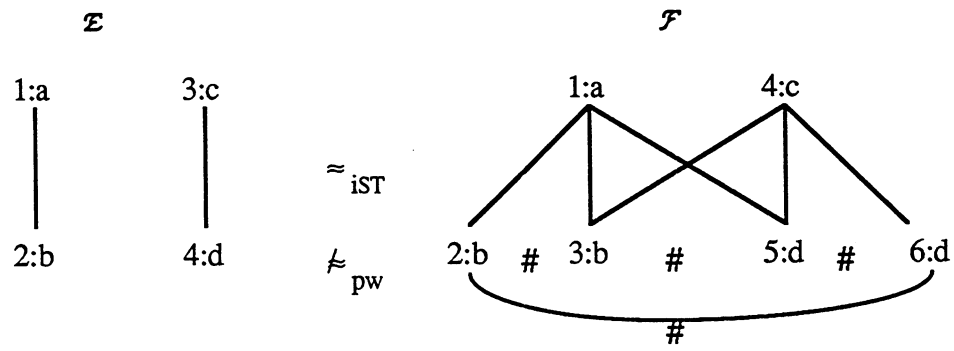


Figure 3.12

Les structures d'événements de la figure 3.5 sont pwST-bisimilaitres mais pas pomset bisimilaires donc pas pST-bisimilaires.
La figure 3.13 donne l'exemple de deux structures d'événements qui sont pomset ST-bisimilaires mais pas history preserving bisimilaires donc non hST-bisimilaires: en effet, la transition $\{\emptyset\} \rightarrow_{\mathcal{F}} \{1{:}a, 3{:}a\}$ est imitée dans $\mathcal{E}$ par $\{\emptyset\} \rightarrow_{\mathcal{E}} \{1{:}a, 2{:}a\}$. Maintenant si $\mathcal{E}$ eféctue l'événement 3:b, $\mathcal{F}$ ne peut l'imiter qu'en effctuant l'événement 5:b. Ainsi la transition $\{1{:}a, 2{:}a\} \rightarrow_{\mathcal{E}} \{1{:}a, 2{:}a, 3{:}b\}$ est imitée par $\{1{:}a, 3{:}a\} \rightarrow_{\mathcal{F}} \{1{:}a, 3{:}a, 5{:}b\}$. Il est clair que la configuration $\{1{:}a, 2{:}a, 3{:}b\}$ de $\mathcal{C}(\mathcal{E})$ n'est pas isomorphe à $\{1{:}a, 3{:}a, 5{:}b\}$ de $\mathcal{C}(\mathcal{F})$.

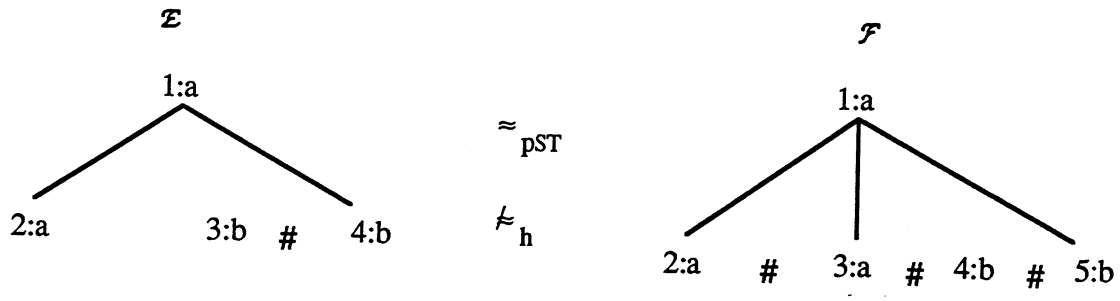


Figure 3.13

Une propriété importante des ST-bisimulations est qu'elles sont toutes compatibles avec le raffinement d'actions.

Proposition 3.4 [Vog 90]
Soient $\mathcal{E}$ et $\mathcal{F} \in \mathcal{E}_{prim.}$, raf un raffinement, $\alpha \in$ {interleaving, pomset, partial word, history preserving }

$$\mathcal{E} \approx_{\alpha ST} \mathcal{F} \Rightarrow raf(\mathcal{E}) \approx_{\alpha ST} raf(\mathcal{F})$$

Pour faire la preuve de cette proposition, nous avons besoin du lemme suivant qui relie les ST-configurations d'une structure d'événements $\mathcal{E}$ aux ST-configurations de raf($\mathcal{E}$).

Lemme 3.1 [Vog 90]
Soit $\mathcal{E}$ une structure d'événements, raf un raffinement,
i) $(\widetilde{C}, \widetilde{P}) \in \mathcal{P}(E_{raf(\mathcal{E})}) \times \mathcal{P}(E_{raf(\mathcal{E})})$ est une ST-configuration de raf($\mathcal{E}$) si et seulement si :
$\widetilde{C} = \{(e, e') \mid e \in C, e' \in C_e\}$, $\widetilde{P} = \{(e, e') \mid e \in C, e' \in P_e\}$ où :
Si P est le sous ensemble de C défini par $P_e = E_{raf(e)}$ si et seulement si $e \in P$ alors:

- (C, P) est une ST-configuration de $\mathcal{E}$,
- (C_e, P_e) est une ST-configuration de raf(e) pour $e \in C$.

ii) D'après i) $(\widetilde{C}, \widetilde{P}) \in \delta(raf(e))$ détermine de façon unique (C, P) et (C_e, P_e)

iii) D'après ii) $(\widetilde{C}, \widetilde{P})$ et $(\widetilde{C}', \widetilde{P}')$ déterminent (C, P), (C', P'), (C_e, P_e) et (C'_e, P'_e) alors

- $(\widetilde{C}, \widetilde{P}) \rightarrow_{raf(\mathcal{E})} (\widetilde{C}', \widetilde{P}')$ si et seulement si $(C, P) \rightarrow_{\mathcal{E}} (C', P')$ et $(C_e, P_e) \rightarrow (C'_e, P'_e)$ pour $e \in C$.

Preuve (proposition 3.4)

Soit R : $\mathcal{E} \approx_{\alpha ST} \mathcal{F}$, une αST bisimulation entre $\mathcal{E}$ et $\mathcal{F}$.

On définit $\widetilde{R} = \{((\widetilde{C}, \widetilde{P}), (\widetilde{D}, \widetilde{Q}), \tilde{f})) \in \delta(raf(\mathcal{E})) \times \delta(raf(\mathcal{F})) \times \mathcal{P}(E_{raf(\mathcal{E})} \times E_{raf(\mathcal{F})}) \mid \exists ((C,P), (D,Q), f) \in R$ et $\tilde{f} : \widetilde{C} \rightarrow \widetilde{D}$ est une bijection telle que $\tilde{f}(e, e') = (f(e), e')$ et $\tilde{f}(\widetilde{P}) = \widetilde{Q}\}$

Montrons que $\widetilde{R}$ est une αST-bisimulation entre $raf(\mathcal{E})$ et $raf(\mathcal{F})$.

i) $((\emptyset, \emptyset), (\emptyset, \emptyset), \emptyset) \in \widetilde{R}$ puisque $((\emptyset, \emptyset), (\emptyset, \emptyset), \emptyset) \in R$.

ii) Supposons $((\widetilde{C}, \widetilde{P}), (\widetilde{D}, \widetilde{Q}), \tilde{f})) \in \widetilde{R}$ donc il existe $((C,P), (D,Q), f) \in R$. Par construction $\tilde{f}$ est une bijection de $\widetilde{C}$ dans $\widetilde{D}$ telle que $\tilde{f}(e, e') = (f(e), e')$ et $\tilde{f}(\widetilde{P}) = \widetilde{Q}$.

Montrons d'abord que $\tilde{f}$ préserve l'étiquetage: $l_{raf(\mathcal{F})}(\tilde{f}(e, e') = l_{raf(\mathcal{F})}(f(e), e') = l_{raf(f(e))}(e') = l_{raf(e)}(e') = l_{raf\mathcal{E})}(e, e')$.

1) dans le cas α = history, il faut que $\tilde{f} : \widetilde{C} \rightarrow \widetilde{D}$ soit un isomorphisme:

$(d, d') <_{raf(\mathcal{E})} (e, e') \Leftrightarrow d, e \in C$ et $d <_{\mathcal{E}} e$ ou bien $d = e$ et $d' <_{raf(e)} e' \Leftrightarrow f(d) < f(e)$ ou bien $f(e) = f(d)$ et $d' <_{raf(f(e))} e'$, comme f est un isomorphisme de C dans D, nous avons $(f(d), d') <_{raf(\mathcal{F})} (f(e), e') \Leftrightarrow \tilde{f}(d, d') = \tilde{f}(e, e')$. Donc $\tilde{f}$ est bien un isomorphisme de $\widetilde{C}$ dans $\widetilde{D}$.

iii) Supposons $(\widetilde{C}, \widetilde{P}) \rightarrow_{raf(\mathcal{E})} (\widetilde{C}', \widetilde{P}')$ donc $\widetilde{C} \subseteq \widetilde{C}'$ et $\widetilde{P} \subseteq \widetilde{P}'$. D'après le lemme 3.1 il existe (C', P'), (C'_e, P'_e) des ST-configurations telles que $(C, P) \rightarrow_{\mathcal{E}} (C', P')$ et $(C_e, P_e) \rightarrow C'_e, P'_e)$. Comme R est une αST-bisimulation entre $\mathcal{E}$ et $\mathcal{F}$, il existe D', Q', f' avec les propriétés désirées. On définit $\widetilde{D}'$, $\widetilde{Q}'$ et $\tilde{f}'$ comme suit :

$\widetilde{D}' = \{(f'(e), e') \mid (e, e') \in \widetilde{C}'\}$

$\widetilde{Q}' = \{(f'(e), e') \mid (e, e') \in \widetilde{P}'\}$

$\tilde{f}' = \{((e, e'),(f'(e), e')) \mid (e, e') \in \widetilde{C}'\}$

Il est clair que $(\widetilde{D}', \widetilde{Q}')$ est une ST-configuration de $raf(\mathcal{F})$ car $e \in P'$ implique $f'(e) \in Q'$ par construction de R. Par construction de $\widetilde{R}$, $((\widetilde{C}', \widetilde{P}'), (\widetilde{D}', \widetilde{Q}'), \tilde{f}') \in \widetilde{R}$. Il reste à prouver que :

- $\tilde{f}' \lceil \widetilde{C} = \tilde{f}$. Pour $(e, e') \in \widetilde{C}$, $\tilde{f}'(e, e') = (f'(e), e')$ (par construction) $= (f(e), e')$ (car $f' \lceil C = f$) $= \tilde{f}(e, e')$.

• $(\tilde{D}, \tilde{Q}) \rightarrow_{raf(\mathcal{F})} (\tilde{D}', \tilde{Q}')$. Soit $(d, e') \in \tilde{D}$, donc $d \in D$, il existe donc $e \in C$ tel que $f(e) = d$. La transition $(C, P) \rightarrow_{\mathcal{E}} (C', P')$ implique $e \in C'$ et $(C_e, P_e) \rightarrow (C'_e, P'_e)$ implique que $e' \in C'_e$ donc $(d, e') = (f(e), e') = (f'(e), e') \in \tilde{D}'$, donc $\tilde{D} \subseteq \tilde{D}'$. De façon similaire on montre que $\tilde{Q} \subseteq \tilde{Q}'$

2) Cas où α = partial word et α = pomset

Rappelons que $\tilde{f}' \lceil \tilde{C} = \tilde{f}$ et que $\tilde{f}(\tilde{P}) = \tilde{Q}$, donc $\tilde{f}'(\tilde{C}' - \tilde{P}) = \tilde{D}' - \tilde{Q}$. Si $e \in P$, $P_e = E_{raf(e)}$, donc tous les éléments (e, e') de $\tilde{C}$ sont dans $\tilde{P}$. Donc si $(e, e') \in \tilde{C}' - \tilde{P}$, $e \in C'-P$. Ceci nous permet de vérifier facilement que $\tilde{f}'^{-1}$ est un homomorphisme de $\tilde{D}' - \tilde{Q}$ dans $\tilde{C}' - \tilde{P}$ dans le cas α = partial word, et $\tilde{f}'^{-1}$ est un isomorphisme dans le cas α = pomset. □

Ainsi, les ST-versions des diverses bisimulations évoquées, sont toutes compatibles avec le raffinement d'actions. De plus, Vogler [Vog 90] a montré que les αST-bisimulations pour $\alpha \in$ {interleaving, pomset, history preserving} sont les plus grandes congruences pour le raffinement d'actions qui respectent respectivement l'interleaving, la pomset et la history preserving bisimulation.

Vogler dans [Vog 90], a donné une autre caractérisation de la iST et de la h-ST bisimulation en utilisant la notion classique de configurations et ce grâce à la notion d'éléments maximaux.

Théorème 3.1 [Vog 90]

Deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ sont αST-bisimilaires pour $\alpha \in$ {interleaving, history preserving} si et seulement s'il existe une relation symétrique $R \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{F}) \times \mathcal{P}(E_{\mathcal{E}} \times E_{\mathcal{F}}) \cup \mathcal{C}(\mathcal{F}) \times \mathcal{C}(\mathcal{E}) \times \mathcal{P}(E_{\mathcal{F}} \times E_{\mathcal{E}})$ telle que

i) $(\emptyset, \emptyset, \emptyset) \in R$

ii) Si $(C, D, f) \in R$ alors

• $f : C \rightarrow D$ est une bijection qui preserve l'étiquetage, et c'est un isomorphisme pour l'ordre si α = history preserving.

• Pour toute transition $C \rightarrow C'$, il existe D', f' tels que:

a) $D \rightarrow D'$ avec $(C', D', f') \in R$ et $f' \lceil C = f$,

b) si $e \in C'$ et e est maximal dans C' alors $f'(e)$ est maximal dans D' ou bien $e \in C$ et $f(e)$ n'est pas maximal dans D.

Un corollaire intéressant de cet théorème est que la iST-bsimulation est plus fine que la step bisimulation et que la hST-bisimulation coincide avec la history preserving bisimulation.

Corollaire 3.1

i) $\mathcal{E} \approx_{iST} \mathcal{F} \Rightarrow \mathcal{E} \approx_{s} \mathcal{F}$

ii) $\mathcal{E} \approx_{hST} \mathcal{F} \Leftrightarrow \mathcal{E} \approx_{h} \mathcal{F}$

Preuve

i) Soit R : $\mathcal{E} \approx_{iST} \mathcal{F}$ ayant les propriétés enoncées dans le théorème 3.1. On montre que R est une step bisimulation.

- $(\emptyset, \emptyset) \in R$ puisque $(\emptyset, \emptyset, \emptyset) \in R$

- Si (C, D, f) $\in$ R, supposons $C \rightarrow_{z} C'$ telle que C'- C est un step (ensemble d'événements indépendants). Il existe D', f' tels que $D \rightarrow_{f} D'$, f' $\lceil C = f$ et (C', D', f') $\in$ R. Donc f'(C'-C) = D'-D. Les éléments de C'-C sont maximaux dans C', leurs images par f' sont maximales dans D', donc D'-D est un step. Comme f' est une bijection de C' dans D' qui preserve l'étiquetage, D' - D est donc le même step que C' - C .

ii) $\Rightarrow$: évident

$\Leftarrow$: soit R : $\mathcal{E} \approx_{h} \mathcal{F}$ une history preserving bisimulation entre $\mathcal{E}$ et $\mathcal{F}$. On vérifie facilement que R est une hST-bisimulation : il suffit de vérifier la propriété b) du théorème 1. Si (C, D, f) $\in$ R, f est un isomorphisme entre C et D. Les éléments maximaux de C ont des images maximales dans D. □

3.6 Conclusions

Dans ce chapitre, nous avons étudié les équivalences de temps arborescent sur les structures d'événements premières, modèle de base pour le vrai parallélisme. Nous avons montré que les ST-sémantiques sont des équivalences qui conviennent si on désire concevoir des systèmes parallèles à différents niveaux d'abstraction (raffiner). Cette notion de ST-sémantiques a été adapté aux équivalences de "failures" dans [AE 91].

Le but de ce chapitre étant de donner les bases nécessaires pour les résultats que nous développons dans le chapitre suivant, tout au long de notre étude, nous n'avons pas traité les situations où certaines actions sont invisibles ou silencieuses. Nous renvoyons le lecteur intéressé par le traitement de ce dernier cas à [Vog 90]. Néanmoins, nous remarquons qu'à l'exception du point ii) du corollaire 3.1 ($\mathcal{E} \approx_{hST} \mathcal{F} \Leftrightarrow \mathcal{E} \approx_{h} \mathcal{F}$), tous les résultats énoncés sont

valables dans le cas où l'on considère que les systèmes peuvent effectuer des actions silencieuses ou non observables .

Chapitre 4

Bisimulations avant arrière sur les structures d'événements premières

4.1 Introduction

Les structures d'événements premières [NPW 81]constituent un modèle de base pour le vrai parallèlisme. Le comportement des systèmes est décrit à l'aide d'ordre partiel entre événements. Cet ordre formalise la notion de causalité entre événements. Deux événements non ordonnés peuvent avoir lieu en parallèle. Dans le chapitre 3, nous avons étudié diverses équivalences de bisimulation sur les structures d'événements.

Récemment, De Nicola, Montanari, et Vaandrager [DNMV 90] ont introduit un nouveau critère de comparaison de processus. Les auteurs proposent une équivalence, appelée bisimulation avant arrière. Dans cette équivalence, les processus se simulent non seulement en faisant des pas en avant (transitions classiques) mais aussi en faisant des retours en arrière le long de leurs histoires respectives. Les auteurs ont travaillé sur les modèles séquentiels (les graphes), nous avons rappelé quelques uns de leurs résultats dans le chapitre 2.

Dans la section 2 de ce chapitre, nous définissons la bisimulation avant arrière sur les structures d'événements. Nous généralisons, de façon uniforme, cette notion à la step-, partial word- et pomset- bisimulation. Nous comparons les nouvelles équivalences aux bisimulations classiques ainsi qu'aux ST-bisimulations. Il s'avère essentiellement, que la partial word et la pomset bisimulation avant arrière coïncident avec la history preserving bisimulation [Che 92a], [Che92b].

Par ailleurs, il est reconnu que la logique temporelle constitue un formalisme adéquat pour exprimer les propriétés des systèmes parallèles ou réactifs. Aussi il existe dans le domaine la sémantique du parallèlisme plusieurs travaux reliant une équivalence comportementale de processus à une logique temporelle. Le résultat le plus connu étant celui des travaux de Hennessy et Milner [HM 85] où ils ont formulé la notion d'adéquation d'une logique par rapport à une équivalence de processus : la bisimulation distingue exactement ce que la logique HML distingue.
Dans le même esprit que HML, des logiques temporelles ont été proposées sur les modèles de vrai parallélisme comme par exemple dans [KP 87], [LT 87], [LRT 89], [Pen 88], [Rei 89]. Cependant, à ce jour, il n'existe pas à notre connaissance un résultat d'adéquation pour la history preserving bisimulation. Par conséquent, dans la section 4 de ce chapitre, nous proposons une caractérisation logique de la history preserving bisimulation. La logique l_{bf}, que nous proposons est une variante de HML avec un opérateur du passé et où l'entité observable est généralisée au pomset. Notons que cette logique est très naturelle, elle découle directement de la nouvelle caractérisation de la history preserving bisimulation par la pomset bisimulation avant arrière [CLP 92].

Dans [DNF 90], les auteurs proposent une logique l_p, qui selon eux, caractérisent la weak history preserving bisimulation. Nous comparons les logiques l_p et l_{bf} et nous montrons qu'elles ont exactement le même pouvoir de distinction. Par conséquent l_p caractérise elle aussi la history preserving bisimulation [CLP 92].

4.2 Bisimulations avant arrière sur les structures d'événements premières

Cette section est consacrée à la définition des bisimulations avant arrière sur les structures d'événements premières. Cette notion a été introduite pour la première fois sur les modèles de graphes dans [DNMV 90]. Dans les bisimulations avant arrière, les processus se simulent non seulement en faisant des pas en avant comme c'est le cas de la bisimulation classique, mais

aussi en faisant des pas en arrière. Il est important de souligner que les retours en arrière ne se font pas de façon arbitraire. A partir d'un état, il est possible de faire un retour en arrière uniquement le long du chemin qui a amené le processus dans cet état. Plus exactement le retour se fait le long d'une *histoire*. Une *histoire* est une séquence d'événements qui respectent les relations de causalité entre les événements ainsi que l'ordre exact dans lequel ils ont eu lieu.

4.2.1 Notions de bases

Avant de définir les bisimulations avant arrière, nous présentons quelques notions de base utiles.

Définition 4.1 (histoire)
Soit $\mathcal{E} = (E, \leq, \#, l)$ une structure d'événements première, une séquence d'événements $\sigma = e_1.e_2...e_n$ est appelée histoire de $\mathcal{E}$ si $\forall\ i = 1,...,n$ nous avons $\{e_1, e_2,...,e_{i-1}\} \rightarrow_{\mathcal{E}} \{e_1, e_2,...,e_{i-1}, e_i\}$

Nous écrirons $\Pi(\mathcal{E}) = \{\sigma, \rho, ...\}$ l'ensemble de toutes les histoires de $\mathcal{E}$.
Le comportement d'un système parallèle est décrit par un système de transitions dont les états sont des histoires.

Définition 4.2
Pour toutes histoires σ, σ' de $\Pi(\mathcal{E})$, θ une séquence d'événements
i) $\sigma \xrightarrow{\theta} \sigma'$ si et seulement si $\sigma' = \sigma.\theta$ où $\sigma.\theta$ est la concaténation des séquences σ et θ. σ est alors un préfixe de σ'.
ii) Si $\sigma = e_1.e_2...e_n$, trace(σ) = $l(e_1).l(e_2)...l(e_n)$, longueur(σ) = $|\sigma| = n$, λ est la séquence vide.

Pour définir nos équivalences sémantiques, il est important de connaitre les relations de causalité qui existent entre les événements d'une histoire ou d'une séquence d'événements quelconque.

Définition 4.3
Pour toute structure d'événements $\mathcal{E} = (E, \leq, \#, l)$
i) Si $\sigma = e_1.e_2...e_n$, $C_\sigma = \{e_1, e_2, ..., e_n\}$ est la configuration associée à σ.
ii) Si σ est une séquence d'événements sans conflit, pomset (σ) est la classe d'isomorphisme de la structure d'événements $\mathcal{E}$ restreinte aux événements de σ.
iii) pomset(σ) est un step si les événements de σ sont deux à deux indépendants.

4.2.2 Bisimulations avant arrière

Définition 4.4 (bisimulation avant arrière)
Deux structures d'événements $\mathcal{E}$ =(E, ≤, #, l) et $\mathcal{F}$= (F, ≤, #, l') sont interleaving bisimilaires "avant arrière" s'il existe une relation symétrique $R \subseteq \Pi(E) \times \Pi(\mathcal{F}) \cup \Pi(\mathcal{F}) \times \Pi(\mathcal{E})$, (dite interleaving bisimulation avant arrière) telle que :
i) $(\lambda, \lambda) \in R$

ii) Pour tout $(\sigma, \rho) \in R$ et toute transition $\sigma \xrightarrow{\theta} \sigma'$ telle que $|\theta| = 1$, il existe θ', ρ' tels que $\rho \xrightarrow{\theta'} \rho'$, $l(\theta)=l'(\theta')$ et $(\sigma', \rho') \in R$.

iii) Pour tout $(\sigma, \rho) \in R$ et toute transition $\sigma' \xrightarrow{\theta} \sigma$ telle que $|\theta| = 1$, il existe θ', ρ' tels que $\rho' \xrightarrow{\theta'} \rho$, $l(\theta)=l'(\theta')$ et $(\sigma', \rho') \in R$.

On notera $\mathcal{E} \approx_{ibf} \mathcal{F}$ (bf pour back and forth) quand $\mathcal{E}$ et $\mathcal{F}$ sont deux structures d'événements interleaving bisimilaires avant arrière.

Notons que la relation de bisimulation avant arrière porte sur les histoires. Dans [GKP 92], les auteurs proposent une définition de bisimulation avant arrière sur les configurations.

Exemple La figure 4.1 donne un exemple de deux structures d'événements qui se bisimulent en avant et en arrière ($\approx_{ibf}$).

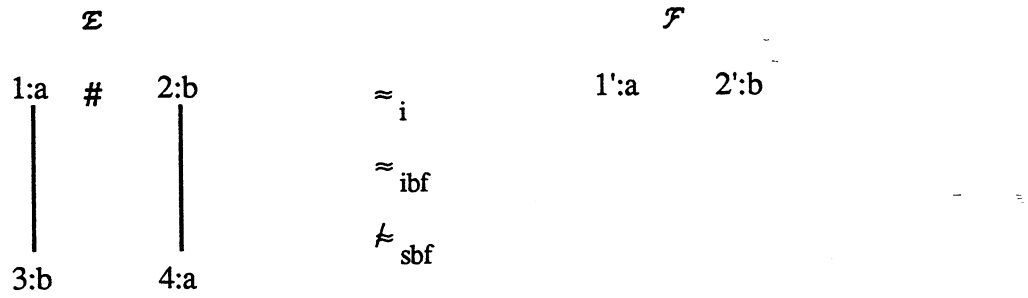


Figure 4.1

Définition 4.5 (step bisimulation avant arrière)

Deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ sont step bisimilaires "avant arrière" s'il existe une relation symétrique $R \subseteq \Pi(E) \times \Pi(\mathcal{F}) \cup \Pi(\mathcal{F}) \times \Pi(\mathcal{E})$, (dite step bisimulation avant arrière) telle que :

i) $(\lambda, \lambda) \in R$

ii) Pour tout $(\sigma, \rho) \in R$ et toute transition $\sigma \xrightarrow{\theta} \sigma'$ où pomset(θ) est un step, il existe θ', ρ' tels que $\rho \xrightarrow{\theta'} \rho'$, pomset($\theta$) = pomset($\theta'$) et $(\sigma', \rho') \in R$.

iii) Pour tout $(\sigma, \rho) \in R$ et toute transition $\sigma' \xrightarrow{\theta} \sigma$ où pomset(θ) est un step, il existe θ', ρ' tels que $\rho' \xrightarrow{\theta'} \rho$, pomset($\theta$) = pomset($\theta'$) et $(\sigma', \rho') \in R$.

On notera $\mathcal{E} \approx_{sbf} \mathcal{F}$ quand $\mathcal{E}$ et $\mathcal{F}$ sont step bisimilaires avant arrière.

Exemple La figure 4.2 donne un exemple de deux structures d'événements qui se bisimulent en avant en effectuant les mêmes step et qui se bisimulent en arrière en faisant des retours arrière via les mêmes step le long de leurs histoires respectives.

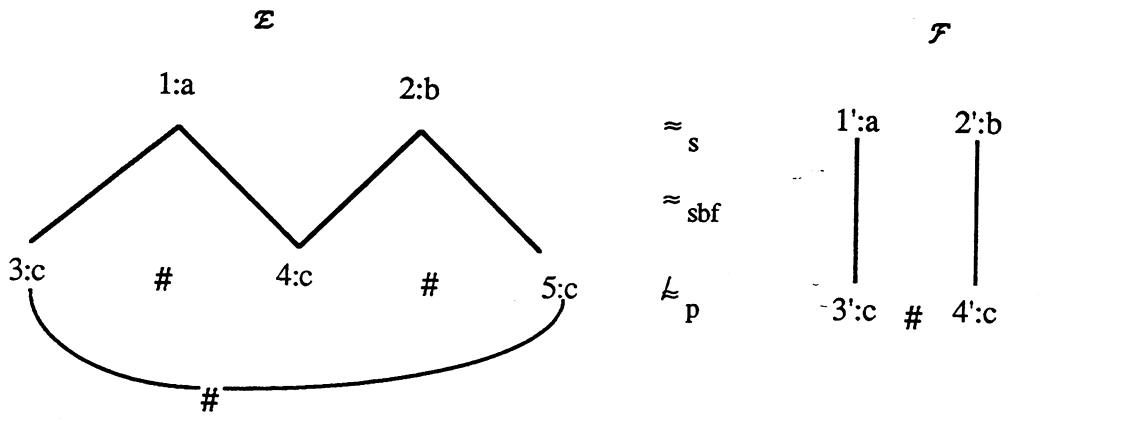


Figure 4.2

La définition 4.5 est généralisée en considérant des transitions par des pomsets. Ainsi la partial word et la pomset bisimulation avant arrière sont définies comme suit :

Définition 4.6 (partial word et pomset bisimulation avant arrière)
Deux structures d'événements $\mathcal{E}$ et $\mathcal{F}$ sont α-bisimilaires "avant arrière" pour $\alpha \in$ {partial word, pomset} s'il existe une relation symétrique $R \subseteq \Pi(E) \times \Pi(\mathcal{F}) \cup \Pi(\mathcal{F}) \times \Pi(\mathcal{E})$, dite α-bisimulation avant arrière telle que :

i) $(\lambda, \lambda) \in R$

ii) Pour tout $(\sigma, \rho) \in R$ et toute transition $\sigma \xrightarrow{\theta} \sigma'$, il existe ρ', θ' tels que $\rho \xrightarrow{\theta'} \rho'$, $(\sigma', \rho') \in R$, pomset$(\theta') \sqsubseteq$ pomset (θ) si α = partial word et pomset (θ') = pomset (θ) sinon.

iii) Pour tout $(\sigma, \rho) \in R$ et toute transition $\sigma' \xrightarrow{\theta} \sigma$, il existe ρ', θ' tels que $\rho' \xrightarrow{\theta'} \rho$, $(\sigma', \rho') \in R$, pomset$(\theta') \sqsubseteq$ pomset (θ) si α = partial word et pomset (θ') = pomset (θ) sinon.

On notera $\mathcal{E} \approx_{pwbf} \mathcal{F}$ (respectivement $\mathcal{E} \approx_{pbf} \mathcal{F}$) quand $\mathcal{E}$ et $\mathcal{F}$ sont partial word (respectivement pomset) bisimilaires avant arrière.

Exemple Les structures d'événements modélisant les agents CCS $a|(b+c) + a|b + b|(a+c)$ et $a|(b+c) + b|(a+c)$ sont pomset bisimilaires avant arrière.

Il est facile de constater que toute bisimulation avant arrière est plus fine que sa version avant. De plus la pomset avant arrière est plus fine que la step avant arrière, qui elle même est plus fine que l'interleaving avant arrière.
La proposition suivante stipule également que la partial word avant arrière coïncide avec la pomset bisimulation avant arrière [Sch 92].

Proposition 4.1
Soient $\mathcal{E}$ et $\mathcal{F}$ deux structures d'événements :
i) $\mathcal{E} \approx_{\alpha bf} \mathcal{F} \Rightarrow \mathcal{E} \approx_{\alpha} \mathcal{F}$ pour $\alpha \in$ {interleaving, step, partial word, pomset }
ii) $\mathcal{E} \approx_{pwbf} \mathcal{F} \Leftrightarrow \mathcal{E} \approx_{pbf} \mathcal{F}$
iii) $\mathcal{E} \approx_{pbf} \mathcal{F} \Rightarrow \mathcal{E} \approx_{sbf} \mathcal{F} \Rightarrow \mathcal{E} \approx_{ibf} \mathcal{F}$.

Preuve
i) et iii) sont des conséquences directes des définitions 4.4, 4.5 et 4.6.
ii) nous reproduisons la preuve proposée par Ph. Schnoebelen [Sch 92]
1)"$\Leftarrow$" is triviale
2) "$\Rightarrow$" Supposons $R : \mathcal{E} \approx_{pwbf} \mathcal{F}$, montrons que R est une pomset bisimulation avant arrière (pbf).

- Clairement $(\lambda, \lambda) \in R$
- Supposons $(\sigma, \rho) \in R$ et $\sigma \xrightarrow{\theta} \sigma'$, alors il existe ρ' et θ' tels que $\rho \xrightarrow{\theta'} \rho'$, pomset$(\theta') \sqsubseteq$ pomset(θ) et $(\sigma', \rho') \in R$. La propriété de transfert en avant de la pbf-bisimulation est vérifiée si on montre que pomset (θ) = pomset (θ'). Or $(\sigma', \rho') \in R$ implique que si $\rho \xrightarrow{\theta'} \rho'$

est une transition en arrière valide depuis ρ', $\sigma \xrightarrow{\theta} \sigma'$ est aussi un retour en arrière valide depuis σ' avec pomset(θ) $\sqsubseteq$ pomset(θ'), donc pomset (θ) = pomset (θ').

• Maintenant, si $\sigma' \xrightarrow{\theta} \sigma$, $(\sigma, \rho) \in R$ implique qu'il existe ρ', θ' tels que $\rho' \xrightarrow{\theta'} \rho$ avec pomset (θ') $\sqsubseteq$ pomset(θ) et $(\sigma', \rho') \in R$. Or $(\sigma', \rho') \in R$ et $\rho' \xrightarrow{\theta'} \rho$ impliquent qu'il existe θ" et σ" tels que $\sigma' \xrightarrow{\theta''} \sigma''$, $(\sigma'', \rho) \in R$ et pomset(θ") $\sqsubseteq$ pomset(θ').

$(\sigma'', \rho) \in R$ implique pomset(θ') $\sqsubseteq$ pomset(θ"), $(\sigma, \rho) \in R$ implique pomset(θ) $\sqsubseteq$ pomset(θ') donc pomset(θ) $\sqsubseteq$ pomset(θ") , donc pomset(θ) = pomset(θ') □

L'implication i) de la proposition 4.1 est stricte dans le cas où $\alpha \in$ {step, partial word, pomset}. En effet, les structures d'événements de la figure 4.3 sont step bisimilaires mais pas step bisimilaires en avant et en arrière (sbf). La transition $\lambda \rightarrow$ 1':a.2':b n'est pas possible dans $\mathcal{F}$ en tant que step. Cependant $\mathcal{F}$ peut effectuer $\lambda \xrightarrow{1'} 1':a \xrightarrow{2'} 1':a.2':b$. Ce comportement est imité dans $\mathcal{E}$ par les transitions $\lambda \xrightarrow{1} 1:a \xrightarrow{2} 1:a.2:b$. σ= 1:a.2:b est un step dans $\mathcal{E}$. Ainsi $\mathcal{E}$ peut faire un retour en arrière via le step σ. Mais ce comportement n'est pas possible dans $\mathcal{F}$.
Les structures d'événements de la figure 4.4 sont pomset (partial word) bisimilaires mais pas pomset avant arrière bisimilaires. La transition $\lambda \xrightarrow{1'3'}$ 1':a.3':a dans $\mathcal{F}$ est imitée par $\lambda \xrightarrow{1\,2}$ 1:a.2:a dans $\mathcal{E}$. Puis si $\mathcal{E}$ effectue l'événement 3, nous avons $\lambda \xrightarrow{1\,2} 1:a.2:a \xrightarrow{3} 1:a.2:a.3:b$. Pour imiter $\mathcal{E}$, $\mathcal{F}$ peut seulement effectuer l'événement 5 c.à.d. : $\lambda \xrightarrow{1'3'} 1:a.3:a \xrightarrow{5'} 1':a.3':a.5':b$. $\mathcal{E}$ et $\mathcal{F}$ ont effectue les mêmes pomsets en avant, à savoir a—a (a suivi de a) et b. $\mathcal{E}$ et $\mathcal{F}$ ont effectué respectivement les séquences σ = 1:a.2:a.3:b et ρ = 1':a.3':a.5':b. Aussi $\lambda \xrightarrow{\sigma} \sigma$ et $\lambda \xrightarrow{\rho} \rho$ sont respectivement des transitions valides dans $\mathcal{E}$ et $\mathcal{F}$. Comme pomset(σ)≠pomset(ρ), σ et ρ ne peuvent effectuer des retours arrière dans leurs histoires respectives via les mêmes pomsets.

Les implications de iii) sont strictes; En effet, les structures d'événements de la figure 4.1 sont interleaving avant arrière bisimilaires mais pas sbf bisimilaires puisqu'elles ne sont même pas step bisimilaires (seul $\mathcal{F}$ peut effectuer le step a b).
Les structures d'événements de la figure 4.2 qui sont step bisimilaires en avant et en arrière, ne sont pas pbf bisimilaires (puisque même pas pomset bisimilaires).

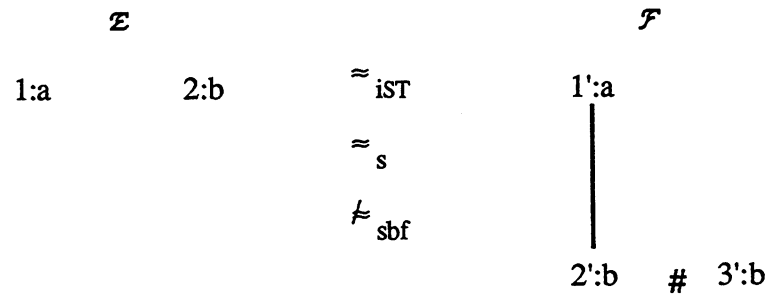


Figure 4.3

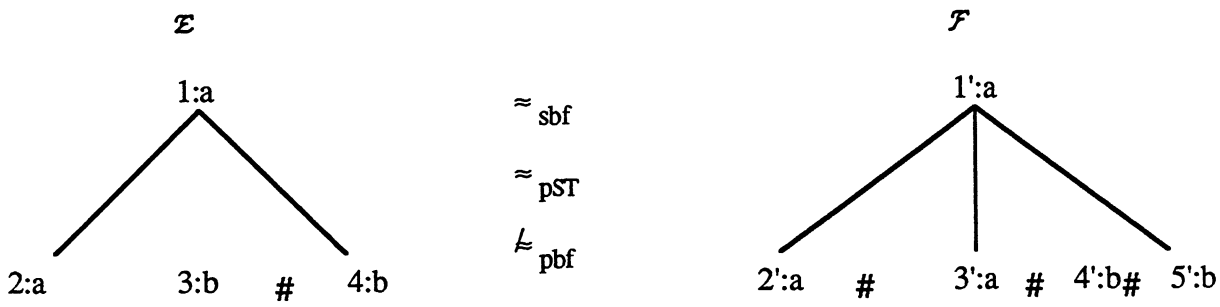


Figure 4.4

4.2.3 Comparaison avec les bisimulations classiques

Dans cette section, nous comparons les équivalences avant arrière aux bisimulations classiques.

Dans le cas où les processus effectuent au plus une action à chaque pas de calcul (interleaving bisimulation), le retour en arrière le long des histoires n'augmente pas le pouvoir de distinction : la version avant arrière de l'interleaving bisimulation coincide avec l'interleaving bisimulation classique.

Proposition 4.2
Soient $\mathcal{E}$ et $\mathcal{F}$ deux structures d'événements
$\mathcal{E} \approx_{ibf} \mathcal{F} \Leftrightarrow \mathcal{E} \approx_{i} \mathcal{F}$

Preuve
Similaire à celle de la proposition 2.5 du chapitre 2. Rappelons que cette proposition énonce le même résultat dans le cas des modèles de graphes. □

Proposition 4.3

i) $\approx_{sbf}$ et $\approx_{pw}$ ne sont pas comparables

ii) $\approx_{sbf}$ et $\approx_{p}$ ne sont pas comparables.

Preuve

i) Les exemples de la figure 4.5 sont partial word bisimilaires mais pas sbf bisimilaires : quand $\mathcal{F}$ effectue l'événement 1', $\mathcal{E}$ l'imite en effectuant l'événement 1. A ce niveau si $\mathcal{F}$ effectue l'événement 2', $\mathcal{E}$ effectue 2. Ainsi la transition $\lambda \xrightarrow{1'2'} \rho = 1':b.2':a$ de $\mathcal{F}$ est imitée par $\lambda \xrightarrow{2\,1} \sigma = 2:b.1:a$ dans $\mathcal{E}$. $\mathcal{E}$ peut faire un retour en arrière via le step $\sigma = 2:b.1:a$ alors que pomset(ρ) n'est pas un step dans $\mathcal{F}$. Donc $\mathcal{E}$ et $\mathcal{F}$ ne sont pas step bisimilaires avant arrière.
La figure 4.6 donne un exemple de deux structures d'évènements sbf bisimilaires mais pas partial word bisimilaires. En effet, supposons que $\mathcal{E}$ effectue les transitions $\lambda \xrightarrow{1\,3} 1:a.3:c \xrightarrow{2\,4} 1:a.3:c.2:b.4:c$, $\mathcal{F}$ l'imite en effectuant $\lambda \xrightarrow{1'3'} 1':a.3':c \xrightarrow{2'5'} 1':a.3':c.2':b.5':c$. Ainsi, $\lambda \xrightarrow{\sigma} \sigma = 1:a.3:c.2:b.4:c$ et $\lambda \xrightarrow{\rho} \rho = 1':a.3':c.2':b.5':c$ sont des transitions valides dans $\mathcal{E}$ et $\mathcal{F}$ respectivement. Si p et q désignent respectivement pomset (σ) et pomset (ρ), nous avons $\{\emptyset\} \xrightarrow{p} C_\sigma$ est imitée par $\{\emptyset\} \xrightarrow{q} C_\rho$ avec q qui n'est pas moins séquentiel que p puisque dans q l'évènement 1' est une cause de 5'. Une situation similaire se présente dans le cas où $\mathcal{F}$ imite $\mathcal{E}$ en effectuant ρ = 1':a.4':c.2':b.6':c.

ii) est une conséquence du point i). Les structures d'événements de la figure 4.6 qui sont sbf bisimilaires, ne sont pas pomset bisimilaires : seul $\mathcal{F}$ peut effectuer un pomset où a et b sont toutes les deux des actions prérequises pour l'action c. □

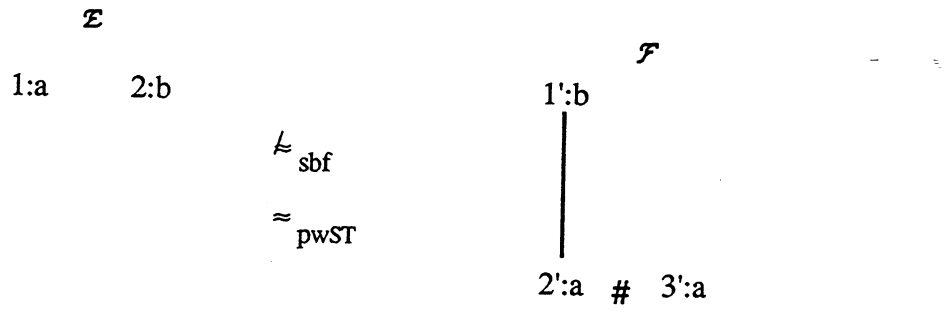


Figure 4.5

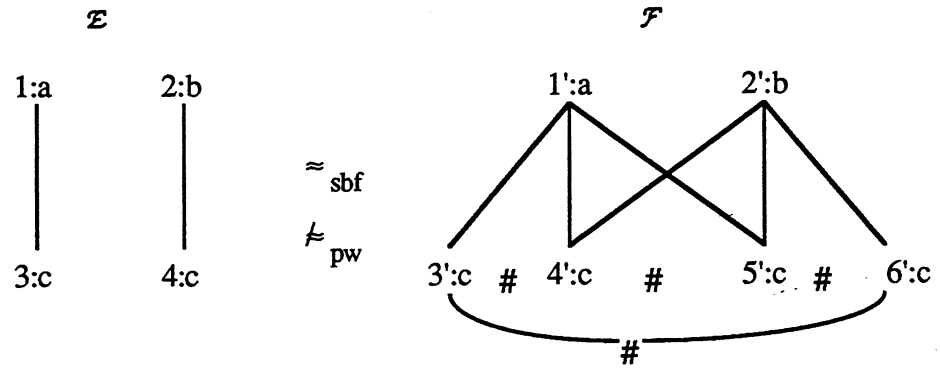


Figure 4.6

Un résultat intéressant est que la version avant arrière de la pomset bisimulation est plus forte que la pomset bisimulation classique. Plus précisément, elle coïncide avec la history preserving bisimulation. Ainsi, nous obtenons une nouvelle caractérisation de la history preserving bisimulation.

Proposition 4.4

Soient $\mathcal{E}$ et $\mathcal{F}$ deux structures d'événements

$\mathcal{E} \approx_{pbf} \mathcal{F} \Leftrightarrow \mathcal{E} \approx_{h} \mathcal{F}$

Preuve

Soient $\mathcal{E}$ et $\mathcal{F}$ deux structures d'événements. Pour une histoire $\sigma \in \Pi(\mathcal{E})$ telle que $|\sigma|=n$, nous écrivons C_σ la configuration $\{\sigma(1),...,\sigma(n)\}$, $C_{\sigma i}$ est la configuration associée au préfixe $\sigma^i = \sigma(1).\sigma(2)...\sigma(i)$ de σ. D_ρ et $D_{\rho i}$ désignent les notions correspondantes quand $\rho \in \Pi(\mathcal{F})$.

Pour $\sigma \in \Pi(\mathcal{E})$, $\rho \in \Pi(\mathcal{F})$ telles que $\text{trace}(\sigma) = \text{trace}(\rho)$, f^ρ_σ dénote la bijection $f : C_\sigma \rightarrow D_\rho$ telle que $f(\sigma(i)) = \rho(i)$ pour $i = 1...n$

1) "$\Rightarrow$" Supposons $R: \mathcal{E} \approx_{pbf} \mathcal{F}$. On définit $R' \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{F}) \times \mathcal{P}(E_\mathcal{E} \times E_\mathcal{F}) \cup \mathcal{C}(\mathcal{F}) \times \mathcal{C}(\mathcal{E}) \times \mathcal{P}(E_\mathcal{F} \times E_\mathcal{E})$, comme la plus petite relation contenant $(C_\sigma, D_\rho, f^\rho_\sigma)$ pou tout $(\sigma,\rho) \in R$ tel que f^ρ_σ est un isomorphisme entre pomset(σ) et pomset (ρ). Montrons que $R' : \mathcal{E} \approx_h \mathcal{F}$

- Il est évident que $(\emptyset,\emptyset,\emptyset) \in R'$puisqu'il correspond à $(C_\lambda, D_\lambda, f^\lambda_\lambda)$ et que $(\lambda, \lambda) \in R$

Vérifions la propriété de transfert:

•Supposons $(C_\sigma, D_\rho, f^\rho_\sigma) \in R'$. Sans perte de généralité, nous pouvons supposer que $C_\sigma \in \mathcal{C}(\mathcal{E})$. Pour vérifier la propriété de transfert de la history preserving bisimulation, il suffit de considérer les transitions par un seul événement. Si $C_\sigma \rightarrow_{\mathcal{E}} C_{\sigma'.e}$, il s'ensuit que $\sigma \xrightarrow{e} \sigma'=\sigma.e$, la pomset bisimilarité avant arrière implique qu'il existe $\rho \xrightarrow{e'} \rho'=\rho.e'$ avec $(\sigma',\rho') \in R$. Nous avons donc $D_\rho \rightarrow_{\mathcal{F}} D_{\rho'}$. Il reste à vérifier que $f^{\rho'}_{\sigma'}$ est un isomorphisme (évidemment, $f^{\rho'}_{\sigma'}$ étend f^ρ_σ et préserve l'étiquetage). Supposons que $f^{\rho'}_{\sigma'}$ n'est pas un isomorphisme, alors soit j le plus grand indice tel que $\sigma(j) <_{\mathcal{E}} e$ et $\rho(j) <_{\mathcal{F}} e'$(ou vice versa), nous avons donc $\text{pomset}(\sigma(j)...\sigma(n).e) \neq \text{pomset}(\rho(j)...\rho(n).e')$ puisque $\text{pomset}(\sigma(j+1)...\sigma(n).n) = \text{pomset}(\rho(j+1)...\rho(n).e')$, il s'ensuit que le retour en arrière $\sigma(1)...\sigma(j-1) \xrightarrow{\theta} \sigma' = \sigma(1)...\sigma(n).e$ dans $\mathcal{E}$ ne peut être imité par $\rho(1)...\rho(j-1) \xrightarrow{\theta'} \rho' = \rho(1)...\rho(n).e'$ dans $\mathcal{F}$ avec $\text{pomset}(\theta) = \text{pomset}(\theta')$. Ce qui contredit le fait que $(\sigma',\rho') \in R$. Ainsi $f^{\rho'}_{\sigma'}$ est nécessairement un isomorphisme.

2) "⇐" Supposons $R: \mathcal{E} \approx_h \mathcal{F}$. On définit $R' \subseteq \Pi(\mathcal{E}) \times \Pi(\mathcal{F}) \cup \Pi(\mathcal{F}) \times \Pi(\mathcal{E})$ par $(\sigma,\rho) \in R'$ssi

- $|\sigma| = |\rho|$
- $\forall\ i = 0,..., |\sigma|\ (C_{\sigma i}, D_{\rho i}, f^{\rho i}_{\sigma i}) \in R$

Montrons que R': $\mathcal{E} \approx_{pbf} \mathcal{F}$

- R' est symétrique puisque R est symétrique.
- $(\lambda,\lambda) \in R'$ puisque $(C_\lambda, D_\lambda, f^\lambda_\lambda)$ est $(\emptyset,\emptyset, \emptyset)$ et est dans R.

- Supposons $(\sigma,\rho) \in R'$ and $\sigma \xrightarrow{\theta} \sigma'$. Pour vérifier la propriété de transfert, nous considérons d'abord le cas où $|\theta| = 1$.

  a) Si $\sigma \xrightarrow{e} \sigma'$ alors $C_\sigma \rightarrow_{\mathcal{E}} C_{\sigma'}$ et comme $(\sigma,\rho) \in R'$ nous avons $(C_\sigma, D_\rho, f^\rho_\sigma) \in R$ et la history preserving bisimilarité implique qu'il existe D', f' tels que $D_\rho \rightarrow_{\mathcal{F}} D'$ with $(C_{\sigma'}, D', f') \in R$ pour une bijection f' qui étend f^ρ_σ . D' est nécesairement un $D_{\rho.e'}$ avec $\rho \xrightarrow{e'} \rho'$ une transition valide et f'(e)=e' alors $(\sigma.e,\rho.e') \in R'$.

b) Si $\sigma \xrightarrow{\theta} \sigma'$ tel que $\theta = e_1.e_2...e_n$, alors $C_\sigma \rightarrow_{\mathcal{E}} C_{\sigma.e1} \rightarrow_{\mathcal{E}} ... \rightarrow_{\mathcal{E}} C_{\sigma.e1...en}$ est imitée par $D_\rho \rightarrow_{\mathcal{F}} D_{\rho.e'1} ... \rightarrow_{\mathcal{F}} D_{\rho.e'1...e'n}$ avec $(\sigma.e_1...e_k , \rho.e'_1....e'_k) \in R'$ pour k = 1...n comme ceci a été prouvé dans a). Il reste à prouver que pomset($e_1...e_k$) = pomset($e'_1...e'_k$). La h-bisimilarité implique qu'il existe un isomorphisme $C_{\sigma'}$ -C_σ and $D_{\rho'}$ - D_ρ .

• Si $\sigma' \xrightarrow{\theta} \sigma$ est une transition en arrière, il existe $\rho' \xrightarrow{\theta'} \rho$ avec $|\theta| = |\theta'|$ et $(\sigma', \rho') \in R'$ (par définition deR'). Puis nous vérifions facilement que pomset (θ)= pomset (θ'): puisque $(C_\sigma, D_\rho, f^\rho_\sigma) \in R$ et f^ρ_σ est un isomorphisme de C_σ dans D_ρ tel que $f^\rho_\sigma (C_\sigma - C_{\sigma'}) = D_\rho - D_{\rho'}$.

□

Notons que le résultat de la proposition 4.4 se généralise facilement aux structures d'événements stables. On montre ceci en utilisant l'identité entre la history preserving bisimulation et l'équivalence causale sur les structures d'événements stables [Ace 91].

4.2.4 Comparaison aves les ST-bisimulations

Dans toutes les équivalences évoquées ci dessus, les actions que peuvent effectuer les processus sont considérées atomiques. Rappelons que le concept d'atomicité peut être formalisé par la notion de raffinement d'actions. Le raffinement consiste à remplacer une action par un comportement quelconque. Nous avons vu au chapitre 3 que les ST-sémantiques sont les équivalences les mieux adaptées quand on désire concevoir des systèmes parallèles à différents niveaux d'abstraction.

La section qui suit est consacrée à la comparaison entre les bisimulations avant arrière et les ST-bisimulations.
Avant de comparer les nouvelles équivalences aux ST-bisimulations, il est intéressant de noter que si nous définissions la step ST-bisimulation, elle coïnciderait avec la iST bisimulation.

Proposition 4.5

$\approx_{iST}$ et $\approx_{sbf}$ ne sont pas comparables.

Preuve

Les exemples de la figure 4.3 sont iST bisimilaires mais pas sbf bisimilaires.
La figure 4.7 contient un exemple de deux structures d'événements sbf équivalentes mais pas iST équivalentes. En effet supposons que $\mathcal{E}$ et $\mathcal{F}$ sont iST-bisimilaires par une relation R. Ainsi

$((\emptyset, \emptyset), (\emptyset, \emptyset), \emptyset) \in$ R. Quand $\mathcal{E}$ effectue l'événement 1 : $(\emptyset, \emptyset) \rightarrow_{\mathcal{E}}(\{1{:}a\}, \emptyset)$, $\mathcal{F}$ l'imite en effectuant $(\emptyset, \emptyset) \rightarrow_{\mathcal{F}}(\{1'{:}a\}, \emptyset)$. Quand $\mathcal{E}$ effectue l'événement 2, la transition $(\{1{:}a\}, \emptyset) \rightarrow_{\mathcal{E}}(\{1{:}a, 2{:}b\}, \emptyset)$ est imitée dans $\mathcal{F}$ par $(\{1'{:}a\}, \emptyset) \rightarrow_{\mathcal{F}}(\{1'{:}a, 2'{:}b\}, \emptyset)$. $\mathcal{F}$ a la possibilité

d'effectuer c aussitôt que b est terminée. En effet, $(\{1'{:}a, 2'{:}b\}, \emptyset) \rightarrow_{\mathcal{F}}(\{1'{:}a, 2'{:}b\}, \{2'{:}b\})$ $\rightarrow_{\mathcal{F}}(\{1'{:}a, 2'{:}b, 3'{:}c\}, \{2'{:}b\})$ est une suite de transitions valides dans $\mathcal{F}$. La dernière transition de $\mathcal{F}$ ne peut être imitée dans $\mathcal{E}$, puisque dans $\mathcal{E}$ a et b sont tous les deux des causes de c. Donc $(\{1{:}a, 2{:}b\}, \emptyset)$ et $(\{1'{:}a, 2'{:}b\}, \emptyset)$ ne sont pas reliés par R. Notons que le symbole + est utilisé pour exprimer le comportement non déterministe d'un système. Ainsi $\mathcal{E} + \mathcal{F}$ signifie que tous les événements de $\mathcal{E}$ sont en conflit avec tous les évènements de $\mathcal{F}$ et vice versa. □

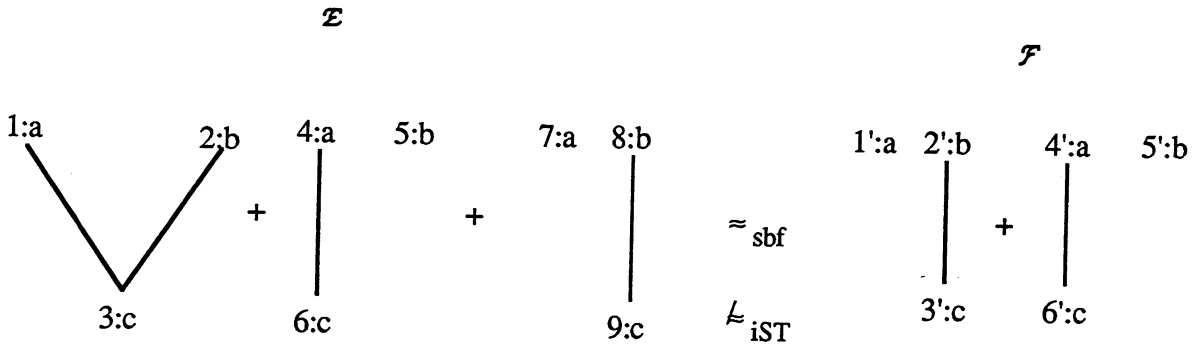


Figure 4.7

La pomset ST-bisimulation est plus fine que la step bisimulation avant arrière.

Proposition 4.6

$\mathcal{E} \approx_{pST} \mathcal{F} \Rightarrow \mathcal{E} \approx_{sbf} \mathcal{F}$

Preuve

Soient $\mathcal{E}$ et $\mathcal{F}$ deux structures d'événements. Pour une histoire $\sigma \in \Pi(\mathcal{E})$ telle que $|\sigma| = n$, nous écrivons C_σ la configuration $\{\sigma(1),...,\sigma(n)\}$ et P_σ la configuration $C_{\sigma i}$ où i est le plus petit indice tel que $C_\sigma - C_{\sigma i} = \{\sigma(i+1),...,\sigma(n)\}$ est un step. D_ρ et Q_ρ sont les notions correspondantes quand $\rho \in \Pi(\mathcal{F})$.

Pour $\sigma \in \Pi(\mathcal{E})$, $\rho \in \Pi(\mathcal{F})$ tel que $\text{trace}(\sigma) = \text{trace}(\rho)$, f_σ^ρ dénote la bijection $f : C_\sigma \rightarrow D_\rho$ telle que $f(\sigma(i)) = \rho(i)$ pour $i = 1...n$

Supposons R: $\mathcal{E} \approx_{pST} \mathcal{F}$, on définit $R' \subseteq \Pi(\mathcal{E}) \times \Pi(\mathcal{F}) \cup \Pi(\mathcal{F}) \times \Pi(\mathcal{E})$ par $\{(\sigma,\rho)\}$ si :

- $|\sigma| = |\rho|$
- $\forall\ i = 0...|\sigma|\quad ((C_{\sigma i}, P_{\sigma i}),(D_{\rho i}, Q_{\rho i}), f_{\sigma i}^{\rho i}) \in R$

. Il est clair que R' est symétrique..

. $(\lambda,\lambda) \in R'$ puisque $(C_\lambda, D_\lambda, f_\lambda^\lambda)$ est $(\emptyset,\emptyset, \emptyset)$ et est dans R.

. Supposons $(\sigma,\rho)\in R'$ et $\sigma \xrightarrow{\theta} \sigma'$ (supposons $\sigma \in \Pi(\mathcal{E})$). Pour vérifier la propriété de transfert, nous considérons d'abord le cas où $|\theta| = 1$.

a) Si $\sigma \xrightarrow{e} \sigma'$alors $(C_\sigma, P_\sigma) \rightarrow_{\mathcal{E}} (C_{\sigma'}, P_{\sigma'})$. Comme $(\sigma,\rho)\in R'$, nous avons $((C_\sigma,P_\sigma), (D_\rho, Q_\rho), f_\sigma^\rho) \in R$ et la pomset ST-bisimilarité entre $\mathcal{E}$ et $\mathcal{F}$ implique qu'il existe une transition $(D_\rho, Q_\rho)\rightarrow_{\mathcal{F}}(D', Q')$ avec $((C_{\sigma'}, P_{\sigma'}), (D',Q'), f') \in R$ pour une bijection f' qui étend f_σ^ρ. Nécessairement D' est un $D_{\rho.e'}$ avec $\rho \xrightarrow{e'} \rho'$ une transition valide et f'(e)=e'.

Alors $(\sigma.e, \rho.e') \in R'$ si nous prouvons que Q' est en effet $Q_{\rho.e'}$. Mais la pST-bisimilarité implique que f' est un isomorphisme de $C_{\sigma.e} - P_\sigma$ dans $D_{\rho.e'} - Q_\rho$ avec $f'(P_{\sigma.e}) = Q'$. Finalement Q' doit être $Q_{\rho.e'}$.

b) Si $\sigma \xrightarrow{\theta} \sigma'$ tel que $\theta=e_1.e_2...e_n$ et pomset (θ) est un step, alors $(C_\sigma, P_\sigma) \rightarrow_{\mathcal{E}} (C_{\sigma.e1}, P_{\sigma.e1}) \rightarrow_{\mathcal{E}} ... \rightarrow_{\mathcal{E}} (C_{\sigma.e1...en}, P_{\sigma.e1...en})$ est imité par $(D_\rho, Q_\rho) \rightarrow_{\mathcal{F}} (D_{\rho.e'1}, Q_{\rho.e'1}) \rightarrow_{\mathcal{F}} ... \rightarrow_{\mathcal{F}} (D_{\rho.e'1...e'n}, Q_{\rho.e'1...e'n})$ avec $(\sigma.e_1...e_k, \rho.e'_1...e'_k)\in R'$ pour k = 1...n comme il a été prouvé dans a). Il reste à montrer que $e'_1...e'_k$ est un step valide après ρ. Ceci est facile puisque nous avons un isomorphisme entre $C_{\sigma'} - P_{\sigma'}$ et $D_{\rho'} - Q_{\rho'}$ avec $e_1...e_n\in C_{\sigma'} - P_{\sigma'}$ par définition de $P_{\sigma'}$.

. Si $\sigma' \xrightarrow{\theta} \sigma$ est une transition en arrière via le step pomset(θ), alors il existe $\rho' \xrightarrow{\theta'} \rho$ avec $|\theta| = |\theta'|$ et $(\sigma',\rho')\in R'$ (par définition deR'). Nous avons seulement à vérifier que pomset(θ') est un step: c'est le cas puisque $((C_\sigma, P_\sigma), (D_\rho, Q_\rho, f_\sigma^\rho) \in R$ et f_σ^ρ est un isomorphisme entre $C_\sigma - P_\sigma$ et $D_\rho - Q_\rho$. Si pomset(θ) est un step, $\theta \subseteq C_\sigma - P_\sigma$ alors $f_\sigma^\rho(\theta) = \theta'$ est nécessairement un step dans $\mathcal{F}$. □

L'implication de la proposition 4.6 est stricte : les structures d'événements de la figure 4.2 sont sbf bisimilaires, mais pas pomset bisimilaires. Seule $\mathcal{E}$ peut effectuer un pomset où a et b sont toutes les deux des causes de c.

4.2.5 Classification

Nous terminons cette section par le théorème suivant qui rassemble les implications entre équivalences que nous avons présentées dans ce chapitre.

Théorème 4.1

La figure 4.8 regroupe toutes les implications (omettant celles dérivables par transitivité) qui existent entre les équivalences traitées.

Preuve

Les propositions 1, 2, 3, 4, 5 et 6 montrent qu'aucune implication ne peut être ajoutée entre les bisimulations traitées, puisque nous avons donné suffisamment d'exemples dans le chapitre 3 qui montrent qu'aucune implication ne peut être ajoutée entre les α-bisimulations et les αST-bisimulations pour $\alpha \in$ {interleaving, partial word, pomset, history preserving} □

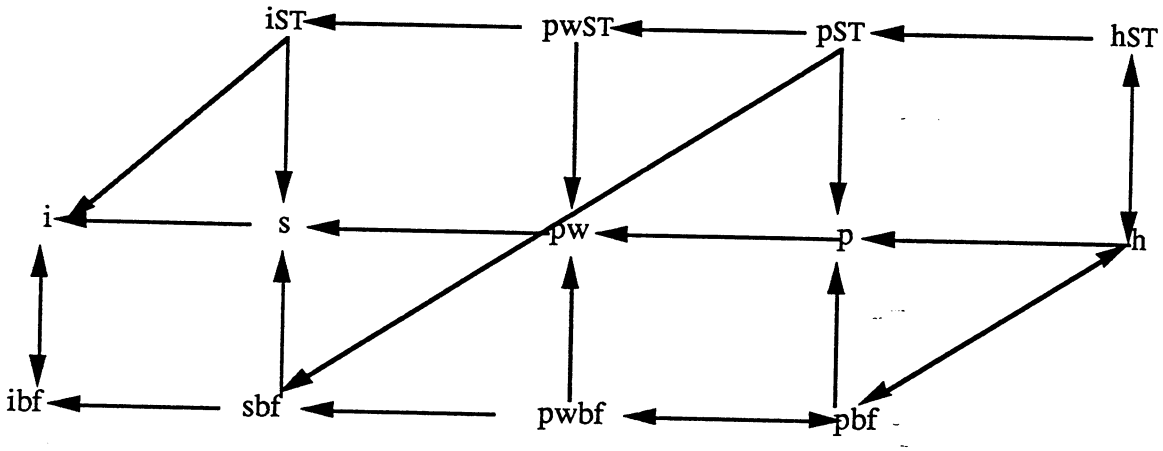


Figure 4.8

4.3 Raffinement d'actions

Ici, nous étudions le comportement des bisimulations avant arrière vis à vis de l'opération de raffinement d'actions.

Proposition 4.7

Soient $\mathcal{E}$ et $\mathcal{F}$ deux structures d'événements de E_{prim}, raf : Act $\rightarrow E_{prim}$ -{0} une fonction de raffinement :

i) $\mathcal{E} \approx_{pbf} \mathcal{F} \Rightarrow raf(\mathcal{E}) \approx_{pbf} raf(\mathcal{F})$

ii) $\mathcal{E} \approx_{pwbf} \mathcal{F} \Rightarrow raf(\mathcal{E}) \approx_{pwbf} raf(\mathcal{F})$

iii) $\approx_{ibf}$ et $\approx_{sbf}$ ne sont pas compatibles avec le raffinement d'actions.

Preuve

i) et ii) Les propositions 4.1 et 4.4 stipulent que la partial word avant-arrière coincide avec la pomset bisimulation avant arrière, qui elle même coincide avec la history preserving bisimulationd (dans le cas sans τ). Or il est bien connu que la history preserving bisimulation est compatible avec le raffinement d'actions. Donc la partial word et la pomset bisimulation avant arrière sont des congruences pour le raffinement d'actions.

iii) $\approx_{ibf}$ n'est pas une congruence pour le raffinement d'actions puisqu'elle coïncide avec l'interleaving bisimulation (non compatible avec le raffinement). La sbf bisimulation n'est pas compatible avec le raffinement d'actions : les structures d'événements de la figure 4.9 (qui est la figure 4.7 où l'action a est raffinée en a—a) ne sont pas sbf bisimilaires, puisqu'elles ne sont même plus interleaving bisimilaires. □

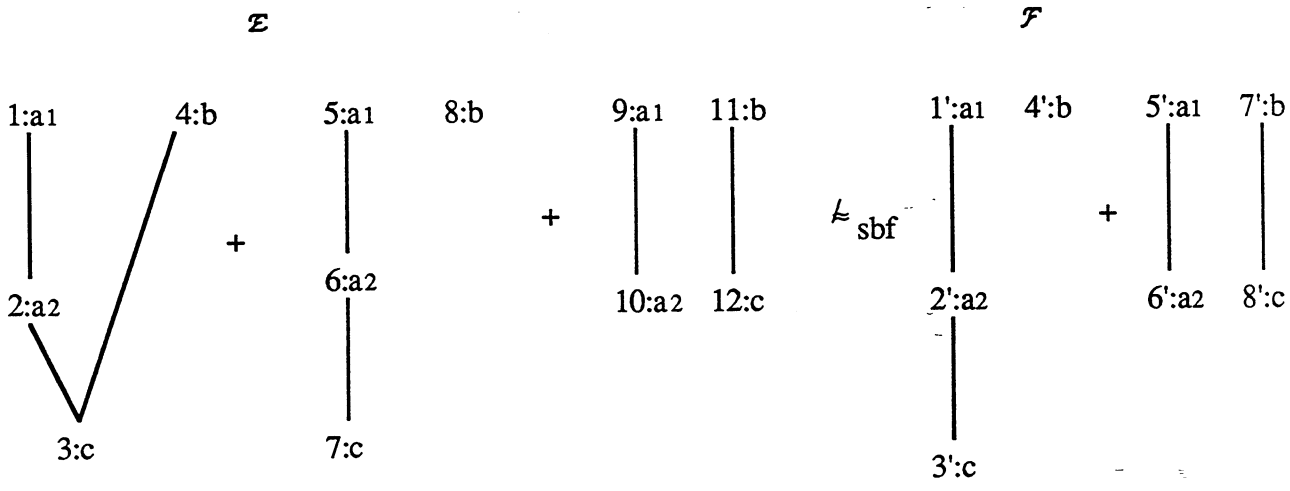


Figure 4.9

4.4 Caractérisations logiques

De nombreuses logiques temporelles ont été définies pour la spécification et la vérification de systèmes parallèles. Ces logiques induisent des équivalences entre processus : deux processus équivalents satisfont exactement les mêmes formules. Il est évident que les équivalence induites

par les logiques doivent être compatibles avec les équivalences comportementales des processus c.à.d. la logique doit confondre tout ce que l'équivalence comportementale confond. En d'autres termes si $\approx_{log}$ et $\approx_{comp}$ désignent respectivement les équivalences logique et comportementale, il est indispensable d'avoir : $\approx_{comp} \Rightarrow \approx_{log}$. Dans le cas où l'implication est une équivalence, on dit que la logique est *adéquate* pour l'équivalence comportementale. On parle également de *caractérisation logique* de l'équivalence sémantique.

La nouvelle caractérisation de la history preserving bisimulation (proposition 4.4),suggère de façon très naturelle une caractérisation logique de cette équivalence.
La logique utilisée l_{bf}, est une variante de la logique HML où nous autorisons des modalités du genre $<p>\varphi$ et $<\!\!\leftarrow\!\!p>\varphi$ où p est un pomset et $\leftarrow$ un opérateur du passé similaire à celui introduit dans [DNV 90] pour caractériser la bisimulation de branchement [vGW 89].

Par ailleurs, dans [DNF 90], les auteurs proposent une logique, l_p, qui caractérise la weak history preserving bisimilation. Nous montrons que la logique l_{bf} s'injecte facilement dans la logique l_p. Ainsi l_p a au moins le même pouvoir de distinction que l_{bf}. Comme de plus, nous donnons une traduction de l_p dans l_{bf}, l_p caractérise donc la history preserving bisimulation et non la weak history preserving bisimulation.

4.4.1 La logique modale l_{bf}

Ici, nous présentons la logique l_{bf} qui suppose donné un ensemble Act = {a,b,...} d'actions.

Définition 4.7 (Syntaxe de l_{bf})
Les formules de la logique l_{bf} sont données par la grammaire suivante :
$\varphi, \varphi' ::= T \mid \neg\varphi \mid \varphi \wedge \varphi' \mid <p>\varphi \mid <\!\!\leftarrow\!\!p>\varphi$
où p est un pomset sur Act.

Nous interprétons les formules de l_{bf} sur les histoires. Intuitivement, $<p>\varphi$ signifie qu'il est possible d'effectuer le pomset p et atteindre une histoire où φ est vraie. $<\!\!\leftarrow\!\!p>\varphi$ signifie qu'on peut faire un retour en arrière via le pomset p et atteindre une histoire où φ est vraie.

Définition 4.8 (Sémantique de l_{bf})
Soit $\mathcal{E}$ une structure d'événements, σ une histoire de $\mathcal{E}$, on définit une relation $\models_{\mathcal{E}} \subseteq \Pi(\mathcal{E}) \times l_{bf}$ par :

- $\sigma \models_{\mathcal{E}} T$ toujours
- $\sigma \models_{\mathcal{E}} \neg\varphi$ ssi $\sigma \not\models_{\mathcal{E}} \varphi$
- $\sigma \models_{\mathcal{E}} \varphi \wedge \varphi'$ ssi $\sigma \models_{\mathcal{E}} \varphi$ et $\sigma \models_{\mathcal{E}} \varphi'$
- $\sigma \models_{\mathcal{E}} \langle p \rangle \varphi$ ssi il existe $\sigma \xrightarrow{p} \sigma'$ telle que $\sigma' \models_{\mathcal{E}} \varphi$
- $\sigma \models_{\mathcal{E}} \langle \leftarrow p \rangle \varphi$ ssi il existe $\sigma' \xrightarrow{p} \sigma$ telle que $\sigma' \models_{\mathcal{E}} \varphi$

Le nom de la structure d'événements (dans $\models_{\mathcal{E}}$) sera laissé implicite quand cela n'introduit pas d'ambiguités.
Pour $A \subseteq Act$ et φ une formule de l_{bf}, on écrira $\langle A \rangle \varphi$ pour $\vee_{a \in A} \langle a \rangle \varphi$ et $\langle \leftarrow A \rangle \varphi$ pour $\vee_{a \in A} \langle \leftarrow a \rangle \varphi$. On écrira aussi $[A]\varphi$ pour $\neg \langle A \rangle \neg \varphi$.

La logique l_{bf} définit une équivalence, notée $\sim_{lbf}$, entre les structures d'événements : deux structures d'événements sont $\sim_{lbf}$ équivalentes si et seulement si elles satisfont exactement le même ensemble de formules, c.à.d. si elles sont indistinguables par la logique l_{bf}.

Définition 4.9 (équivalence l_{bf})
Pour toutes structures d'événements $\mathcal{E}$ et $\mathcal{F}$ de E_{prim}
i) $\mathcal{E}$ satisfait φ, noté $\mathcal{E} \models \varphi$ ssi $\lambda \models_{\mathcal{E}} \varphi$
ii) $\mathcal{E} \sim_{lbf} \mathcal{F}$ ssi ($\mathcal{E} \models \varphi \Leftrightarrow \mathcal{F} \models \varphi$) pour toute formule φ de l_{bf}.

Le théorème suivant relie la logique l_{bf} à la pomset bisimulation avant arrière : La logique l_{bf} distingue exactement ce que $\approx_{pbf}$ distingue. Dans la littérature on parle dans ce cas de résultat d'adéquation.

Théorème 4.2 (adéquation)
Soient $\mathcal{E}$ et $\mathcal{F}$ deux structures d'événements premières finies:
$\mathcal{E} \approx_{pbf} \mathcal{F} \Leftrightarrow \mathcal{E} \sim_{lbf} \mathcal{F}$

Preuve
1) "$\Rightarrow$" Supposons R une pomset bisimulation avant arrière entre $\mathcal{E}$ et $\mathcal{F}$. On montre par induction sur la structure des formules φ de l_{bf} que $(\sigma, \rho) \in R$ implique ($\sigma \models \varphi$ ssi $\rho \models \varphi$).

- Les cas T, $\neg\varphi$, $\varphi \wedge \varphi'$ sont évidents.

• $<p>\varphi$: Supposons $\sigma \models <p>\varphi$. Alors il existe $\sigma \xrightarrow{p} \sigma'$ avec $\sigma' \models \varphi$. Comme R est une $\approx_{pbf}$, il existe $\rho \xrightarrow{p} \rho'$ telle que $(\sigma', \rho') \in R$. Par hypothèse d'induction $\sigma' \models \varphi$ ssi $\rho' \models \varphi$. Donc $\rho \models <p>\varphi$.

• $<\!\!\leftarrow p>\varphi$: Se traite de façon similaire.

Comme $(\lambda, \lambda) \in R$, nous avons bien R: $\mathcal{E} \sim_{lbf} \mathcal{F}$

2) "$\Leftarrow$" : Supposons R : $\mathcal{E} \sim_{lbf} \mathcal{F}$. Par un raisonnement par l'absurde, on montre que R est une pomset bisimulation avant arrière.
Si R n'est pas une $\approx_{pbf}$, trois cas sont alors possibles :

• a) $(\lambda, \lambda) \notin R$. Il existe donc $\varphi \in l_{bf}$ telle que $\lambda \models_{\mathcal{E}} \varphi$ et $\lambda \not\models_{\mathcal{F}} \varphi$ ou vice versa. Ceci contredit $\mathcal{E} \sim_{lbf} \mathcal{F}$. Donc (λ, λ) est nécessairement dans R.

• b) pour $(\sigma, \rho) \in R$, $\sigma \xrightarrow{p} \sigma'$ nous avons $\forall\, \rho \xrightarrow{p} \rho'$, $(\sigma', \rho') \notin R$. Soit $\{\rho_1, \rho_2, ...\} = \{\rho' \mid \rho \xrightarrow{p} \rho'\}$. $\forall i$, il existe une formule φ_i de l_{bf} telle que $\sigma' \not\models_{\mathcal{F}} \varphi_i$ et $\rho_i \models \varphi_i$. Donc $\rho \models [p] \vee_i \varphi_i$ et $\sigma \not\models [p] \vee_i \varphi_i$. Ce qui contredit $(\sigma, \rho) \in R$.

•c) Le cas $(\sigma, \rho) \in R$, $\sigma' \xrightarrow{p} \sigma$ est similaire à b) . □

Corollaire 4.1
Soient $\mathcal{E}$ et $\mathcal{F}$ deux structures d'événements :
$\mathcal{E} \sim_{lbf} \mathcal{F} \Leftrightarrow \mathcal{E} \approx_h \mathcal{F}$

Exemple : Dans la figure 4.10 les stuctures d'événements qui ne sont pas history preserving, ne vérifient pas les mêmes formules l_{bf}. Par exemple, seul $\mathcal{F} \models <a>\,[b]\, <\!\!\leftarrow a.b>T$: En effet, dans $\mathcal{F}$, il est possible qu'après avoir effectué une action a, quelle que soit la transition par b, $\mathcal{F}$ aurait effectué un pomset a.b (a est une cause de b). Il est clair que ce comportement n'est pas possible dans $\mathcal{E}$: il suffit de considérer le cas où $\mathcal{E}$ choisit d'effectuer l'événement 3 après l'événement 1.

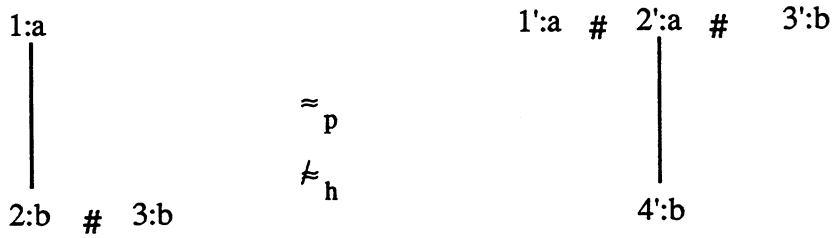


Figure 4.10

4.4.2 Une autre caractérisation logique de la history preserving bisimulation

A ce jour il n'existe pas, à notre connaissance, d'autres caractérisations logiques pour la history preserving bisimulation. Lors de nos recherches de travaux dans cette direction, nous avons noté que dans [DNF 90], les auteurs proposent une logique, notée l_p, qui pour les auteurs caractérise la weak history preserving bisimulation. Dans cette section, nous prouvons que la logique l_p caractérise elle aussi la history preserving bisimulation. Après les définitions de la syntaxe et de la sémantique de l_p, nous proposons une traduction plutôt évidente de l_{bf} dans l_p, puis une transformation de l_p dans l_{bf}.

4.4.2.1 La logique l_p

Définition 4.10 (Syntaxe de l_p)
Les formules de la logique l_p sont données par la grammaire suivante :

$$\varphi, \varphi' ::= T \mid q \mid \neg\varphi \mid \varphi \wedge \varphi' \mid X\varphi \mid X^{-1}\varphi \mid \forall\varphi$$

où q est une proposition atomique correspondant au pomset q sur Act.

Nous interprétons l_p sur des modèles (π, σ) où σ est une histoire et π une histoire maximale qui est une extension possible de σ.
Le quantificateur $\forall$ donne à l_p la possibilité de branchement. Il permet de quantifier sur tous les futurs possibles ou extensions possibles d'une histoire σ.
Une proposition atomique q est vraie dans l'état (π, σ) si le pomset q constitue le dernier ensemble d'événements effectués le long de σ.
Nous noterons $\sigma \leq \pi$ quand σ est un préfixe de π.

Définition 4.11(Sémantique de l_p)
Soit $\mathcal{E}$ une structure d'événements, $\pi, \sigma \in \Pi(\mathcal{E})$ avec π une histoire maximale et $\sigma \leq \pi$. On définit $\models_{\mathcal{E}} \subseteq \Pi(\mathcal{E}) \times \Pi(\mathcal{E}) \times l_p$ par :

- $(\pi, \sigma) \models_{\mathcal{E}} T$ toujours
- $(\pi, \sigma) \models_{\mathcal{E}} q$ ssi il existe $\sigma = \sigma'.\theta$ avec pomset $(\theta) = q$
- $(\pi, \sigma) \models_{\mathcal{E}} \neg\varphi$ ssi $(\pi, \sigma) \not\models_{\mathcal{E}} \varphi$
- $(\pi, \sigma) \models_{\mathcal{E}} \varphi \wedge \varphi'$ ssi $(\pi, \sigma) \models_{\mathcal{E}} \varphi$ et $(\pi, \sigma) \models_{\mathcal{E}} \varphi'$
- $(\pi, \sigma) \models_{\mathcal{E}} X\varphi$ ssi $\pi = \sigma.e.\theta$ et $(\pi, \sigma.e) \models_{\mathcal{E}} \varphi$
- $(\pi, \sigma) \models_{\mathcal{E}} X^{-1}\varphi$ ssi $\sigma = \sigma'.e$ et $(\pi, \sigma') \models_{\mathcal{E}} \varphi$
- $(\pi, \sigma) \models_{\mathcal{E}} \forall\varphi$ ssi pour toutes les histoires maximales π' telles que $\sigma \leq \pi'$, $(\pi', \sigma) \models\varphi$

On écrira $\sigma\models_{\mathcal{E}} \varphi$ quand $(\pi, \sigma) \models_{\mathcal{E}} \forall\varphi$ et $\mathcal{E} \models \varphi$ si et seulement si $\lambda\models_{\mathcal{E}} \varphi$

Comme l_{bf} la logique l_p définit une équivalence, notée $\sim_{lp}$, sur les structures d'événements, définie formellement comme suit :

Définition 4.12 (équivalence l_p)
Pour toutes structures d'événements $\mathcal{E}$ et $\mathcal{F}$:
$\mathcal{E} \sim_{lp} \mathcal{F}$ ssi $(\mathcal{E} \models \varphi \Leftrightarrow \mathcal{F} \models \varphi)$ pour toute formule φ de l_p.

4.4.2.2 Une traduction de l_{bf} dans l_p

Dans cette section, nous montrons que l_p a moins le même pouvoir de distinction que l_{bf}

Théorème 4.3 (l_p au moins aussi expressive que l_{bf})
Il existe une application $f : l_{bf} \rightarrow l_p$ telle que pour toute structure d'événements $\mathcal{E}$, pour toute histoire σ de $\mathcal{E}$ et pour toute formule φ de l_{bf} : $\sigma\models\varphi$ ssi $\sigma\models f(\varphi)$

Preuve
Nous écrivons $\exists\varphi$ pour $\neg\forall\neg\varphi$, X^n (respectivement X^{-n}) pour n emboitements de l'opérateur X (respectivement X^{-1}). $|q|$ est la taille du pomset q ou la cardinalité du multiensemble des actions de q.
f est définie inductivement comme suit :

- $f(T) = T$
- $f(\neg\varphi) = \neg f(\varphi)$
- $f(\varphi \wedge \varphi') = f(\varphi) \wedge f(\varphi')$
- $f(<q>\varphi) = \exists X^{|q|}\ (f(\varphi) \wedge q)$
- $f(<\leftarrow q>\varphi) = q \wedge X^{-|q|}\ f(\varphi)$

Il est facile de vérifier par induction sur la structure des formules, que f a bien la propriété énoncée dans le théorème 4.3. □

Exemple Les structures d'événements de la figure 4.10 qui sont distinguables par la logique l_{bf} sont aussi distinguables par la logique l_p. Il est facile de constater. que seule $\mathcal{F}$ satisfait $\varphi \equiv \exists X(a \wedge \forall X(a.b))$.

L'exemple de la figure 4.11 montre que les deux structures d'événements qui sont weak history preserving bisimilaires sont distinguables par la logique l_p. En effet, seule $\mathcal{E}$ satisfait $\phi \equiv \forall X(\exists XXa.b)$. En d'autres termes, dans $\mathcal{E}$ après avoir effectué un événement (n'importe lequel), il est toujours possible d'effectuer le pomset a.b (a cause de b). Ce comportement n'est pas possible dans $\mathcal{F}$ quand $\mathcal{F}$ effectue l'événement 2 en premier. Par conséquent L_p ne peut caractériser la weak history preserving bisimulation. Le théorème 4.3 prouve que l_p a au moins le même pouvoir de dinstinction que l_{bf}.

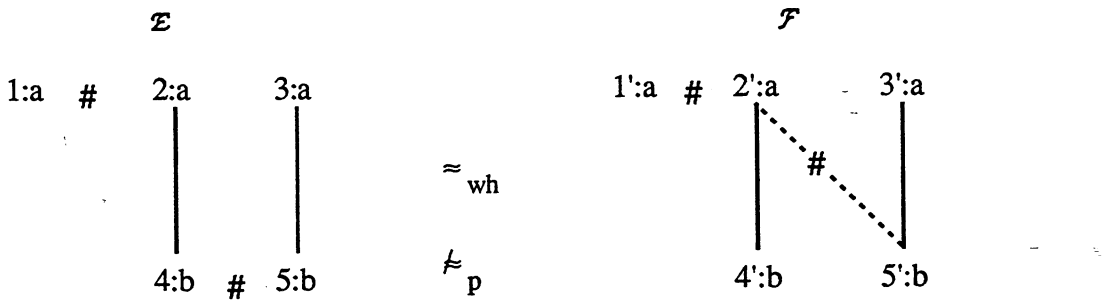


Figure 4.11

La transformation qui suit montre que l_p n'est pas plus expressive que l_{bf}.

4.4.2.3 Une traduction de l_p dans l_{bf}

Dans cette section, nous proposons une transformation de l_p dans l_{bf}. Cette traduction n'est pas directe[CLS 92]. Cependant les formules de l_p dites de branchement peuvent être vues comme des formules l_{bf}.

Définition 4.13 (formule de branchement)
Une formule de l_p est dite de branchement, si toute occurrence de l'opérateur X(next) est immédiatement sous la portée d'un quantificateur $\forall$ ou $\exists$.

Il est intéressant de noter que toute formule de branchement est une formule d'état, formule dont la valeur de vérité en (π, σ) ne dépend que de σ. Les formules d'état ne dépendent pas du chemin choisi.

Proposition 4.8
Toute formule φ de branchement de l_p est équivalente à une formule φ' de l_{bf}.

Preuve
Dans φ de branchement, nous substituons toute occurrence de :

- q : proposition atomique par $<\!\!\leftarrow q\!>\!T$
- X^{-1} par $<\!\!\leftarrow Act\!>$
- $\forall X$ par $[Act]$
- $\exists X$ par $<\!Act\!>$

Ceci nous fournit une formule $\varphi' \in l_{bf}$, équivalente à φ.

Exemple La formule $\varphi \equiv \exists X(a \wedge \forall X(a.b))$ se transforme en $\varphi' \equiv <\!Act\!>(<\!\!\leftarrow a\!>\!T \wedge [Act]<\!\!\leftarrow a.b\!>\!T)$ de l_{bf}.

Pour traduire l_p dans l_{bf}, nous devons traduire toute formule d'état de l_p en une formule de branchement. Cette traduction se fait en plusieurs étapes. Pour commencer, nous présentons quelques notations utiles. Nous écrivons δ pour $\neg$ XT. Par conséquent, $\sigma \models_{\mathcal{E}} \delta$ ssi σ est maximale dans $\mathcal{E}$.

Définition 4.14

Une formule φ de l_p est dite :

i) linéaire si elle ne comporte pas de quantificateur $\forall$ ou $\exists$,

ii) pur futur (respectivement pur passé) si elle ne contient pas d'opérateur X^{-1} (respectivement X).

iii) Strictement du futur, si toute proposition atomique *q* est sous la portée d'un X.

Lemme 4.1

Toute formule linéaire φ de l_p est équivalente à une formule dite séparée qui est une combinaison booléenne de formules pur passé ou pur futur c.à.d de la forme:

$$\wedge_i(\vee_j \varphi^-_{i,j} \vee \vee_k \varphi^+_{i,k})$$

où les $\varphi^-_{i,j}$ sont linéaires pur passé et $\varphi^+_{i,k}$ sont pur futur.

Preuve

Nous utilisons les règles de réécriture suivantes :

- $XX^{-1}\varphi \rightarrow XT \wedge \varphi$ (1)
- $X\neg X^{-1}\varphi \rightarrow XT \wedge \neg\varphi$ (2)
- $X^{-1}X\varphi \rightarrow X^{-1}T \wedge \varphi$ (3)
- $X^{-1}\neg X\varphi \rightarrow X^{-1}T \wedge \neg\varphi$ (4)

En utilisant les propriétés de distributivité de X et X^{-1} (telles que $X(\varphi \wedge \varphi') \equiv X(\varphi) \wedge X(\varphi')$) et les lois booléennes habituelles, toute formule φ peut être transformée de façon à lui appliquer une des règles (1)- (4). Clairement, ce système de réécriture termine et préserve l'équivalence et la linéarité des formules. La forme normale conjonctive ne contient que des formules pur passé ou pur futur. □

Nous venons de voir comment traduire une formule linéaire de l_p en une formule équivalente (en forme normale conjonctive, dite séparée) qui ne contient que des formules pur passé ou pur futur. Maintenant, nous nous posons le problème de séparation de formules de la forme $\forall\varphi$ où φ est linéaire.

Lemme 4.2

Toute formule $\forall\varphi$ de l_p, où φ est linéaire est équivalente à une formule séparée de la forme :

$$\wedge_i (\vee_j \varphi^-_{i,j} \vee \psi_i)$$

où les $\varphi^-_{i,j}$ sont linéaires pur passé et les ψ_i sont des formules de branchement pur futur.

Preuve

D'après le lemme 4.1, il suffit de considérer φ de la forme $\wedge_i (\vee_j \varphi^-_{i,j} \vee \vee_k \varphi^+_{i,k})$. Alors $\forall\varphi \equiv$ $\wedge_i \forall(\vee_j \varphi^-_{i,j} \vee \vee_k \varphi^+_{i,k})$. Or $\forall(\varphi^- \vee \varphi) \equiv \varphi^- \vee \forall\varphi$ quand φ^- est pur passé. Nous avons donc $\forall\varphi$ $\equiv \wedge_i (\vee_j \varphi^-_{i,j} \vee \forall\vee_k \varphi^+_{i,k})$.

Il reste à montrer comment transformer une formule $\forall\psi$ où ψ est pur futur en une formule de branchement. La preuve se fait par induction sur la profondeur de ψ c.à.d. sur le nombre maximal d'opérateurs X emboités. ψ peut être mise sous la forme générale $\psi \equiv \wedge_r(q_{r,1} \vee q_{r,2}$ $...\vee\neg q'_{r,1}\vee \neg q'_{r,2}...\vee X\psi_{r,1}\vee X\psi_{r,2}...\vee\neg X\psi'_{r,1}\vee\neg X\psi'_{r,2}....)$. Comme $\neg X\psi'_{r,i} \equiv \neg XT \vee X\neg\psi'_{r,i}$ et $X(\psi) \vee X(\psi') \equiv X(\psi\vee\psi')$, il est possible de réécrire ψ en :

$\psi \equiv \wedge_r(q_{r,1} \vee q_{r,2} ...\vee\neg q'_{r,1}\vee \neg q'_{r,2}...\vee X(\psi_{r,1} \vee \psi_{r,2}....))$ avec des $q_{r,i}$ éventuellement égaux à δ. Donc $\forall\psi \equiv \wedge_s(q_{s,1} \vee q_{s,2} ...\vee\neg q'_{s,1}\vee \neg q'_{s,2}...\vee\forall X\forall(\psi_{s,1} \vee \psi_{s,2}....))$. Par hypothèse d'induction $\forall(\psi_{s,1} \vee \psi_{s,2}....)$ peut être transformée en une formule de branchement ψ_s. Donc $\forall\psi \equiv \wedge_s(q_{s,1} \vee$ $q_{s,2} ...\vee\neg q'_{s,1}\vee \neg q'_{s,2}...\vee\forall X\psi_s)$ est une formule de branchement. □

Les lemmes 4.1 et 4.2 suffisent pour transformer n'importe quelle formule $\forall\varphi$ de l_p.
On procède par induction sur les sous formules :

- Si φ est linéaire, on applique le lemme 4.2.
- Dans le cas plus général, φ n'est pas linéaire, c.à.d. elle contient des sous formules de la forme $\forall\varphi_i$, i =1...
- Les $\forall\varphi_i$ sont des formules d'états, nous substituons une proposition atomique pour chacune d'entre elles.

- On applique le lemme 4.2 à $\forall\varphi$ qui donne une formule de branchement à l'exception des sous formules $\forall\varphi_i$. Par hypothèse d'induction ces dernières sont transformables en des formules de branchement ψ_i, i = 1....
- On remplace les $\forall\varphi_i$, i =1...par les ψ_i, i =1...et on obtient une formule de branchement φ' équivalente $\forall\varphi$.

Corollaire 4.2
Il existe une application $g : l_p \rightarrow l_{bf}$ telle que pour toute structure d'événements $\mathcal{E}$, pour toute histoire σ de $\mathcal{E}$, et pour toute formule φ de l_p : $\sigma \models \varphi$ ssi $\sigma \models g(\varphi)$

Preuve
Soit $\varphi \in l_p$. Quand elles sont interprétées sur les histoires, les formules $\forall\varphi$ et φ sont équivalentes. Il suffit donc de considérer les formules φ de l_p quantifiées par $\forall$. Nous savons transformer toute formule $\forall\varphi$ de l_p en une formule de branchement ψ. Or d'après la proposition 4.8, il existe $\psi' \in l_{bf}$ équivalente à ψ. □

Exemple la formule $\forall (Xq_1 \vee X(q_2 \wedge \neg \chi^{-1}\forall Xq_3))$ se traduit en $[Act] (q_1 \vee q_2) \wedge [Act]T \wedge ([Act] q_1 \vee \neg [Act] q_3)$

4.5 Conclusions

Dans ce chapitre, nous avons adapté la notion de bisimulation avant arrière aux structures d'événements. Nous avons comparé les nouvelles équivalences aux bisimulations classiques et aux ST-bisimulations. Nous avons étudié leurs comportements vis à vis de l'opérateur de raffinement d'actions. Cette étude a conduit à une nouvelle caractérisation de la history preserving bisimulation.
Par ailleurs, nous avons proposé deux caractérisations logiques de la history preserving bisimulation. La première à l'aide de la logique l_{bf} qui découle directement du résultat de la proposition 4.4 où on prouve que la pomset bisimulation avant arrière coïncide avec la history preserving bisimulation. La deuxième logique est appelée l_p. Alors qu'elle a été proposé dans [DNF 90] pour la weak history preserving bisimulation, nous avons prouvé qu'elle a exactement le même pouvoir de distinction que l_{bf}.

Diverses extensions à ce travail sont à envisager :

- Il serait intéressant d'étudier les bisimulations avant arrière sur les structures d'événements dans le cas où on autorise des actions invisibles (non observables de l'extérieur).
- Chercher les plus grandes équivalences compatibles avec le raffinement d'actions et contenues dans les bisimulations avant arrière.
- Chercher des caractérisations logiques pour d'autres équivalences sur les modèles de vrai parallèlisme.

Conclusions

Résultats

Dans ce travail, nous avons étudié le comportement des équivalences de bisimulation vis à vis de l'opération de raffinement d'actions dans les systèmes de transitions. La bisimulation forte où toutes les actions sont considérées observables ou visibles de l'extérieur est compatible avec le raffinement d'actions.
Les τ-,η-, Δ-bisimulation et la bisimulation de branchement sont des équivalences qui traitent les cas où certaines actions sont invisibles ou non observabes de l'extérieur. Parmi ces quatre équivalences, la bisimulation de branchement et la Δ-bisimulation sont compatibles avec le raffinement d'actions. Aussi, nous avons montré que la quasi-branching bisimulation et la Δ-bisimulation sont les plus grandes congruences par rapport au raffinement et contenues respectivement dans la η-bisimulation et la τ-bisimulation.

Dès lors que l'étude des équivalences comportementales était bien maîtrisée dans le modèle des systèmes de transition, on a cherché à étudier les modèles opérationnels plus généraux incorporant la notion d'actions concurrentes. Dans ce cadre, nous avons étudié les équivalences de bisimulation sur les structures d'événements qui constituent un modèle de base pour le vrai parallèlisme. Nous avons défini la notion de bisimulation avant arrière sur les structures d'événements premières. Nous avons généralisé cette définition, de façon uniforme à la step, pomset, partial word et pomset bisimulation. Nous avons comparé ces nouvelles équivalences

aux bisimulations classiques ainsi qu'aux ST-bisimulations. Il s'est avéré essentiellement que l'interleaving bisimulation avant arrière n'augmente pas le pouvoir de distinction de la bisimulation classique (interleaving). La step bisimulation avant arrière est plus fine que la step bisimulation classique et contient la pomset-ST-bisimulation. La partial word bisimulation avant arrière coïncide avec la version avant arrière de la pomset bisimulation qui elle même coïncide avec la history preserving bisimulation (équivalence compatible avec le raffinement d'actions). Il est important de souligner que ces nouvelles caractérisations contribuent à une compréhension plus approfondie des équivalences comportementales. Par exemple, la history preserving bisimulation a été définie de diverses façons (pas toujours très simples à comprendre) par plusieurs auteurs [RT 88], [DDNM 89], [vGG 89], [BDKP 91].

Dans ce travail, nous avons aussi contribué aux travaux qui lient les équivalences induites par une logique modale aux équivalences comportementales. Dans la dernière partie de la thèse, nous proposons une caractérisation logique l_{bf}, pour la history preserving bisimulation. Cette logique généralise la logique HML de Hennessy Milner : l'entité observable étant le pomset. Il s'agit d'une logique avec des modalités du genre $\langle p \rangle \varphi$ et $\langle\langle\leftarrow p\rangle\varphi$ où p est pomset et $\leftarrow$ un opérateur du passé. L'introduction d'opérateurs du passé dans les logiques permet d'analyser les états antérieurs d'un calcul. Notons que ce résultat d'adéquation découle très naturellement de la nouvelle caractérisation de la history preserving bisimulation par la pomset avant arrière. A ce jour, il n'existe pas à notre connaissance d'autres caractérisations logiques pour la history preserving bisimulation. Par contre, dans [DNV 90], les auteurs proposent une logique l_P, qu'ils affirment adéquate pour la weak history preserving bisimulation. Dans la dernière section du chapitre 4, nous avons prouvé que la logique l_P a exactement le même pouvoir de distinction que la logique l_{bf}. Ainsi, la logique l_p caractérise elle aussi la history preserving bisimulation et non la weak history preserving bisimulation.

Perspectives

• Le problème de raffinement étant d' actualité, il serait intéressant de chercher les plus grandes équivalences compatibles avec le raffinement et contenues dans les bisimulations avant arrière.

• Il serait également intéressant d'étudier les bisimulations avant arrière sur les modèles de vrai parallélisme où les systèmes peuvent effectuer des actions invisibles ou non observables de l'extérieur.

• Chercher des caractérisations logiques pour d'autres équivalences sur les modèles de vrai parallélisme. Plus particulièrement, pour celles qui sont basées sur la notion de ST-sémantiques. Ces équivalences semblent avoir de bonnes propriétés et sont cependant actuellement peu explorées.

Bibliographie

[AB 84] D. Austry and G. Boudol. *Algèbre de Processus et Synchronisation.* Theoretical Computer Science, 30(1): 91-131, 1984.

[Ace 91] L. Aceto. *History preserving, causal and mixed ordering equivalences for stable event structures (note).* Technical Memo HPL-PSC-91-28. Hewlett Packard Laboratories, Pisa Center, Pisa 11. To appear in Fundamenta Informaticae.

[AE 91] L. Aceto and U. Engberg . *Failures Semantics for a Simple Process Language with Refinement.* In Proc. 11th FSTTCS, LNCS 560, pp. 89-108. Dec 1991

[AH 89] L. Aceto and M. Hennessy. *Torwards Action-Refinement in Process Algebras.* In Proc. 4th Annual Symp. on Logic in Computer Science (LICS' 89), Asilomar, California, IEEE Computer Society Press, 138-145, 1989.

[AH 90] L. Aceto and M. Hennessy. *Adding Action Refinement to a Finite Process Algebra.* Computer 6/90, University of Sussex, Brighton, GB, November 1990.

[Apt 81] K. R. Apt. *Ten years of Hoare's Logic : A survey.* Part I. ACM Transactions on Programming Langages and Systems. 3(4), pp. 431-484, October 1981.

[BBK 87] J. C. M Baeten, J.A. Bergstra, and J.W. Klop. *On the Consistency of Koomen's fair Abstraction rule.* Theoretical Computer Science, 51(1): 129-176, 1987.

[BC 87] G. Boudol and I. Castellani. *On the Semantics of Concurrency : Partial Orders and Transition Systems*.. In Proc. CAAP'87, Pisa, LNCS 249, pp 123-137. Springer Verlag 1987.

[BC 88] G. Boudol and I. Castellani. *Concurrency and Atomicity*. Theoretical Computer Science 59 (1-2), 25-84. 1988.

[BC 89] G. Boudol and I. Castellani. *Permutation of transitions : An event Structure Semantics for CCS and SCCS* . In Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency. Noordwijkerhoot, LNCS354, pp 411-427, Springer Verlag. 1989

[BCG 88] M. C. Bowne, E. M. Clarke and O. Grümberg. *Characterizing Finite Kripke Structures in Propositional Temporal Logic*. Theoretical Computer Science 59(1988) 115-131.

[BDKP 89] E. Best, R. Devillers, A. Kiehn, and L. Pomello. *Concurrent bisimulations in Petri Nets*. Acta Informatica, 28, pp 231-264, 1991

[BvG 87] J. C. M. Baeten and R. J. Van Glabbeek. *Another Look at Abstraction in Process Algebra*. (Extended Abstract). In Proc. 14th ICALP, Karlsruhe, LNCS 267, pp 84-94. Springer Verlag, July 1987.

[BHR 84] S. D. Brookes, C. A. R Hoare, and A.W.Roscoe. *A Theory of Communicating Sequential Processes*. Journal of the ACM 31(3),pp 560-599, July 1984

[BK 84] J. A. Bergstra and J. W. Klop.*Process Algebra for Synchronous Communication*. Information and Control, 60 : 109- 137, 1984.

[BK 85] J. A. Bergstra and J. W. Klop. *Algebra of Communicating Processes with Abstractions*..Theoretical Computer Science. 37(1),pp. 77-121, 1985.

[BMP 83] M. Ben Ari, Z. Manna and A. Pnueli. *The Temporal Logic of Branching Time*. Acta Informatica 20, 207-226, 1983.

[Bou 85] G.Boudol. *Calculs de Processus et Vérification.* RR424, INRIA, Sophia-Antipolis, 1985.

[CDMP 87] L. Castellano, G. De Michelis, and L. Pomello. *Concurrency versus interleaving: An instructive example.* Bull. EATCS 31, pp 12-15, 1987

[CE 81] E. M. Clarke and E. A. Emerson. *Design and Synthesis of Synchronization Skeletons using Branching Time Logic.* Proc. IBM workshop on logics of programs. LNCS 131, 52-71, 1981.

[CES 83] E. Clarke, E. A. Emerson, and A. P. Sistla. *Automatic Verification of Finite State Concurrent Systems Using Temporal Logic.* In Proc.10th. Annual Symp. On Principles of Programming Langages. 1983.

[CFM 83] I. Castellani, P. Franceschi and U. Montanari. *Labeled Event Structures : A Model for ObsevableConcurrency.* In Formal Description of Programming Concepts II (D. Bjorner , Ed.), North Holland, Amsterdam, 1983, pp. 383-400.

[Che 92a] F. Cherief. *Back and Forth Bisimulations on Prime event Structures.* In Proc. PARLE'92, Paris, LNCS 605, pp 843-858, Springer Verlag, June 1992.

[Che 92b] F. Cherief. *Investigations of Back and Forth Bisimulations on Prime event Structures* . Computers and Artificial Intelligence, Vol. 11, 1992, No 5, 481- 496.

[CLS 92] F. Cherief, F. Laroussinie, and S. Pinchinat. *Modal Logics with Past for True Concurrency.* To appear

[CS91] F. Cherief and Ph. Schnoebelen. *τ-bisimulations and full abstration for refinement of actions.* Inf. Proc. Letters. 40, November 1991, pp 219-222.

[CS 93] F. Cherief and Ph. Schnoebelen. Quasi-branching Bisimulation. To appear.

[DNF90] R. De Nicola, G.L Ferrari . *Observational Logics And Concurrency Models.* In Proc. 10th Conf. On Fondations of Software Technology and Theoretical Computer Science, Bangalore, India, LNCS 472, pp 301-315

[DNV90] R. De Nicola, F. Vaandrager. *Three Logics For Branching Bisimulation.* (extended abstract). In Proc. 5th IEEE Symp. Logic in Computer Science, Philadelphia, PA, pp 118-129, June 1990.

[DNMV90] R. De Nicola, U. Montanari, and F. W. Vaandrager. *Back and Forth Bisimulations*. In Proc. CONCUR'90, Amsterdam, LNCS 458, pp 152-165. Springer-Verlag, August 1990.

[DDNM 88] P. Degano, R. De Nicola, and U. Montanari. *A Distributed Operational Semantics for CCS based on Condition/Event Systems*. Acta Informatica 26, pp. 59-91. 1988.

[DD 89] Ph. Darondeau and P. Degano. *Causal Trees*. In Proc. 11th ICALP LNCS 372, pp. 234-248.1989.

[DDNM 89] R. Degano, R. De Nicola and U. Montanari. *Partial Ordering Description of Nondeterministic Concurrent Systems*. In Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency. Noordwijkerhoot, LNCS 354, pp 438-466, Springer Verlag. 1989.

[Dev 88] R. Devillers. *On The definition of a bisimulation notion based on partial words.* Petri Net newsletter 29, Gesellschaft für Informatik., Bonn, pp. 16-19

[Dev 90] R. Devillers. *Maximality Preserving Bisimulation.* Tech. Report LIT-214, Lab. Informatique Théorique, Université libre de Bruxelles,March 1990.

[DN 87] R. De Nicola. *Extensional Equivalences for Transition Systems*. Acta Informatica 24, pp. 211-237, 1987.

[EH 83] E. A. Emerson and J. Y. Halpern. *'Sometimes 'and 'not never' Revisited* : On Branching Versus Linear Time, In Proc. 10th ACM Symposium on Principles of Programming Langages. 1983.

[Fer 90] J. C. Fernandez. *An implementation of an efficient algorithm for Bisimulation Equivalence*. Science of Computer Programming. 13 (2-3): 219-236, May 1990.

[Flo 67] R. W. Floyd. *Assigning Meanings to Programs*. In Proc. Symposium on Applied Mathematics, AMS, 19 (1967).

[vG 90a] R. J. Van. Glabbeek. *Comparative Concurrency Semantics and Refinement of actions*. PhD thesis. Free University of Amsterdam, May 1990.

[vG 90b] 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.

[vGG 89] R.J. v. Glabbeek , U. Goltz. *Refinement of actions in causality based models*. In Stepwise Refinement of Distributed Systems. Models, Formalisms, Correctness, Mook, LNCS 430, pp 267-300. Springer-Verlag, May 1989.

[vGG 90] R.J. Van. Glabbeek , and U. Goltz. *Refinement of Action in Causality Based Models*. In Proc. of the Rex Workshop on Stepwise Refinement of Distributed Systems : Models, Formalism, Correctness, Mook, The Netherlands, LNCS 430, pp. 267-300, Springer Verlag, 1990.

[Gor 91] R. Gorrieri. *Refinement, Atomicity, and Transactions for Process Description Languages*. PhD Thesis. Universita di Pisa- Genova- Udine.

[GR 83] U. Goltz and W. Reisig. *The Non-sequential Behaviour of Petri Nets*. Information and Computation 57, pp. 125-147, 1983.

[Gra 81] J. Grabowski. *On Partial Langages*. Fundamenta Informaticae 4(2), 427-498, 1981.

[vGW 89] R.J. Van Glabbeek and W. P. Weijland. *Refinement in Branching Time Semantics*. In Proc. AMAST Conf., Iowa City, pp 197-201, 1989. Available as CWI Report CS-8922.

[Hen 88] M. Hennessy. *Axomatising Finite Concurrent Processes*. SIAM Journal on Computing, 17(5), pp 997- 1017, October 1988.

[HM 80] M. Hennessy and R. Milner. *On Observing nondeterminism and concurrency*. In Proceedings ICALP 80 (J.W. de Bakker and J.van Leeuwen eds.) LNCS 85, Springer Verlag, pp 299-309, 1980. A preliminary version of :

[HM 85] M. Hennessy and R. Milner. *Algebraic Laws for Nondeterminism and Concurrency*. Journal of the ACM 32(1), pp. 137-161.1985

[HS 85] M. Hennessy and C. Stirling. The power of the future perfect in program logics. Information and Control. pp 67: 23-52, 1985.

[Hoa 69] C. A. R. Hoare. *An Axiomatic Basis for Computer Programming*. Communications of the ACM, 12(10), pp. 576-583,October 1969.

[Hoa 78] C. A. R. Hoare. *Communicating Sequential Processes*. Communications of the ACM, 21(8), 666-677, August 1978.

[Hoa 84] C. A. R. Hoare. *Communicating Processes*. On the construction of programs.. (R.Mc Keag and A. Mc Naghton Eds.) Cambridge University Press, pp 229- 254, 1980.

[Hoa 85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall Int., 1985.

[JM 92] L. Jategoanker and A. Meyer. *Testing Equivalences for Petri Nets with Action Refinement*. Preliminary Report. Concur 92, Stony Brook, NY, USA 1992.

LNCS 630, pp 17-31.

[Kel 76] R.M Keller. *Formal verification of parallel programs*. Communication of the ACM, 19(7) : 371-384, July 1976.

[KP 87] S. Katz and D. Peled. *Interleaving set Temporal Logic*, In Proc.6th ACM Symp. On Distributed Computing . 1987.

[Lam 80] L. Lamport. *"Sometimes" is sometimes "Not Never"*. In Proc. 7th ACM Symp. Principles of Programming Languages. Las Vegas, pages 174- 185. January 1980.

[Lam 86] L. Lamport. On Interprocess communication. I. Distributed Comp. I: 77-85, 1986

[LRT 89] K. Lodaya and R. Ramanujam, and P. S. Thiagarajan. A Logic for Distributed Systems. n Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency. Noordwijkerhoot, LNCS 354

[LT 87] K. Lodaya and P. S. Thiagarajan. *A Modal Logic for a subclass of Event Structures*. CALP'87, LNCS 267, pp290-303. 1987.

[Maz 87] A. Mazurkiewicz. *Trace Theory*. In Petri Nets : Application and Relationships to Other Models of Concurrency. LNCS 255, pp. 279-324. Springer Verlag, 1987.

[Mil 80] R. Milner. *A Calculus of Communicating Systems*. LNCS 92. Springer Verlag, 1980.

[Mil 83] R. Milner. *Calculi for Sychrony and Asynchrony*. Theoretical Computer Science 23(3), pp 267-310, 1983.

[Mil 85] G. J. Milne. *CIRCAL and the representation of Communication, Concurrency, and Time*. TOPLAS 7(2), pp. 270-298. 1985.

[Mil 88] R. Milner. *Operational and Algebraic Semantics of Concurrent Processes.* Research Report ECS-LFCS-88-46, Lab. For Foundations of Computer Science.

[Mil 89] R. Milner. *Communication and Concurrency.* Prentice Hall Int. 1989.

[Mou 92] L. Mounier. *Méhodes de Vérification de Spécifications Comportemantales : Etude et mise en oeuvre.* Thèse de Doctorat . I.N.P. de Grenoble, France ,1992.

[MS 81] U. Montanari and C. Simonelli. *On Distinguishing Between Concurrency and Nondeterminism.* Proc. Ecole de Printemps on Concurrency and Petri Nets, Colleville sur Mer, 1980, available as Tech. Report. PFI-Cnet N°7, 1981.

[Nel 89] M. Nielsen, U. Engberg, and K. Larsen. *Partial Order Semantics for True Concurrency.* In Linear Time, Branching Time and Partial Order in Logic and Models of Concurrency. Noordwijkerhout, 1988, LNCS 354 : 523-548, 1989

[NPW81] M. Nielsen, G. D. Plotkin, and G. Winskel. *Petri nets, event structures and domains I.* Theor. Comput. Sci., 13:85-108, 1981.

[OGvG 88] E. R. Olderog, U. Goltz, and R.J . Van Glabbeek. Combining Compositionality and Concurrency. Summary of a GMD-Workshop, Königswinter, March 1988. Arbeitpapier der GMD 320, Sankt Augustin.

[Par 81] D. M. R. Park. *Concurrency and Automata on Infinite Sequences.* In Proc. 5th GI Conf. On Theor.Comp. Sci.(P. Deussen ed.), LNCS 104, pp 147-183. Springer Verlag. 1981

[Pen 88] W. Penczek. *A Temporal Logic for Event Structures.* Fundamenta Informaticae 11, pp 297-326, 1988.

[Pet 84] C. A. Petri. *Introduction to Net Theory* . In W. Brauer (Ed.), Proc. Net Theory and Application. LNCS 84. Springer Verlag, 1984.

[Plo 81] G. Plotkin. *A Structural Approach to Operational Semantics*. Technical Report DAIMI FN-19, Aarhus University, Department of Computer Science. Aarhus 1981.

[Pnu 77] A. Pnueli. *The Temporal Logic of Programs*. In Proc. 18th IEEE Symp. Foundations of Computer Science, Providence, pages 46-57, November 1977.

[Pnu 85] 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, 1985.

[Pnu 86] A. Pnueli. *Specification and development of reactive systems*. In Proc. IFIP 86, pages 845-858. North Holland. 1986.

[Pom 85] L. Pomello. *Some equivalence notions for concurrent systems. An overview*. In Advances in Petri Nets 1985, LNCS 222, pages 381-400. Springer Verlag, 1985.

[Pra 86] V. R. Pratt. Modelling *Concurrency with Partial Orders*. International Journal of Parallel Programming. 15(1),pp. 33-71, 1986.

[PT 87] R. Paige and R. E. Tarjan. *Three Partition Refinement Algorithms*. SIAM Journal on Computing, 16(6), pp 973-989, December 1987..

[QS 83] J. P Queille and J. Sifakis. *Fairness and Related Properties in Transition Systems A Temporal Logic to deal with fairness*.Acta Informatica, 19 : 195-220, 1983.

[Rei 89] W. Reisig. *Torwards a Temporal Logic of Causality a Choice in Distributed Systems*. In Linear Time, Branching Time and Partial Order in Logic and Models of Concurrency. Noordwijkerhout, 1988, LNCS 354 , 1989

[RT88] A. Rabinovich and B. A. Trakhtenbrot. *Behaviour structures and nets*. Fundamenta Informaticae, 11:357-404, 1988.

[Sch 90] Ph. Schnoebelen . *Sémantique du parallélisme et logique temporelle. Application au Langage FP2*. Thèse de Doctorat. I.N.P. de Grenoble, France, 1990.

[Sch 92] Ph. Schnoebelen . *Private Communication*

[Vog90] W. Vogler. *Bisimulation and action refinement.* In Proc. STACS'91, Hamburg, LNCS 480, pp 309-321, Springer Verlag. February 91.

[Wal 88] D. J. Walker. *Bisimulation and Divergence.* In Proc. 3rd IEEE Symp. Logic in Computer Science, Edinburgh, July 1988.

[Wei 89] W. P. Weijland. *Synchrony and Asynchrony in Process Algebra.* PhD Thesis, University of Amsterdam, June 1989.

[Win 80] G. Winskel. *Events in Computation.* Ph.D. Thesis, University of Edinburgh, CST-10-80, (1980).

[Win 87] G. Winskel. *Event Structures.* In Advances in Petri Nets 1987, (G.Rozenberhg, Ed.) LNCS 266, Springer Verlag, Heidelberg, 1987, pp. 187-223.

[Win 89] 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.

Grenoble, le 29 Septembre 1992

ECOLE DOCTORALE

Affaire suivie par
Tél : 76 57 Michèle SIMEON
N/Réf. : 4525

Objet :

AUTORISATION de SOUTENANCE

Vu les dispositions de l'arrêté du **30 MARS 1992** relatif aux Etudes Doctorales

Vu les rapports de présentation de :

Mr DARANDEAU Philippe — Prof. IRISA RENNES
Mme GOLTZ — Prof Intitut Für Informatik MARIENBURGER

Mme HADJERES Epouse CHERIEF Ferroudja

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***

Pour le Président de l'INPG
et par délégation
le Directeur de l'Ecole Doctorale
J.L. LACOUME

Résumé : Dans ce travail, nous nous intéressons essentiellement aux trois points de vue importants adoptés dans la sémantique du parallélisme: l'étude d'équivalences comportementales compatibles avec le raffinement d'actions, les modèles dits de "vrai parallélisme", et les liens qui existent entre les logiques modales et les équivalences comportementales.

Parmi les équivalences de bisimulation qui traitent les situations où certaines actions sont invisibles ou non observables de l'extérieur, la τ-bisimulation ou équivalence observationnelle de Milner et la η-bisimulation ne sont pas compatibles avec le raffinement d'actions. La bisimulation de branchement et la Δ-bisimulation sont compatibles avec le raffinement d'actions. Ceci nous a suggéré la recherche d'équivalences compatibles avec le raffinement et contenues dans la τ-bisimulation ou la η-bisimulation

Aussi, nous avons défini la notion de bisimulation avant arrière sur les structures d'événements premières qui constituent un modèle de base pour le vrai parallèlisme. Nous avons généralisé cette nouvelle définition à la step, partial word et pomset bisimulation avant arrière. Nous avons comparé ces nouvelles équivalences aux autres équivalences de bisimulation sur les structures d'événements premières. Il s'est avéré essentiellement que la pomset et la partial word bisimulations avant arrière coïncident avec la history preserving bisimulation. Partant de ce résultat, nous avons proposé une logique l_{bf}, adéquate à la history preserving bisimulation. Nous avons aussi montré que la loqique l_p de [DNF 90] caractérise elle aussi la history preserving bisimulation.

Mots Clés: Sémantique du Parallélisme, Bisimulation, Actions invisibles, Logique modale, Logique temporelle, Systèmes de transitions, Structures d'événements premières.

Abstract : In this work, we are interested in the three main viewpoints used for the specification of parallel programs: the study of behavioural equivalences that are well behaved against refinement of actions, the so called truly concurrent models, and the link between modal logics and behavioural equivalences. Among all the equivalences that deal with silent actions, τ-bisimulation or Milner's observational equivalence and η-bisimulation are not well behaved w.r.t. refinement of actions while branching and Δ-bisimulations are. This led us looking for the coarsest aquivalences that are well behaved w.r.t refinement and contained in τ-bisimulation or η-bisimulation.

We also propose a definition of back and forth bisimulation on prime event structures. We show that our proposal can be adapted in a uniform way to yield step, partial word, and pomset back and forth bisimulations. It turns out that partial word and pomset back and forth bisimulations both coïncide with history preserving bisimulation.Using this new result, we propose a logic l_{bf}, that narurally characterizes history preserving bisimulation. We also prove that the l_p logic of [DNF 90] also characterizes history preserving bisimulation.

Key Words: Semantics of Concurrency, Bisimulation, Silent actions, Modal logic, Temporal logic, Transition systems, Prime event structures.