Composition Parallèle et Synchronisation de Systèmes à Evénements Discrets

Wojciech Frączak

Remerciements

Je remercie vivement Jos Baeten, Professeur à l'Université Technologique d'Eindhoven, de l'intérêt qu'il a manifesté pour ce travail et de la compétence qu'il m'a apportée. Ses critiques constructives et suggestions m'ont permis d'améliorer la qualité de la version finale de cette thèse, dont il m'a fait l'honneur d'être rapporteur.

Je suis très reconnaissant à Raymond Devillers, Professeur à l'Université Libre de Bruxelles d'avoir accepté d'être mon rapporteur et du temps qu'il a consacré à la lecture très attentive et patiente des versions précédentes de ce travail. Grâce à ses conseils et suggestions les aspects techniques et rédactionnels de cette thèse ont beaucoup gagné en qualité.

Je remercie Eike Best, Professeur à l'Université de Hildesheim, d'avoir accepté de faire partie du jury. Les contacts que j'ai pu avoir avec lui durant ma thèse m'ont été très précieux.

Je remercie également Laurence Puel et Antoine Petit de l'honneur qu'ils m'ont fait de participer au jury.

Ce travail a été effectué sous la direction d'Elisabeth Pelz. Durant toutes les années de notre collaboration, j'ai tout particulièrement apprécié la confiance qu'elle m'a témoignée, ses conseils, ses encouragements et la liberté qu'elle m'a toujours laissée. C'est aussi grâce à elle que j'ai pu bénéficier de contacts fructueux avec d'autres chercheurs, notamment dans le cadre du projet européen Caliban. Qu'elle soit assurée de toute ma reconnaissance.

Merci à Jean Fanchon, ami et partenaire de discussion précieux, qui a toujours su m'écouter en partageant avec moi son savoir et son enthousiasme pour la recherche. C'est lors de nos nombreux échanges que certaines de mes idées ont pu naître ou se cristalliser.

Je remercie chaleureusement Marie-Rose Gonçalves qui a bien voulu relire intégralement cette thèse et qui m'a apporté son aide dans le travail de rédaction.

Je remercie individuellement tous les collègues et amis que j'ai au LRI, et en particulier : Christophe, Fabrice, Hanna, Isabelle, Robert, Serena, Sylvie pour avoir rendu ces années agréables.

Enfin, un grand merci à ma famille, à ma femme Lidia pour ses encouragements et son soutien sans faille et à mes parents qui ont cru en moi.

Contents

Introduction

Notre travail concerne la spécification et la modélisation de *systèmes réactifs*. Un système réactif est, d'un point de vue abstrait, un objet capable d'interagir avec son *environnement* par un ensemble de *primitives de communication*. Des exemples de tels systèmes peuvent être :

1. une simple machine à café, ou un distributeur de billets, qui interagit avec un client via un ensemble de signaux bien précis : introduire la carte ou mettre une pièce, composer le code ou appuyer sur un bouton pour faire son choix, etc ;

2. un micro-processeur qui interagit avec une mémoire et des portes d'entrées/-sorties par des signaux électriques ;

3. un ordinateur (avec son système d'exploitation) qui communique avec l'environnement via des périphériques tels qu'un terminal, une imprimante, ou d'autres.

En fait, n'importe quel objet d'utilisation quotidienne peut être vu comme un système réactif.

La spécification d'un système réactif a pour but de caractériser son *comportement* vis-à-vis de l'environnement, qui lui-même est traité comme un système réactif. Ceci dit, il faut maintenant préciser par quels moyens les systèmes peuvent interagir. Une approche habituellement adoptée est de considérer que les systèmes communiquent via des événements atomiques, discrets dans le temps et dans l'espace, événements qui représentent les (ou résultent des) changements d'états des systèmes.

Bien évidemment, la spécification d'un système réactif doit être précise et non ambiguë, elle doit donc être effectuée dans un *modèle mathématique* précis. Là où l'on utilise les mathématiques pour des disciplines scientifiques ou d'ingénierie, un modèle unique ne peut généralement pas représenter le phénomène d'une façon absolument complète. Tel est le cas dans la modélisation des systèmes réactifs. Parmi les modèles les plus répandus pour les systèmes réactifs, on trouve :

Systèmes de transitions. Les systèmes de transitions constituent un des modèles les plus simples pour les systèmes réactifs à événements discrets. Un système de transitions est un graphe orienté dont les noeuds représentent des états et les arcs représentent les transitions possibles entre les états. L'intérêt principal de ces systèmes vient de leur simplicité et des méthodes d'analyse qui existent sur ces modèles, permettant la preuve de propriétés sur le comportement du système spécifié. Les systèmes de transitions permettent de décrire des systèmes *non–déterministes*, c'est-à-dire avec plusieurs transitions possibles à partir d'un même état, mais la modélisation de systèmes parallèles se voit traditionnellement limitée à l'approche dite de l'*entrelacement*[1], où le parallélisme de deux événements est interprété comme un choix indéterministe entre les deux mises en séquence possibles des événements. Il existe cependant des extensions des systèmes de transitions, comme par exemple les "Systèmes de Transitions Concurrents" [Sta86, Sta87] (ang. *Concurrent Transition Systems*), "Systèmes de Transitions Asynchrones" [Bed88] (ang. *Asynchronous Transition Systems*) et "Systèmes de Transitions Localisés" [Fan95], où une information complémentaire sur la concurrence/indépendance/localisation des transitions est ajoutée.

Réseaux de Petri. L'avantage des réseaux de Petri (proposés pour la première fois dans [Pet62]) par rapport aux systèmes de transitions est qu'ils permettent d'exprimer le parallélisme et la distribution des événements de façon naturelle, en plus du non-déterminisme. De manière générale, un réseau de Petri, est constitué de *places* et de *transitions* reliées entre eux, et d'une distribution de *jetons* sur les places. Les places et les transitions représentent respectivement les états et les événements potentiels du système modélisé. L'état global du système est représenté par un "marquage" de places (distribution de *jetons* dans les places). Une place représente un état local (une partie de l'état global du système) qui devient effectif (actif) si un jeton s'y trouve. Un changement d'état s'effectue par le *franchissement* d'une des transitions en n'impliquant en général qu'une partie des jetons qui se trouvent dans les places reliées à la transition. Ainsi, deux transitions qui impliquent deux parties disjointes du réseau seront considérées comme indépendantes, et si l'état du système (distribution de jetons) permet le franchissement des deux transitions, alors elles pourront être franchies en parallèle (ou concurremment).

La théorie des réseaux de Petri offre une vaste variété de classes de réseaux, comme les *réseaux élémentaires*, les *réseaux Place/Transition*, ou les *réseaux de haut niveau.*

Algèbres de processus. Une algèbre de processus peut être vue comme un

[1] De récents travaux de Rob van Glabbeek sur "*History Preserving Process Graphs*" montrent cependant qu'un régime supplémentaire dans la construction et l'interprétation des systèmes de transitions permet de sortir du cadre de la modélisation du parallélisme par l'entrelacement.

> langage abstrait de spécification des comportements de systèmes réactifs. Comme tout langage, une algèbre de processus est munie d'une syntaxe (signature) bien précise et, généralement, très simple.
>
> R. Milner [Mil78], C.A.R Hoare [Hoa78] et P.E. Lauer [LTS79] semblent être les pionniers dans l'élaboration de tels langages. Beaucoup ont suivi leurs approches, ce qui en fait a donné naissance à une nouvelle branche de l'informatique théorique, que l'on appelle aujourd'hui "algèbre de processus", et qui consiste en l'étude des systèmes réactifs par rapport à un ensemble d'opérateurs de base qui servent à les définir. Le terme "algèbre de processus" vient du fait que les opérateurs sont vus comme des opérations d'une algèbre dont le domaine est un ensemble de comportements appelés *processus*.

Un processus définit les relations qui existent entre les événements (occurrences d'actions), c'est-à-dire la *dépendance causale*, la *concurrence* ou le *conflit*. Dans cette thèse, nous introduisons un nouveau type de dépendance entre les événements, appelée *simultanéité*. Deux événements sont en *dépendance de simultanéité* si aucun d'eux ne peut se produire sans l'autre. Cette relation nous mène à élargir la notion d'événement et à le considérer comme un ensemble d'actions s'exécutant simultanément, et non plus seulement comme une occurrence d'une seule action. Un tel événement sera appelé *multi-action*. Nous reconsidérons plusieurs notions élaborées pour les événements simples (occurrence d'une seule action) dans le contexte des multi-actions et nous étudions l'impact de l'introduction des multi-actions dans les modèles de systèmes réactifs concurrents. Le choix de multi-actions comme événements de base sera justifié dans les différents chapitres.

Dans le cas des systèmes réactifs, l'opération la plus importante est l'opération de composition, qui permet de regrouper des sous-systèmes en un nouveau système plus complexe. Il existe plusieurs façons de définir la composition de systèmes réactifs, non seulement selon le choix du modèle mais aussi selon le choix des primitives de communication, c'est-à-dire le choix des actions. Les actions étiquettent les transitions d'un système de transitions ou d'un réseau de Petri, ou elles constituent l'ensemble des *objets de base* dans une algèbre de processus.

La question se pose alors de déterminer s'il existe une approche générale qui soit capable d'unifier des approches déjà existantes, et qui en même temps soit simple et intuitive. Nous proposons une approche basée sur les *structures de synchronisation*, une classe de structures relationnelles sur des *actions atomiques*, qui a pour objectif d'uniformiser les définitions des opérateurs liés à la composition. Nous examinons aussi la possibilité d'extension d'une structure de synchronisation aux *multi-actions*.

Une application des structures de synchronisation et de leurs propriétés algébriques est présentée dans le contexte des systèmes de transitions, où nous analysons comment on peut composer de tels systèmes. Deux cas de figures sont analysés : les systèmes de transitions avec étiquetage par actions atomiques, et les systèmes de transitions étiquetés par multi-actions (ang. *Multi-labeled Transition Systems* ou `MTS`).

Une autre application des notions liées aux structures de synchronisation est proposée pour une re-formulation intéressante de deux opérateurs de l'Algèbre des Petri-Box : la *synchronisation* et la *restriction*, en les rendant paramétrables par des structures de synchronisation quelconques. Cette extension élargit considérablement la puissance expressive de l'algèbre sans perte de ses propriétés générales.

Cette thèse est composée de trois parties :

I **Cadre général et notions de base**

Dans la première partie, nous introduisons les *systèmes de transitions*, les *réseaux de Petri* et les *algèbres de processus*. Les définitions introduites sont largement motivées par la suite de la thèse, et elles ne doivent donc pas être considérées comme une présentation du domaine, mais plutôt comme une présentation et une définition des notations utilisées dans la thèse. Néanmoins, dans les Section I.3.1 et I.3.2, on trouvera une présentation générale des algèbres de processus.

II **Composition Parallèle et Synchronisation**

Dans cette partie, nous introduisons les *structures de synchronisation*, une classe de structures relationnelles sur des *actions atomiques*. Nous analysons aussi la possibilité d'extension d'une structure de synchronisation aux *multi-actions*. Ensuite, nous présentons une application des structures de synchronisation et de leurs propriétés algébriques dans le contexte des systèmes de transitions (Chapitre II.5) et dans le contexte des réseaux de Petri (Chapitre II.6).

III **Algèbre de Processus sur des Multi-actions**

Dans la dernière partie, nous proposons une nouvelle algèbre de processus (appelée MAPA : ang. *Multi-Action Process Algebra*) permettant d'intégrer la dépendance de simultanéité grâce à l'utilisation des multi-actions. La syntaxe de MAPA inclut seulement trois opérateurs : le *préfixage*, la *composition parallèle* et la *restriction*. Les principales caractéristiques de MAPA sont les suivantes : premièrement, un processus peut être *préfixé* par une multi-action ; deuxièmement, la *composition* de deux processus est accompagnée par une spécification explicite des restrictions de communication, formalisée par la notion de *canaux de communication*.

Un canal de communication est caractérisé par le type et le nombre d'actions qu'il peut véhiculer simultanément. Deux processus connectés par un canal peuvent seulement communiquer via des multi-actions permises par le canal. Nous définissons des opérations sur les canaux qui, par exemple, permettent de renverser la direction d'un canal, ou de grouper deux canaux en un seul.

Une sémantique opérationnelle de MAPA est proposée de manière standard via des "règles de dérivation" à la Plotkin. Dans le cas de MAPA, où le "choix" n'existe pas en tant qu'opération primitive, il est possible de prendre en compte différents ensembles de règles de dérivation, chaque ensemble définissant une sémantique opérationnelle légèrement différente. Nous analysons quelques exemples de telles sémantiques et nous en choisissons une qui, selon les critères proposés, est "la plus opérationnelle".

En plus des équivalences standards, telles que l'*isomorphisme de graphes de processus*, l'*équivalence forte*, l'*équivalence faible* des processus, nous proposons une nouvelle équivalence appelée l'*équivalence sémantique*, inspirée de l'*équivalence par échec* de Hoare mais caractérisée de manière différente.

L'absence du *choix* comme opérateur primitif garantit que l'équivalence faible est une congruence. Nous démontrons que, par rapport à l'équivalence faible, toutes les sémantiques opérationnelles analysées sont équivalentes.

A la fin de la dernière partie, nous montrons que MAPA possède la "puissance expressive la plus générale" dans le sens que tout *graphe de processus récursivement énumérable* peut être décrit par un terme de notre algèbre, modulo l'équivalence faible.

L'introduction des multi-actions permet d'obtenir une algèbre de processus extrêmement puissante, avec toutes les propriétés habituelles, à partir de trois opérations primitives seulement.

Part I

Cadre général et notions de base

Chapter 1

Systèmes de Transitions

Les systèmes de transitions constituent un des modèles les plus simples des systèmes réactifs à événements discrets. Un système de transitions est un graphe orienté dont les noeuds représentent des états et les arcs représentent les transitions possibles entre les états. L'intérêt principal des systèmes de transitions vient de leur simplicité et des méthodes d'analyse qui existent sur ces systèmes et qui permettent la preuve de propriétés sur le comportement du système modélisé [Arn92].

Définition I.1 Un *système de transitions (étiqueté)* est un triplet $(\mathtt{St}, \mathtt{Lab}, \rightarrow)$, où $\mathtt{St}$ et $\mathtt{Lab}$ sont respectivement deux ensembles d'*états* et d'*étiquettes*, et $\rightarrow\ \subseteq \mathtt{St} \times \mathtt{Lab} \times \mathtt{St}$ est un ensemble de transitions étiquetées. Une transition (s, α, s') est habituellement notée $s \xrightarrow{\alpha} s'$. □

Intuitivement, une transition $s \xrightarrow{\alpha} s'$ signifie que le système se trouvant dans l'état s peut évoluer vers l'état s', sous une impulsion externe, représentée par l'étiquette α. Un exemple de système de transitions est fourni dans la Figure 1.1. Ce système est composé de sept états $s_1, \ldots, s_7$ et de plusieurs transitions éti-

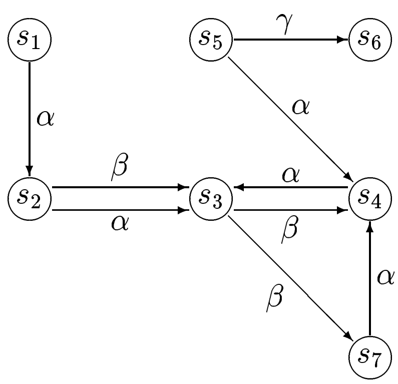
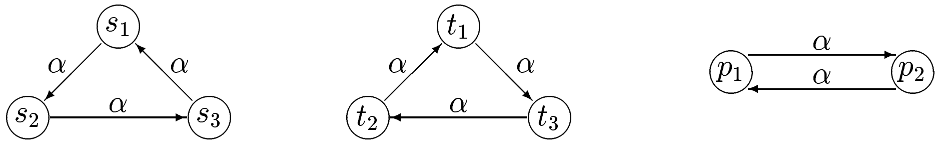


Figure 1.1: Un système de transitions.

quetées par α, β ou γ. Notons que tout état n'est pas nécessairement accessible d'un autre état. Par exemple, les états s_5 et s_6 ne pourront pas être atteints si le système démarre à l'état s_1. Le système présenté dans la Figure 1.1, s'il se trouve par exemple dans l'état s_1, ne peut évoluer que vers l'état s_2 en exécutant l'action α . Après cela, le système peut choisir entre l'action α ou β, mais l'état d'arrivée ne dépendra pas de ce choix, et sera toujours s_3. Une fois dans l'état s_3, le système ne pourra exécuter que l'action β et arriver soit à l'état s_4 soit à l'état s_7, etc.

Selon l'intuition présentée dans l'introduction, dans le comportement d'un système réactif, seules les (occurrences d') actions sont considérées, les états jouant seulement un rôle auxiliaire. On peut se demander alors, si deux systèmes de transitions différents peuvent avoir le même comportement. Prenons, par exemple, les systèmes de transitions $\mathtt{TS}_1$, $\mathtt{TS}_2$ et $\mathtt{TS}_3$ de la Figure 1.2.



$\mathtt{TS}_1$ $\mathtt{TS}_2$ $\mathtt{TS}_3$

Figure 1.2: Systèmes de transitions à même comportement.

Bien que ces trois systèmes soient différents ($\mathtt{TS}_1$ diffère de $\mathtt{TS}_2$ par les noms des états), leur comportement semble être le même. Dans tous les cas, il consiste en l'exécution de l'action α un nombre arbitraire de fois, indépendamment de l'état de départ.

Dans la section suivante, nous allons présenter quelques approches qui tentent de trouver de bons critères pour la comparaison de systèmes vis-à-vis de leurs comportements.

Définition I.2 Un *graphe de processus* est un quadruple $(\mathtt{St}, \mathtt{Lab}, \rightarrow, s)$, où $(\mathtt{St}, \mathtt{Lab}, \rightarrow)$ est un système de transitions et $s \in \mathtt{St}$ est un *état initial.* De plus, nous exigeons que : $\forall x \in \mathtt{St} \quad (s, x) \in \stackrel{*}{\rightarrow}$, où $\stackrel{*}{\rightarrow}$ dénote la clôture réflexive et transitive de la relation $\{(x, y) \in \mathtt{St} \times \mathtt{St} \mid \exists \alpha \in \mathtt{Lab} \ : \ x \stackrel{\alpha}{\rightarrow} y\}$. $\square$

Un *graphe de processus* est un système de transitions avec un état sélectionné, appelé *état initial*, qui représente l'état actuel du système modélisé. De plus, pour des raisons techniques, nous exigeons que tout état d'un graphe de processus soit accessible à partir de cet état initial. On ne demande pas, par contre, que l'ensemble des étiquettes des transitions effectivement présentes dans le graphe de processus soit égal à $\mathtt{Lab}$.

Soit $\mathtt{TS} = (\mathtt{St}, \mathtt{Lab}, \rightarrow)$ un système de transitions. A tout état $x \in \mathtt{St}$, nous associons un graphe de processus $\mathtt{graph}(x)$ défini par :

$$\mathtt{graph}(x) \stackrel{\text{def}}{=} (\mathtt{St}_x, \mathtt{Lab}, \rightarrow_x, x) \ ,$$

où $\mathtt{St}_x \stackrel{\mathrm{def}}{=} \{y \in \mathtt{St} \mid x \stackrel{*}{\rightarrow} y\}$, et où $\rightarrow_x$ est la relation de transition ($\rightarrow$) restreinte aux états $\mathtt{St}_x$. Intuitivement, $\mathtt{graph}(x)$ décrit la partie du TS, accessible à partir de l'état x.

1.1 Relations d'équivalence

Dans la mesure où nous nous intéressons aux comportements des systèmes réactifs, la représentation d'un comportement (processus) par un graphe de processus paraît trop détaillée. C'est pour cela que plusieurs notions d'équivalence ont été proposées (cf. [vG90, NH84, Nic87, Pom86, AD93]). Nous allons en présenter les plus connues, avant de les comparer entre elles. Pour uniformiser les définitions, nous supposerons toujours que deux processus P et Q sont représentés par deux graphes de processus (pas forcément disjoints) d'un même système de transitions. Dans ce contexte, une équivalence entre les processus peut être ramenée à une équivalence entre les états d'un système de transitions, qui sont en fait les états initiaux respectifs des graphes de processus considérés.

1.1.1 Isomorphisme ($\equiv$)

L'équivalence entre deux états s_1 et s_2 d'un système de transitions TS par *isomorphisme de graphes de processus* (ou simplement par *isomorphisme*) signifie que les graphes de processus accessibles de s_1 et de s_2 peuvent différer seulement par l'identité de leurs états.

Définition I.3 Deux graphes de processus, $G_1 = (\mathtt{St}_1, \mathtt{Lab}_1, \rightarrow_1, s_1)$ et $G_2 = (\mathtt{St}_2, \mathtt{Lab}_2, \rightarrow_2, s_2)$, sont *isomorphes* s'il existe une bijection $\varphi : \mathtt{St}_1 \mapsto \mathtt{St}_2$, telle que :

1. $\varphi(s_1) = s_2$.

2. $\forall p, q \in \mathtt{St}_1 \;\; p \stackrel{\alpha}{\rightarrow}_1 q \Leftrightarrow \varphi(p) \stackrel{\alpha}{\rightarrow}_2 \varphi(q)$.

Deux états p et q d'un système de transitions étiqueté TS sont *isomorphes*, ce qui sera noté $p \equiv q$, si et seulement si les graphes de processus $\mathtt{graph}(p)$ et $\mathtt{graph}(q)$ sont isomorphes. □

Dans le système de transitions $\mathtt{TS}_1$ de la Figure 1.2, tous les états sont *isomorphes* : $s_1 \equiv s_2 \equiv s_3$. La même figure montre que cette équivalence distingue des systèmes que l'on ne voudrait peut-être pas distinguer : dans le système TS, obtenu par union disjointe de $\mathtt{TS}_1$, $\mathtt{TS}_2$ et $\mathtt{TS}_3$, les états s_1 et p_1 ne sont pas *isomorphes*.

1.1.2 Equivalence par traces ($\simeq_t$)

Soient $\texttt{TS} = (\texttt{St}, \texttt{Lab}, \rightarrow)$ un système de transitions étiqueté et $p \in \texttt{St}$ un état. Nous définissons l'*ensemble des traces*[1] de l'état p, noté $\texttt{traces}(p)$, comme le sous-ensemble suivant de mots finis sur $\texttt{Lab}$ ($traces(p) \subseteq \texttt{Lab}^*$) : Un mot $\omega \in \texttt{Lab}^*$ est une trace de p ($\omega \in \texttt{traces}(p)$), si une des conditions suivantes est vérifiée :

1. $\omega = \varepsilon$ (mot vide),
2. si $\omega = \alpha_1\alpha_2 \ldots \alpha_n$ alors il existe des états $p_1, \ldots, p_n \in \texttt{St}$, tels que :

$$p \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} p_n \quad \text{(noté aussi :} \quad p \xrightarrow{\ \omega\ } p_n)$$

Définition I.4 Deux états p et q d'un système de transitions $\texttt{TS}$ sont *équivalents par traces*, noté $p \simeq_t q$, si et seulement si $\texttt{traces}(p) = \texttt{traces}(q)$. □

L'équivalence par traces est basée sur l'idée que deux systèmes doivent être considérés comme équivalents s'ils permettent le même ensemble d'observations, où une observation est constituée de la suite d'actions qui peuvent être exécutées par le système. Cette définition rend équivalents les trois systèmes de la Figure 1.2, mais également les états s_1, s_2 et s_3 du système de transitions de la Figure 1.3, même si à partir de l'état s_3 un blocage supplémentaire (état s_4) est possible par rapport aux états s_1 et s_2.

L'exemple montre que les cas de bloquages ne sont pas bien traités par l'équivalence par trace. C'est pour cela que l'équivalence par traces est rarement considérée comme la sémantique adéquate pour des systèmes réactifs.

Néanmoins, l'équivalence par traces joue un rôle important dans le domaine de la modélisation des systèmes réactifs, puisqu'il existe plusieurs résultats très généraux[2] qui utilisent cette équivalence dans la formulation et/ou dans les preuves.

1.1.3 Equivalence forte ($\simeq$)

Définition I.5 Soit $\texttt{TS} = (\texttt{St}, \texttt{Lab}, \rightarrow)$ un système de transitions étiqueté. Une *simulation forte* S dans $\texttt{TS}$ est une relation binaire sur les états ($S \subseteq \texttt{St} \times \texttt{St}$), telle que $\forall p, q \in \texttt{St}$, $\forall \alpha \in \texttt{Lab}$, si $(p, q) \in S$ et $p \xrightarrow{\alpha} p'$ alors :

$$\exists q' \in \texttt{St} : \ q \xrightarrow{\alpha} q' \wedge (p', q') \in S \ .$$

Si S est symétrique, alors on l'appelle *bisimulation forte*. □

[1] Attention : les traces dont il s'agit ici ne sont pas les "traces" au sens de Mazurkiewicz.

[2] Par exemple, le résultat de F. Vaandrager [Vaa92], qui prouve qu'il n'existe pas une algèbre de processus avec une sémantique opérationnelle "effective" capable de décrire tout graphe de processus récursif, modulo l'équivalence par trace, signifie aussi que cela n'est pas possible pour toute équivalence inclues dans l'équivalence par traces, telle que par exemple, l'*équivalence forte*.

La plus grande bisimulation forte, appelée *équivalence forte*, existe (voir eg. [Mil89]); c'est en fait l'union de toutes les bisimulations fortes :

$$\simeq \stackrel{\text{def}}{=} \bigcup\{S \mid S \text{ est une bisimulation forte}\}$$

L'*équivalence forte*, $\simeq$, est une relation symétrique, réflexive et transitive ; c'est donc bien une relation d'équivalence.

Dans le système de transitions de la Figure 1.3, les états s_1 et s_2 sont en relation d'*équivalence forte*. L'état s_3 est en équivalence par traces avec s_1 et s_2 mais pas en équivalence forte.

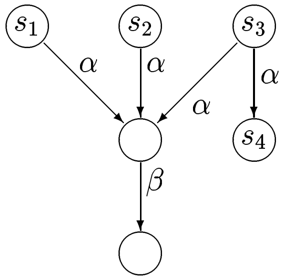


Figure 1.3: Equivalence forte.

1.1.4 Equivalence faible ($\approx$)

L'équivalence faible est une extension de l'équivalence forte. Cette notion permet de prendre en compte les actions qui sont internes au système et ne peuvent pas être observées. Donc, lors de la comparaison, on voudrait que ce type d'actions ne soit pas pris en compte. Une action interne (invisible) se distingue par son étiquette, traditionnellement notée τ.

Pour donner la définition de la simulation faible, nous devons définir la relation $\overset{*}{\rightsquigarrow}$ sur les états d'un système de transition $\mathtt{TS} = (\mathtt{St}, \mathtt{Lab}, \rightarrow)$, qui exprime la possibilité d'évolution interne (invisible) d'un système, et ensuite la relation $\Longrightarrow \subseteq \mathtt{St} \times \mathtt{Lab} \times \mathtt{St}$.

Soit $\mathtt{TS} = (\mathtt{St}, \mathtt{Lab}, \rightarrow)$ un système de transitions étiqueté, et $\tau \in \mathtt{Lab}$ l'étiquette des actions internes. Nous définissons :

1. $\overset{*}{\rightsquigarrow}$ est la clôture réflexive–transitive de $\{(p, q) \in \mathtt{St} \times \mathtt{St} \mid p \overset{\tau}{\rightarrow} q\}$,

2. $p \overset{\alpha}{\Longrightarrow} q$ ssi $\begin{cases} \exists p', q' \in \mathtt{St} \;\; p \overset{*}{\rightsquigarrow} p' \wedge p' \overset{\alpha}{\rightarrow} q' \wedge q' \overset{*}{\rightsquigarrow} q & \text{pour } \alpha \neq \tau \\ p \overset{*}{\rightsquigarrow} q & \text{pour } \alpha = \tau \end{cases}$

Définition I.6 Soit $\mathtt{TS} = (\mathtt{St}, \mathtt{Lab}, \rightarrow)$ un système de transitions, où toute action invisible est étiquetée par $\tau \in \mathtt{Lab}$. Une *simulation faible* S dans $\mathtt{TS}$

est une relation sur les états, $S \subseteq \mathtt{St} \times \mathtt{St}$, telle que $\forall p, q \in \mathtt{St}$, $\forall \alpha \in \mathtt{Lab}$, si $(p, q) \in S$ et $p \overset{\alpha}{\rightarrow} p'$ alors :

$$\exists q' \in \mathtt{St} : \ q \overset{\alpha}{\Longrightarrow} q' \wedge (p', q') \in S \ .$$

Si S est symétrique, alors on l'appelle *bisimulation faible.* $\square$

La plus grande bisimulation faible, appelée, *équivalence faible*, existe (voir eg. [Mil89]) ; c'est l'union de toutes les bisimulations faibles :

$$\simeq \ \stackrel{\text{def}}{=} \bigcup \{S \mid S \text{ est une bisimulation faible}\}$$

L'*équivalence faible*, $\approx$, est symétrique, réflexive et transitive ; c'est donc bien une relation d'équivalence.

1.1.5 Equivalence par échec ($\approx_f$)

L'équivalence par échec suppose que la seule manière d'analyser les processus est d'interagir avec eux. Deux processus sont en équivalence par échec s'ils peuvent (s'ils risquent de) se "bloquer" exactement dans les même situations. Par "bloquer" nous comprenons une situation où le processus ne peut accepter aucune des actions proposées par l'environnement.

Soit p un état d'un système de transitions étiqueté $\mathtt{TS} = (\mathtt{St}, \mathtt{Lab}, \rightarrow)$ et $\tau \in \mathtt{Lab}$ dénote les actions invisibles. Un ensemble fini $X \subseteq \mathtt{Lab} \setminus \{\tau\}$ est un *ensemble de refus* de p, dénoté par $X \in \mathtt{Refusals}(p)$, si et seulement si, pour tout $\alpha \in X$, il n'existe pas d'état q tel que $p \overset{\alpha}{\Longrightarrow} q$.

Soit $\mathtt{TS} = (\mathtt{St}, \mathtt{Lab}, \rightarrow)$ un système de transitions étiqueté. Une paire $\langle \omega, X \rangle$ composée d'un mot ω sur $\mathtt{Lab} \setminus \{\tau\}$ et d'un ensemble $X \subseteq \mathtt{Lab} \setminus \{\tau\}$ est appelée *paire d'échec* d'un état p de $\mathtt{St}$, si et seulement si il existe un état $q \in \mathtt{St}$ et un mot ω' sur $\mathtt{Lab}$, tels que :

1. $\omega = \omega'|_{\mathtt{Lab} \setminus \{\tau\}}$, c.-à-d. que la projection du mot ω' sur l'alphabet $\mathtt{Lab} \setminus \{\tau\}$ donne le mot ω,
2. $p \xrightarrow{\omega'} q$,
3. $X \in \mathtt{Refusals}(q)$.

Si nous notons l'existence d'une suite $p \overset{\tau}{\Longrightarrow} p_0 \overset{\alpha_1}{\Longrightarrow} p_1 \overset{\alpha_2}{\Longrightarrow} \ldots \overset{\alpha_n}{\Longrightarrow} p_n$ par $p \overset{\omega}{\Longrightarrow} q$, où $q = p_n$ et $\omega = \alpha_1 \alpha_2 \ldots \alpha_n$, alors la définition d'une paire d'échec d'un état p, peut s'exprimer de la manière suivante : $\langle \omega, X \rangle$ est une paire d'échec de p si et seulement si il existe un état q tel que $X \in \mathtt{Refusals}(q)$ et $p \overset{\omega}{\Longrightarrow} q$.

Définition I.7 Deux états p et q d'un système de transitions $\mathtt{TS} = (\mathtt{St}, \mathtt{Lab}, \rightarrow)$ sont *équivalents par échec*, noté $p \approx_f q$, si et seulement si les ensembles de paires d'échec de ces deux états sont égaux. $\square$

Dans la Figure 1.4 les états s_1, s_2 et s_3 sont équivalents par échec, mais ne sont ni dans l'équivalence forte ni dans l'équivalence faible : $s_1 \approx_f s_2 \approx_f s_3$, $s_1 \not\approx s_2$, $s_1 \not\approx s_3$ et $s_2 \not\approx s_3$.

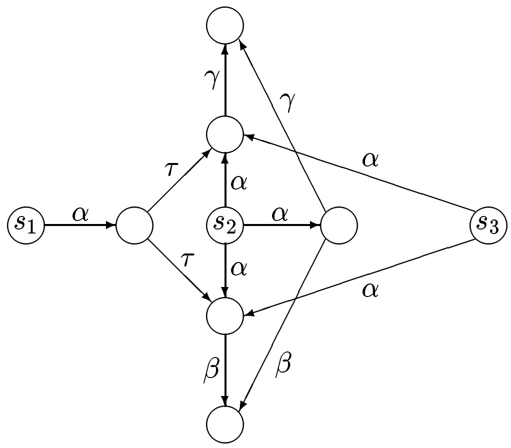


Figure 1.4: Equivalence par échec.

1.1.6 Hiérarchie des relations d'équivalences

La définition de l'équivalence par échec présentée ici est celle proposée dans [BHR84], où l'on se limite aux ensembles de refus finis. On peut se demander alors ce qui change si l'on ne se restreint pas à des ensembles finis. Notons par $\approx_f^{\mathcal{C}}$ l'équivalence par échec dont l'ensemble de refus appartient à une classe $\mathcal{C}$ de sous-ensembles de $\mathtt{Lab}$. Par exemple, si par ∞ nous notons la classe de tous les sous-ensembles de $\mathtt{Lab}$, alors $\approx_f^{\infty}$ dénote l'équivalence par échec, où l'ensemble de refus est un sous-ensemble quelconque de $\mathtt{Lab}$. Nous remarquons la propriété suivante, qui découle directement de la définition de l'équivalence par échec. Soient $\mathcal{A}$, $\mathcal{B}$ deux classes de sous-ensembles de $\mathtt{Lab}$:

$$\mathcal{A} \subseteq \mathcal{B} \;\Rightarrow\; \approx_f^{\mathcal{B}} \;\subseteq\; \approx_f^{\mathcal{A}} \;.$$

Considérons les classes de sous-ensembles de $\mathtt{Lab}$ suivantes : la classe des ensembles finis, notée $\mathcal{F}$, la classe des ensembles récursifs, notée $\mathcal{R}$, la classe des ensembles récursivement énumérables, notée $\mathcal{E}$, et la classe de tous les sous-ensembles de $\mathtt{Lab}$, notée ∞. Bien évidemment, on a les inclusions suivantes : $\mathcal{F} \subseteq \mathcal{R} \subseteq \mathcal{E} \subseteq \infty$, et donc la proposition suivante est trivialement vraie.

Proposition I.1 *Soit* $\mathtt{TS} = (\mathtt{St}, \mathtt{Lab}, \rightarrow)$ *un système de transitions. Pour tous les états* $p, q \in \mathtt{St}$ *nous avons :*

1. $p \approx_f^{\infty} q \;\Rightarrow\; p \approx_f^{\mathcal{E}} q$
2. $p \approx_f^{\mathcal{E}} q \;\Rightarrow\; p \approx_f^{\mathcal{R}} q$
3. $p \approx_f^{\mathcal{R}} q \;\Rightarrow\; p \approx_f^{\mathcal{F}} q$ *(Remarquons que $\approx_f^{\mathcal{F}}$ coïncide avec $\approx_f$.)*

Dans la proposition ci-dessous, nous rappelons des résultats bien connus sur la hiérarchie entre les relations d'équivalence définies ici.

Proposition I.2 *Soit* $\mathtt{TS} = (\mathtt{St}, \mathtt{Lab}, \rightarrow)$ *un système de transitions. Pour tous les états* $p, q \in \mathtt{St}$ *nous avons :*

1. $p \equiv q \Rightarrow p \simeq q$
2. $p \simeq q \Rightarrow p \simeq_t q$
3. $p \simeq q \Rightarrow p \approx q$
4. $p \approx q \Rightarrow p \approx_f^\infty q$

Preuve. Nous prouvons ici seulement le point 4. Pour les autres preuves, se rapporter à [vG90] et à [Nic87].

4. Il suffit de montrer que si $p \approx q$ et $\langle \omega, X \rangle$ est une paire de refus de p, alors c'est aussi une paire de refus de q.

   Si $p \approx q$, alors les ensembles $\{\alpha \in \mathtt{Lab} \mid p \stackrel{\alpha}{\Longrightarrow} p'\}$ et $\{\alpha \in \mathtt{Lab} \mid q \stackrel{\alpha}{\Longrightarrow} q'\}$ sont égaux, donc $\mathtt{Refusals}(p) = \mathtt{Refusals}(q)$.

   Si $\langle \omega, X \rangle$ est une paire d'échec de p, alors il existe p' tel que $X \in \mathtt{Refusals}(p')$ et $p \stackrel{\omega}{\Longrightarrow} p'$, c.-à-d. qu'il existe une suite :

   $$p \stackrel{\tau}{\Longrightarrow} p_0 \stackrel{\alpha_1}{\Longrightarrow} p_1 \stackrel{\alpha_2}{\Longrightarrow} p_2 \ldots \stackrel{\alpha_n}{\Longrightarrow} p'$$

   telle que $\omega = \alpha_1 \alpha_2 \cdots \alpha_n$. Par définition de l'équivalence faible, si $p \approx q$, alors il doit y avoir une suite :

   $$q \stackrel{\tau}{\Longrightarrow} q_0 \stackrel{\alpha_1}{\Longrightarrow} q_1 \stackrel{\alpha_2}{\Longrightarrow} q_2 \ldots \stackrel{\alpha_n}{\Longrightarrow} q'$$

   avec $p \approx q$, $p_0 \approx q_0$, $p_1 \approx q_1$, $\ldots$, $p' \approx q'$. Puisque $p' \approx q'$, alors $\mathtt{Refusals}(p') = \mathtt{Refusals}(q')$, donc $\langle \omega, X \rangle$ est une paire d'échec de q.

■

Notons qu'il existe beaucoup d'autres équivalences établies dans le contexte des systèmes de transitions, telles que, par exemple, l'*équivalence par tests* [NH84], *completed trace*, *ready trace/simulation*, etc. Pour une présentation plus générale, voir [Pom86, Nic87, vG90].

Chapter 2

Réseaux de Petri

Les réseaux de Petri ont été introduits en 1962 par C.A. Petri [Pet62]. Ils constituent l'un des modèles les plus étudiés pour la description des systèmes parallèles (voir par exemple [Rei85, VNCG92]). Il existe de nombreuses classes de tels réseaux ; nous allons donc d'abord préciser quelle classe nous utiliserons dans ce travail.

Définition I.8 Un *réseau de Petri* est un triplet $N = (S, T, W)$, où S et T sont deux ensembles disjoints, respectivement de *places* et de *transitions*, et W est une *fonction de poids* qui, à chaque couple (*place, transition*) ou (*transition, place*) associe un entier, $W : ((S \times T) \cup (T \times S)) \mapsto \texttt{Nat}$. □

On appelle *marquage* d'un réseau $N = (S, T, W)$ toute application M de l'ensemble des places du réseau dans l'ensemble des entiers, $M : S \mapsto \texttt{Nat}$. Autrement dit, un marquage de N est un multi-ensemble sur S (voir l'Annexe A pour les définitions et les notations concernant les multi-ensembles). Intuitivement, cette application décrit une distribution de *jetons* sur les places du réseau. Si $M(p) = k$, on dit que la place p contient k jetons. On appelle *réseau marqué* un couple (N, M), où N est un réseau de Petri et M est un marquage de N.

L'aspect dynamique d'un réseau marqué s'exprime par la *règle de franchissement* d'une transition, laquelle fait évoluer le marquage du réseau. On dit que le marquage M_2 est accessible dans $N = (S, T, W)$ par une transition $t \in T$ à partir d'un marquage M_1, ce que l'on notera $M_1 \gg_N^t M_2$, s'il existe un marquage M du réseau N, tel que pour toute place $s \in S$:

$$M_1(s) = M(s) + W(s, t) \ \wedge \ M_2(s) = M(s) + W(t, s) \ .$$

Si l'on utilise la notation ${}^\bullet t$ et $t^\bullet$ pour décrire respectivement les *pré-conditions* et les *post-conditions* d'une transition t d'un réseau $N = (S, T, W)$, qui sont définies comme les marquages de N (c.-à-d. les multi-ensembles sur S) suivants :

$$\forall s \in S \quad {}^\bullet t(s) \ \stackrel{\text{def}}{=} \ W(s, t) \quad \wedge \quad t^\bullet(s) \ \stackrel{\text{def}}{=} \ W(t, s) \ ,$$

alors la *règle de franchissement* peut s'écrire :

$$M_1 \gg_N^t M_2 \;\Leftrightarrow\; \exists M \in \mu_f(S)\; M_1 = M + {}^\bullet t \wedge M_2 = M + t^\bullet$$

La notation pour les *pré-conditions* et les *post-conditions* peut être utilisée pour les places et, dans ce cas, elle décrit des multi-ensembles de transitions du réseau :

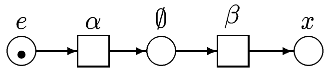
$$\forall t \in T \quad {}^\bullet s(t) \stackrel{\text{def}}{=} W(t,s) \quad \wedge \quad s^\bullet(t) \stackrel{\text{def}}{=} W(s,t)\ .$$

2.1 Réseaux étiquetés

Si les transitions et/ou les places d'un réseau de Petri $N = (S, T, W)$ sont étiquetées par des éléments d'un ensemble d'étiquettes, alors on dit que le réseau est étiqueté. Ici, nous allons présenter un type particulier de tels réseaux, que d'ailleurs nous appellerons dans la suite *réseaux étiquetés.* Ce sont les réseaux utilisés dans l'*Algèbre des Petri Box* (voir Section I.3.4). Ils ont l'avantage d'être compositionnels.

Un réseau étiqueté est un quadruple (S, T, W, λ), où (S, T, W) est un réseau de Petri et λ est une fonction d'étiquetage qui concerne aussi bien les transitions que les places du réseau. Les places sont étiquetées par leur *statut* : e, $\emptyset$ ou x, et les transitions par un ensemble d'étiquettes (que nous détaillerons plus tard).

Le statut e (de l'ang. *entry*) d'une place signifie que c'est une *place d'entrée* (c.-à-d. qu'elle contient un jeton au départ). Le statut x (de l'ang. *exit*) signifie que c'est une *place de sortie* (c.-à-d. que, si tout va bien, elle prend un jeton à la fin). Le statut $\emptyset$ dénote une *place interne.* Par exemple, considérons la figure suivante :



Le réseau a une place d'entrée, une place interne et une place de sortie. De plus, il a deux transitions, étiquetées respectivement par α et β.

Tout réseau étiqueté $B = (S, T, W, \lambda)$ doit vérifier les conditions suivantes :

1. Toute transition, $t \in T$, a au moins une place d'entrée et une place de sortie, c.-à-d.
$$\forall t \in T \quad {}^\bullet t \neq \emptyset \neq t^\bullet$$
2. Il y a au moins une place de sortie et une place d'entrée, c.-à-d.
$$\exists s, s' \in S \quad \lambda(s) = e \;\wedge\; \lambda(s') = x$$
3. Il n'y a pas d'arc qui entre dans une place d'entrée ni d'arc qui sorte d'une place de sortie, c.-à-d. que,
$$\forall s \in S \quad (\lambda(s) = e \Rightarrow {}^\bullet s = \emptyset) \;\wedge\; (\lambda(s) = x \Rightarrow s^\bullet = \emptyset)$$

Toutes ces restrictions imposées aux réseaux étiquetés vont se justifier lorsque nous parlerons des opérations de composition des réseaux de ce type (Section I.3.4).

2.2 Relations d'équivalence sur les réseaux

On remarquera qu'un réseau de Petri $N = (S, T, W)$ et la règle de transition définissent un système de transitions $\mathtt{TS}(N) = (\mu_f(S), T, \gg_N)$ dont les états sont les marquages du réseau, c.-à-d. tous les multi-ensembles finis sur S, dont les étiquettes sont les éléments de T, et dont la relation de transition est définie par la règle de franchissement.

L'étiquetage des transitions dans les réseaux étiquetés s'étend, de manière naturelle, au système de transitions correspondant, c.-à-d. qu'une transition $M \gg_N^t M'$ va être remplacée par $M \gg_N^{\lambda(t)} M'$, où λ est la fonction d'étiquetage du réseau.

Ainsi, toutes les équivalences sur les systèmes de transitions (comme définies dans le Chapitre I.1) peuvent s'appliquer aux réseaux de Petri (voir par exemple [BD87, Pom86, PRS92]). Mais les réseaux de Petri ne sont pas uniquement une technique de description de systèmes de transitions. Ils définissent en plus, de manière très intuitive, la relation de *concurrence* sur les *événements* représentés par les transitions. Un marquage M d'un réseau N permet l'exécution concurrente de deux transitions t_1 et t_2 si la somme multi-ensembliste des préconditions de t_1 et de t_2 est incluse dans M, c.-à-d. ${}^\bullet t_1 + {}^\bullet t_2 \subseteq M$. D'ailleurs, il existe beaucoup de sémantiques de réseaux de Petri, comme les sémantiques d'ordre partiel ("processes" [BD87], de "mots partiels" [Pel89, Vog91, Vog92]) et de simultanéité ("step semantics" [RV83]), qui explorent la "vraie concurrence".

Il existe aussi des relations d'équivalence sur les réseaux, dites *structurelles* (voir par exemple [Des90, Ber86]) qui ne font pas directement intervenir la règle de franchissement. Elles sont d'autant plus intéressantes qu'elles sont souvent basées sur des relations de réduction (transformation) des réseaux, ce qui donne directement des algorithmes de décision pour déterminer si deux réseaux sont équivalents.

2.2.1 Relation de duplication

Comme exemple d'une équivalence structurelle, nous définissons ici la *relation de duplication* sur les réseaux étiquetés.

Définition I.9 Deux réseaux de Petri étiquetés $B_1 = (S_1, T_1, W_1, \lambda_1)$ et $B_2 = (S_2, T_2, W_2, \lambda_2)$ sont en relation de *duplication*, $\simeq_D$, si et seulement si il existe une relation $\rho \subseteq (S_1 \times S_2) \cup (T_1 \times T_2)$ telle que :

1. ρ préserve l'étiquetage : $(x, y) \in \rho \Rightarrow \lambda_1(x) = \lambda_2(y)$.

2. ρ est partout définie et surjective :

   $\forall x_1 \in S_1 \cup T_1 \;\; \exists x_2 \in S_2 \cup T_2 \;\; (x_1, x_2) \in \rho$ et

   $\forall x_2 \in S_2 \cup T_2 \;\; \exists x_1 \in S_1 \cup T_1 \;\; (x_1, x_2) \in \rho.$

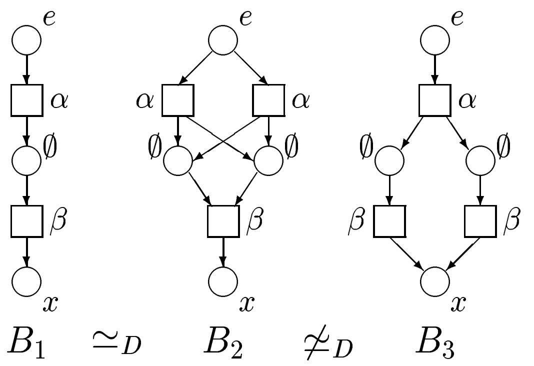


Figure 2.1: Equivalence par duplication de réseaux étiquetés.

3. Si $(s_1, s_2) \in \rho$ et $(t_1, t_2) \in \rho$,
   alors $W_1(s_1, t_1) = W_2(s_2, t_2)$ et $W_1(t_1, s_1) = W_2(t_2, s_2)$.

□

Nous appelons *Petri-box* (ou *Box*) toute classe d'équivalence de réseaux étiquetés par rapport à $\simeq_D$. Si B est un réseau étiqueté, alors par $[B]$ nous dénoterons le *Box* qui le contient. Tout *Box* a un représentant canonique (à un isomorphisme près) dans lequel il n'y a pas d'éléments dupliqués. Dans la Figure 2.1, les réseaux B_1 et B_2 sont équivalents ($\simeq_D$), mais le réseau B_3 ne l'est pas. De plus, B_1 et B_3 sont des représentants canoniques de leurs classes respectives.

2.3 Autres classes de réseaux de Petri

La théorie des réseaux de Petri offre une vaste variété de classes de réseaux, en particulier des classes de réseaux dit "de bas niveau" comme les *réseaux élémentaires* [RT86] et les *réseaux Place/Transition* [Bra83].

Un des défauts de ces domaines est que de nombreux problèmes ne peuvent pas être analysés à cause de la taille trop importante des réseaux associés. C'est pour faire face au problème d'explosion combinatoire de la taille qu'ont été introduites différentes extensions de réseaux dans lesquels les jetons peuvent être "colorés", ce qui veut dire qu'un jeton peut encoder de l'information. Un marquage d'une place d'un tel réseau n'est plus un entier qui représente le nombre de jetons dans la place, mais c'est un multi-ensemble de jetons "colorés" présents dans la place. Généralement, tout réseau de "haut niveau" (avec des jetons colorés) peut être "déplié" en un réseau de "bas niveau" (ce dernier étant souvent infini en nombre de places et/ou de transitions).

A l'heure actuelle, il existe plusieurs sortes de réseaux de haut niveau, comme, par exemple, les "Réseaux Colorés" [Jen86], les "Réseaux Prédicat/-Transition" [Gen91], les "Réseaux Algébriques" [Vau85, Vau86], les "M-nets" [BFF+95b, BFF+95a], etc.

Chapter 3

Algèbres de Processus

Une *algèbre de processus* est un langage abstrait de programmation qui permet de représenter des systèmes à événements discrets (appelés ici *processus*), et qui est muni d'une sémantique permettant de modéliser leurs comportements.

On décrit un processus par une expression constituée à partir d'un ensemble d'éléments de base (les processus élémentaires à comportement atomique) combinés par des opérateurs comportementaux qui expriment les différentes façons d'assembler des composants : composition parallèle, composition séquentielle, synchronisation, etc. La sémantique (dénotationnelle ou opérationnelle) d'une telle expression est en général définie inductivement sur sa structure par celles de ses composants.

Les différentes algèbres se distinguent syntaxiquement par des alphabets de processus élémentaires (actions atomiques) et des ensembles d'opérateurs différents. Ces distinctions prennent leur effet dans les différentes sémantiques en différenciant, par exemple, les types de communication : point-à-point, broadcast, synchrone ou asynchrone.

Dans ce qui suit, nous décrivons d'abord quelques opérateurs de base de différentes algèbres de processus, à savoir les opérateurs de composition parallèle, de changement d'interface, de composition séquentielle, de composition par choix, ainsi que les constructions de base permettant de définir des processus à comportements infinis. Tous ces opérateurs sont définis par des règles d'inférence "à la Plotkin" (voir [HP80]). Nous abordons ensuite la sémantique qui en découle, appelée SOS. Dans la Section I.3.3, nous résumons l'histoire du domaine des algèbres de processus. A la fin de ce chapitre, nous présentons un peu plus en détail l'*Algèbre de Petri-Box*, un exemple d'algèbre de processus munie d'une sémantique à base de réseaux de Petri étiquetés.

3.1 Opérateurs de base

Toute algèbre de processus est définie d'abord par une signature Σ paramétrée par un ensemble d'*actions atomiques*. Les opérateurs de la signature permettent de combiner des processus entre eux, le comportement du processus résultant dépendant de celui de ses composants.

Une manière généralement utilisée pour caractériser le comportement d'un processus est la suivante. Une *transition* $P \xrightarrow{\alpha} Q$ dénote que le processus représenté par le terme P peut évoluer en Q via l'action α. Pour décrire qu'un processus construit par un opérateur $op(P_1, \ldots, P_n)$ peut exécuter une action α en évoluant en Q, à condition que certains de ses sous-processus sont capables d'exécuter certaines actions α_i, noté $P_i \xrightarrow{\alpha_i} P_i'$, nous utilisons la notation suivante :

$$\frac{P_i \xrightarrow{\alpha_i} P_i' \text{ pour certains } i \in \{1, \ldots, n\}}{op(P_1, \ldots, P_n) \xrightarrow{\alpha} Q}$$

Plus généralement, une règle de la forme $\frac{\texttt{Prémisses}}{\texttt{Conclusion}}$, où `Prémisses` est un ensemble fini de *transitions* et `Conclusion` est une *transition*, dit que la transition `Conclusion` est possible si toutes les transitions de `Prémisses` sont possibles. En particulier la règle $\frac{}{\texttt{Conclusion}}$, où l'ensemble `Prémisses` est vide, indique que la transition `Conclusion` est toujours possible (c.-à-d. constitue un axiome).

Une règle est souvent accompagnée par une condition supplémentaire `Cond` qui précise le domaine de l'utilisation de la règle :

$$\frac{P_i \xrightarrow{\alpha_i} P_i' \text{ pour certains } i \in \{1, \ldots, n\}}{op(P_1, \ldots, P_n) \xrightarrow{\alpha} Q} \ \texttt{Cond}$$

La règle peut être utilisée (est effective) si la condition `Cond` est vérifiée.

3.1.1 Composition parallèle

La *composition parallèle* (ou *composition*) est un opérateur clé dans les modèles de systèmes réactifs. Il permet de regrouper (dans l'espace et dans le temps) plusieurs systèmes en un seul en les plaçant juste côte-à-côte et en les laissant s'exécuter en collaboration. La définition de l'opération de composition varie beaucoup d'un auteur à l'autre. Elle dépend en effet de l'intuition que l'on a pour les actions atomiques.

Prenons, par exemple, la composition parallèle proposée par Robin Milner dans le cadre de CCS [Mil80, Mil89], où les systèmes peuvent communiquer par la notion de *liens* (ang. *links*). Une action est en fait interprétée soit comme l'envoi soit comme la réception d'un signal par un lien. Toute communication est considérée comme synchrone et binaire (point-à-point). La composition de deux systèmes P et Q, notée $P|Q$, représente un système composé de P et Q qui s'exécutent en parallèle, avec possibilité de communication. Si par $P \xrightarrow{\alpha} P'$ nous représentons la possibilité d'évolution d'un système P en P' via l'interaction α, la composition $P|Q$, peut être exprimée par les règles suivantes :

$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'} \qquad \frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\beta} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \beta = \overline{\alpha}$$

La première règle dit que si P peut évoluer sous l'action α en P', alors le processus $P|Q$ peut évoluer sous l'action α en $P'|Q$. La deuxième règle décrit la même loi, mais pour l'argument droit de la composition. La dernière règle exprime la possibilité de communication entre les deux systèmes et dans ce cas les actions α et β doivent être *complémentaires* (ce qui dans CCS est dénoté par $\bar{}$). Puisque les communications sont point-à-point, l'action résultante τ devient interne et donc invisible de l'extérieur, car aucun autre processus ne pourra la voir.

Dans CSP [BHR84], la composition parallèle, en plus de regrouper les deux processus en un seul, précise certaines contraintes de communication entre les deux systèmes. Si P et Q sont deux processus, et A un ensemble d'actions, alors $P|_A Q$ décrit le nouveau système composé de P et Q, qui se synchronisent sur les éléments de A. Les règles suivantes donnent la définition de cet opérateur :

$$\frac{P \xrightarrow{\beta} P'}{P|_A Q \xrightarrow{\beta} P'|_A Q}\ \beta \notin A \qquad \frac{Q \xrightarrow{\beta} Q'}{P|_A Q \xrightarrow{\beta} P|_A Q'}\ \beta \notin A \qquad \frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\alpha} Q'}{P|_A Q \xrightarrow{\alpha} P'|_A Q'}\ \alpha \in A$$

Notons que dans la composition à la CSP on a la possibilité d'une synchronisation multiple, c.-à-d. que plus de deux systèmes peuvent se synchroniser sur une action.

3.1.2 Opérateurs de changement de l'interface

La composition parallèle est accompagnée d'opérateurs qui décrivent (modifient) l'interface du système. Ces opérateurs, comme d'ailleurs la composition elle-même, sont directement dépendants de l'intuition derrière une action. Dans CCS, on dispose de la *restriction* ($P\backslash\alpha$) et du *renommage* ($P[\alpha/\beta]$) :

$$\frac{P \xrightarrow{\beta} P'}{P\backslash\alpha \xrightarrow{\beta} P'\backslash\alpha}\ \alpha \neq \beta \qquad \frac{P \xrightarrow{\beta} P'}{P[\alpha/\beta] \xrightarrow{\alpha} P'[\alpha/\beta]} \qquad \frac{P \xrightarrow{\gamma} P'}{P[\alpha/\beta] \xrightarrow{\gamma} P'[\alpha/\beta]}\ \gamma \neq \beta$$

tandis que dans CSP on dispose de l'*abstraction* (ang. *concealment*) (P/α)[1] :

$$\frac{P \xrightarrow{\beta} P'}{P/\alpha \xrightarrow{\beta} P'/\alpha}\ \alpha \neq \beta \qquad \frac{P \xrightarrow{\alpha} P'}{P/\alpha \xrightarrow{\tau} P'/\alpha}$$

Intuitivement, la restriction $P\backslash\alpha$ interdit la communication du système P avec l'environnement via l'action α, le renommage $P[\alpha/\beta]$ renomme chaque occurrence de β en α et l'abstraction P/α rend l'action α invisible de l'extérieur (ce qui peut être exprimé en utilisant l'opérateur de *renommage* de la syntaxe de CCS par $P[\tau/\alpha]$). La différence entre la restriction et l'abstraction est que la restriction bloque toutes les évolutions de P via α, alors que par l'abstraction elles sont transformées en évolutions silencieuses.

[1] En fait, les symboles utilisés dans la littérature pour désigner la restriction dans CCS et pour désigner l'abstraction dans CSP sont identiques. Nous avons donc décidé de remplacer ici le symbole \ par / pour dénoter l'opérateur d'abstraction.

3.1.3 Mise en séquence

L'idée de la composition séquentielle est, sans doute, la plus répandue dans des langages de programmation à l'heure actuelle (comme PASCAL, C/C++, BASIC, et tous les langages de bas niveaux), dans lesquels, en gros, un programme est une suite d'instructions (programmes élémentaires). Par exemple, si α, β sont des instructions, la composition séquentielle, notée ici $\alpha;\beta$, représente un programme qui exécute d'abord l'instruction α, et ensuite, après avoir terminé α, l'instruction β. La mise en séquence des instructions dans le cadre de la programmation séquentielle peut être généralisée facilement à des programmes quelconques. Soient P et Q deux programmes séquentiels, c.-à-d. des programmes construits à partir d'un ensemble d'instructions élémentaires (chaque instruction élémentaire α représente l'exécution d'une action atomique) à l'aide de l'opérateur binaire ";" (si P, Q sont des programmes séquentiels, alors $P;Q$ l'est aussi). La sémantique de la mise en séquence peut être définie par les règles suivantes :

$$\frac{}{\alpha \xrightarrow{\alpha} \surd} \qquad \frac{P \xrightarrow{\alpha} P'}{P;Q \xrightarrow{\alpha} P';Q} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P;Q \xrightarrow{\alpha} Q'}\; P = \surd$$

où $\surd$ dénote un programme séquentiel terminé et α est un programme élémentaire demandant l'exécution de l'action α.

Les mêmes règles ne peuvent pas être directement utilisées pour définir la composition séquentielle des systèmes distribués. Prenons par exemple $P;Q$ où $P = \alpha|\beta$, et $Q = \gamma$. Intuitivement, la composition séquentielle $P;Q$ de ces deux systèmes décrit un nouveau système dont le comportement consiste à simuler l'exécution de P jusqu'à ce qu'il se termine, et ensuite à simuler Q. Le problème est de préciser la signification de la *terminaison* dans un environnement distribué. Dans notre exemple, nous aurions envie de dire que P a terminé si les deux sous-programmes séquentiels α et β ont terminé, mais, malheureusement, nous ne pourrons jamais (en n'utilisant que les règles déjà mentionnées) dériver $P \to \ldots \to \surd$, donc nous ne pourrons pas non plus exécuter le sous-programme Q. De plus, dans le cas général, il faut aussi considérer les problèmes dûs à un éventuel interblocage du système. Est-ce que, par exemple, un blocage doit être considéré comme la terminaison ? D'ailleurs, qu'est ce qu'un blocage ? Pour remédier à ce problème, nous pouvons considérer un prédicat "`fin`" sur les (états des) systèmes, qui déterminera si le système a terminé ou non :

$$\frac{P \xrightarrow{\alpha} P'}{P;Q \xrightarrow{\alpha} P';Q} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P;Q \xrightarrow{\alpha} Q'}\; \mathtt{fin}(P)$$

Il reste, bien sûr, le problème de la définition du prédicat "`fin`". Néanmoins, nous voudrions sûrement avoir au moins $\mathtt{fin}(\surd)$ et $\mathtt{fin}(\surd|\surd)$.

La solution couramment adoptée à tous les problèmes liés à la terminaison est de remplacer l'opération de mise en séquence par le *préfixage*, opération qui permet de "préfixer" un système P par une action α (programme élémentaire). Le système résultant, noté $\alpha.P$, est défini par une seule règle (sans prémisses) :

$$\frac{}{\alpha.P \overset{\alpha}{\to} P}$$

Il est important de remarquer que même le préfixage $\alpha.P$ est une opération plutôt abstraite, dans la mesure où l'on décrit des systèmes réellement distribués. Dans le cas où P est constitué de plusieurs sous-systèmes $P_1, \ldots, P_n$ s'exécutant en parallèle, l'opération $\alpha.P$ signifie le démarrage indépendant des activités P_i, après l'exécution de α, et cache en fait des communications internes au système.

3.1.4 Composition par choix

L'opérateur de *composition par choix*, souvent noté $+$, est un des opérateurs les plus usités des algèbres de processus. La composition par choix de deux systèmes P et Q, notée $P + Q$, décrit intuitivement un système qui peut se comporter soit comme P, soit comme Q.

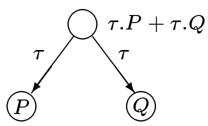
Si P peut dériver en P', et Q en Q', alors le système $P + Q$ peut dériver en P' ou en Q' en exécutant respectivement une des actions initiales de P ou de Q. Cette intuition peut s'exprimer par les deux règles suivantes :

$$\frac{P \overset{\alpha}{\to} P'}{P + Q \overset{\alpha}{\to} P'} \qquad\qquad \frac{Q \overset{\beta}{\to} Q'}{P + Q \overset{\beta}{\to} Q'}$$

Cette formalisation du choix est cependant un peu ambiguë. Nous ne savons pas si c'est le système lui-même qui choisit une des évolutions possibles en écartant les autres à l'avance (*Choix Interne*), ou si c'est l'environnement qui, par le choix de l'action, détermine ce choix (*Choix Externe*). Dans ce dernier cas, le système est censé exécuter n'importe laquelle des actions initiales des sous-systèmes dès que l'action est demandée par l'environnement, en évoluant par la suite comme un seul des composants.

Une des solutions possibles est de représenter explicitement l'évolution interne du système en introduisant l'action invisible, τ. Le système passe d'un état à un autre soit de manière visible et contrôlable, cas dans lequel ce passage est signalé (étiqueté) par une action visible, différente de τ, soit de manière interne, donc incontrôlable, et dans ce cas le passage est étiqueté par l'action τ. On supposera alors que le choix est toujours externe, sauf pour l'action τ.

Par exemple, la composition par choix de P et Q avec "non-déterminisme interne" peut alors être représentée par $\tau.P + \tau.Q$. Un tel système choisit de manière interne de dériver, via l'action τ, vers P ou vers Q, ce qui peut être représenté par le graphe de transitions suivant :



Somme indexée

Il existe aussi une forme plus compacte de la composition par choix, appelée *somme indexée* :

$$\sum_{i \in I} P_i$$

où I est un ensemble d'indices. Intuitivement, le système $\sum_{i \in I} P_i$ dénote la composition par choix de tous les systèmes P_i $(i \in I)$. Cette notation permet de décrire la composition par choix d'un ensemble infini de systèmes grâce à la définition suivante :

$$\frac{P_k \xrightarrow{\alpha} P'_k}{\sum_{i \in I} P_i \xrightarrow{\alpha} P'_k} \; k \in I$$

Notons que, par convention, si l'ensemble d'indices est vide ($I = \emptyset$), la somme indexée, souvent notée "`stop`", décrit un système qui ne peut rien faire.

Dans la mesure où I est fini et non-vide ($I = \{1, 2, \ldots, n\}$), si l'on admet que $+$ est associatif et commutatif, la somme indexée peut être considérée comme une simple abréviation :

$$\sum_{i \in I} P_i \;\stackrel{abr.}{=}\; P_1 + P_2 + \ldots + P_n$$

Une forme particulière de somme indexée est la forme suivante :

$$\sum_{i \in I} \alpha_i . P_i$$

où l'on "garde" (préfixe) chaque sous-système P_k $(k \in I)$ par une action α_k. La règle de transition d'un tel système peut s'exprimer par :

$$\frac{}{\sum_{i \in I} \alpha_i . P_i \xrightarrow{\alpha_k} P_k} \; k \in I$$

Choix interne et choix externe

Une autre solution possible est de considérer explicitement deux opérateurs différents de composition par choix [BHR84] :

1. Choix Interne $(+_i)$, où le choix est pris localement :

$$\frac{}{P +_i Q \xrightarrow{\tau} P} \qquad \frac{}{P +_i Q \xrightarrow{\tau} Q}$$

2. Choix Externe $(+_e)$, où c'est l'environnement qui choisit :

$$\frac{P \xrightarrow{\tau} P'}{P +_e Q \xrightarrow{\tau} P' +_e Q} \qquad \frac{Q \xrightarrow{\tau} Q'}{P +_e Q \xrightarrow{\tau} P +_e Q'}$$

$$\frac{P \xrightarrow{\alpha} P'}{P +_e Q \xrightarrow{\alpha} P'} \; \alpha \neq \tau \qquad \frac{Q \xrightarrow{\beta} Q'}{P +_e Q \xrightarrow{\beta} Q'} \; \beta \neq \tau$$

Outre la composition par choix explicite, il existe dans les systèmes distribués plusieurs sources de "*Non-Déterminisme*" qui viennent de la nature même de ces systèmes. Prenons, par exemple, le processus CCS suivant :

$$((\alpha.P \mid \alpha.Q) \mid \overline{\alpha}.R) \backslash \alpha$$

dont le graphe de transition se présente ainsi:

$$\begin{tikzcd}[column sep=small] & \bigcirc\ ((\alpha.P \mid \alpha.Q) \mid \overline{\alpha}.R) \backslash \alpha \arrow[dl, "\tau"'] \arrow[dr, "\tau"] & \\ ((P \mid \alpha.Q) \mid R) \backslash \alpha\ \bigcirc & & \bigcirc\ ((\alpha.P \mid Q) \mid R) \backslash \alpha \end{tikzcd}$$

Le système peut, par choix interne, évoluer vers la branche gauche ou droite, en exécutant l'une des deux actions invisibles.

3.1.5 Systèmes à comportement infini

Les opérations définies jusqu'à présent ne permettent pas de décrire les systèmes à comportement infini. En effet, tout système spécifié par un terme sur ces opérateurs arrive toujours au bout d'un nombre de pas fini, à un état dit de blocage, où plus aucune transition n'est possible. Les opérations d'*itération*, de *réplication* ou de *récursion*, que l'on verra dans la suite, nous permettent de spécifier des comportements infinis.

Itération

Si P est un terme et $k \geq 1$, nous notons P^k les k mises en séquence de P :

$$P^k \stackrel{\text{def}}{=} \underbrace{P; P; \ldots; P}_{k\text{–fois}}$$

Nous pouvons facilement imaginer une adaptation des trois opérations d'itération classiques de la théorie de langage P^ω–itération infinie de P, P^+–une ou plusieurs répétitions de P, et *–zéro ou plusieurs répétitions de P :

- P^ω – défini par un nombre infini d'itérations de P :

  $$\frac{P \xrightarrow{\alpha} Q}{P^\omega \xrightarrow{\alpha} Q; P^\omega}$$

  Puisque nous voulons un nombre infini d'itérations, le prédicat $\texttt{fin}(P^\omega)$ devrait être fixé à faux.

- P^+ – défini par :

  $$\frac{P \xrightarrow{\alpha} Q}{P^+ \xrightarrow{\alpha} Q; P^+}$$

  Le prédicat $\texttt{fin}(P^+)$ devrait coïncider avec $\texttt{fin}(P)$.

- P^* – défini comme P^+, mais tel que $\mathtt{fin}(P^*)$ est toujours vrai. Cela signifie que, si $\mathtt{fin}(P)$ est vrai, alors $P^* = P^+$.

D'autres sortes d'itération existent (voir par exemple [BBP94]) où deux formes originales d'itérations dans le cadre de l'algèbre de processus, (P^*Q) et $(P^\#Q)$, sont introduites.

Réplication

L'opérateur unaire "$!P$" de "réplication", proposé dans [Mil91], représente un système dont le comportement consiste à exécuter en parallèle un nombre infini de copies du système argument P, ce qui peut être représenté par un terme infini "$P|P|P|\ldots$". Voici les deux règles qui définissent la réplication :

$$\frac{P \xrightarrow{\alpha} Q}{!P \xrightarrow{\alpha} Q|\,!P} \qquad \frac{P|\,!P \xrightarrow{\alpha} Q}{!P \xrightarrow{\alpha} Q}$$

Récursion

La manière la plus puissante et la plus souvent utilisée pour la description des systèmes réactifs infinis est sans doute la *récursion*. Dans la littérature, on trouve plusieurs formes de récursion, mais la plus simple et la plus compacte est la suivante :

$$\mathtt{rec}\ X\ \mathtt{in}\ P$$

où X est une *variable* et P est un terme avec, éventuellement, des variables. Intuitivement, le terme $\mathtt{rec}\ X\ \mathtt{in}\ P$ est une représentation finie d'un terme $P[X \leftarrow R]^\infty$ (généralement) infini obtenu par les remplacements de toutes les occurrences libres de X dans P par le terme lui-même, une infinité de fois. Une occurrence de X est libre dans P si elle n'est pas sous la construction $\mathtt{rec}\ X$. La règle d'inférence suivante est souvent utilisée, car elle préserve la représentation finie pour toute évolution finie du système :

$$\frac{P[X \leftarrow \mathtt{rec}\ X\ \mathtt{in}\ P] \xrightarrow{\alpha} Q}{\mathtt{rec}\ X\ \mathtt{in}\ P \xrightarrow{\alpha} Q}$$

où $P[X \leftarrow R]$ représente la substitution de toute occurrence libre de X dans P par le terme R (une fois).

3.2 Sémantique

Les règles d'inférence $\frac{\mathtt{Prémisses}}{\mathtt{Conclusion}}$ (ou plus généralement $\frac{\mathtt{Prémisses}}{\mathtt{Conclusion}}\,\mathtt{Cond}$) que nous avons si abondamment utilisées dans la description des différents opérateurs, constituent ce que l'on appelle la *Sémantique Opérationnelle Structurelle* (SOS, ang. *Structured Operational Semantics*). Plus précisément, comme sémantique d'une expression P construite sur la signature Σ, nous prenons le graphe de processus $\texttt{graph}(P)$ dans le système de transitions $\texttt{TS} = (T_\Sigma, A, \rightarrow)$, où T_Σ est l'ensemble de tous les termes de la signature Σ, A est l'ensemble des actions atomiques, et $\rightarrow$ est la relation de transition définie comme la plus petite partie de $T_\Sigma \times A \times T_\Sigma$ engendrée (prouvable) par les règles d'inférence.

A l'heure actuelle, la sémantique SOS peut être considérée presque comme un domaine de recherche à part entière [HP80, Plo81, dS84, BIM88, GV92].

3.2.1 Equivalences sur les processus

Une fois la syntaxe et sa sémantique SOS fixée, on peut se demander sous quelles conditions deux systèmes (c.-à-d. deux termes de la signature) peuvent être considérés comme équivalents. Bien évidemment, du point de vue théorique, toutes les relations d'équivalence mentionnées dans la Section I.1.1 peuvent être directement adoptées, mais satisfont-elles nos intuitions ?

Malheureusement, les équivalences mentionnées ci-dessus ne sont pas toujours des *congruences*. Autrement dit, deux processus équivalents ne pourront pas toujours être interchangés dans un système qui les utilise, sans changement du comportement du système (par rapport à la même équivalence). Il semble important de remarquer le fait qu'une équivalence (définie sur le domaine sémantique des systèmes de transitions) dépend à la fois de sa définition et du choix des opérations de l'algèbre. Il nous semble pourtant que les recherches d'équivalences sur les systèmes réactifs ont été menées sur les systèmes de transitions qui les modélisaient, avec une intuition forte pour les primitives de communication (étiquettes), mais sans remise en question des opérations de base qui empêchaient la relation d'être une congruence.

Prenons, par exemple, l'équivalence faible comme définie dans [Mil89]. Cette équivalence n'est pas une congruence dans CCS, à cause de l'opérateur de choix $(+)$. Néanmoins, la somme indexée "gardée" $\sum_{i \in I} \alpha_i . P_i$, un cas particulier de choix, préserve l'équivalence faible. La question que l'on peut alors se poser est de savoir si c'est l'équivalence ou l'opération de choix qu'il faut revoir. La réponse n'est pas évidente. Dans la troisième partie de cette thèse, nous proposons une algèbre de processus dont la signature ne contient aucun opérateur de composition par choix et dans laquelle l'équivalence faible est justement une congruence. Nous montrons que le choix est généralement dérivable des autres opérateurs proposés, et que la puissance expressive de l'algèbre est préservée.

3.3 Bref historique des algèbres de processus

Robin Milner, avec son Algèbre de Systèmes Communicants (CCS, ang. *Calculus of Communicating Systems*), est généralement considéré comme l'initiateur du domaine de l'algèbre de processus, dont la naissance est souvent associée à son livre [Mil80]. En 1985, C.A.R. Hoare publie le livre *Communicating Sequential Processes*, où il propose un nouveau langage pour la spécification de comportements de systèmes réactifs, CSP. Par rapport à CCS, le langage CSP diffère surtout par l'intuition pour l'*action*. En effet, la considération d'une action par Milner comme l'envoi ou la réception d'un signal est remplacée dans CSP par une approche plus abstraite, mais plus expressive, qui permet la synchronisation multiple (plus que deux processus peuvent se synchroniser sur une action). La conséquence directe de cela est qu'une grande classe de comportements est beaucoup plus simple à définir en CSP qu'en CCS. Les deux algèbres de processus diffèrent aussi par le domaine sémantique. La sémantique de CCS est définie en termes de classes d'équivalence sur systèmes de transitions engendrés par des règles de dérivations SOS, tandis que la sémantique d'un processus CSP est exprimée par l'ensemble de *paires d'échec* de ce processus.

Une autre approche pour la description et l'analyse de systèmes réactifs dans un cadre purement algébrique a été développée à Amsterdam sous le nom de ACP (ang. *Algebra of Communicating Processes*) [BK84, BW90]. La différence majeure de cette algèbre de processus par rapport à CCS ou CSP, est que sa sémantique est donnée uniquement par des *axiomes*, ce qui donne la liberté du choix du domaine sémantique pour les termes, pourvu qu'ils vérifient les axiomes.

Plusieurs autres travaux ont été menés, soit pour donner une nouvelle sémantique d'une algèbre de processus déjà existante, grâce à des domaines sémantiques différents [Win83, vGV87, BC87, Hen88, BC88, Tau89, DD90, Kri91, Ruq93, Tau89], soit pour définir une nouvelle algèbre directement adaptée à un domaine spécifique [Mil85, Bes86, BB90, BDH92, BDG95].

3.4 Algèbre des Petri-Box (PBC)

Comme nous allons plus tard appliquer nos résultats dans le cadre d'une algèbre particulière, à savoir PBC, nous l'introduisons ici aussi brièvement que possible mais avec les détails nécessaires pour la suite.

L'algèbre des Petri-Box (PBC ang. *Petri-Box Calculus*) est un résultat important du projet ESPRIT BRA 3148 DEMON [BDH92]. Cette algèbre a été développée, entre autre, dans le but de donner une sémantique compositionnelle à des langages de programmation concurrents. La sémantique de PBC consiste dans le domaine des *Petri-Box*. Les Petri-Box (*Box*) sont des classes d'équivalence sur des réseaux de Petri étiquetés. En fait, les Box peuvent être vus comme des "boîtes noires" dont l'étiquetage joue le rôle de l'interface grâce auquel les Box peuvent être combinés entre eux.

3.4.1 Syntaxe

PBC est particulièrement riche en opérateurs comportementaux, que nous allons énumérer un par un. Les objets de base (processus élémentaires) de PBC sont des *multi-actions*, plus précisément des multi-ensembles finis sur les éléments d'un ensemble prédéfini, `Act`, d'actions atomiques. Une multi-action représente l'exécution simultanée de ses éléments. La multi-action vide modélise de façon naturelle l'action interne (τ).

La syntaxe de PBC, dont les termes s'appellent *Box-expressions*, est paramétrée par deux ensembles :

1. l'ensemble `Act` d'actions atomiques, et
2. l'ensemble `Vars` de variables, appelées *étiquettes hiérarchiques*.

L'ensemble `Act` d'actions atomiques est muni d'une bijection $\bar{ } : \mathtt{Act} \mapsto \mathtt{Act}$ telle que $\overline{a} \neq a$ et $\overline{\overline{a}} = a$, comme dans CCS.

Définition I.10 (Box–Expression)

1. Un multi-ensemble fini 'm' sur `Act` est une Box–expression.
2. Une variable X de `Vars` est une Box–expression.
3. Si E, E_1, E_2 et E_3 sont des Box–expressions, $a \in \mathtt{Act}$ et φ est une application $\mathtt{Act} \mapsto \mathtt{Act}$, alors
   (a) $E_1; E_2$ - mise en séquence,
   (b) $E_1 \;\square E_2$ - choix,
   (c) $[E_1 * E_2 * E_3]$ - itération,
   (d) $E_1 \| E_2$ - composition parallèle,
   (e) E **sy** a - synchronisation,
   (f) E **rs** a - restriction,
   (g) $[E \;:\; a]$ - scoping,
   (h) $E[\varphi]$ - renommage,
   (i) $E[X \leftarrow E_1]$ - raffinement,
   (j) **rec** X **in** E - récursion

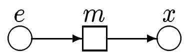
   sont des Box–expressions.

$\square$

3.4.2 Sémantique

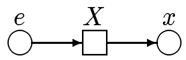
Au niveau sémantique, toute Box–expression possède un réseau étiqueté associé. En général, la sémantique par réseaux des Box–expressions est donnée par une fonction qui à chaque Box–expression E associe un réseau étiqueté $B(E)$. Ainsi, l'ensemble d'opérations sémantiques sur les réseaux correspond exactement aux opérateurs donnés dans la Définition I.10, dont le domaine des étiquettes de transitions (non spécifié au Chapitre I.2) est égale à $\mu_f(\mathtt{Act}) \cup \mathtt{Vars}$, où $\mu_f(\mathtt{Act})$ dénote tous les multi-ensembles finis sur `Act`.

Multi-action. Une multi-action 'm' est représentée par le réseau suivant :



Le réseau est composé d'une place d'entrée, d'une place de sortie et d'une *transition de communication* étiquetée par m.

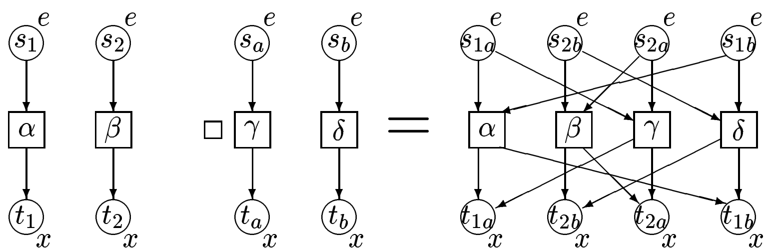
Action hiérarchique. Une variable 'X' est représentée par le réseau suivant :



Toute transition avec une étiquette hiérarchique est appelée *transition d'abstraction*, pour la différencier des transitions de communication, qui sont toutes étiquetées par une multi-action.

Composition Séquentielle. La composition séquentielle de deux réseaux B_1 et B_2 définit un nouveau réseau $(B_1; B_2)$ en unifiant (par produit Cartésien) les places de sortie de B_1 avec les places d'entrée de B_2.

Choix. La composition par choix de deux réseaux B_1 et B_2 définit un nouveau réseau $B_1 \;\square B_2$, dont les places d'entrée et de sortie sont obtenues par le produit Cartésien des places d'entrée de B_1 avec les places d'entrée de B_2, et par le produit Cartésien des places de sortie de deux réseaux. Par exemple :



Itération. Intuitivement, l'itération $[B_1 * B_2 * B_3]$ décrit l'exécution de B_1 (initialisation), suivie de l'exécution de B_2 (boucle) zéro, une ou plusieurs fois, suivie de B_3 (terminaison). La Figure 3.1 montre trois constructions possibles.

Composition Parallèle. La composition parallèle de deux réseaux étiquetés les met tout simplement côte-à-côte.

Synchronisation. La synchronisation est un des éléments les plus intéressants de PBC, et aussi un des éléments les plus discutés. Beaucoup de propositions différentes ont été envisagées [BDH92, Dev93, FK94]. Intuitivement, la synchronisation d'un réseau B génère de nouvelles transitions, qui représentent les synchronisations de certaines transitions de B. Le Chapitre II.6 est entièrement consacré à la synchronisation et à la restriction dans PBC.

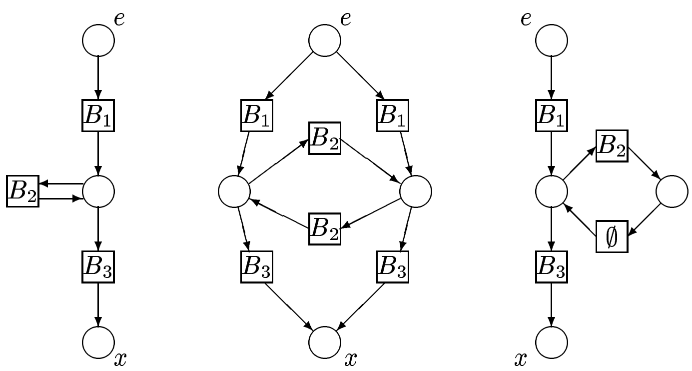


Figure 3.1: Trois constructions schématiques pour l'itération.

Restriction. La restriction d'un réseau par rapport à une action (ou à un ensemble d'actions) enlève du réseau toute transition dont l'étiquette contient l'action (une des actions) en question. Dans le Chapitre II.6 nous en parlerons en détail.

Scoping. Le scoping $[B : a]$ est une abréviation pour $(B \ \mathtt{sy} \ a) \ \mathtt{rs} \ a$. On peut le voir comme la déclaration d'un canal a local à B.

Renommage. Le renommage d'un réseau consiste en un renommage des étiquettes de transitions de communication (c.-à-d. des multi-actions) selon une fonction de renommage donnée.

Raffinement. Le raffinement $B[X \leftarrow B']$ consiste en un remplacement de toutes les transitions étiquetées par la variable X par le réseau B'.

Récursion. La récursion consiste en une suite infinie de raffinements. Elle peut créer des réseaux infinis (en nombre de places et de transitions). La généralisation de la définition du raffinement et de la récursion de [BDH92] a été proposée dans [BDE93].

Comme nous l'avons déjà mentionné, le domaine sémantique de PBC est constitué de classes d'équivalence, appelées *Box*, sur les réseaux étiquetés, engendrées par la relation de duplication, $\simeq_D$ (la définition de $\simeq_D$ est la même que celle donnée dans la Section I.2.1, mais on exige en plus que la relation ρ soit bijective sur toutes les transitions hiérarchiques, ce qui veut dire que les transitions hiérarchiques ne sont jamais considérées comme des duplications l'une de l'autre). Un des résultats de [BDH92] est que la sémantique de PBC est consistante avec Box, c'est-à-dire que les opérations sur les réseaux ne dépendent pas du choix du représentant de la classe :

$$\left\{\begin{array}{ccc} B_1 & \simeq_D & B'_1 \\ B_2 & \simeq_D & B'_2 \\ B_3 & \simeq_D & B'_3 \end{array}\right\} \Rightarrow \left\{\begin{array}{rcl} B_1; B_2 & \simeq_D & B'_1; B'_2 \\ B_1 \ \square B_2 & \simeq_D & B'_1 \ \square B'_2 \\ [B_1 * B_2 * B_3] & \simeq_D & [B'_1 * B'_2 * B'_3] \\ B_1 \parallel B_2 & \simeq_D & B'_1 \parallel B'_2 \\ B_1 \ \mathtt{sy}\ a & \simeq_D & B'_1 \ \mathtt{sy}\ a \\ B_1 \ \mathtt{rs}\ a & \simeq_D & B'_1 \ \mathtt{rs}\ a \\ [B_1 : a] & \simeq_D & [B'_1 : a] \\ B_1[\varphi] & \simeq_D & B'_1[\varphi] \\ B_1\,[X \leftarrow B_2] & \simeq_D & B'_1\,[X \leftarrow B'_2] \\ \mathtt{rec}\ X\ \mathtt{in}\ B_1 & \simeq_D & \mathtt{rec}\ X\ \mathtt{in}\ B'_1 \end{array}\right\}$$

Cela prouve directement la consistance de la sémantique :

$$B(\mathtt{op}(E_1, \ldots, E_n)) \ \simeq_D \ \mathtt{op}(B(E_1), \ldots, B(E_n)) \ ,$$

pour toutes les opérations op de PBC.

Il existe aussi une sémantique SOS de PBC [Kou94, KEB94].

Discussion

L'Algèbre des Petri-Box constitue le noyau de PEP (ang. *Programming Environment based on Petri nets*). PEP [BF95] est un environnement d'aide à la conception, à la modélisation et à la vérification de systèmes informatiques. PEP intègre un langage parallèle $\mathrm{B(PN)}^2$ [BH93] dont la sémantique est définie en termes de PBC. Plusieurs programmes d'analyse et de vérification (généralement basés sur des algorithmes de “model-checking” sur des réseaux de Petri) y sont intégrés.

Un des atouts de l'Algèbre des Petri-Box est la facilité d'intégration de nouveaux objets, dans le but, par exemple, de représenter des données. En effet, il suffit de trouver (à la main) un réseau de Petri pour représenter l'objet voulu et de l'ajouter comme une nouvelle constante de l'algèbre, comme par exemple les définitions de *data-box* et de *channel-box* dans $\mathrm{B(PN)}^2$ [BH93].

La sémantique en termes de réseaux de Petri de bas niveau crée souvent des réseaux de taille très importante donc difficilement maniables. Pour remédier à ce problème, nous avons récemment proposé un nouveau domaine sémantique de PBC en termes de réseaux de haut niveau, appelés M-nets [BFF^+95b, BFF^+95a] (voir aussi [KP95, DK95, Kla95] pour un essai d'une généralisation de M-nets).

Part II

Composition Parallèle et Synchronisation

Dans cette partie, nous nous intéressons aux opérateurs liés à la *composition* dans des formalismes dédiés à la spécification, à la modélisation et à l'analyse des *systèmes réactifs*, tels que les *systèmes de transitions* et les *réseaux de Petri*. Cette partie est consacrée à la spécification formelle de différentes primitives de synchronisation. Nous proposons une formalisation des notions intuitives liées aux primitives de communication que nous appelons *actions atomiques*.

Dans le premier chapitre, nous proposons une classe de structures relationnelles sur l'ensemble des actions, appelées *structures de synchronisation*, qui décrivent de manière formelle et uniforme la capacité de synchronisation de chaque action.

Une première application des structures de synchronisation et de ses propriétés algébriques est présentée dans le deuxième chapitre de cette partie, dans le contexte des *systèmes de transitions*.

Le chapitre suivant est entièrement consacré à la synchronisation et à la restriction dans l'*Algèbre des Petri-Box* (PBC), généralisées par des structures de synchronisation. Nous avons élargi la flexibilité de l'interprétation de ces opérateurs par rapport aux définitions d'origine, en les rendant paramétrables par une structure de synchronisation sur des actions atomiques.

Chapter 4

Primitives de communication

Les capacités de communication d'un processus sont liées aux capacités de synchronisation de ses événements élémentaires qui sont encodées par des étiquettes. De telles étiquettes sont appelées *actions*. Parfois, une structure particulière sur les actions, qui définit leurs capacités de synchronisation, est fixée pour un modèle donné, comme par exemple dans CCS [Mil80, Mil89], CSP [BHR84, Hoa85], ou dans PBC [BDH92]. Dans d'autres approches, cette structure est dotée d'une opération binaire arbitraire (avec quelques restrictions raisonnables), par exemple, l'opération 'γ' dans ACP [BW90, BB93] ou '$\bullet$' dans [Win83, Win86].

Pour assurer un traitement uniforme des primitives de communication, nous considérons une classe de structures relationnelles sur les actions, appelées *structures de synchronisation*. Nous proposons une relation d'équivalence sur les actions, appelée *bisimulation d'action*, qui identifie les actions non-distinguables par rapport aux opérations qui en dépendent, telles que la synchronisation ou la restriction. Nous définissons aussi quelques sous-ensembles consistants d'actions avec de belles propriétés, qui, à leur tour, vont garantir l'exactitude des opérations dérivées sur les processus.

Nous proposons une extension de la structure de synchronisation permettant de grouper les actions dans des multi-ensembles d'actions, appelés *multi-actions*. L'idée sous-jacente à la multi-action est qu'elle représente une exécution simultanée de plusieurs actions (ses composantes), comme dans [Mil85, BDH92]. Ce point de vue s'oppose à la signification intuitive des actions dans MEIJE [AB84] ou SCCS [Mil83], où une exécution simultanée de deux actions mène toujours à leur synchronisation.

L'utilisation des multi-actions à la place de simples actions a plusieurs motivations. D'abord, les multi-actions permettent de passer de la sémantique de l'entrelacement (ang. *interleaving semantics*) à la sémantique par pas (ang. *step semantics*) de façon très naturelle, voir par exemple [TV89]. Deuxièmement, les multi-actions nous permettent de considérer un nouveau type de dépendance entre les événements, que nous appelons *dépendance de simultanéité* (voir la dernière partie de la thèse).

4.1 Structures de synchronisation

Soient $B \subseteq Q \times Q$ une relation binaire et $R \subseteq Q \times Q \times Q$ une relation ternaire sur les éléments d'un ensemble Q. Nous introduisons les notations fonctionnelles $B(x)$ et $R(x, y)$, où $x, y \in Q$, pour dénoter les sous-ensembles suivants de Q :

$$\begin{aligned} B(x) &\stackrel{\text{def}}{=} \{z \in Q \mid (x, z) \in B\} \\ R(x, y) &\stackrel{\text{def}}{=} \{z \in Q \mid (x, y, z) \in R\} \end{aligned}$$

Définition II.1
Une *structure de synchronisation*, $\mathcal{S}$, est un triplet $(\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$, où $\mathtt{Act}$ est un ensemble d'*actions*, $\mathtt{Syn}$ est une relation ternaire de *synchronisation* sur les actions, et $\mathtt{Hide}$ est une relation unaire d'*abstraction* sur les actions (c.-à-d. $\mathtt{Hide} \subseteq \mathtt{Act}$). La relation de synchronisation $\mathtt{Syn}$ doit être *commutative* dans le sens fonctionnel :

$$\forall x, y \in \mathtt{Act} \;\; \mathtt{Syn}(x, y) = \mathtt{Syn}(y, x)$$

□

Un triplet (x, y, z) dans $\mathtt{Syn}$ dénote que l'action x peut se synchroniser avec y en créant l'action z.

Cette définition est proche de la définition d'*algèbre de synchronisation* (ang. *synchronization algebra*) proposée par Glynn Winskel [Win83, Win86]. Mais, tandis que cette algèbre a été conçue pour exprimer la synchronisation dans les modèles basés sur les "actions singletons", notre structure relationnelle est destinée à travailler aussi dans le contexte des *multi-actions*. L'algèbre de synchronisation à la Winskel est un quadruple $(\mathtt{Act}, \bullet, *, 0)$, où $\bullet$ est une opération binaire, commutative et associative sur les actions de $\mathtt{Act}$, et $*$, 0 sont des constantes (actions virtuelles) qui représentent respectivement l'*absence* d'action et l'*échec* (action impossible). La relation de synchronisation $\mathtt{Syn}$ d'une structure de synchronisation $(\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ joue le rôle de l'opération $\bullet$, mais elle permet d'exprimer aussi une sorte de non-déterminisme, puisqu'une synchronisation de deux actions peut potentiellement donner plusieurs résultats. Nous avons besoin de ce non-déterminisme si nous voulons exprimer de manière directe les différentes possibilités de synchronisation des multi-actions. Par exemple, si une multi-action peut se synchroniser avec une autre multi-action via deux actions a ou b, la synchronisation peut utiliser l'action a ou b, exclusivement ou simultanément. De plus, nous n'exigeons pas que la relation $\mathtt{Syn}$ soit "associative[1]".

La relation d'abstraction, $\mathtt{Hide}$, désigne les actions qui peuvent s'exécuter de manière asynchrone. Pendant l'encapsulation d'un système, les actions de $\mathtt{Hide}$

[1]Par l'associativité de la relation $\mathtt{Syn}$ nous comprenons :

$$\forall \alpha, \beta, \gamma \in \mathtt{Act} \quad \bigcup_{x \in \mathtt{Syn}(\alpha,\beta)} \mathtt{Syn}(x, \gamma) \;=\; \bigcup_{x \in \mathtt{Syn}(\beta,\gamma)} \mathtt{Syn}(\alpha, x) \;.$$

peuvent devenir locales, invisibles de l'extérieur, sans pour autant provoquer des blocages. Dans une algèbre de synchronisation à la Winskel, cela correspond à la synchronisation non nulle d'une action avec $*$. L'idée sous-jacente est que : *une action qui peut s'exécuter sans synchronisation avec une autre action ne provoque pas de blocages durant l'encapsulation*[2]*, dans la mesure où l'encapsulation consiste en la constatation de l'impossibilité de certaines synchronisations.* La relation d'abstraction Hide peut donc être interprétée dans une algèbre de synchronisation par :

$$\texttt{Hide} \stackrel{\text{def}}{=} \{x \in \texttt{Act} \mid x \bullet * \neq 0\} \ .$$

Une action x telle que $x \notin \texttt{Hide}$ et $\forall y\ \texttt{Syn}(x,y) = \emptyset$ dénote une action qui ne peut jamais s'exécuter, et donc qui représente un blocage (dans une algèbre de synchronisation, elle est notée par la constante 0).

Exemple II.1 La structure de synchronisation pour CCS peut être définie par : $\mathcal{S}_M = (\texttt{Act}_M, \texttt{Syn}_M, \texttt{Hide}_M)$, où $\texttt{Act}_M$ est un ensemble d'actions $\{a, b, \ldots\}$ munies de leurs actions complémentaires $\{\overline{a}, \overline{b}, \ldots\}$ et d'une action *invisible* $\{\tau\}$. Seules deux actions complémentaires peuvent se synchroniser en formant l'action invisible τ, et τ ne peut se synchroniser avec aucune action. D'autre part, seule l'action τ peut s'exécuter de manière asynchrone. Donc, la relation de synchronisation $\texttt{Syn}_M$ est composée des triplets $(x, \overline{x}, \tau)$ et $(\overline{x}, x, \tau)$ pour toute action $x \in \{a, b, \ldots\}$, et la relation d'abstraction est un singleton $\texttt{Hide}_M = \{\tau\}$.

4.1.1 Simulations et bisimulations d'actions

Définition II.2 Soit $\mathcal{S} = (\texttt{Act}, \texttt{Syn}, \texttt{Hide})$ une structure de synchronisation. Une *simulation d'actions* dans $\mathcal{S}$ est une relation binaire sur Act, $B \subseteq \texttt{Act} \times \texttt{Act}$, telle que, si $(a, b) \in B$, alors:

1. $a \in \texttt{Hide} \Rightarrow b \in \texttt{Hide}$, et

2. $\forall x \in \texttt{Act}\ \forall y \in \texttt{Syn}(a, x)\ \exists y' \in \texttt{Syn}(b, x) \quad (y, y') \in B$.

Si B est aussi symétrique, on l'appelle *bisimulation d'actions.* □

Remarquons que, pour toute structure de synchronisation $\mathcal{S}$, la plus grande bisimulation d'actions (notée $\approx_A$) existe. En fait, c'est l'union de toutes les bisimulations d'actions dans $\mathcal{S}$. $\approx_A$ est une relation symétrique, réflexive et transitive, c'est donc une relation d'équivalence sur les actions de $\mathcal{S}$.

Intuitivement, deux actions équivalentes, $a \approx_A b$, ne se distinguent pas par la possibilité ou l'impossibilité de synchronisation avec d'autres actions. Cela veut dire que, si nous supposons qu'observer une occurrence d'action c'est se

[2] Par "encapsulation" d'un système réactif nous comprenons une sorte de spécification de l'environnement dans lequel le système va évoluer. En général, cette spécification indiquera quels types d'interactions pourront être demandées au système par l'environnement.

synchroniser avec elle, nous ne serons jamais capables de distinguer les actions équivalentes. Au niveau des systèmes réactifs, cela se traduit par le fait que deux systèmes qui ne diffèrent que par des actions équivalentes par $\approx_A$ seront équivalents dans des équivalences qui ne font pas intervenir explicitement les noms d'actions mais qui se basent sur les opérations de composition de systèmes, comme par exemple l'*équivalence sémantique* ($\sim$) introduite dans la Section III.9.4.3 de la troisième partie de cette thèse.

Soit $\mathcal{S} = (\texttt{Act}, \texttt{Syn}, \texttt{Hide})$, une structure de synchronisation. Nous définissons le *quotient* de $\mathcal{S}$, noté $\mathcal{S}/_{\approx_A}$, comme

$$\mathcal{S}/_{\approx_A} \stackrel{\text{def}}{=} (\texttt{Act}/_{\approx_A}, \texttt{Syn}/_{\approx_A}, \texttt{Hide}/_{\approx_A})$$

où

- $\texttt{Act}/_{\approx_A} \stackrel{\text{def}}{=} \{[a]_{\approx_A} \mid a \in \texttt{Act}\}$, où $[a]_{\approx_A}$ est la classe d'équivalence générée par l'action a, c.-à-d. : $[a]_{\approx_A} \stackrel{\text{def}}{=} \{x \in \texttt{Act} \mid a \approx_A x\}$,
- $\texttt{Syn}/_{\approx_A} \stackrel{\text{def}}{=} \{([a]_{\approx_A}, [b]_{\approx_A}, [c]_{\approx_A}) \mid (a, b, c) \in \texttt{Syn}\}$,
- $\texttt{Hide}/_{\approx_A} \stackrel{\text{def}}{=} \{[a]_{\approx_A} \mid a \in \texttt{Hide}\}$.

Remarquons que le quotient $\mathcal{S}/_{\approx_A}$ d'une structure de synchronisation $\mathcal{S}$ est une structure de synchronisation. Une structure de synchronisation quotient, $\mathcal{S}/_{\approx_A}$, a quelques propriétés évidentes mais utiles comme, par exemple, que $\mathcal{S}/_{\approx_A}$ est isomorphe à $(\mathcal{S}/_{\approx_A})/_{\approx_A}$, et que deux actions différentes dans $\mathcal{S}/_{\approx_A}$ ne sont jamais équivalentes. On peut considérer $\mathcal{S}/_{\approx_A}$ comme une représentation optimale de $\mathcal{S}$. Ainsi, on peut définir la relation d'équivalence suivante sur les structures de synchronisation.

Définition II.3 Deux structures de synchronisation $\mathcal{S}$ et $\mathcal{S}'$ sont équivalentes, noté $\mathcal{S} =_{\mathcal{S}} \mathcal{S}'$, si et seulement si leur quotients, $\mathcal{S}/_{\approx_A}$ et $\mathcal{S}'/_{\approx_A}$ sont isomorphes (c.-à-d. ils sont identiques modulo une bijection entre leurs actions). $\square$

Exemple II.2 *Soit* $\mathcal{S} = (\texttt{Act}, \texttt{Syn}, \texttt{Hide})$*, une structure de synchronisation où :*

1. $\texttt{Act} = \{w, r, w_r\}$
2. $\texttt{Syn} = \{(w, r, w_r), (w_r, r, w_r), (r, w, w_r), (r, w_r, w_r)\}$
3. $\texttt{Hide} = \{w, w_r\}$

La structure de synchronisation $\mathcal{S}$ peut être interprétée ainsi : elle définit trois actions : w – pour "envoi", r – pour "réception", et w_r – pour le résultat de la synchronisation de w avec r. Cette structure de synchronisation permet plusieurs "réceptions simultanées" (un nombre arbitraire d'actions r) d'un "envoi" (une action w), car w peut se synchroniser avec un r en donnant comme résultat w_r ($w_r \in \mathtt{Syn}(w, r)$), qui peut, à son tour, se synchroniser avec un deuxième r en donnant un nouveau w_r ($w_r \in \mathtt{Syn}(w_r, r)$), et ainsi de suite. S'il y a au moins une réception pour un envoi, alors le résultat de la synchronisation est l'action w_r, qui représente "un envoi qui a été reçu".

Intuitivement, il y a une différence entre l'action w et l'action w_r, mais nous n'avons pas la possibilité de distinguer ces deux actions par une opération de synchronisation ou d'abstraction ; les deux actions se comportent exactement de la même façon.

Pour le montrer, il suffit de vérifier que $w \approx_A w_r$, c.-à-d. de trouver une bisimulation d'actions, B, telle que $(w, w_r) \in B$. Prenons

$$B = \{(w, w_r), (w_r, w), (w, w), (w_r, w_r), (r, r)\} .$$

Nous remarquons que B est en réalité la plus grande bisimulation d'actions, c.-à-d. que c'est $\approx_A$ dans $\mathcal{S}$. Donc, $\mathcal{S}/_{\approx_A} = (\mathtt{Act}/_{\approx_A}, \mathtt{Syn}/_{\approx_A}, \mathtt{Hide}/_{\approx_A})$ où

1. $\mathtt{Act}/_{\approx_A} = \{[w]_{\approx_A}, [r]_{\approx_A}\}$, avec $[w]_{\approx_A} = [w_r]_{\approx_A} = \{w, w_r\}$ et $[r]_{\approx_A} = \{r\}$.
2. $\mathtt{Syn}/_{\approx_A} = \{([w]_{\approx_A}, [r]_{\approx_A}), ([r]_{\approx_A}, [w]_{\approx_A})\}$
3. $\mathtt{Hide}/_{\approx_A} = \{[w]_{\approx_A}\}$

Notons que cela n'est plus le cas si, par exemple, la relation d'abstraction $\mathtt{Hide}$ de $\mathcal{S}$ ne contient pas w, c-à-d. si un envoi (w) ne peut pas s'exécuter sans au moins une réception. Dans ce cas, $\approx_A \;=\; \{(x, x) \mid x \in \mathtt{Act}\}$, c.-à-d. c'est l'identité.

4.1.2 Ensembles de communication et aspects

Une action b est une *"co-action"* d'une action a si a et b peuvent se synchroniser. Une action c est une *"sy-action"* d'une action a s'il existe une action b telle que a et b peuvent se synchroniser en c. Les notions de sy-actions et de co-actions peuvent être étendues aux ensembles d'actions.

Définition II.4 Soit $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ une structure de synchronisation, a une action ($a \in \mathtt{Act}$), et Q un ensemble d'actions ($Q \subseteq \mathtt{Act}$). Nous définissons :

1. $\mathtt{coa}(a) \stackrel{\text{def}}{=} \{y \in \mathtt{Act} \mid \exists z \in \mathtt{Act},\ (a, y, z) \in \mathtt{Syn}\}$ (co-actions)
2. $\mathtt{coa}(Q) \stackrel{\text{def}}{=} \bigcup_{x \in Q} \mathtt{coa}(x)$
3. $\mathtt{sya}(a) \stackrel{\text{def}}{=} \{z \in \mathtt{Act} \mid \exists y \in \mathtt{Act},\ (a, y, z) \in \mathtt{Syn}\}$ (sy-actions)
4. $\mathtt{sya}(Q) \stackrel{\text{def}}{=} \bigcup_{x \in Q} \mathtt{sya}(x)$

□

Proposition II.1 *Soit $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ une structure de synchronisation. Pour tous les ensembles d'actions $Q_1 \subseteq \mathtt{Act}$ et $Q_2 \subseteq \mathtt{Act}$ nous avons :*

1. $\mathtt{coa}(Q_1 \cup Q_2) = \mathtt{coa}(Q_1) \cup \mathtt{coa}(Q_2)$
2. $\mathtt{sya}(Q_1 \cup Q_2) = \mathtt{sya}(Q_1) \cup \mathtt{sya}(Q_2)$
3. $Q_1 \subseteq Q_2 \;\Rightarrow\; \mathtt{coa}(Q_1) \subseteq \mathtt{coa}(Q_2) \wedge \mathtt{sya}(Q_1) \subseteq \mathtt{sya}(Q_2)$

Preuve. Conséquences directes des définitions. ■

Définition II.5 Soit $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ une structure de synchronisation. Nous appelons *invisible* une action $a \in \mathtt{Act}$ si $\mathtt{coa}(a) = \emptyset$ et $a \in \mathtt{Hide}$. □

Notons que si a et b sont des actions invisibles, alors $a \approx_A b$, c.-à-d. qu'il existe au plus une classe d'équivalence d'actions invisibles. Cette classe va être dénotée par $\mathtt{Inv}$, avec $\mathtt{Inv} = \emptyset$ s'il n'y a aucune action invisible. D'autres classes intéressantes peuvent aussi être distinguées, comme, par exemple, $\mathtt{Deadlock} = \{x \in \mathtt{Act} \mid x \notin \mathtt{Hide}, \mathtt{coa}(x) = \emptyset\}$.

Définition II.6 Un *ensemble de communication* de $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ est un sous-ensemble A de $\mathtt{Act}$, tel que $\mathtt{coa}(\mathtt{coa}(A)) \subseteq A$. Si un ensemble de communication A vérifie $\mathtt{coa}(\mathtt{coa}(A)) = A$, alors il est appelé *ensemble propre de communication*. □

Remarquons que si A est un ensemble de communication, alors $A \backslash \{a \mid \mathtt{coa}(a) = \emptyset\}$ est un ensemble propre de communication. Inversement, si un ensemble de communication A contient une action a telle que $\mathtt{coa}(a) = \emptyset$, alors A n'est pas propre. Cela veut dire que l'on pourrait redéfinir un ensemble propre de communication, comme un ensemble de communication sans actions à $\mathtt{coa}$ vide.

Définition II.7 Soit $\mathcal{S}$ une structure de synchronisation. Un *aspect* de $\mathcal{S}$ est un ensemble A d'actions de $\mathcal{S}$, qui vérifie $\mathtt{sya}(A) \cup \mathtt{coa}(A) \subseteq A$. □

La notion d'*ensemble de communication* a pour but de permettre de formaliser de manière abstraite et cohérente les *canaux de communication* à deux bouts. Un tel canal reliant deux systèmes réactifs P et Q définit en fait les contraintes de communication entre les deux systèmes. Une approche naturelle est de sélectionner l'ensemble des actions qui peuvent être utilisées pour cette communication. Supposons qu'un canal reliant les deux systèmes permet la transmission à partir de P vers Q des actions d'un ensemble A. Quelles sont alors les actions qui peuvent être transmises de Q vers P ? A priori, ce seront toutes les actions qui peuvent se synchroniser avec les actions de A, c'est-à-dire les actions de $\mathtt{coa}(A)$. Maintenant, si l'on essaie de spécifier la même connection, mais cette fois-ci du côté de Q, en disant que l'ensemble des actions qui peuvent être transmises du système Q vers le système P est $\mathtt{coa}(A)$, on pourrait ne pas obtenir la même chose, puisque $\mathtt{coa}(\mathtt{coa}(A))$ n'est pas forcément égal ou inclus à A. La propriété d'un ensemble de communication A qui dit que $\mathtt{coa}(\mathtt{coa}(A)) \subseteq A$ nous garantit que la connection de P vers Q via A autorise les mêmes synchronisations que la connection de Q vers P via $\mathtt{coa}(A)$ (ainsi que la connection de P vers Q via $\mathtt{coa}(\mathtt{coa}(A))$). Ainsi, un ensemble de communication décrit intuitivement un canal de communication à deux bouts (synchronisations binaires).

La notion d'*aspect* joue le rôle d'ensemble de communication symétrique et transitif, c.-à-d. lorsque le résultat d'une synchronisation peut lui-même être incorporé dans une nouvelle synchronisation.

Proposition II.2 *Pour toute structure de synchronisation* $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ *et pour toutes les actions* $a, b \in \mathtt{Act}$ *nous avons :*

1. *Tout aspect est un ensemble de communication.*

2. *La classe des ensembles de communication et la classe des ensembles propres de communication sont closes pour* $\cup$, $\cap$, $\setminus$, $\mathtt{coa}$ *(mais non pour* $\mathtt{sya}$*).*

3. *La classe des aspects est close pour* $\cup$ *et* $\cap$ *(mais pas pour* $\setminus$, $\mathtt{coa}$ *et* $\mathtt{sya}$*).*

4. *L'ensemble de toutes les co-actions d'un ensemble de communication est un ensemble propre de communication.*

5. *Si* $a \approx_A b$, *alors* $a \in A \Leftrightarrow b \in A$ *pour tout ensemble de communication propre* A.

Preuve. En utilisant la Proposition II.1.

1. On a $\mathtt{sya}(A) \cup \mathtt{coa}(A) \subseteq A$, donc $B = \mathtt{coa}(A) \subseteq A$, ce qui implique
$\mathtt{coa}(\mathtt{coa}(A)) = \mathtt{coa}(B) \subseteq \mathtt{coa}(A) \subseteq A$.

2. ($\cup$) Si $\mathtt{coa}(\mathtt{coa}(A)) \subseteq A$ et $\mathtt{coa}(\mathtt{coa}(B)) \subseteq B$ (c.-à-d. que A et B sont des ensembles de communication), alors $\mathtt{coa}(\mathtt{coa}(A \cup B)) \subseteq A \cup B$. De même, si $\mathtt{coa}(\mathtt{coa}(A)) = A$ et $\mathtt{coa}(\mathtt{coa}(B)) = B$ (c.-à-d. que A et B sont des ensembles propres de communication), alors $\mathtt{coa}(\mathtt{coa}(A \cup B)) = A \cup B$.

   ($\cap$) Soient A et B, deux ensembles de communication.
   Nous prouvons d'abord que $x \in \mathtt{coa}(\mathtt{coa}(A \cap B)) \Rightarrow x \in A \cap B$ (c.-à-d. $\mathtt{coa}(\mathtt{coa}(A \cap B)) \subseteq A \cap B$).
   $x \in \mathtt{coa}(\mathtt{coa}(A \cap B))$ implique qu'il existe $y \in \mathtt{coa}(A \cap B)$ tel que $x \in \mathtt{coa}(y)$. $y \in \mathtt{coa}(A \cap B)$ implique qu'il existe $z \in A \cap B$ tel que $y \in \mathtt{coa}(z)$. Puisque A et B sont des ensembles de communication, $x \in \mathtt{coa}(\mathtt{coa}(z)) \subseteq A$ et $x \in \mathtt{coa}(\mathtt{coa}(z)) \subseteq B$, donc $x \in \mathtt{coa}(\mathtt{coa}(z)) \subseteq A \cap B$.

Ensuite, nous prouvons que si $A \cap B$ n'est pas un ensemble propre de communication, alors A et B ne le sont pas non plus.

S'il existe un $x \in A \cap B$ tel que $x \notin \mathtt{coa}(\mathtt{coa}(A \cap B))$, alors $\forall y \in \mathtt{coa}(x)$ $coa(y) = \emptyset$. On peut en déduire que $\mathtt{coa}(x) = \emptyset$, car par commutativité de $\mathtt{Syn}$ nous avons $\forall y \in \mathtt{coa}(x)$ que $x \in \mathtt{coa}(y)$. Dans ce cas, ni A ni B ne sont ensembles propres de communication puisque $x \notin \mathtt{coa}(\mathtt{coa}(A))$ et $x \notin \mathtt{coa}(\mathtt{coa}(B))$.

($\setminus$) Soient A et B, deux ensembles de communication.

Nous prouvons d'abord que $x \in \mathtt{coa}(\mathtt{coa}(A \setminus B)) \Rightarrow x \in A \setminus B$ (c.-à-d. $\mathtt{coa}(\mathtt{coa}(A \setminus B)) \subseteq A \setminus B$).

$x \in \mathtt{coa}(\mathtt{coa}(A \setminus B))$ implique qu'il existe $y \in \mathtt{coa}(A \setminus B)$ tel que $x \in \mathtt{coa}(y)$, et $y \in \mathtt{coa}(A \setminus B)$ implique qu'il existe $z \in A \setminus B$ tel que $y \in \mathtt{coa}(z)$. Puisque A et B sont des ensembles de communication, $x \in \mathtt{coa}(\mathtt{coa}(z)) \subseteq A$ et $x \notin B$ car $z \notin B$ implique $\mathtt{coa}(\mathtt{coa}(z)) \cap B = \emptyset$. D'où finalement $x \in A \setminus B$.

Ensuite, nous prouvons que si $A \setminus B$ n'est pas un ensemble propre de communication, alors A ne l'est pas non plus.

S'il existe un $x \in A \setminus B$ tel que $x \notin \mathtt{coa}(\mathtt{coa}(A \setminus B))$, alors $\forall y \in \mathtt{coa}(x)$ $\mathtt{coa}(y) = \emptyset$. On peut en déduire que $\mathtt{coa}(x) = \emptyset$, car par commutativité de $\mathtt{Syn}$ nous avons $\forall y \in \mathtt{coa}(x)$ $x \in \mathtt{coa}(y)$. Dans ce cas, A n'est pas un ensemble propre de communication puisque $x \notin \mathtt{coa}(\mathtt{coa}(A))$.

($\mathtt{coa}$) Soit A un ensemble de communication, c.-à-d. $\mathtt{coa}(\mathtt{coa}(A)) \subseteq A$. Par la monotonie de $\mathtt{coa}$, Prop. II.1(3), $\mathtt{coa}(\mathtt{coa}(\mathtt{coa}(\mathtt{coa}(A)))) \subseteq \mathtt{coa}(\mathtt{coa}(A))$. Si A est un ensemble propre, alors directement

$A = \mathtt{coa}(\mathtt{coa}(A)) = \mathtt{coa}(\mathtt{coa}(\mathtt{coa}(\mathtt{coa}(A))))$.

$\neg$($\mathtt{sya}$) Par contre-exemple. Soit $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ une structure de synchronisaton, où $\mathtt{Act} = \{a, b, c, d\}$, $\mathtt{Hide} = \{d\}$ et $\mathtt{Syn}$ est défini par le tableau suivant :

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & \{b\} & \emptyset & \emptyset & \emptyset \\ b & \emptyset & \{d\} & \{d\} & \emptyset \\ c & \emptyset & \{d\} & \{d\} & \emptyset \\ d & \emptyset & \emptyset & \emptyset & \emptyset \end{array} \tag{4.1}$$

L'ensemble $\{a\}$ est un ensemble de communication tandis que $\mathtt{Syn}(\{a\}) = \{b\}$ ne l'est pas.

3. Conséquence directe de la définition. En utilisant la structure de synchronisation juste définie (4.1), nous vérifions que, même si l'ensemble $A = \{b, c, d\}$ est un aspect, les ensembles $\mathtt{Act} \setminus A$, $\mathtt{coa}(A)$ et $\mathtt{sya}(\mathtt{Act})$ ne sont pas des aspects.

4. Puisque $a \in \mathtt{coa}(b)$ si et seulement si $b \in \mathtt{coa}(a)$ pour toutes les actions a et b (voir Déf. II.1), alors $A \setminus \{a \mid \mathtt{coa}(a) = \emptyset\} = \mathtt{coa}(\mathtt{coa}(A))$ pour tout ensemble de communication A. $\mathtt{coa}(\mathtt{coa}(\mathtt{coa}(A))) \subseteq \mathtt{coa}(A)$ puisque $\mathtt{coa}(\mathtt{coa}(A)) \subseteq A$, donc $\mathtt{coa}(A)$ est un ensemble de communication. Par définition, $\mathtt{coa}(A)$ ne contient que les actions qui peuvent se synchroniser, donc $\mathtt{coa}(A) \cap \{a \mid \mathtt{coa}(a) = \emptyset\} = \emptyset$, c.-à-d. $\mathtt{coa}(A)$ est un ensemble propre de communication.

5. Supposons un ensemble propre de communication A tel que $a \in A$. Par définition d'un ensemble propre de communication, il doit exister une action c telle que $c \in \mathtt{coa}(a)$. Puisque $a \approx_A b$, alors $c \in \mathtt{coa}(b)$, donc $\{a, b\} \subseteq \mathtt{coa}(c)$. Par transitivité, $b \in \mathtt{coa}(\mathtt{coa}(a))$, c.-à-d. $b \in \mathtt{coa}(\mathtt{coa}(A))$. Par définition, $A = \mathtt{coa}(\mathtt{coa}(A))$, donc $b \in A$.

■

En résumé, étant donnée une structure de synchronisation $(\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$, nous avons défini trois types particuliers de sous-ensembles de $\mathtt{Act}$: les *ensembles de communication*, les *ensembles propres de communication* et les *aspects*. Un ensemble de communication peut être considéré comme une caractérisation d'une connexion point-à-point entre deux systèmes réactifs. Il contient toutes les actions qui peuvent être engagées de l'un des deux côtés de la connexion pour une communication entre les deux systèmes. Les actions à partir de l'autre côté de la connexion peuvent toujours être recalculées par l'opération `coa`. Les ensembles propres de communication ont en plus une propriété qui dit que deux connexions qui se décrivent par deux ensembles propres de communication différents sont effectivement différentes.

Dans la mesure où l'on ne s'intéresse pas aux directions de communications point-à-point ou que l'on considère des synchronisations multiples, les ensembles de communication sont remplacés par des aspects. Un aspect peut être vu comme une sous-structure de la structure de synchronisation $\mathcal{S}$ de départ, qui préserve fidèlement les capacités de toutes ses actions.

Exemple II.3 *Dans le cas de CCS muni de la structure de synchronisation définie dans l'Exemple II.1, les notions abstraites de* co-*actions,* sy-*actions, ensembles de communication et aspects sont très simples. Pour tout $x \in \{a, b, \dots\}$ nous avons :*

1. $\mathtt{coa}(x) = \{\overline{x}\}$, $\mathtt{coa}(\overline{x}) = \{x\}$, *et* $\mathtt{coa}(\tau) = \{\}$.
2. $\mathtt{sya}(x) = \mathtt{sya}(\overline{x}) = \{\tau\}$, *et* $\mathtt{sya}(\tau) = \{\}$.
3. *Tout sous-ensemble de* $\mathtt{Act}$ *ou de* $\mathtt{Act} \setminus \{\tau\}$ *est respectivement un ensemble de communication ou un ensemble propre de communication.*
4. *Tout aspect A non vide est caractérisé par :*
   (a) $\tau \in A$
   (b) $x \in A \Rightarrow \overline{x} \in A$

4.2 Synchronisation de Multi-actions

Dans cette section, nous analyserons les capacités de synchronisation des *multi-actions*, c'est-à-dire des exécutions simultanées de plusieurs actions atomiques.

Plus formellement, étant donnée une structure de synchronisation $\mathcal{S} = (\texttt{Act}, \texttt{Syn}, \texttt{Hide})$, une *multi-action* sur $\mathcal{S}$ est un multi-ensemble fini sur $\texttt{Act}$, c.-à-d. un élément de $\mu_f(\texttt{Act})$. (Pour les définitions et notations relatives aux multi-ensembles, le lecteur est invité à se reporter à l'Annexe A.)

4.2.1 Relations de synchronisation partielle

Prenons une situation où deux multi-actions m_1 et m_2 se synchronisent sur une action atomique x du côté de m_1 et une action atomique y du côté de m_2, c.-à-d. que $x \in m_1$, $y \in m_2$ et qu'il existe une action z tel que $z \in \texttt{Syn}(x, y)$. Le résultat de cette synchronisation va consister alors en la multi-action $(m_1 + m_2 + \{z\}) \setminus \{x, y\}$, c.-à-d. en l'exécution simultanée de toutes les actions de m_1 et m_2 où les actions x et y se synchronisent en z.

Nous formalisons cette situation par la définition suivante de *relation de synchronisation de multi-action à lien unique* (ang. *one-link multi-action synchronization relation*).

Définition II.8 Soit A un ensemble de communication d'une structure de synchronisation $\mathcal{S} = (\texttt{Act}, \texttt{Syn}, \texttt{Hide})$. La *relation de synchronisation de multi-action à lien unique* par rapport à A est l'ensemble $R_A \subseteq \mu_f(\texttt{Act}) \times \mu_f(\texttt{Act}) \times \mu_f(\texttt{Act})$:

$$R_A \stackrel{\text{def}}{=} \{(m+\{x\}, n+\{y\}, m+n+\{z\}) \mid m, n \in \mu_f(\texttt{Act}),\ (x, y, z) \in \texttt{Syn},\ x \in A\}$$

□

Notons que la *relation de synchronisation de multi-action à lien unique* R_A n'est pas forcément commutative même si $\texttt{Syn}$ l'est par définition, car, en général, $x \in A$ n'implique pas $\texttt{coa}(x) \subseteq A$.

En développant cette idée, nous pouvons considérer plusieurs sortes de relations de synchronisation de multi-actions, en variant le *type* et le *nombre de liens* (c.-à-d. le nombre de synchronisations élémentaires au niveau d'actions atomiques). Le *type* spécifie quelles actions atomiques vont pouvoir se synchroniser ; cela va faire intervenir la notion d'ensembles de communication. Le *nombre de liens* spécifie combien de paires d'actions atomiques peuvent être impliquées dans une synchronisation à la fois.

Par exemple, pour la structure de synchronisation de CCS (Exemple II.1), la synchronisation de deux multi-actions $\{a, b\}$ et $\{\overline{a}, \overline{b}\}$ peut donner :

- $\{\tau, b, \overline{b}\}$, si a et $\overline{a}$ se synchronisent;
- $\{a, \overline{a}, \tau\}$, si b et $\overline{b}$ se synchronisent; ou
- $\{\tau, \tau\}$, si les deux paires, $(a, \overline{a})$ et $(b, \overline{b})$ se synchronisent simultanément.

Nous considérons plusieurs types de relations de synchronisation sur multi-actions par rapport au nombre de liens : R_A – à lien unique (Définition II.8); R_A^k – à un nombre k fixé de liens; et R_A^+ – à un nombre arbitraire de liens.

Nous définissons aussi la relation R_A^0, qui exprime une exécution simultanée mais sans synchronisation de deux multi-actions. Formellement, une fois la *relation de synchronisation* R_A donnée, nous définissons les relations R_A^k (pour $k \geq 0$), R_A^+ et R_A^* comme suit :

1. $R_A^0 \stackrel{\text{def}}{=} \{(m, m', m+m') \mid m, m' \in \mu_f(\mathtt{Act})\}$ simultanéité
2. $R_A^1 \stackrel{\text{def}}{=} R_A$ synchronisation à lien unique
3. $R_A^k \stackrel{\text{def}}{=} R_A +_3 R_A^{k-1}$ pour $k > 1$ synchronisation à k liens
4. $R_A^+ \stackrel{\text{def}}{=} \bigcup_{k \geq 1} R_A^k$ synchronisation multi-lien
5. $R_A^* \stackrel{\text{def}}{=} \bigcup_{k \geq 0} R_A^k$ (ou $R_A^* = R_A^+ \cup R_A^0$)

où l'opération $+_3$ dans le contexte des relations ternaires P et Q sur des multi-ensembles, est définie par (voir aussi l'Annexe A) :

$$P +_3 Q \stackrel{\text{def}}{=} \{(x_1 + y_1, x_2 + y_2, x_3 + y_3) \mid (x_1, x_2, x_3) \in P, (y_1, y_2, y_3) \in Q\}$$

Indépendamment de la structure de synchronisation $\mathcal{S}$ sur les actions atomiques, les relations de synchronisation partielle sont riches en propriétés.

Proposition II.3 *Pour tous les ensembles de communication A, B d'une structure de synchronisation $\mathcal{S}$, et pour tous $k, l \geq 0$:*

1. $R_A^0 = R_B^0 = R_A^0 +_3 R_B^0$
2. $(n_1, n_2, m) \in R_A^k \;\Leftrightarrow\; (n_2, n_1, m) \in R_{\mathtt{coa}(A)}^k$
3. *Si* $\mathtt{coa}(A) \subseteq A$ *alors* R_A^k *est commutatif au sens fonctionnel de Déf. II.1.*
4. $R_{A \cup B}^k = \bigcup_{0 \leq i \leq k} (R_A^i +_3 R_B^{k-i})$
5. $R_{A \cup B}^* = R_A^* +_3 R_B^*$

Preuve.

1. Directement par définition. Un triplet (n_1, n_2, m) est en relation de *simultanéité* R_A^0, si et seulement si $m = n_1 + n_2$, indépendamment de l'ensemble de communication A.

   $R_A^0 \subseteq R_A^0 +_3 R_A^0$, car $(\emptyset, \emptyset, \emptyset) \in R_A^0$

   $R_A^0 +_3 R_A^0 \subseteq R_A^0$. Si $(n_1, n_2, m) \in R_A^0 +_3 R_A^0$ alors il existe des multi-actions n_1', n_1'', n_2', n_2'', telles que $n_1' + n_1'' = n_1$, $n_2' + n_2'' = n_2$ et $m = n_1' + n_1'' + n_2' + n_2''$. C.-à-d. $m = n_1 + n_2$, et donc $(n_1, n_2, m) \in R_A^0$.

2. Par induction sur k. Pour $k = 0$ la formule est vraie puisque la somme multi-ensemble est commutative. Pour $k = 1$ la formule est vraie puisque la relation de synchronisation $\mathtt{Syn}$ est commutative (voir Définition II.1) et que $b \in \mathtt{coa}(a) \Leftrightarrow a \in \mathtt{coa}(b)$.

   Hypothèse : Supposons que $(n_1, n_2, m) \in R_A^k \Leftrightarrow (n_2, n_1, m) \in R_{\mathtt{coa}(A)}^k$ pour tout $k \leq l$ pour un $l \geq 1$.

   $(n_1, n_2, m) \in R_A^{l+1}$ si et seulement si il existe des multi-actions n_1', n_2', m', n_1'', n_2'', m'' tels que $n_1' + n_1'' = n_1$, $n_2' + n_2'' = n_2$, $m' + m'' = m$, $(n_1', n_2', m') \in R_A$ et $(n_1'', n_2'', m'') \in R_A^l$. Par hypothèse, $(n_2', n_1', m') \in R_{\mathtt{coa}(A)}$ et $(n_2'', n_1'', m'') \in R_{\mathtt{coa}(A)}^l$. Par définition de $R_{\mathtt{coa}(A)}^{l+1}$, nous avons $(n_2' + n_2'', n_1' + n_2'', m' + m'') = (n_2, n_1, m) \in R_{\mathtt{coa}(A)}^{l+1}$.

3. Conséquence du point précédent et du fait que $A \subseteq B$ implique $R_A^k \subseteq R_B^k$.

   Remarquons que, par définition, tout aspect A est un ensemble de communication qui vérifie $\mathtt{coa}(A) \subseteq A$, donc R_A^k est commutatif pour tout aspect A.

4. Par induction sur k.

   Pour $k = 0$ la formule $R_{A\cup B}^k = \bigcup_{0 \leq i \leq k} (R_A^i +_3 R_B^{k-i})$ est évidemment vraie, puisque $R_{A\cup B}^0 = R_A^0 +_3 R_B^0$.

   Pour $k = 1$: D'abord nous montrons que $R_A +_3 R_B^0 = R_A$.

   $R_A \subseteq R_A +_3 R_B^0$, car $(\emptyset, \emptyset, \emptyset) \in R_B^0$

   Si $(n_1, n_2, m) \in R_A +_3 R_B^0$ alors il existe des actions $x \in A$, $y, z \in \mathtt{Act}$, et des multi-actions n_1', n_1'', n_2', n_2'' telles que $(x, y, z) \in \mathtt{Syn}$, $n_1' + \{x\} + n_1'' = n_1$, $n_2' + \{y\} + n_2'' = n_2$ et $n_1' + n_2' + \{z\} + n_1'' + n_2'' = m$. D'où

   $(n_1, n_2, m) = (n_1' + n_1'' + \{x\}, n_2' + n_2'' + \{y\}, n_1' + n_1'' + n_2' + n_2'' + \{z\}) \in R_A$.

   Donc, directement $R_{A\cup B}^1 = R_A \cup R_B = (R_A^1 +_3 R_B^0) \cup (R_A^0 +_3 R_B^1)$.

   Hypothèse : Supposons que $R_{A\cup B}^k = \bigcup_{0 \leq i \leq k} (R_A^i +_3 R_B^{k-i})$ pour tout $k \leq l$ et pour un $l \geq 1$ quelconque.

   L'opération $+_3$ dans le contexte de relations ternaires sur des multi-ensembles est commutative, associative et distributive par rapport à l'union : $P +_3 Q = Q +_3 P$, $(P +_3 Q) +_3 R = P +_3 (Q +_3 R)$ et $(P \cup Q) +_3 R = (P +_3 R) \cup (Q +_3 R)$ (voir l'Annexe A).

$$
\begin{aligned}
R_{A\cup B}^{l+1} &= R_{A\cup B} +_3 R_{A\cup B}^l \\
&\overset{\text{par Hyp.}}{=} (R_A \cup R_B) +_3 \bigcup_{0 \leq i \leq l} (R_A^i +_3 R_B^{l-i}) \\
&= \bigcup_{0 \leq i \leq l} (R_A^{i+1} +_3 R_B^{l-i}) \cup \bigcup_{0 \leq i \leq l} (R_A^i +_3 R_B^{l+1-i}) \\
&= \bigcup_{1 \leq i \leq l+1} (R_A^i +_3 R_B^{l+1-i}) \cup \bigcup_{0 \leq i \leq l} (R_A^i +_3 R_B^{l+1-i}) \\
&= \bigcup_{0 \leq i \leq l+1} (R_A^i +_3 R_B^{l+1-i})
\end{aligned}
$$

5. Nous prouvons que $R^*_{A\cup B} = R^*_A +_3 R^*_B$ en montrant que $R^*_{A\cup B} \supseteq R^*_A +_3 R^*_B$ et $R^*_{A\cup B} \subseteq R^*_A +_3 R^*_B$.

   $\underline{R^*_{A\cup B} \supseteq R^*_A + R^*_B}$: Si un triplet $(n_1, n_2, m) \in R^*_A +_3 R^*_B$ alors il existe $k, l \geq 0$ tels que $(n_1, n_2, m) \in R^k_A +_3 R^l_B$ donc $(n_1, n_2, m) \in R^{k+l}_{A\cup B} \subseteq R^*_{A\cup B}$.

   $\underline{R^*_{A\cup B} \subseteq R^*_A +_3 R^*_B}$: Dans l'autre sens, si $(n_1, n_2, m) \in R^*_{A\cup B}$ alors il existe un k, tel que $(n_1, n_2, m) \in R^k_{A\cup B}$ et donc il doit exister l, l' tels que $l+l' = k$ et $(n_1, n_2, m) \in R^l_A +_3 R^{l'}_B$. Puisque l'opération $+_3$ sur les relations ternaires de multi-ensembles est monotone, alors $(n_1, n_2, m) \in R^*_A +_3 R^*_B$.

   De la même manière on peut prouver : $R^+_{A\cup B} = \left(R^+_A +_3 R^+_B\right) \cup R^+_A \cup R^+_B$. ■

Une conséquence de la proposition est que l'on a $R^k_A = R_A + R^{k-1}_A$ pour tout $k > 0$, de sorte que la différence entre la définition de R^1_A et de R^k_A pour $k > 1$ disparaît.

4.2.2 Auto-synchronisation

Jusqu'à présent, la communication était permise seulement entre les actions de deux multi-actions différentes. Si nous permettons les communications entre les actions d'une même multi-action (*auto-synchronisation*), nous pouvons définir un nouveau type de relations de synchronisation pour l'exprimer. Dans ce cas, une multi-action m représente la possibilité d'exécution simultanée de toutes ses actions atomiques. Une telle exécution peut être accompagnée par des synchronisations entre ces actions. Les synchronisations ne doivent pas forcément être binaires. On peut, par exemple, imaginer que trois ou quatre actions peuvent synchroniser entre elles en donnant comme résultat une seule action atomique. Une manière directe d'exprimer ce type de synchronisation est de considérer les couples (m, n) de multi-ensembles d'actions m et n, et d'interpréter cela par : *les actions de m peuvent se synchroniser, en donnant la multi-action n comme résultat* [Dev94]. Néanmoins, nous pensons que cela rompt avec l'intuition sous-jacente aux systèmes réactifs, seuls capables d'exécuter des actions, et qui, par leur définition, n'admettent que des synchronisations avec leur environnement, donc des synchronisations binaires. Même si nous considérons un ensemble d'agents qui constituent un certain système réactif, chaque agent divise le système en deux parties : lui-même et les autres agents.

Cela veut dire qu'une synchronisation multiple entre plusieurs actions atomiques, même si le résultat est considéré comme instantané, sera construite de manière incrémentale. Par exemple, prenons trois systèmes P, Q et R qui se synchronisent tous sur une action, qui, vue de l'extérieur, est l'action a. Dans ce cas, nous supposerons que les trois systèmes se sont engagés respectivement dans trois actions a_P, a_Q et a_R, telles que :

1. soit a_P a pu se synchroniser avec a_Q en générant une action b telle que la synchronisation de b avec a_R donnait a ;

2. soit a_P a pu se synchroniser avec a_R en générant une action b telle que la synchronisation de b avec a_Q donnait a ;

3. soit a_Q a pu se synchroniser avec a_R en générant une action b telle que la synchronisation de b avec a_P donnait a.

Les ensembles de communication, vus comme des extrémités de connexions binaires, ne semblent plus être adéquats pour la caractérisation de contraintes pour l'auto-synchronisation qui ne se limite pas aux synchronisations binaires. C'est pour cela que, dans le contexte des synchronisations "transitives", nous les remplaçons par des *aspects*.

Définition II.9 Soit $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ une structure de synchronisation, et A un aspect de $\mathcal{S}$. Nous définissons la relation d'*auto-synchronisation* $R^@_A$ par rapport à A, comme la plus petite relation binaire sur les multi-actions, $R^@_A \subseteq \mu_f(\mathtt{Act}) \times \mu_f(\mathtt{Act})$, telle que :

1. $(m, m) \in R^@_A$, pour toute multi-action m.

2. Si $(m, n_1 + n_2) \in R^@_A$ et $(n_1, n_2, m') \in R_A$, alors $(m, m') \in R^@_A$.

$\square$

Autrement dit, une paire de multi-actions (m, m') appartient à $R^@_A$ s'il existe une suite (éventuellement vide) de *pas d'auto-synchronisation* dans l'aspect A qui mène de la multi-action m à m'. Un pas d'auto-synchronisation dans A peut être défini comme :

$$n + \{a, b\} \quad \xrightarrow{A} \quad n + \{c\},$$

où $n \in \mu_f(\mathtt{Act})$, $a, b, c \in A$ et $c \in \mathtt{Syn}(a, b)$.

On peut voir les pas d'auto-synchronisation comme des *"réactions"* entre des *"molécules"* au sens de la *"Chemical Abstract Machine"* de Berry et Boudol [BB90], où les *"molécules"* sont des actions atomiques et les *"lois de réactions"* (ang. *reaction laws*) sont définies via la structure de synchronisation.

Proposition II.4 *Pour tous les aspects A, B d'une structure de synchronisation $\mathcal{S}$ nous avons :*

1. $R^@_A \cdot R^@_A = R^@_A$,

   où $\cdot$ dénote la composition standard de deux relations binaires (c.-à-d. $(\alpha, \beta) \in R^@_A \cdot R^@_A$ si et seulement si il existe un γ tel que $(\alpha, \gamma) \in R^@_A$ et $(\gamma, \beta) \in R^@_A$).

2. $R^@_{A \cup B} = R^@_A +_2 R^@_B$.

   L'opération $+_2$ est définie de manière analogue à $+_3$, mais dans le contexte des relations binaires sur multi-ensembles (voir aussi l'Annexe A).

3. $A \subseteq B \Rightarrow R_A^{@} \subseteq R_B^{@}$.

4. $R_A^{@} = R_A^{@} +_2 \mathbf{I} = R_A^{@} +_2 R_A^{@}$

où $\mathbf{I}$ *(pour Identité) est défini par* $\mathbf{I} \stackrel{def}{=} \{(\alpha, \alpha) \mid \alpha \in \mu_f(\mathtt{Act})\}$.

Preuve de la Proposition II.4

1. $\underline{R_A^{@} \cdot R_A^{@} \subseteq R_A^{@}}$ **:** Remarquons d'abord que $(\alpha, \beta) \in R_A^{@}$ si et seulement si il existe une suite (éventuellement vide) de pas d'auto-synchronisation dans A, qui mène de α à β :

$$\alpha \xrightarrow{A} \alpha_1 \xrightarrow{A} \ldots \xrightarrow{A} \alpha_k \xrightarrow{A} \beta$$

pour certaines multi-actions $\alpha_1, \ldots, \alpha_k$.
Si $(m, n') \in R_A^{@}$ et $(n', n) \in R_A^{@}$, alors

$$m \xrightarrow{A} \ldots \xrightarrow{A} n' \xrightarrow{A} \ldots \xrightarrow{A} n.$$

Ce qui montre qu'il existe une suite de pas d'auto-synchronisation dans A qui mène de m à n. Donc $(m, n) \in R_A^{@}$.

$\underline{R_A^{@} \subseteq R_A^{@} \cdot R_A^{@}}$ **:** Immédiat, puisque $\mathbf{I} \subseteq R_A^{@}$ pour tout A.

2. Nous montrons que $R_{A \cup B}^{@} \subseteq R_A^{@} +_2 R_B^{@}$ et que $R_A^{@} +_2 R_B^{@} \subseteq R_{A \cup B}^{@}$:

$\underline{R_{A \cup B}^{@} \subseteq R_A^{@} +_2 R_B^{@}}$ **:** Si $(m, n) \in R_{A \cup B}^{@}$ alors il existe des multi-actions m_1, $\ldots$, m_k telles que la suite

$$m \xrightarrow{A \cup B} m_1 \xrightarrow{A \cup B} \ldots \xrightarrow{A \cup B} m_k \xrightarrow{A \cup B} n$$

est une suite de pas d'auto-synchronisation.
Notons m^A la multi-action m restreinte aux actions de A, c.-à-d. :

$$\forall x \in \mathtt{Act} \quad m^A(x) \stackrel{\text{def}}{=} \begin{cases} m(x) & \text{si } x \in A \\ 0 & \text{sinon} \end{cases}$$

Si nous notons m^B la multi-action $m \setminus m^A$, alors nous pouvons réécrire la suite de pas d'auto-synchronisation par :

$$m^A + m^B \xrightarrow{A \cup B} m_1^A + m_1^B \xrightarrow{A \cup B} \ldots \xrightarrow{A \cup B} m_k^A + m_k^B \xrightarrow{A \cup B} n^A + n^B$$

Puisque pour un pas d'auto-synchronisation

$$m_i^A + m_i^B \xrightarrow{A \cup B} m_{i+1}^A + m_{i+1}^B$$

soit $m_i^A = m_{i+1}^A$ et dans ce cas nous avons $m_i^B \xrightarrow{B} m_{i+1}^B$, soit $m_i^B = m_{i+1}^B$ et donc $m_i^A \xrightarrow{A} m_{i+1}^A$. Donc,

$$m^A \xrightarrow{A} \ldots \xrightarrow{A} n^A$$

et

$$m^B \xrightarrow{B} \ldots \xrightarrow{B} n^B$$

Nous pouvons alors conclure que $(m^A, n^A) \in R_A^{@}$ et $(m^B, n^B) \in R_B^{@}$ et donc $(m^A + m^B, n^A + n^B) \in R_A^{@} +_2 R_B^{@}$.

$\underline{R^@_A +_2 R^@_B \subseteq R^@_{A\cup B}}$: Si $(m,n) \in R^@_A$ et $(m',n') \in R^@_B$ alors $(m,n) \in R^@_{A\cup B}$ et $(m',n') \in R^@_{A\cup B}$ et donc $(m+m', n+m') \in R^@_{A\cup B}$ et $(n+m', n+n') \in R^@_{A\cup B}$. Par la même proposition (point 1) nous avons $(m+m', n+n') \in R^@_{A\cup B}$.

3. Nous montrons que, si $A \subseteq B$ alors $(m,n) \in R^@_A \Rightarrow (m,n) \in R^@_B$. Immédiat.

4. Remarquons que $R^@_\emptyset = \mathbf{I}$, puisque $R_\emptyset = \emptyset$ (c.-à-d. la relation de synchronisation à un lien par rapport à l'aspect vide est vide).

   $R^@_A = R^@_{A\cup\emptyset} \overset{\text{par point 2}}{=} R^@_A +_2 R^@_\emptyset = R^@_A +_2 \mathbf{I}$

   $R^@_A = R^@_{A\cup A} \overset{\text{par point 2}}{=} R^@_A +_2 R^@_A$

■

Remarquons toutefois, que la *"généralisation"* des synchronisations en allouant constamment des auto-synchronisations peut devenir une restriction pour un système, car une partie du système responsable de l'exécution d'une multi-action n'est jamais capable de contrôler (donc d'empêcher) les communications au sein de la multi-action.

4.2.3 Relations d'abstraction partielle

Nous considérons aussi une classe de relations binaires particulières sur les multi-actions, appelées *relations d'abstraction partielle* (ang. *partial hiding relations*). De telles relations vont exprimer la possibilité de restreindre les capacités de synchronisation des multi-actions par rapport à un ensemble de communication. Une multi-action m peut être cachée (ang. *hidden*) par rapport à un ensemble de communication A, si toutes les actions de l'ensemble A qui apparaissent dans la multi-action m peuvent s'exécuter de manière asynchrone (c.-à-d. sont dans `Hide`).

La motivation de considérer les relations binaires dans le contexte des multi-actions est de permettre une abstraction *partielle*.

Définition II.10 Soit A, un ensemble de communication de $\mathcal{S} = (\texttt{Act}, \texttt{Syn}, \texttt{Hide})$. Une *relation d'abstraction partielle* par rapport à A est un ensemble $H_A \subseteq \mu_f(\texttt{Act}) \times \mu_f(\texttt{Act})$ qui, pour toute action $x \in \texttt{Act}$, satisfait :

$$(m,n) \in H_A \Leftrightarrow \begin{cases} x \notin A & \Rightarrow \quad n(x) = m(x) \\ x \in A & \Rightarrow \quad n(x) = 0 \ \wedge \ (m(x) = 0 \ \vee \ x \in \texttt{Hide}) \end{cases}$$

n est alors appelée la *version abstraite* de m par rapport à A. □

La relation d'abstraction partielle H_A peut être vue comme une fonction partielle $H_A : \mu_f(\texttt{Act}) \mapsto \mu_f(\texttt{Act})$, puisqu'une multi-action ne peut avoir qu'une seule version abstraite (par rapport à A).

La proposition suivante justifie l'utilisation de l'adjectif "partiel" dans le nom de la relation.

Proposition II.5 *Pour tous les ensembles de communication A et B de $\mathcal{S}$, et toutes les multi-actions $m, m' \in \mu_f(\texttt{Act})$ nous avons :*

$$(m, m') \in H_{A\cup B} \quad \Leftrightarrow \quad \exists n \in \mu_f(\texttt{Act}), \quad (m, n) \in H_A \wedge (n, m') \in H_B$$

ou, si l'on utilise la notation des algèbres de relations : $H_{A\cup B} = H_A \cdot H_B$.

Preuve. Puisque toute *relation d'abstraction partielle* est une fonction partielle, alors il suffit de vérifier que :

1. $H_{A\cup B}(m)$ est définie si et seulement si $H_B(H_A(m))$ l'est aussi.
   Facile, puisque $H_{A\cup B}(m)$ et $H_B(H_A(m))$ sont définies si et seulement si :
   $\forall x \in \texttt{Act} \;\; x \in (A \cup B) \setminus \texttt{Hide} \;\Rightarrow\; m(x) = 0$.
2. Si $H_{A\cup B}(m)$ est défini, alors $H_{A\cup B}(m) = H_B(H_A(m))$. Immédiat. ■

4.2.4 Structures de synchronisation sur des multi-actions

Etant donnée une *structure de synchronisation* $\mathcal{S} = (\texttt{Act}, \texttt{Syn}, \texttt{Hide})$ sur les actions atomiques, nous avons élaboré le concept de structure relationnelle sur les éléments de $\mu_f(\texttt{Act})$, c.-à-d. sur les *multi-actions*, en définissant plusieurs relations de *synchronisation* et d'*abstraction*. Mais, évidemment, cette structure relationnelle ne peut pas être directement considérée comme une structure de synchronisation sur des multi-actions. Nous n'avons pas une seule relation de synchronisation mais plusieurs "relations de synchronisation partielle", et nous avons plusieurs "relations (binaires) d'abstraction partielle", au lieu d'une relation unaire.

On peut envisager plusieurs manières de transformer cette nouvelle structure de relations sur des multi-actions en une structure de synchronisation. Nous proposons ici deux transformations :

Transformation naturelle. La transformation "naturelle" d'une structure de synchronisation $\mathcal{S} = (\texttt{Act}, \texttt{Syn}, \texttt{Hide})$, notée $\mathcal{M}^R(\mathcal{S})$, définit la structure de synchronisation suivante :

$$\mathcal{M}^R(\mathcal{S}) \stackrel{\text{def}}{=} (\mu_f(\texttt{Act}), R^*_{\texttt{Act}}, \mu_f(\texttt{Hide}))$$

La structure de synchronisation $\mathcal{M}^R(\mathcal{S})$ préserve toutes les capacités potentielles de synchronisation des multi-actions de $\mathcal{S}$, dans le sens où tous les cas de figure envisagés pour la synchronisation d'une paire de multi-actions seront possibles dans la structure $\mathcal{M}^R(\mathcal{S})$. Remarquons que même si $\texttt{Hide}$ est vide, $\mu_f(\texttt{Hide})$ contient au moins un élément, la multi-action vide.

Cependant, la transformation $\mathcal{M}^R$ semble ne pas être parfaite. Le résultat de la double application de la transformation, c'est-à-dire la structure de synchronisation sur les "multi-multi-actions" $\mathcal{M}^R(\mathcal{M}^R(\mathcal{S}))$ n'est pas en accord total avec notre intuition. Par exemple, la synchronisation des "multi-multi-actions" $\{\{a,b\}\}$ et $\{\{\overline{a}\},\{\overline{b}\}\}$ dans la double transformation naturelle $\mathcal{M}^R(\mathcal{M}^R(\mathcal{S}_M))$ de la structure de synchronisation $\mathcal{S}_M$ pour CCS, comme définie dans l'Exemple II.1, n'est pas tout à fait la même que la synchronisation de $\{a,b\}$ et $\{\overline{a},\overline{b}\}$ dans $\mathcal{M}^R(\mathcal{S}_M)$, même si intuitivement les multi-multi-actions représentent les exécutions simultanées respectivement de a et b et de $\overline{a}$ et $\overline{b}$. Le problème est dû à l'atomicité de la multi-action $\{a,b\}$ dans $\mathcal{M}^R(\mathcal{S}_M)$.

Transformation via auto-synchronisation. Une autre transformation possible d'une structure de synchronisation $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ en une structure de synchronisation sur multi-actions de $\mathcal{S}$ est la transformation basée sur la relation d'*auto–synchronisation* :

$$\mathcal{M}^@(\mathcal{S}) \stackrel{\text{def}}{=} (\mu_f(\mathtt{Act}), \{(\alpha,\beta,\gamma) \mid (\alpha+\beta,\gamma) \in R^@_{\mathtt{Act}}\}, \mu_f(\mathtt{Hide}))$$

On peut alors prouver qu'une multi-action $\{\alpha_1,\ldots,\alpha_i\}$ est équivalente dans $\mathcal{M}^@(\mathcal{S})$ (par rapport à $\approx_A$) à la multi-action singleton $\{\alpha_1+\ldots+\alpha_i\}$. En conséquence, on a

$$\mathcal{M}^@(\mathcal{S}) \ =_{\mathcal{S}} \ \mathcal{M}^@(\mathcal{M}^@(\mathcal{S}))\ .$$

En général, en passant d'une structure de synchronisation aux multi-actions, nous créons des multi-actions, dont certaines ont les mêmes capacités de synchronisation, même si au niveau des actions atomiques deux actions différentes ne sont jamais équivalentes. C'est-à-dire que le quotient $\mathcal{M}^@(\mathcal{S})/_{\approx_A}$ n'est pas forcément isomorphe à $\mathcal{M}^@(\mathcal{S})$, même si $\mathcal{S}$ est isomorphe à $\mathcal{S}/_{\approx_A}$.

Proposition II.6 *Soit $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ une structure de synchronisation. Si deux multi-actions m et m' de $\mathcal{S}$ ne diffèrent que par des actions invisibles, alors les deux multi-actions sont équivalentes (par rapport à $\approx_A$) dans $\mathcal{M}^R(\mathcal{S})$ et dans $\mathcal{M}^@(\mathcal{S})$. C'est-à-dire, que :*

$$\forall m \in \mu_f(\mathtt{Act})\ \forall n \in \mu_f(\mathtt{Inv})\quad m \approx_A m+n$$

Preuve.

Nous prouvons que $B = \{(m, m+n), (m+n, m) \mid m \in \mu_f(\mathtt{Act}), n \in \mu_f(\mathtt{Inv})\}$ est une bisimulation d'actions dans $\mathcal{M}^R(\mathcal{S})$ et dans $\mathcal{M}^@(\mathcal{S})$.

1. Si $(\alpha,\beta) \in B$ et $\alpha \in \mu_f(\mathtt{Hide})$, alors $\beta \in \mu_f(\mathtt{Hide})$. Immédiat.

2. Si $(\alpha,\beta) \in B$ et $(\alpha,x,y) \in R^*_{\mathtt{Act}}$ (respectivement $(\alpha+x,y) \in R^@_{\mathtt{Act}}$), alors il existe y' tel que $(\beta,x,y') \in R^*_{\mathtt{Act}}$ (respectivement $(\beta+x,y') \in R^@_{\mathtt{Act}}$) et $(y,y') \in B$. Evident. ■

La conséquence immédiate de la proposition est que les actions invisibles, $\mathtt{Inv}$, d'une structure de synchronisation $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$, peuvent être "ignorées" au niveau des multi-actions. Cela veut dire que l'on peut ne considérer que des multi-actions construites sur les actions "visibles", c'est-à-dire les éléments de $\mu_f(\mathtt{Act} \setminus \mathtt{Inv})$, étant donné que pour toute multi-action $m \in \mu_f(\mathtt{Act})$ il existe l'action $\widetilde{m} \in \mu_f(\mathtt{Act} \setminus \mathtt{Inv})$, définie par :

$$\widetilde{m}(x) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{si } x \in \mathtt{Inv} \\ m(x) & \text{sinon} \end{cases}$$

et qui est équivalente (par rapport à $\approx_A$) à m. Dans ce cas les relations de synchronisation et de restriction partielle doivent être restreintes aux éléments de $\mu_f(\mathtt{Act} \setminus \mathtt{Inv})$.

Soit Q un ensemble de k-uplets de multi-actions ($Q \subseteq (\mu_f(\mathtt{Act}))^k$). Nous définissons $\widetilde{Q}$ comme l'ensemble de k-uplets de multi-actions ($\widetilde{Q} \subseteq (\mu_f(\mathtt{Act} \setminus \mathtt{Inv}))^k$) suivant :

$$\widetilde{Q} \stackrel{\text{def}}{=} \{(\widetilde{m_1}, \widetilde{m_2}, \ldots, \widetilde{m_k}) \mid (m_1, m_2, \ldots, m_k) \in Q\} \ .$$

Les équations suivantes prouvent la consistance et la cohérence de l'approche qui au niveau des multi-actions ne considère que les actions "visibles", avec l'approche générale présentée auparavant :

1. $\widetilde{R_A} = \{(m + \{x\}, n + \{y\}, m + n + \widetilde{\{z\}}) \mid m, n \in \mu_f(\mathtt{Act} \setminus \mathtt{Inv}),\ (x, y, z) \in \mathtt{Syn},\ x \in A\}$, c.-à-d. que l'action z n'est ajoutée au résultat que si elle est "visible".

2. $\widetilde{R_A^0} = \{(m, m', m + m') \mid m, m' \in \mu_f(\mathtt{Act} \setminus \mathtt{Inv})\}$,

3. $\widetilde{R_A^1} = \widetilde{R_A}$,

4. $\widetilde{R_A^k} = \widetilde{R_A} +_3 \widetilde{R_A^{k-1}}$ pour $k > 1$,

5. $\widetilde{R_A^+} = \bigcup_{k \geq 1} \widetilde{R_A^k}$,

6. $\widetilde{R_A^*} = \bigcup_{k \geq 0} \widetilde{R_A^k}$,

7. $\widetilde{H_A} = \{(n, m) \mid n \in \mu_f(\mathtt{Act} \setminus \mathtt{Inv}),\ (n, m) \in H_A\}$.

En effet, souvent, dans les formalismes qui utilisent des multi-actions, comme par exemple SCCS, MEIJE ou PBC, les actions invisibles ne sont pas représentées dans les multi-actions.

4.3 Commentaires

Dans ce chapitre, nous avons proposé une formalisation possible pour la description des primitives de communication de systèmes réactifs, basée sur la notion de structures de synchronisation. Nous avons introduit les notions d'*ensembles de communication* et d'*aspects* à l'aide desquelles nous pouvons exprimer de manière abstraite et cohérente différents types de contraintes de communication. Nous avons aussi analysé les possibilités de synchronisation des exécutions simultanées de plusieurs actions, appelées multi-actions, par introduction d'une série de relations de synchronisation sur des multi-actions.

Nous avons proposé une approche originale pour analyser les instructions atomiques d'un point de vue “horizontal”, c'est-à-dire par rapport à leurs possibilités d'interaction avec d'autres instructions.

En effet, après l'introduction par R. Milner du calcul CCS [Mil80], deux différents types d'opérateurs de programme auraient pu être distingués : les opérateurs “verticaux”, qui définissent le flux de control, comme la *récursion*, le *préfixage*, le *choix* ou la *composition parallèle*, et les opérateurs “horizontaux”, qui définissent le mode d'interaction, comme les *actions*, la *restriction* et le *renommage*.

Nous pensons que les constructeurs primitifs “horizontaux” sont généralement un peu négligés par rapport aux constructeurs “verticaux”. Néanmoins, il y a déjà eu des propositions pour développer l'aspect “horizontal”, par exemple dans le concept de CCS avec passage de valeurs [Mil89], ou dans les travaux de Joachim Parrow [Par90]. On peut aussi mentionner les algèbres de processus telles que CIRCAL [Mil85], SCCS [Mil83], MEIJE [AB84] ou PBC [BDH92, Dev93, FK94].

Entre autres, les structures de synchronisation peuvent aussi être utilisées dans le contexte du raffinement–abstraction d'actions atomiques, puisque rien n'empêche de considérer une action comme un processus. Ainsi, une synchronisation de deux telles actions représente une exécution “coopérante” de deux processus. Bien sûr, une telle approche demande une réflexion plus approfondie que celle que nous avons pu mener dans cette thèse.

Les relations de synchronisation et d'abstraction partielle ici définies vont être essentielles pour établir des versions paramétrées d'opérations liées à la composition parallèle dans des systèmes de transitions étiquetées par des multi-actions (Chapitre II.5) et dans PBC (Chapitre II.6). Grâce aux propriétés de ces relations (Propositions II.3, II.4, II.5), nous pourrons prouver la *commutativité* et l'*idempotence* des opérateurs de synchronisation et de restriction partielles, dans chacune des deux approches.

Chapter 5

Composition de Systèmes de Transitions

Dans ce chapitre, nous montrons comment la notion de *structures de synchronisation* peut être utilisée dans le contexte de systèmes de transitions étiquetées. Nous allons considérer deux cas de figure : les systèmes de transitions étiquetés par des actions atomiques et les systèmes de transitions étiquetés par des multi-actions. Dans le premier cas, nous allons redéfinir deux opérations : la composition (avec synchronisation) et la restriction. Dans le cas des systèmes de transitions étiquetés par des multi-actions nous avons la possibilité de séparer la "composition" et la "synchronisation" en deux opérations distinctes, ce qui nous amène à définir trois opérations : la composition (sans synchronisation), la synchronisation et la restriction.

5.1 Systèmes de transitions simples

Par système de transitions simple ($\mathtt{TS}$) nous comprendrons un système de transitions étiqueté, dont les étiquettes sont les actions atomiques d'une structure de synchronisation $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ donnée.

5.1.1 Composition parallèle avec synchronisation

Intuitivement, la composition parallèle avec synchronisation de deux systèmes réactifs construit un nouveau système en rassemblant les deux systèmes et en leur permettant de collaborer via des synchronisations. Dans la mesure où les systèmes réactifs sont représentés par des systèmes de transitions étiquetés, et que les primitives de communication, encodées dans les étiquettes, sont directement définies par une structure de synchronisation, la composition parallèle avec synchronisation de deux systèmes produira un nouveau système, qui va être représenté par un système de transitions dont les éléments vont être calculés à partir des éléments de systèmes de transitions représentant les systèmes de départ. Un exemple d'une telle construction est donné dans la Figure 5.1. L'ensemble des états du système $\mathtt{TS}_1 \mid \mathtt{TS}_2$ est obtenu par le produit Cartésien des états des systèmes $\mathtt{TS}_1$ et $\mathtt{TS}_2$. L'ensemble des transitions du système $\mathtt{TS}_1 \mid \mathtt{TS}_2$ est constitué des transitions de $\mathtt{TS}_1$ (flèches horizontales), des transitions de $\mathtt{TS}_2$ (flèches verticales), et de nouvelles transitions résultant des synchronisations entre les deux systèmes. Dans la Figure 5.1, nous avons représenté une nouvelle transition issue de la synchronisation par une flèche en pointillés. Nous créons une telle transition pour chaque synchronisation possible entre α et β, c.-à-d. pour tout $\gamma \in \mathtt{Syn}(\alpha, \beta)$.

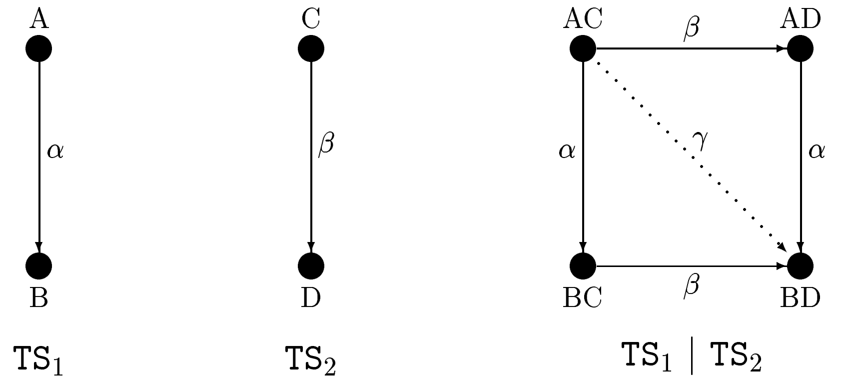


Figure 5.1: Composition parallèle de deux systèmes de transitions simples.

Définition II.11 Soient $\mathtt{TS}_1 = (\mathtt{St}_1, \mathtt{Act}, \rightarrow_1)$ et $\mathtt{TS}_2 = (\mathtt{St}_2, \mathtt{Act}, \rightarrow_2)$ deux systèmes de transitions simples sur une structure de synchronisation $\mathcal{S}$. La composition parallèle avec synchronisation de ces deux systèmes, $\mathtt{TS}_1 \mid \mathtt{TS}_2$, représente le système de transitions simple suivant :

$$\mathtt{TS}_1 \mid \mathtt{TS}_2 \stackrel{\text{def}}{=} (\mathtt{St}, \mathtt{Act}, \rightarrow)$$

où :

1. $\mathtt{St} \stackrel{\text{def}}{=} \mathtt{St}_1 \times \mathtt{St}_2$
2. $(s_1, s_2) \xrightarrow{\alpha} (s'_1, s'_2)$ ssi une des conditions suivantes est satisfaite :
   (a) $s_2 = s'_2$ et $s_1 \xrightarrow{\alpha}_1 s'_1$,
   (b) $s_1 = s'_1$ et $s_2 \xrightarrow{\alpha}_2 s'_2$,
   (c) $s_1 \xrightarrow{\alpha_1}_1 s'_1$, $s_2 \xrightarrow{\alpha_2}_2 s'_2$ et $\alpha \in \mathtt{Syn}(\alpha_1, \alpha_2)$.

□

Remarquons que la composition ainsi définie est toujours commutative mais n'est pas forcément associative, ce qui est dû à la structure de synchronisation qui peut ne pas être "associative". Pour que la composition ainsi définie soit associative, la condition nécessaire et suffisante est que pour toutes actions $\alpha, \beta, \gamma \in \mathtt{Act}$ les deux ensembles suivants soient égaux :

$$\bigcup_{x \in \mathtt{Syn}(\alpha,\beta)} \mathtt{Syn}(x, \gamma) \quad = \quad \bigcup_{x \in \mathtt{Syn}(\beta,\gamma)} \mathtt{Syn}(\alpha, x) \ .$$

Dans Définition II.11, nous supposons qu'il n'y a aucune restriction dans la communication entre les deux systèmes composants. Il nous semble intéressant d'envisager une possibilité de spécifier de telles contraintes. La notion d'*ensembles de communication* peut y être directement utilisée. La composition parallèle avec synchronisation restreinte à un ensemble de communication Q de deux systèmes réactifs S_1 et S_2, notée $S_1 \mid_Q S_2$, va représenter la composition parallèle avec synchronisation de S_1 et S_2, mais dont les actions utilisées par S_1 dans la communication avec S_2 seront limitées aux éléments de Q.

Définition II.12
Soient $\mathtt{TS}_1 = (\mathtt{St}_1, \mathtt{Act}, \rightarrow_1)$ et $\mathtt{TS}_2 = (\mathtt{St}_2, \mathtt{Act}, \rightarrow_2)$ deux systèmes de transitions simples sur une structure de synchronisation $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$, et Q un ensemble de communication de $\mathcal{S}$. La composition parallèle avec synchronisation restreinte à Q de ces deux systèmes, $\mathtt{TS}_1 \mid_Q \mathtt{TS}_2$, représente le système de transitions simple suivant :

$$\mathtt{TS}_1 \mid_Q \mathtt{TS}_2 \stackrel{\text{def}}{=} (\mathtt{St}, \mathtt{Act}, \rightarrow)$$

où :

1. $\mathtt{St} \stackrel{\text{def}}{=} \mathtt{St}_1 \times \mathtt{St}_2$
2. $(s_1, s_2) \stackrel{\alpha}{\rightarrow} (s_1', s_2')$ ssi une des conditions suivantes est satisfaite :
   (a) $s_2 = s_2'$ et $s_1 \stackrel{\alpha}{\rightarrow}_1 s_1'$,
   (b) $s_1 = s_1'$ et $s_2 \stackrel{\alpha}{\rightarrow}_2 s_2'$,
   (c) $\alpha_1 \in Q$, $s_1 \stackrel{\alpha_1}{\rightarrow}_1 s_1'$, $s_2 \stackrel{\alpha_2}{\rightarrow}_2 s_2'$ et $\alpha \in \mathtt{Syn}(\alpha_1, \alpha_2)$.

□

Au premier abord, une telle définition de $S_1 \mid_Q S_2$ peut sembler étrange, d'autant plus qu'elle n'est pas a priori symétrique, puisque l'ensemble de communication Q s'applique uniquement à l'argument gauche, c.-à-d. au système S_1. Pourtant, il suffit de se rappeler de la définition d'un ensemble de communication pour remarquer que limiter les actions de S_1 aux éléments de Q dans la communication avec S_2 revient au même que limiter les actions de S_2 aux actions de $\mathtt{coa}(Q)$ dans la communication avec S_1. Cela se résume par la proposition suivante.

Proposition II.7 *Soient* $\mathtt{TS}_1$ *et* $\mathtt{TS}_2$ *deux systèmes de transitions simples sur une structure de synchronisation* $\mathcal{S}$*, et* Q *un ensemble de communication de* $\mathcal{S}$*. La composition avec synchronisation restreinte à* Q *est commutative (modulo isomorphisme de graphes et "*$\mathtt{coa}$*–tisation"), c.-à-d. que :*

$$\mathtt{TS}_1 \;|_Q\; \mathtt{TS}_2 = \mathtt{TS}_2 \;|_{\mathtt{coa}(Q)}\; \mathtt{TS}_1$$

Le résultat est une conséquence directe de la Proposition II.3–2. On notera aussi les propriétés suivantes :

1. La composition avec synchronisation restreinte à un aspect A est commutative (par Prop. II.3(3)) :

$$\mathtt{TS}_1 \;|_A\; \mathtt{TS}_2 = \mathtt{TS}_2 \;|_A\; \mathtt{TS}_1 \ .$$

2. La composition avec synchronisation restreinte à un ensemble Q de communication est équivalente (modulo isomorphisme de graphes) à la composition avec synchronisation restreinte à l'ensemble propre de communication $\mathtt{coa}(\mathtt{coa}(Q))$ (par Prop. II.7) :

$$\mathtt{TS}_1 \;|_Q\; \mathtt{TS}_2 = \mathtt{TS}_1 \;|_{\mathtt{coa}(\mathtt{coa}(Q))}\; \mathtt{TS}_2 \ .$$

3.

$$\mathtt{TS}_1 \;|_{\mathtt{Act}}\; \mathtt{TS}_2 = \mathtt{TS}_1 \;|\; \mathtt{TS}_2 \ .$$

5.1.2 Restriction

L'opération de restriction appliquée à un système a pour but de préciser les contraintes de communication du système avec son environnement. C'est une opération d'encapsulation qui rend certaines actions internes au système. Comme pour la composition avec synchronisation restreinte, la restriction va être paramétrée par un ensemble de communication Q. La manière la plus simple est de considérer chaque transition du système ayant une étiquette α dans Q et de la supprimer si α ne peut pas s'exécuter de manière asynchrone ($\alpha \notin \mathtt{Hide}$), et sinon, c.-à-d. si $\alpha \in \mathtt{Hide}$, de remplacer α par une action invisible. Cette approche exige l'existence d'une action invisible (comme définie dans la Def. II.5) dans la structure de synchronisation $\mathcal{S}$ sous-jacente.

Pour définir la restriction sans faire référence à une action invisible (qui peut ne pas exister dans $\mathcal{S}$, ou n'être pas unique) nous proposons la définition suivante.

Définition II.13 Soient $\mathtt{TS} = (\mathtt{St}, \mathtt{Act}, \rightarrow)$ un système de transitions simple sur une structure de synchronisation $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$, et Q un ensemble de communication de $\mathcal{S}$. La restriction de TS par rapport à Q, notée $\mathtt{TS}\backslash Q$ définit le système de transitions étiqueté suivant :

$$\mathtt{TS}\backslash Q \stackrel{\text{def}}{=} (\mathtt{St}, \mathtt{Act}, \rightarrow_1)$$

où la relation de transition $\rightarrow_1$ est la restriction de la relation $\rightarrow_{\mathtt{Hide}}$ (comme définie juste après) aux transitions dont les étiquettes ne sont pas dans Q :

$$s \xrightarrow{\alpha}_1 s' \stackrel{\text{def}}{\iff} s \xrightarrow{\alpha}_{\mathtt{Hide}} s' \text{ et } \alpha \notin Q \ .$$

La relation $\rightarrow_{\mathtt{Hide}}$ est la plus petite partie de $\mathtt{St} \times \mathtt{Act} \times \mathtt{St}$ qui vérifie :

1. $\rightarrow \ \subseteq \ \rightarrow_{\mathtt{Hide}}$
2. si $s \xrightarrow{\alpha}_{\mathtt{Hide}} s'$, $s' \xrightarrow{\beta}_{\mathtt{Hide}} s''$ alors
   (a) si $\alpha \in \mathtt{Hide}$ alors $s \xrightarrow{\beta}_{\mathtt{Hide}} s''$
   (b) si $\beta \in \mathtt{Hide}$ alors $s \xrightarrow{\alpha}_{\mathtt{Hide}} s''$

$\square$

La relation de transition $\rightarrow_{\mathtt{Hide}}$, comme définie dans la Définition II.13, sert à compléter le système de transitions en ajoutant les nouvelles transitions dues à la nature même des actions de Hide, qui peuvent toujours s'exécuter de manière interne.

Dans la Figure 5.2, le système de transitions TS est étiqueté par les actions d'une structure de synchronisation où $\tau \in \mathtt{Hide}$, $a, c, d \notin \mathtt{Hide}$ et $\{\tau\}$ est un ensemble de communication.

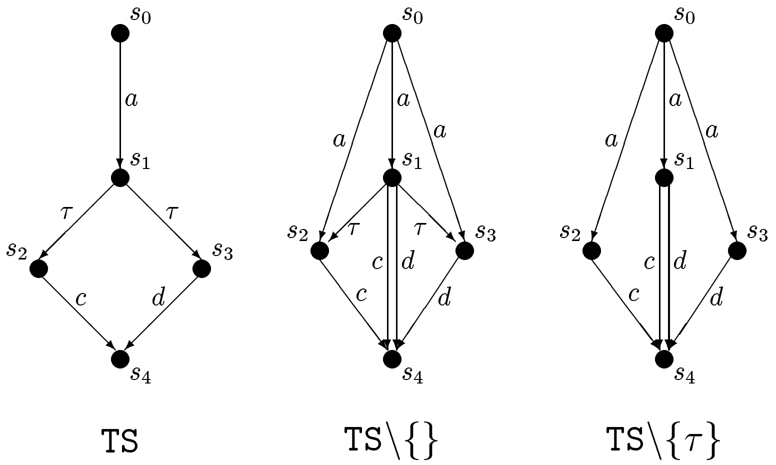


Figure 5.2: Restriction : $\tau \in \mathtt{Hide}$.

Remarquons cependant que si dans TS l'état initial est s_1, dans $\mathtt{TS} \setminus \{\tau\}$ on voudrait dire que les trois états s_1, s_2 et s_3 sont initiaux. Une telle définition de la restriction ne peut donc pas être automatiquement adaptée aux *graphes de processus* comme définis dans la Définition I.2, où un graphe de processus n'a qu'un seul état initial, à moins qu'il n'y ait pas de transitions *internes* (c.-à-d. étiquetées par une action dans Hide) sortant de l'état initial (c.-à-d. si s est un état initial alors $s \xrightarrow{\alpha} s'$ implique que $\alpha \notin \mathtt{Hide}$). On peut aussi changer la définition de *graphe de processus* en autorisant plus qu'un état initial, c'est-à-dire, en remplaçant l'état initial par un ensemble d'états initiaux. Dans ce cas, tout état accessible d'un état initial par une suite d'actions de Hide va aussi être considéré comme état initial. Remarquons aussi que si la structure de synchronisation est celle de CCS (Exemple II.1), alors les trois systèmes de la Figure 5.2 avec l'état initial s_0 sont équivalents par échec mais non par l'équivalence faible.

Nous arrêtons ici notre analyse des opérations sur les systèmes de transitions étiquetés simples, car nous pensons que de tels systèmes ne permettent guère de modéliser des systèmes concurrents de manière satisfaisante. C'est pour cela que nous proposons dans la section suivante une extension de ces systèmes en remplaçant le domaine d'étiquetage des transitions par des *multi-actions*.

5.2 Systèmes de transitions et multi-actions

Les systèmes de transitions étiquetés par des *multi-actions*, c.-à-d. par des multi-ensembles d'actions, sont, du point de vue théorique, un cas particulier de systèmes de transitions simples, puisque nous pouvons toujours considérer une multi-action comme une action atomique d'une structure de synchronisation obtenue par "extension aux multi-actions" (voir Section II.4.2). Néanmoins, la représentation explicite de la simultanéité par multi-actions fait que la composition parallèle de deux systèmes peut se faire de manière beaucoup plus intuitive et plus raffinée.

Dans cette section, nous proposons trois opérations sur les systèmes de transitions étiquetés par des multi-actions : la *composition par produit* (sans synchronisation), la *synchronisation* et la *restriction*.

5.2.1 Opérations sur MTS

Définition II.14 Soit $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ une structure de synchronisation. Un *système de transitions étiqueté avec multi-actions* sur $\mathcal{S}$, MTS, est un système de transitions étiquetés dont les étiquettes sont des multi-ensembles finis sur Act. □

Nous supposons que la structure de synchronisation $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ est donnée et que tous les MTS considérés ont comme étiquettes des multi-actions de $\mu_f(\mathtt{Act})$.

Composition par produit

Soient $\texttt{TS}_1 = (\texttt{St}_1, \mu_f(\texttt{Act}), \rightarrow_1)$ et $\texttt{TS}_2 = (\texttt{St}_2, \mu_f(\texttt{Act}), \rightarrow_2)$ deux $\texttt{MTS}$. La *composition par produit* de $\texttt{TS}_1$ et $\texttt{TS}_2$ est définie par :

$$\texttt{TS}_1 \parallel \texttt{TS}_2 \stackrel{\text{def}}{=} (\texttt{St}, \mu_f(\texttt{Act}), \rightarrow)$$

où :

1. l'ensemble d'états $\texttt{St}$ est le produit Cartésien $\texttt{St}_1 \times \texttt{St}_2$,
2. la relation de transitions $\rightarrow$ contient $(s_1, s_2) \xrightarrow{m} (s'_1, s'_2)$ si et seulement si une des conditions suivantes est satisfaite[1] :
   (a) $s_2 = s'_2$ et $s_1 \xrightarrow{m}_1 s'_1$,
   (b) $s_1 = s'_1$ et $s_2 \xrightarrow{m}_2 s'_2$,
   (c) $s_1 \xrightarrow{n_1}_1 s'_1$, $s_2 \xrightarrow{n_2}_2 s'_2$ et $(n_1, n_2, m) \in R^0$ (voir Section II.4.2.1).

Le $\texttt{MTS}$ résultant de la composition peut exécuter toutes les multi-actions de $\texttt{TS}_1$ et $\texttt{TS}_2$, et en plus les multi-actions qui représentent l'exécution simultanée des multi-actions des deux systèmes. Dans la définition, nous avons noté la somme multi-ensembliste $n_1 + n_2 = m$, par $(n_1, n_2, m) \in R^0$, c.-à-d. comme un élément de la relation de *simultanéité*, pour mettre en évidence le fait que d'autres choix pourraient être facilement envisagés, par exemple, en forçant/en permettant la composition avec synchronisation (voir la Section II.4.2.1 pour la définition de R^0).

Synchronisation

Soit $\texttt{TS}_1 = (\texttt{St}_1, \mu_f(\texttt{Act}), \rightarrow_1)$ un $\texttt{MTS}$ et A un aspect de la structure de synchronisation $\mathcal{S}$. La synchronisation $\texttt{TS}_1\ \texttt{sy}\ A$ du $\texttt{MTS}$ par rapport à A est définie par :

$$\texttt{TS}_1\ \texttt{sy}\ A \stackrel{\text{def}}{=} (\texttt{St}, \mu_f(\texttt{Act}), \rightarrow)$$

où :

1. l'ensemble d'états ne change pas, $\texttt{St} = \texttt{St}_1$;
2. la relation de transitions $\rightarrow$ contient $s_1 \xrightarrow{m} s_2$ si et seulement si
   $\exists n \in \mu_f(\texttt{Act}),\ s_1 \xrightarrow{n}_1 s_2\ \wedge\ (n, m) \in R^{@}_A.$

La synchronisation $\texttt{TS}\ \texttt{sy}\ A$, ajoute dans $\texttt{TS}$ les nouvelles transitions qui représentent les auto-synchronisations de multi-actions de $\texttt{TS}$ dans l'aspect A. Donc, si une transition étiquetée par la multi-action n est dans $\texttt{TS}$, alors pour toute multi-action m telle que $(n, m) \in R^{@}_A$, on ajoute une nouvelle transition dans $\texttt{TS}\ \texttt{sy}\ A$ avec m comme étiquette.

[1] Remarquons que si l'on aurait imposé que pour tout état s il existait une transition $s \xrightarrow{\emptyset} s$ (une boucle via la multi-action vide), les deux premières conditions seraient inutiles.

Restriction

Soit $\texttt{TS}_1 = (\texttt{St}_1, \mu_f(\texttt{Act}), \rightarrow_1)$ un $\texttt{MTS}$. La restriction $\texttt{TS}_1 \texttt{ rs } A$ du $\texttt{MTS}$ par rapport à un aspect A de la structure de synchronisation $\mathcal{S}$, est définie par :

$$\texttt{TS}_1 \texttt{ rs } A \stackrel{\text{def}}{=} (\texttt{St}, \mu_f(\texttt{Act}), \rightarrow)$$

où :

1. l'ensemble d'états ne change pas, $\texttt{St} = \texttt{St}_1$;
2. la relation de transitions $\rightarrow$ contient $s_1 \stackrel{m}{\rightarrow} s_2$ si et seulement si $\exists n \in \mu_f(\texttt{Act}),\ s_1 \stackrel{n}{\rightarrow}_1 s_2 \ \wedge\ (n, m) \in H_A$.

La restriction cache toute action de l'aspect A par le remplacement d'une étiquette de transition par sa version abstraite de A, et si ce n'est pas possible, la transition est supprimée.

5.2.2 Propriétés des opérations sur MTS

Les opérations de *composition*, *synchronisation* et *restriction* de $\texttt{MTS}$ ont des propriétés algébriques intéressantes.

Proposition II.8 *La composition* $\|$ *de* $\texttt{MTS}$ *est associative et commutative (modulo isomorphisme de graphes), c.-à-d. que pour tous les systèmes de transitions sur des multi-actions* $\texttt{TS}_1$*,* $\texttt{TS}_2$ *et* $\texttt{TS}_3$*, nous avons :*

1. $\texttt{TS}_1 \| \texttt{TS}_2 = \texttt{TS}_2 \| \texttt{TS}_1$

2. $(\texttt{TS}_1 \| \texttt{TS}_2) \| \texttt{TS}_3 = \texttt{TS}_1 \| (\texttt{TS}_2 \| \texttt{TS}_3)$

Preuve. Triviale. ∎

Proposition II.9 *L'application de la synchronisation (*$\texttt{sy}$*) est compositionnelle (modulo isomorphisme de graphe), c.-à-d. que pour tout* $\texttt{TS} \in \texttt{MTS}$ *et tous les aspects* A *et* B*, nous avons :*

$$(\texttt{TS sy } A) \texttt{ sy } B = \texttt{TS sy } (A \cup B) \ .$$

Proposition II.10 *L'application de la restriction (*$\texttt{rs}$*) est compositionnelle (modulo isomorphisme de graphe), c.-à-d. que pour tout* $\texttt{TS} \in \texttt{MTS}$ *et tous les aspects* A *et* B*, nous avons :*

$$(\texttt{TS rs } A) \texttt{ rs } B = \texttt{TS rs } (A \cup B) \ .$$

Corollaire II.11 *La synchronisation et la restriction sont commutatives et idempotentes :*

$$\begin{aligned}
(\texttt{TS sy } A) \texttt{ sy } B &= (\texttt{TS sy } B) \texttt{ sy } A\ ,\\
(\texttt{TS sy } A) \texttt{ sy } A &= \texttt{TS sy } A\ ,\\
(\texttt{TS rs } A) \texttt{ rs } B &= (\texttt{TS rs } B) \texttt{ rs } A\ ,\\
(\texttt{TS rs } A) \texttt{ rs } A &= \texttt{TS rs } A\ .
\end{aligned}$$

Proposition II.12 *Soit* TS *un* MTS, *et A et B deux aspects.*

Si $A \subseteq B$, alors :

$$(\texttt{TS rs } B) \texttt{ sy } A = \texttt{TS rs } B\ .$$

Pour faciliter les preuves des Propositions II.9, II.10 et II.12, nous introduisons d'abord la notion de *morphisme* entre MTS et ensuite nous redéfinissons la restriction et la synchronisation en utilisant des morphismes. Cela nous permettra de réduire les preuves des propositions à de simples résolutions d'équations de l'algèbre des relations.

Définition II.15 Soient $\texttt{TS}_1 = (\texttt{St}_1, \mu_f(\texttt{Act}), \rightarrow_1)$, $\texttt{TS}_2 = (\texttt{St}_2, \mu_f(\texttt{Act}), \rightarrow_2)$ deux MTS. Un *morphisme* entre $\texttt{TS}_1$ et $\texttt{TS}_2$ est une paire (ψ, R) (notée $(\psi, R)_{\texttt{TS}_1 \rightarrow \texttt{TS}_2}$) où :

1. R est une relation binaire sur des multi-actions, c.-à-d. $R \subseteq \mu_f(\texttt{Act}) \times \mu_f(\texttt{Act})$;
2. ψ est une bijection de $\texttt{St}_1$ dans $\texttt{St}_2$ qui vérifie pour toutes les multi-actions α et β :
$$p \xrightarrow{\alpha}_1 p' \wedge (\alpha, \beta) \in R \quad \Rightarrow \quad \psi(p) \xrightarrow{\beta}_2 \psi(p')$$

$\square$

On définit aussi la *composition* de deux morphismes par :

$$(\psi_2, R_2)_{\texttt{TS}_2 \rightarrow \texttt{TS}_3} \circ (\psi_1, R_1)_{\texttt{TS}_1 \rightarrow \texttt{TS}_2} \stackrel{\text{def}}{=} (\psi_2 \cdot \psi_1, R_2 \cdot R_1)_{\texttt{TS}_1 \rightarrow \texttt{TS}_3}$$

où $\cdot$ dénote la composition standard de deux fonctions ainsi que la composition standard de deux relations binaires (c.-à-d. $(a, b) \in R_2 \cdot R_1$ si et seulement si il existe un c tel que $(a, c) \in R_1$ et $(c, b) \in R_2$).

Nous appelons un morphisme $(\psi, R)_{\texttt{TS}_1 \rightarrow \texttt{TS}_2}$ *plein*, si le système de transition $\texttt{TS}_2$ n'a que les transitions nécessaires pour que $(\psi, R)_{\texttt{TS}_1 \rightarrow \texttt{TS}_2}$ soit un morphisme, c'est-à-dire que pour toute transition $q \xrightarrow{\beta}_2 q'$ dans $\texttt{TS}_2$, il existe une transition $\psi^{-1}(q) \xrightarrow{\alpha}_1 \psi^{-1}(q')$ dans $\texttt{TS}_1$, telle que $(\alpha, \beta) \in R$.

L'isomorphisme de systèmes de transitions peut alors être exprimé comme suit. *Deux systèmes de transitions* $\texttt{TS}_1$ *et* $\texttt{TS}_2$ *sont isomorphes, noté* $\texttt{TS}_1 = \texttt{TS}_2$, *s'il existe un morphisme plein* $(\psi, \mathbf{I})_{\mathtt{TS}_1 \to \mathtt{TS}_2}$*, où* $\mathbf{I}$ *(pour Identité) est défini par* $\mathbf{I} \stackrel{def}{=} \{(\alpha, \alpha) \mid \alpha \in \mu_f(\mathtt{Act})\}$.

Il n'est pas difficile de vérifier qu'une telle formulation définit effectivement l'isomorphisme de graphes.

En utilisant la notion de morphisme, nous sommes capables de redéfinir la synchronisation et la restriction de `MTS` de la manière suivante.

Soit $\mathcal{S}$ une structure de synchronisation, A un aspect de $\mathcal{S}$, et `TS`, `TS'` deux `MTS` sur $\mathcal{S}$.

1. Le `MTS` `TS'` est (à un isomorphisme près) le résultat de la synchronisation `TS sy` A si et seulement si il existe un morphisme plein $(\psi, R^{@}_A)_{\mathtt{TS} \to \mathtt{TS'}}$, où $R^{@}_A$ est la relation d'auto-synchronisation sur A.

$$\begin{tikzcd} \mathtt{TS} \arrow[r, "{(\psi, R^{@}_A)}"] & \mathtt{TS'} \end{tikzcd}$$

2. Le `MTS` `TS'` est (à un isomorphisme près) le résultat de la restriction `TS rs` A si et seulement si il existe un morphisme plein $(\psi, H_A)_{\mathtt{TS} \to \mathtt{TS'}}$, où H_A est la relation d'abstraction par rapport à A.

$$\begin{tikzcd} \mathtt{TS} \arrow[r, "{(\psi, H_A)}"] & \mathtt{TS'} \end{tikzcd}$$

Preuve de la Proposition II.10

On démontre que le diagramme suivant commute :

$$\begin{tikzcd} & \circ \arrow[dr, "{(g, H_B)}"] & \\ \circ \arrow[ur, "{(f, H_A)}"] \arrow[rr, "{(g \cdot f, H_{A \cup B})}"'] & & \circ \end{tikzcd}$$

Il suffit de prouver que : $H_B \cdot H_A = H_{A \cup B}$, ce qui est un résultat déjà établi (voir Prop. II.5). ■

Preuve de la Proposition II.9

Il suffit de prouver que le diagramme suivant commute :

$$\begin{tikzcd} & \circ \arrow[dr, "{(g, R^{@}_B)}"] & \\ \circ \arrow[ur, "{(f, R^{@}_A)}"] \arrow[rr, "{(g \cdot f, R^{@}_{A \cup B})}"'] & & \circ \end{tikzcd}$$

Il reste à prouver que : $R^{@}_B \cdot R^{@}_A = R^{@}_{A \cup B}$.

$\underline{R^{@}_{B} \cdot R^{@}_{A} \subseteq R^{@}_{A\cup B}}$ **:** Par la Proposition II.4(1), nous avons $R^{@}_{A\cup B} \cdot R^{@}_{A\cup B} \subseteq R^{@}_{A\cup B}$.

Par la Proposition II.4(3), nous avons $R^{@}_{A} \subseteq R^{@}_{A\cup B}$ et $R^{@}_{B} \subseteq R^{@}_{A\cup B}$.

Donc $R^{@}_{B} \cdot R^{@}_{A} \subseteq R^{@}_{A\cup B} \cdot R^{@}_{A\cup B} \subseteq R^{@}_{A\cup B}$.

$\underline{R^{@}_{A\cup B} \subseteq R^{@}_{B} \cdot R^{@}_{A}}$ **:** Par Proposition II.4, nous avons :

$R^{@}_{A} +_2 R^{@}_{B} = R^{@}_{A\cup B}$.

$R^{@}_{A} +_2 R^{@}_{B} \stackrel{\text{def}}{=} \{(x+x', y+y') \mid (x,y) \in R^{@}_{A},\ (x',y') \in R^{@}_{B}\}$

$\Rightarrow\ (x+x', y+x') \in R^{@}_{A} \wedge (x'+y, y'+y) \in R^{@}_{B}$ (par Prop. II.4(4))

$\Rightarrow\ (x+x', y'+y) \in R^{@}_{B} \cdot R^{@}_{A}$

Donc finalement, $R^{@}_{A\cup B} \subseteq R^{@}_{B} \cdot R^{@}_{A}$.

■

Preuve de la Proposition II.12

Il faut prouver que si $A \subseteq B$, alors $R^{@}_{A} \cdot H_B = H_B$.

1. Puisque $\mathbf{I} \subseteq R^{@}_{A}$, $H_B \subseteq R^{@}_{A} \cdot H_B$.

2. Dans l'autre sens, si $(m,n) \in R^{@}_{A} \cdot H_B$, alors il existe une multi-action m' telle que $(m,m') \in H_B$ et $(m',n) \in R^{@}_{A}$. Par définition de H_B, nous savons que $m'(x) = 0$ pour tout $x \in B$. Par supposition, $A \subseteq B$ et donc $(m',n) \in R^{@}_{A}$ seulement si $n = m'$. On peut en déduire que $(m,n) \in R^{@}_{A} \cdot H_B$ seulement si $(m,n) \in H_B$. Donc $H_B \supseteq R^{@}_{A} \cdot H_B$.

■

5.2.3 Un langage de spécification pour `MTS` finis

Les trois opérations proposées, celles de composition, de synchronisation et de restriction, nous permettent déjà de définir un langage $\mathcal{L}(\mathcal{S})$ de spécification de systèmes réactifs. Ce langage est paramétré par une structure de synchronisation $\mathcal{S}$ arbitraire. Les éléments de base du langage seront les systèmes à deux états et à deux transitions étiquetées respectivement par les multi-actions m et n. Intuitivement, le comportement d'un tel système, noté $m \cdot n$, consiste en l'exécution alternée des multi-actions m et n. De tels systèmes pourront alors être composés via ces trois opérations.

Syntaxe

Soit $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ une structure de synchronisation. Un terme (processus) P du langage $\mathcal{L}(\mathcal{S})$ est défini par :

$$P ::= m \cdot n \mid P \parallel P \mid P\ \mathtt{sy}\ A \mid P\ \mathtt{rs}\ A$$

où A est un aspect de $\mathcal{S}$ et m, n sont des multi-actions de $\mathcal{S}$.

Les éléments de base sont des paires de multi-actions. Une telle paire, notée $m \cdot n$, définit un système capable d'exécuter les multi-actions m et n en alternance, en commençant par m. Les éléments de base peuvent ensuite être groupés/modifiés par les trois opérateurs : la *composition* (par produit) ($\|$), la *synchronisation* ($\mathtt{sy}\ A$) et la *restriction* ($\mathtt{rs}\ A$), par rapport à un aspect A de la structure de synchronisation $\mathcal{S}$.

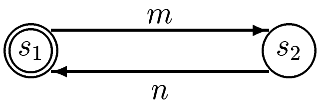
L'ensemble de tous les termes de $\mathcal{L}(\mathcal{S})$ sera noté $\mathcal{P}$.

Sémantique dénotationnelle en termes de graphes de processus

A chaque élément de base, c.-à-d. à chaque terme $m \cdot n$, nous associons le graphe de processus suivant :

$$G(m \cdot n) \stackrel{\text{def}}{=} (\{s_1, s_2\}, \mu_f(\mathtt{Act}), \{(s_1, m, s_2), (s_2, n, s_1)\}, s_1)) \ .$$

Graphiquement, le graphe de processus $G(m \cdot n)$ peut être représenté par :



La composition $P \parallel Q$ est définie de manière compositionnelle, à partir de la sémantique de P et Q, en utilisant la composition parallèle de deux MTS. Soient $G(P) = (\mathtt{TS}_P, s_P)$ et $G(Q) = (\mathtt{TS}_Q, s_Q)$ les graphes de processus associés respectivement aux termes P et Q. C'est-à-dire que $\mathtt{TS}_P$ (resp. $\mathtt{TS}_Q$) est un MTS et s_P (resp. s_Q) est l'état initial de $G(P)$ (resp. $G(Q)$). Alors $G(P \parallel Q)$ est défini comme le graphe de processus $\mathtt{graph}((s_P, s_Q))$ dans $\mathtt{TS}_P \parallel \mathtt{TS}_Q$.

Le modèle de la synchronisation $P\ \mathtt{sy}\ A$ et de la restriction $P\ \mathtt{rs}\ A$ d'un processus P par rapport à un aspect A de $\mathcal{S}$ est défini à partir de la sémantique de P. Soit $G(P) = (\mathtt{TS}_P, s_P)$ le graphe de processus associé au terme P. Alors $G(P\ \mathtt{sy}\ A)$ (resp. $G(P\ \mathtt{rs}\ A)$) est défini comme le graphe de processus $\mathtt{graph}(s_P)$ dans $\mathtt{TS}_P\ \mathtt{sy}\ A$ (resp. dans $\mathtt{TS}_P\ \mathtt{rs}\ A$).

Sémantique opérationnelle

La sémantique de $\mathcal{P}(\mathcal{S})$ peut aussi être exprimée par une sémantique opérationnelle de style SOS, en considérant les règles suivantes :

Multi-Action

$$\frac{}{m \cdot n \xrightarrow{m} n \cdot m}$$

Composition

$$\frac{P \xrightarrow{n_1} P'}{P \parallel Q \xrightarrow{n_1} P' \parallel Q} \qquad \frac{Q \xrightarrow{n_2} Q'}{P \parallel Q \xrightarrow{n_2} P \parallel Q'}$$

$$\frac{P \xrightarrow{n_1} P' \qquad Q \xrightarrow{n_2} Q'}{P \parallel Q \xrightarrow{n_1 + n_2} P' \parallel Q'}$$

Synchronisation

$$\frac{P \xrightarrow{n} P'}{P \texttt{ sy } A \xrightarrow{m} P' \texttt{ sy } A}\ (n,m) \in R_A^{@}$$

Restriction

$$\frac{P \xrightarrow{n} P'}{P \texttt{ rs } A \xrightarrow{m} P' \texttt{ rs } A}\ (n,m) \in H_A$$

La première règle (Multi-Action) dit que le processus $m \cdot n$ peut exécuter la multi-action m, et dans ce cas il devient $n \cdot m$. Les trois règles suivantes (Composition) expriment la composition de deux processus P et Q. Le processus résultant peut éventuellement exécuter les multi-actions de P et Q simultanément, ce qui est exprimé par la somme de multi-ensembles $+$, qui coïncide d'ailleurs avec la relation R^0 (voir Section II.4.2.1). La règle pour la Synchronisation dépend de la relation d'auto-synchronisation $R_A^{@}$ définie dans la Section II.4.2.1. La dernière règle (Restriction) dit que, si un processus P peut exécuter une multi-action n qui peut être cachée par rapport à un aspect A, alors le processus $P \texttt{ rs } A$ peut exécuter m, la version cachée de la multi-action n, c.-à-d. telle que $(n, m) \in H_A$ (voir la Section II.4.2.3).

Les règles de dérivation de type SOS définissent le système de transitions $(\mathcal{P}, \mu_f(\texttt{Act}), \rightarrow)$ sur l'ensemble $\mathcal{P}$ de tous les termes de $\mathcal{L}(\mathcal{S})$. Une transition d'un terme P vers un terme Q via une multi-action m existe, si et seulement si elle peut être prouvée par ces règles. La sémantique d'un terme P est alors définie comme le graphe de processus $\texttt{graph}(P)$ dans le système de transitions $(\mathcal{P}, \mu_f(\texttt{Act}), \rightarrow)$.

Proposition II.13 *Les deux sémantiques sont équivalentes dans le sens où pour tout terme P le graphe de processus $G(P)$ obtenu de manière dénotationnelle est isomorphe avec le graphe de processus $\texttt{graph}(P)$ dans le système de transitions engendré par les règles de dérivation.*

Preuve. Immédiate, par récurrence sur la construction du terme P. ■

Illustrons par un exemple ce qu'on peut faire dans ce langage de spécification.

Exemple II.4 *Supposons que nous voulions décrire le graphe de processus "bool" de la Figure 5.3, où α, β, γ et δ sont des multi-actions de la structure de synchronisation $\mathcal{S}_M$ pour CCS (voir l'Exemple II.1 pour la définition de $\mathcal{S}_M$).*

Pour simplifier les notations, une multi-action sera représentée par un mot simple, ex. "$aa\overline{c}$" pour $\{a, a, \overline{c}\}$, ou "αa" pour $\alpha + \{a\}$, si α est une multi-action. La définition dans $\mathcal{L}(\mathcal{S}_M)$ du graphe de processus `bool` *de la Figure 5.3, modulo l'équivalence faible (c.-à-d. modulo $\approx$ de la Section I.1.1.4, où la multi-action vide, $\emptyset$, est considérée comme invisible), peut être la suivante :*

$$T = \Big(\Big(\alpha\overline{a}\overline{x} \cdot \overline{b}\overline{y} \parallel \beta\overline{b}\overline{x} \cdot \overline{a}\overline{y} \parallel a \cdot b \parallel x \cdot y \parallel \gamma\overline{a} \cdot \delta\overline{b}\Big) \texttt{ sy } A\Big) \texttt{ rs } A$$

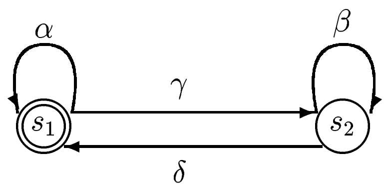


Figure 5.3: Graphe de processus “bool”.

L'aspect A est défini par $\{a, \overline{a}, b, \overline{b}, x, \overline{x}, y, \overline{y}, \tau\}$. Nous supposons aussi que les actions a, $\overline{a}$, b, $\overline{b}$, x, $\overline{x}$, y, $\overline{y}$ ne font pas partie des multi-actions α, β, γ et δ. Les cinq éléments de base du terme T sont représentés dans la Figure 5.4. Après la composition, la synchronisation et la restriction de ces systèmes par

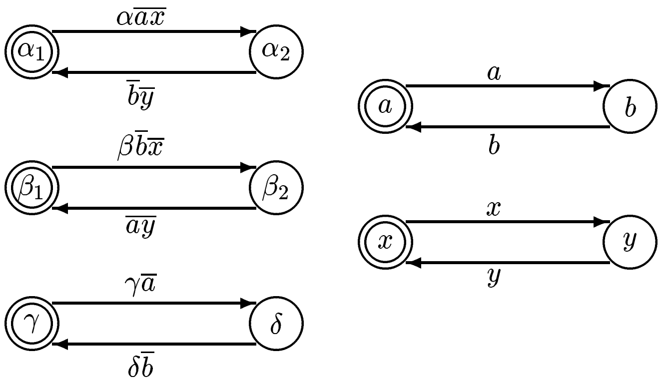


Figure 5.4: Les cinq éléments de base du terme T.

rapport à l'aspect A, nous obtenons le graphe de processus de la Figure 5.5 qui est faiblement équivalent au graphe `bool` *(si nous considérons la multi-action vide $\emptyset$ comme une action interne).*

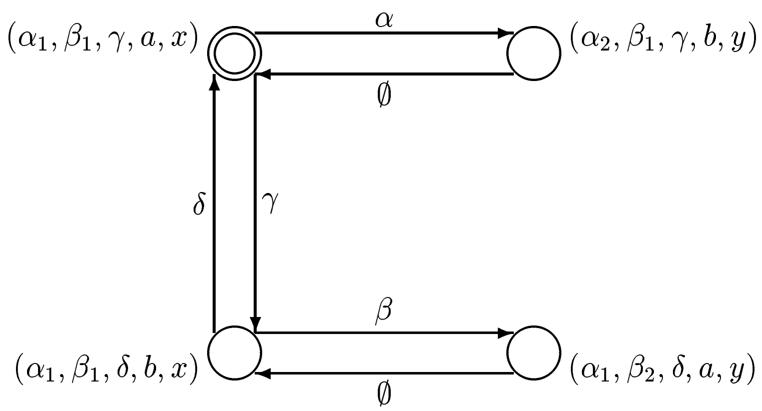


Figure 5.5: Graphe de processus $G(T)$.

Il est également possible de définir un système fortement équivalent à `bool`.

$$\Big(\big(\alpha\overline{a}\overline{x} \parallel \alpha\overline{b}\overline{y} \parallel \beta\overline{a}\overline{y} \parallel \gamma\overline{c}\cdot\delta\overline{c} \parallel \beta\overline{b}\overline{x} \parallel a\cdot b \parallel x\cdot y \parallel c\overline{a} \parallel c\overline{b}\big) \texttt{ sy } C\Big) \texttt{ rs } C$$

où tout terme de base de la forme $m \cdot m$ *a été abrégé par* m*, et où l'aspect* C *est défini comme* $A \cup \{c, \overline{c}\}$*. On remarque tout de même que la taille du système de transitions en nombre d'états, obtenu par le produit Cartésien des états des neuf systèmes à deux états chacun, est importante (*$2^9 = 512$*). Bien évidemment, le nombre d'états du graphe de processus final, après la restriction, diminue considérablement (par suppression des états inaccessibles).*

On peut en déduire que si la structure de synchronisation $\mathcal{S}$ et les propriétés des graphes de base sont suffisamment riches, notre langage semble assez puissant pour décrire n'importe quel `MTS` fini (en nombre d'états et de transitions) modulo l'*équivalence forte*.

Dans ce cas, le langage $\mathcal{L}(\mathcal{S})$ peut alors être vu comme une alternative aux *expressions régulières* (de la théorie de langages) construites via les opérations de *mise en séquence*, d'*union* et d'*itération*, et dont les éléments de base sont les lettres (actions) d'un alphabet. L'avantage du langage $\mathcal{L}(\mathcal{S})$ est que l'équivalence faible telle qu'établie dans la Section I.1.1, est une congruence, ce qui n'est pas le cas pour les expressions régulières, ou plus généralement, pour les formalismes utilisant la composition par choix (c.-à-d. l'union). C'est une voie que l'on pourra explorer dans le futur.

5.3 Commentaires

Dans ce chapitre nous avons analysé l'application des structures de synchronisation dans le contexte de systèmes de transitions étiquetés aussi bien par des actions atomiques que par des multi-actions. Nous avons envisagé quelques opérations de composition qui, d'une part, sont en accord avec notre intuition et, d'autre part, vérifient des propriétés algébriques intéressantes. Toute cette étude va être indirectement explorée dans la dernière partie de cette thèse, où une algèbre de processus, inspirée de celle définie ici, va être proposée, avec un choix beaucoup plus raffiné d'opérations de base et avec une structure particulière de synchronisation.

Chapter 6

Synchronisation et Restriction dans l'Algèbre des Petri-Box

Dans ce chapitre, nous présentons l'application des notions liées aux structures de synchronisation, introduites dans le Chapitre II.4, à la redéfinition intéressante de deux opérateurs de l'Algèbre des Petri-Box, à savoir la *synchronisation* et la *restriction*. Nous détaillons d'abord les définitions originales de ces opérateurs, telles que proposées dans [BDH92], et ensuite nous les redéfinissons en les rendant paramétrables par des structures de synchronisation quelconques.

Enfin, nous montrons que toutes les nouvelles opérations définies via le schéma proposé, préservent toutes les propriétés algébriques qui rendaient les opérations originales tellement attrayantes.

6.1 Synchronisation et Restriction dans PBC

Pour la présentation générale de l'Algèbre des Petri-Box (PBC) le lecteur est invité à se référer à la Section I.3.4. Ici, nous allons seulement détailler les notions nécessaires pour la présente section.

La relation d'équivalence de duplications, $\simeq_D$ (voir Section I.2.2.1), qui définit les objets du domaine sémantique de PBC, appelés *Box*, permet d'ajouter et de supprimer des transitions de communication et des places *dupliquées*, c.-à-d. ayant les mêmes étiquettes et ayant les mêmes connexions.

Dans le contexte des *Box*, toute transition de communication t étant uniquement définie par son étiquette $\lambda(t)$, ses pré-conditions ${}^\bullet t$ et ses post-conditions $t^\bullet$, pourra être identifiée par le triplet $({}^\bullet t, t^\bullet, \lambda(t))$.

Dans ce qui suit, nous nous concentrons seulement sur deux opérateurs de l'algèbre : la *synchronisation* et la *restriction*. Nous n'allons pas non plus considérer les transitions d'abstraction (c.-à-d. les transitions avec une étiquette hiérarchique), qui ne sont pas utilisées par ces opérations (du moins dans notre travail).

Une des originalités de PBC, par comparaison avec d'autres algèbres de processus, est que la composition parallèle et la synchronisation constituent deux opérations bien distinctes.

L'idée intuitive de la synchronisation est qu'elle regroupe certains ensembles de transitions de communication et les connecte entre elles par des *liens de communication*. Un lien entre deux transitions peut être créé chaque fois que deux actions atomiques complémentaires (qui peuvent se synchroniser) font partie des étiquettes de ces deux transitions, respectivement. Dans PBC, les actions complémentaires sont toujours des *paires conjuguées* $(a, \overline{a})$ et leur synchronisation crée toujours une action invisible. Ainsi, par exemple, deux multi-actions $m = \{a, b\}$ et $n = \{\overline{a}, c\}$ pourront se synchroniser via un lien créé par l'action a de m et l'action $\overline{a}$ de n, en donnant la multi-action $\{b, c\}$ comme résultat.

Pendant l'opération de synchronisation, deux transitions de communication peuvent se synchroniser par des liens (dans [BDH92] seulement par un lien unique). Une telle paire de transitions est alors transformée en une nouvelle transition, qui à son tour peut faire partie d'autres synchronisations, même pendant la même opération de synchronisation.

Définition II.16 Soit $B = (S, T, W, \lambda)$ un réseau de Petri étiqueté, et a une action atomique. On définit l'opération de synchronisation standard [BDH92] du réseau B par rapport à l'action a par :

$$B \texttt{ sy } a \stackrel{\text{def}}{=} (S, T', W', \lambda')$$

où T' est le plus petit ensemble de transitions qui vérifie :

1. Si $t \in T$ alors $t \in T'$ avec les mêmes pré-conditions, post-conditions et la même étiquette :

   (a) $\forall s \in S \;\; W'(t, s) \stackrel{\text{def}}{=} W(t, s)$,

   (b) $\forall s \in S \;\; W'(s, t) \stackrel{\text{def}}{=} W(s, t)$,

   (c) $\lambda'(t) \stackrel{\text{def}}{=} \lambda(t)$.

2. Si $t_1, t_2 \in T'$ avec $\lambda'(t_1) = \alpha + \{a\}$ et $\lambda'(t_2) = \beta + \{\overline{a}\}$ alors $t \in T'$, où t est une transition dont les pré- et post-conditions sont la somme multi-ensemble des pré- et post-conditions des deux transitions, respectivement, et dont l'étiquette est $\alpha + \beta$:

   (a) ${}^\bullet t \stackrel{\text{def}}{=} {}^\bullet t_1 + {}^\bullet t_2$,

   (b) $t^\bullet \stackrel{\text{def}}{=} {t_1}^\bullet + {t_2}^\bullet$,

   (c) $\lambda'(t) \stackrel{\text{def}}{=} \alpha + \beta$.

$\square$

La synchronisation d'un réseau B par rapport à une action a ajoute de nouvelles transitions chaque fois qu'il existe deux transitions dont les étiquettes

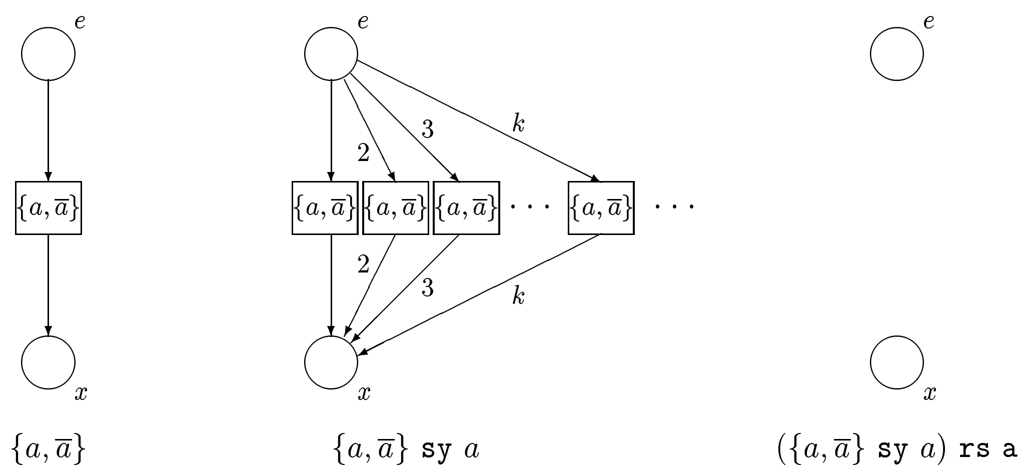


Figure 6.1: Exemple de synchronisation et restriction dans PBC.

comportent respectivement l'action a et son complément $\overline{a}$. L'opération de synchronisation ne donne pas toujours un résultat fini, même si l'argument est fini. Un exemple simple d'une telle situation est illustré dans la Figure 6.1, où une seule transition peut se synchroniser avec un nombre non-borné de ses copies.

La restriction est une opération qui, appliquée à un réseau, réduit les capacités de synchronisation du réseau en enlevant certaines transitions de communication. Plus précisément, comme pour la synchronisation, une restriction est paramétrée par une action et elle consiste à enlever toute transition dont l'étiquette contient l'action en question ou son complément.

Définition II.17 Soit $B = (S, T, W, \lambda)$ un réseau de Petri étiqueté, et a une action atomique. On définit l'opération de restriction du réseau B par rapport à l'action a par :

$$B \;\texttt{rs}\; a \stackrel{\text{def}}{=} (S, T', W', \lambda')$$

où :

1. $T' \stackrel{\text{def}}{=} \{t \in T \mid \lambda(t)(a) = \lambda(t)(\overline{a}) = 0\}$.
2. $W' \stackrel{\text{def}}{=} W$ restreint à $(T' \times S) \cup (S \times T')$.
3. $\lambda' \stackrel{\text{def}}{=} \lambda$ restreint à $T' \cup S$.

□

La synchronisation dans PBC de base (telle que définie dans [BDH92] et présentée ci-dessus) est une extension de la synchronisation de CCS dans le cadre des multi-actions. Cette extension est plutôt restrictive, car elle permet uniquement des synchronisations par un lien unique et elle interdit l'*auto-synchronisation* (c.-à-d. la synchronisation entre les actions d'une même multi-action).

C'est pour cela que la box-expression, $((\{a,\overline{a}\}\ \texttt{sy}\ a)\ \texttt{rs}\ a)$, a comme sémantique le Box "**stop**", qui représente un blocage (voir Figure 6.1).

Une même situation peut se produire quand l'on essaye de construire une synchronisation *multiple* entre deux transitions. Prenons la box-expression suivante :

$$\Big(\Big(\Big(\Big(\{a,b\} \parallel \{\overline{a},\overline{b}\}\Big)\ \texttt{sy}\ a\Big)\ \texttt{sy}\ b\Big)\ \texttt{rs}\ a\Big)\ \texttt{rs}\ b\ .$$

Intuitivement, les deux transitions étiquetées respectivement par $\{a,b\}$ et par $\{\overline{a},\overline{b}\}$ pourraient se synchroniser simultanément par deux liens $(a,\overline{a})$ et $(b,\overline{b})$ en créant une transition invisible étiquetée par $\emptyset$, mais la sémantique de PBC donne un blocage, c.-à-d. le Box "**stop**".

Les résultats principaux des opérations de synchronisation et de restriction, présentés dans [BDH92], se trouvent résumés dans le théorème suivant.

Théorème II.14

1. *L'application de la synchronisation* (`sy`) *est :*
   (a) *commutative :* $B\ \texttt{sy}\ a\ \texttt{sy}\ b \simeq_D B\ \texttt{sy}\ b\ \texttt{sy}\ a$.
   (b) *symétrique par conjugaison :* $B\ \texttt{sy}\ a \simeq_D B\ \texttt{sy}\ \overline{a}$.
   (c) *idempotente :* $B\ \texttt{sy}\ a \simeq_D B\ \texttt{sy}\ a\ \texttt{sy}\ a$.
2. *L'application de la restriction* (`rs`) *est :*
   (a) *commutative :* $B\ \texttt{rs}\ a\ \texttt{rs}\ b \simeq_D B\ \texttt{rs}\ b\ \texttt{rs}\ a$.
   (b) *symétrique par conjugaison :* $B\ \texttt{rs}\ a \simeq_D B\ \texttt{rs}\ \overline{a}$.
   (c) *idempotente :* $B\ \texttt{rs}\ a \simeq_D B\ \texttt{rs}\ a\ \texttt{rs}\ a$.
3. *La synchronisation et la restriction peuvent être inter-changées si appliquées à des actions différentes :*
   (a) *si* $b \notin \{a,\overline{a}\}$ *alors* $B\ \texttt{sy}\ a\ \texttt{rs}\ b \simeq_D B\ \texttt{rs}\ b\ \texttt{sy}\ a$.

Preuve. Voir [BDH92]. ■

6.2 Définitions paramétrées

Dans l'approche que nous allons proposer, nous supposons l'existence d'une structure de synchronisation $\mathcal{S} = (\texttt{Act}, \texttt{Syn}, \texttt{Hide})$, qui décrit les capacités de synchronisation de chaque action atomique. Tout *aspect* de cette structure de synchronisation pourra être utilisé comme paramètre de l'opération de synchronisation ou de restriction, que nous définissons par la suite.

6.2.1 Synchronisation

La définition suivante d'un opérateur alternatif de synchronisation de PBC est une extension de la Définition II.16, dans le sens où elle est paramétrée par une structure de synchronisation quelconque sur les actions atomiques des étiquettes de transitions de communication, et qu'elle autorise des synchronisations par *plusieurs liens*, ainsi que l'*auto-synchronisation*.

Définition II.18 Soient $\mathcal{S}$ une structure de synchronisation, $B = (S, T, W, \lambda)$ un réseau étiqueté dont les transitions sont étiquetées par des multi-actions de $\mathcal{S}$, et A un aspect de $\mathcal{S}$. L'opération de synchronisation $\texttt{Sy}$ du réseau B par rapport à l'aspect A, est définie par :

$$B \;\texttt{Sy}\; A \stackrel{\text{def}}{=} (S, T', W', \lambda')$$

où T' est le plus petit ensemble de transitions qui vérifie :

1. Si $t \in T$ alors $t \in T'$ avec les mêmes pré-conditions, post-conditions et la même étiquette :

   (a) $\forall s \in S \;\; W'(t, s) \stackrel{\text{def}}{=} W(t, s)$,

   (b) $\forall s \in S \;\; W'(s, t) \stackrel{\text{def}}{=} W(s, t)$,

   (c) $\lambda'(t) \stackrel{\text{def}}{=} \lambda(t)$.

2. Si $t_1, t_2 \in T'$ alors pour tout $\alpha \in R_A(\lambda'(t_1), \lambda'(t_2))$, $t \in T'$, où t est une transition dont les pré- et post-conditions sont la somme multi-ensemble des pré- et post-conditions des deux transitions, respectivement, et dont l'étiquette est α :

   (a) ${}^\bullet t \stackrel{\text{def}}{=} {}^\bullet t_1 + {}^\bullet t_2$,

   (b) $t^\bullet \stackrel{\text{def}}{=} {t_1}^\bullet + {t_2}^\bullet$,

   (c) $\lambda'(t) \stackrel{\text{def}}{=} \alpha$.

3. Si $t \in T'$ et $\lambda'(t) = m + n$, alors pour tout $\alpha \in R_A(m, n)$, $t' \in T'$, où t' est une transition dont les pré- et post-conditions sont identiques aux pré- et post-conditions de t, et dont l'étiquette est α :

   (a) ${}^\bullet t' \stackrel{\text{def}}{=} {}^\bullet t$,

   (b) $t'^\bullet \stackrel{\text{def}}{=} t^\bullet$,

   (c) $\lambda'(t') \stackrel{\text{def}}{=} \alpha$.

□

La première règle de la définition fixe l'étape de base de la synchronisation, qui consiste en toutes les transitions de communication de B. Les deux règles suivantes engendrent inductivement des nouvelles transitions.

Cette définition permet l'*auto-synchronisation* (par la règle 3, qui représente *un pas d'auto-synchronisation*, voir Section II.4.2.2). Si nous considérons l'*auto-synchronisation*, nous permettons aussi automatiquement la synchronisation par plusieurs liens. Si la règle 3 est supprimée, alors la synchronisations ainsi définie ne permet ni auto-synchronisations ni synchronisations par *multi-liens*. Dans la Section II.6.2.4, nous décrivons plus en détail comment on peut obtenir, comme cas particulier, la synchronisation standard de PBC. Pour obtenir la synchronisation permettant des multi-liens mais sans auto-synchronisation, il suffit de prendre seulement les deux premières règles et de remplacer dans la deuxième règle la "*relation de synchronisation à lien unique*", R_A, par la "*relation de synchronisation à multi-lien*" R_A^+.

6.2.2 Restriction

L'opération de restriction ($\texttt{Rs}$) dans le contexte des actions atomiques définies par une structure de synchronisations, sera paramétrée par un aspect, comme pour la synchronisation.

Définition II.19 Soient $\mathcal{S}$ une structure de synchronisation, $B = (S, T, W, \lambda)$ un réseau dont les transitions sont étiquetées par des multi-actions de $\mathcal{S}$, et A un aspect de $\mathcal{S}$. L'opération de restriction $\texttt{Rs}$ du réseau B par rapport à l'aspect A, est définie par :

$$B \;\texttt{Rs}\; A \stackrel{\text{def}}{=} (S, T', W', \lambda')$$

où :

1. $T' \stackrel{\text{def}}{=} \{t \in T \mid H_A(\lambda(t)) \text{ est défini }\}$.

2. $W' \stackrel{\text{def}}{=} W$ restreint à $(T' \times S) \cup (S \times T')$.

3. $\lambda'(x) \stackrel{\text{def}}{=} \begin{cases} \lambda(x) & \text{si } x \in S \\ H_A(\lambda(x)) & \text{si } x \in T' \end{cases}$

□

Intuitivement, la restriction par rapport à un aspect A appliquée à un réseau étiqueté B, enlève toutes les capacités de synchronisation du réseau pour cet aspect.

6.2.3 Propriétés des nouvelles opérations

Les propriétés des opérations de synchronisation et de restriction mentionnées dans le Théorème II.14 sont préservées par les nouvelles définitions.

Théorème II.15 *Pour tous les aspects A et B et pour tout réseau étiqueté Σ nous avons :*

$$(\Sigma \ \mathtt{Sy}\ A)\ \mathtt{Sy}\ B \simeq_D \Sigma\ \mathtt{Sy}\ (A \cup B).$$

Avant de présenter la preuve du théorème, remarquons que la commutativité et l'idempotence de l'application de la synchronisation en découlent directement, c.-à-d. :

$$(\Sigma\ \mathtt{Sy}\ A)\ \mathtt{Sy}\ B \simeq_D (\Sigma\ \mathtt{Sy}\ B)\ \mathtt{Sy}\ A$$

$$(\Sigma\ \mathtt{Sy}\ A)\ \mathtt{Sy}\ A \simeq_D \Sigma\ \mathtt{Sy}\ A$$

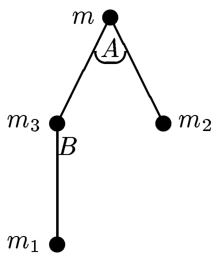
Pour faciliter la preuve, nous introduisons la notion d'*arbre de synchronisation*[1]. Il est intéressant de signaler que cette technique d'utilisation des arbres de synchronisation s'est montrée très puissante et elle a été entretemps utilisée avec succès ailleurs [FK94, KP95, Kla95].

Définition II.20 Soit $\mathcal{S} = (\mathtt{Act}, \mathtt{Syn}, \mathtt{Hide})$ une structure de synchronisation. Un *arbre de synchronisation* sur $\mathcal{S}$ est un arbre binaire étiqueté dont les noeuds sont étiquetés par des multi-actions de $\mathcal{S}$, et les noeuds internes sont en plus "typés" par des aspects de $\mathcal{S}$. Tout noeud interne p de type A et étiqueté par m doit en plus vérifier :

1. si p a un fils unique étiqueté par une multi-action m', alors $m' \xrightarrow{A} m$ est *un pas d'auto-synchronisation*, c.-à-d. qu'il existe des multi-actions n_1, n_2 telles que $n_1 + n_2 = m'$ et $(n_1, n_2, m) \in R_A$ (voir aussi Section II.4.2.2).

2. si p a deux fils étiquetés par m'_1 et m'_2, alors $m \in R_A(m'_1, m'_2)$. □

Intuitivement, un tel arbre dénote la synchronisation de ses feuilles, le père étant étiqueté par le résultat d'une synchronisation entre ses fils dans l'aspect encodé par son type. Par exemple, l'arbre ci-dessous est un arbre de synchronisation sur $\mathcal{S}$, si A et B sont des aspects de $\mathcal{S}$, m_1, m_2, m_3, m sont des multi-actions de $\mathcal{S}$, $m_1 \xrightarrow{B} m_3$ et $m \in R_A(m_3, m_2)$.



[1] Attention : les arbres de synchronisation dont il s'agit ici ne sont pas les arbres de synchronisation au sens de R. Milner.

Remarquons qu'à toute transition t d'un réseau Σ $\mathtt{Sy}$ A nous pouvons associer un arbre de synchronisation (avec uniquement les noeuds de type A), qui "trace" la création de t selon la Définition II.18. Les feuilles sont étiquetées par les étiquettes de transitions de Σ. Le multi-ensemble de transitions de Σ dont les étiquettes marquent les feuilles de l'arbre, détermine sans ambiguïté les connexions de la transition t avec les places du réseau Σ $\mathtt{Sy}$ A. En fait, les pré- et post-conditions de t sont les sommes multi-ensemble de pré- et post-conditions, respectivement, de toutes les transitions de ce multi-ensemble. L'étiquette de t est l'étiquette de la racine de l'arbre de synchronisation.

D'autre part, tout arbre de synchronisation dont les feuilles sont étiquetées par des étiquettes de transitions de Σ, et dont les noeuds internes sont tous de type A, définit une transition t dans Σ $\mathtt{Sy}$ A. Remarquons, qu'à un arbre de synchronisation peuvent correspondre plusieurs transitions, s'il y a plusieurs façons d'associer des transitions de Σ aux feuilles.

La preuve du théorème, c.-à-d. la preuve de :

$$(\Sigma \ \mathtt{Sy} \ A) \ \mathtt{Sy} \ B \simeq_D \Sigma \ \mathtt{Sy} \ (A \cup B)$$

est basée sur le fait que l'ensemble des places des deux réseaux est le même. Il suffit alors de montrer que pour toute transition t d'un réseau il existe une transition t' dans l'autre réseau avec les mêmes pré- et post-conditions et avec la même étiquette. Autrement dit, tout arbre de synchronisation dont les feuilles sont étiquetées par des multi-actions de transitions de Σ et dont les noeuds internes sont de type $A \cup B$ peut être reconfiguré (sans changement de l'ensemble de feuilles et de la racine) en un arbre dont les noeuds sont typés par A ou B et tel qu'un noeud de type B n'est jamais un fils d'un noeud de type A. Cette reconfiguration peut être représentée par le schéma de la Figure 6.2, où m est

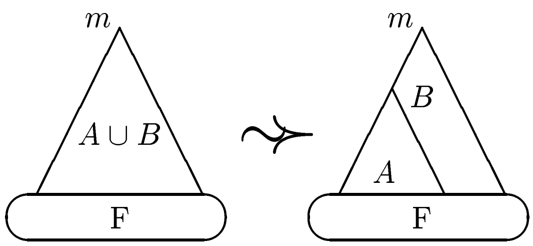
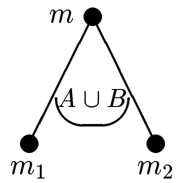


Figure 6.2: Reconfiguration d'un arbre de synchronisation sans changement des feuilles ni de l'étiquette de la racine.

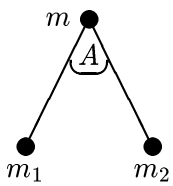
une multi-action et F est un ensemble de feuilles, commun aux deux arbres. La reconfiguration dans l'autre sens est évidente, puisqu'une synchronisation dans l'aspect A implique la possibilité d'avoir la même synchronisation dans tout aspect A' qui contient A, en l'occurrence dans $A \cup B$. Cela veut dire que pour la reconfiguration inverse des arbres de synchronisation il suffit de considérer pour chaque noeud interne le type $A \cup B$ à la place de A ou de B. L'arbre ainsi obtenu est un arbre de synchronisation avec le même étiquetage.

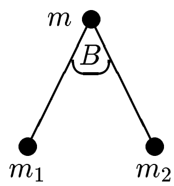
Les lemmes suivants montrent qu'une reconfiguration telle que présentée dans la Figure 6.2 est toujours possible.

Lemme II.16 *Si l'arbre suivant est un arbre de synchronisation*



alors au moins un des arbres suivants est un arbre de synchronisation :

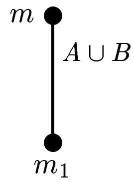




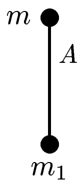
Preuve. Par Proposition II.3(4) nous avons $R_{A\cup B} = R_A \cup R_B$.

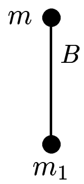
Donc, si $(m_1, m_2, m) \in R_{A\cup B}$, alors (m_1, m_2, m) appartient à R_A ou à R_B. ■

Lemme II.17 *Si l'arbre suivant est un arbre de synchronisation*



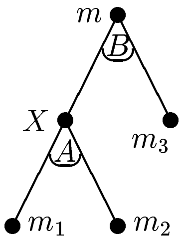
alors au moins un des deux arbres suivants est un arbre de synchronisation :



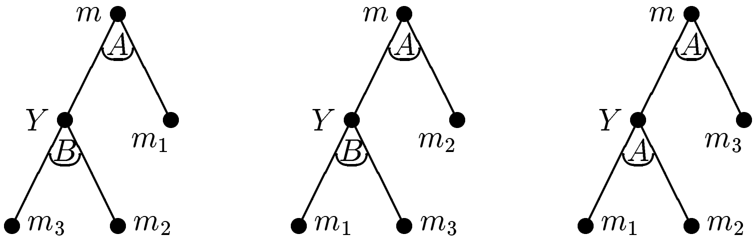


Preuve. Si $m_1 \xrightarrow{A\cup B} m$, alors, par définition, il existe deux multi-actions n_1, n_2 telles que $m' = n_1 + n_2$ et $m \in R_{A\cup B}(n_1, n_2)$. Par Proposition II.3(4), nous avons $R_{A\cup B} = R_A \cup R_B$. Puisque $(n_1, n_2, m) \in R_{A\cup B}$, alors (n_1, n_2, m) appartient à R_A ou à R_B. Donc, $m_1 \xrightarrow{A} m$ ou $m_1 \xrightarrow{B} m$. ■

Lemme II.18 *Si l'arbre suivant est un arbre de synchronisation*



alors il existe une multi-action Y, telle qu'au moins un des arbres suivants est un arbre de synchronisation :



Preuve. Par définition d'un arbre de synchronisation, $X \in R_A(m_1, m_2)$ et $m \in R_B(X, m_3)$. Cela signifie qu'il existe des multi-actions m', m'_1, m'_2, X', X'', m'_3 et des actions a, b, c, x, y, z telles que :

- $\{a, b, c\} \subseteq A$ et $c \in \mathtt{Syn}(a, b)$
- $\{x, y, z\} \subseteq B$ et $z \in \mathtt{Syn}(x, y)$
- $m'_1 + \{a\} = m_1$, $m'_2 + \{b\} = m_2$, $X' + \{c\} = X$
- $X'' + \{x\} = X$, $m'_3 + \{y\} = m_3$, $m' + \{z\} = m$
- $X' = m'_1 + m'_2$, $m' = X'' + m'_3$

Il y a deux cas possibles :

1. $c = x$:

   Dans ce cas, $\{x, y, z\} \subseteq A$ (car $c \in A$, $x = c$, $y \in \mathtt{coa}(c)$, $z \in \mathtt{sya}(c)$), $X' = X''$ et pour $Y = X$ le dernier arbre est un arbre de synchronisation (mais remarquons qu'il se peut que $a, b \notin B$).

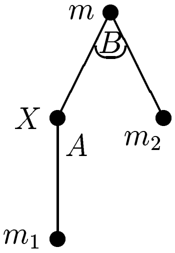
2. $c \neq x$:

   Dans ce cas, $x \in X' = m'_1 + m'_2$.

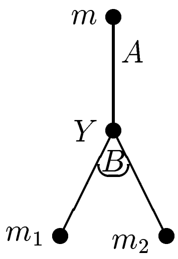
   Soit $x \in m'_1$ et dans ce cas, pour $Y = (m_1 + m'_3) \setminus \{x\}$, le deuxième arbre est un arbre de synchronisation,

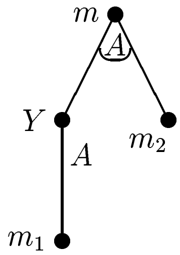
   soit $x \in m'_2$ et dans ce cas, pour $Y = (m_2 + m'_3) \setminus \{x\}$, le premier arbre est un arbre de synchronisation. ■

Lemme II.19 *Si l'arbre ci-dessous est un arbre de synchronisation :*



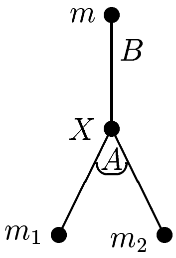
alors il existe une multi-action Y, telle qu'au moins un des arbres suivants est un arbre de synchronisation :



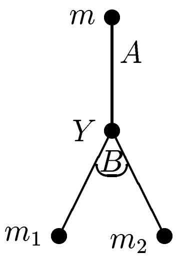


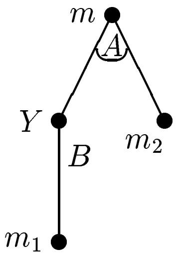
Preuve. Similaire à la précédente. ∎

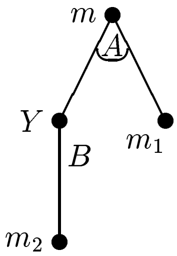
Lemme II.20 *Si l'arbre ci-dessous est un arbre de synchronisation*

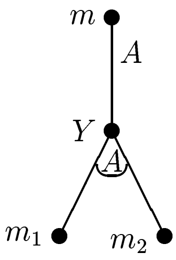


alors il existe une multi-action Y, telle qu'au moins un des arbres suivants est un arbre de synchronisation :



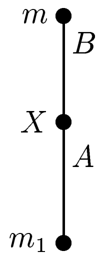




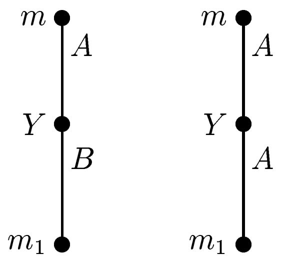


Preuve. Similaire. ■

Lemme II.21 *Si l'arbre ci-dessous est un arbre de synchronisation*



alors il existe une multi-action Y *telle qu'au moins un des arbres ci-dessous est un arbre de synchronisation :*



Preuve. Similaire. ■

Preuve du Théorème II.15

Les six lemmes nous disent que pour chaque arbre de synchronisation dont les noeuds internes sont typés par $A \cup B$, construit sur un ensemble F de feuilles, et dont la racine est étiquetée par une multi-action m, il existe un arbre de synchronisation, avec la même multi-action à la racine et construit sur le même ensemble F de feuilles, dont les noeuds internes sont typés par A ou B et tel qu'il n'existe pas de noeud de type A qui a comme fils un noeud de type B.

Nous pouvons aussi construire un arbre de synchronisation sur le même ensemble de feuilles F et avec la même étiquette à la racine, en procédant de la manière suivante (voir aussi Figure 6.2) :

1. d'abord, on génère une "forêt" d'arbres avec des noeuds de type A

2. et ensuite, on génère un arbre avec des noeuds de type B à partir des sommets de la "forêt".

Cela prouve que pour n'importe quelle transition du réseau Σ `Sy` $(A \cup B)$ il existe une transition dans le réseau (Σ `Sy` A) `Sy` B avec la même étiquette et les mêmes pré- et post-conditions, l'ensemble de places étant le même pour les deux réseaux. L'inverse est trivialement vrai puisque toute synchronisation dans l'aspect A ou dans l'aspect B est toujours possible dans l'aspect $A \cup B$.

Donc, on peut conclure que $(\Sigma \ \mathtt{Sy}\ A)\ \mathtt{Sy}\ B \simeq_D \Sigma\ \mathtt{Sy}\ (A \cup B)$. ∎

Pour l'opération de restriction ($\mathtt{Rs}$) nous avons un résultat similaire.

Théorème II.22 *Pour tous les aspects A et B, et pour tout réseau étiqueté Σ nous avons :*

$$(\Sigma\ \mathtt{Rs}\ A)\ \mathtt{Rs}\ B \simeq_D \Sigma\ \mathtt{Rs}\ (A \cup B).$$

Bien évidemment, la commutativité et l'idempotence de la restriction découlent directement du théorème :

$$(\Sigma\ \mathtt{Rs}\ A)\ \mathtt{Rs}\ B \simeq_D (\Sigma\ \mathtt{Rs}\ B)\ \mathtt{Rs}\ A$$

$$(\Sigma\ \mathtt{Rs}\ A)\ \mathtt{Rs}\ A \simeq_D \Sigma\ \mathtt{Rs}\ A$$

Preuve du Théorème II.22 Puisque l'ensemble de places des deux réseaux est le même (comme pour la synchronisation), il suffit de prouver que pour toute transition de l'un des réseaux il existe une transition dans l'autre réseau avec la même étiquette et les mêmes pré- et post-conditions. En effet, par la Proposition II.5, nous savons que l'ensemble des transitions des deux réseaux est le même. ∎

Nous obtenons aussi le résultat suivant pour les applications alternées de la synchronisation et de la restriction.

Théorème II.23 *Pour tous les aspects A et B, et pour tout réseau étiqueté Σ nous avons :*

Si $A \subseteq B$, alors

$$(\Sigma\ \mathtt{Rs}\ B)\ \mathtt{Sy}\ A \simeq_D \Sigma\ \mathtt{Rs}\ B$$

Preuve du Théorème II.23

Il suffit de remarquer que dans $\Sigma\ \mathtt{Rs}\ A$ il n'y a pas de transitions étiquetées par une multi-action contenant un élément de A, donc ni la règle 2 ni la règle 3 de la Définition II.18 ne peuvent être appliquées, ce qui signifie que l'ensemble des transitions reste le même dans $(\Sigma\ \mathtt{Rs}\ B)\ \mathtt{Sy}\ A$ que dans $\Sigma\ \mathtt{Rs}\ B$.

∎

6.2.4 Simulation de la synchronisation de PBC

La structure de synchronisation qui définit les actions atomiques pour PBC, est celle de CCS (voir l'Exemple II.1), notée $\mathcal{S}_M$. Dans PBC, il est possible de faire une synchronisation par rapport à chaque paire conjuguée d'actions $(a, \overline{a})$, qui dans notre approche correspond à l'aspect $\{a, \overline{a}, \tau\}$. L'action τ est une action *invisible* (voir Définition II.5). Dans PBC, cette action n'apparaît jamais explicitement dans les étiquettes des transitions. Ce procédé pourrait être choisi aussi dans notre approche en utilisant les versions restreintes (avec

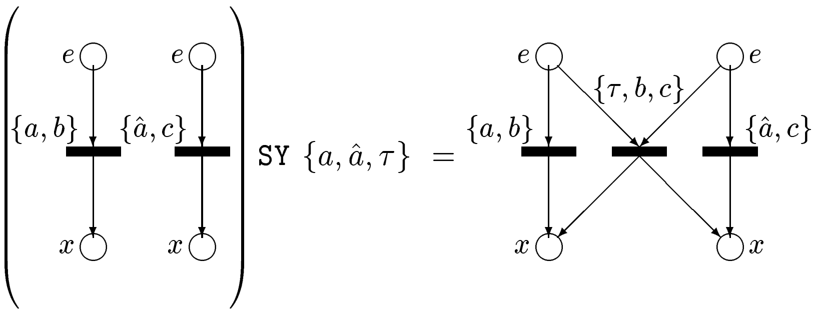
($\tilde{\ }$) comme défini dans la Section II.4.2.4) des relations de synchronisation et de restriction partielle. Nous avons, cependant, préféré de laisser les τ dans les étiquettes comme témoins des synchronisations. Cela veut dire que, dans l'approche proposée nous obtenons une simulation de la synchronisation de PBC modulo une bisimulation d'actions, à savoir celle engendrée par l'équation $m = m + \{\tau\}$.

Nous avons déjà mentionné que la synchronisation ($\mathtt{Sy}$) de la Définition II.18 permet l'auto-synchronisation, ce qui n'est pas le cas pour la synchronisation ($\mathtt{sy}$) de PBC (Définition II.16). Néanmoins, il suffit d'enlever la règle 3 de la Définition II.18 pour interdire l'auto-synchronisation. La nouvelle synchronisation $\mathtt{SY}$, définie dans la Définition II.18, dont la dernière règle est supprimée, dans le contexte de $\mathcal{S}_M$ simule la synchronisation $\mathtt{sy}$ de PBC (modulo *bisimulation d'actions*).

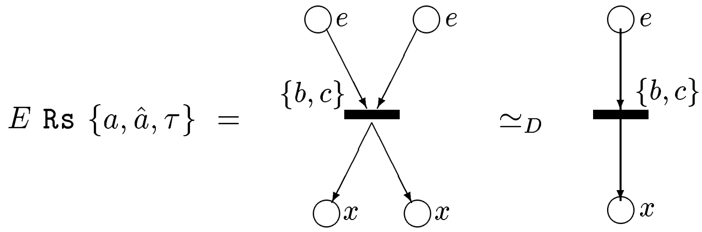
Exemple II.5 *En utilisant l'opération de synchronisation* $\mathtt{SY}$ *avec la structure de synchronisation* $\mathcal{S}_M$ *de l'Exemple II.1, la Box-expression*

$$E \;=\; (\{a,b\} \parallel \{\hat{a},c\}) \;\mathtt{SY}\; \{a,\hat{a},\tau\}$$

se traduit en :



La restriction de E par rapport à l'aspect $\{a,\hat{a},\tau\}$ *donne ensuite :*



6.3 Commentaires

Dans ce chapitre, nous avons développé une application particulière des structures de synchronisation pour obtenir une nouvelle définition de l'opération de synchronisation dans l'Algèbre des Petri-Box. La nouvelle définition, en plus d'être paramétrable par des structures de synchronisation quelconques, diffère aussi de la définition originale par la possibilité d'*auto-synchronisation* et de synchronisation par liens multiples. Dans tous les cas de figure, la nouvelle synchronisation préserve les principales propriétés algébriques, telles que la commutativité et l'idempotence.

En utilisant la structure de synchronisation pour la description des actions atomiques, nous pouvons, par exemple, modéliser la lecture simultanée dans une case de mémoire partagée, ou un broadcast, ce qui n'est pas possible dans l'ancienne version de la synchronisation de box se limitant aux actions à la CCS.

Supposons que nous voulions représenter dans PBC une variable $\mathtt{V}$ de type $\mathtt{D}$, c.-à-d. qu'elle prend ses valeurs dans un ensemble de valeurs $\mathtt{D}$. De plus, nous voudrions que la lecture simultanée de $\mathtt{V}$ soit possible. Pour cela, nous définissons d'abord l'ensemble d'actions, $\mathtt{Act}_\mathtt{V}$, via lesquelles nous pourrions lire et écrire une valeur dans la variable : $\mathtt{Act}_\mathtt{V} \stackrel{\text{def}}{=} \{\mathtt{V}?(x), \mathtt{V}!(x) \mid x \in \mathtt{D}\}$. L'action $\mathtt{V}!(x)$ dénoterait l'écriture de la valeur x dans la variable $\mathtt{V}$, et l'action $\mathtt{V}?(x)$ dénoterait la lecture de la valeur x de la variable $\mathtt{V}$.

Il nous reste à fixer la relation de synchronisation, $\mathtt{Syn}_\mathtt{V}$, et la relation d'abstraction, $\mathtt{Hide}_\mathtt{V}$, sur les actions $\mathtt{Act}_\mathtt{V}$:

- $\mathtt{Syn}_\mathtt{V} \stackrel{\text{def}}{=} \{(\mathtt{V}?(x), \mathtt{V}!(x), \mathtt{V}!(x)), (\mathtt{V}!(x), \mathtt{V}?(x), \mathtt{V}!(x)) \mid x \in \mathtt{D}\}$,
- $\mathtt{Hide}_\mathtt{V} \stackrel{\text{def}}{=} \{\mathtt{V}!(x) \mid x \in \mathtt{D}\}$.

Le triplet $(\mathtt{Act}_\mathtt{V}, \mathtt{Syn}_\mathtt{V}, \mathtt{Hide}_\mathtt{V})$ constitue une structure de synchronisation qui peut être ajouté (de manière disjointe) à n'importe quelle autre structure de synchronisation. Bien évidemment, dans la structure de synchronisation ainsi obtenue, l'ensemble $\mathtt{Act}_\mathtt{V}$ reste un aspect.

Grâce aux Théorèmes II.15 et II.22, nous pouvons grouper les aspects dans le but d'augmenter le niveau d'abstraction, ce qui est très utile pour l'intégration du traitement de données dans PBC ou $\mathrm{B(PN)}^2$. Cela nous permet d'éviter l'utilisation des méta-constructions pour exprimer, par exemple, la synchronisation sur toutes les valeurs potentielles d'une variable (surtout de type composé, comme un article (ang. *record*) ou un tableau). Dans notre approche, il suffit de déclarer un aspect comme l'union correspondante.

Chapter 7

Conclusions

Dans cette partie, nous avons proposé une classe de structures relationnelles, appelées *structures de synchronisation*, qui formalisent des notions intuitives liées aux *primitives de communication* d'un système réactif. Une structure de synchronisation définit l'ensemble des primitives de synchronisation (appelées *actions*) et précise comment les actions peuvent se synchroniser entre elles. Nous avons aussi analysé les capacités de synchronisations des *multi-actions*, qui consistent en l'exécution simultanée de plusieurs actions. Pour cela, nous avons défini deux types de relations entre les multi-actions : des *relations de synchronisation partielle* et des *relations d'abstraction partielle*, qui peuvent être utilisées pour la définition de compositions parallèles dans différents modèles de la concurrence. Afin de permettre la compositionalité, nous avons introduit la notion d'*ensembles de communication* et d'*aspects* comme des ensembles consistants d'actions qui peuvent être traités séparément. Les ensembles de communication peuvent être considérés comme des caractérisations des connexions binaires (point-à-point). Les aspects, une classe particulière d'ensembles de communication, permettent, de façon très naturelle, une analyse partielle des processus. Cela semble être une bonne idée de séparer les actions dans des ensembles indépendants, afin de pouvoir se focaliser uniquement sur quelques-unes d'entre elles pendant l'analyse des systèmes réactifs.

Quelques exemples d'applications des notions liées aux structures de synchronisation ont été présentées dans le contexte de systèmes de transitions et dans le contexte de réseaux de Petri.

Part III

Algèbre de Processus sur des Multi-actions

Les algèbres de processus peuvent être utilisées comme langages de spécification des comportements des systèmes réactifs, appelés *processus*, qui sont composés d'*actions* discrètes. En fait, un processus définit les relations entre ses *événements* (occurrences d'actions), qui sont le plus souvent la *dépendance causale*, la *concurrence* et le *conflit*. Dans cette partie, nous introduisons un nouveau type de dépendance entre les événements, appelé la *simultanéité*. Deux événements sont en "dépendance de simultanéité" quand l'un des événements ne peut se produire sans l'autre. Cette relation nous mène à étendre la notion d'événement et à le considérer comme un ensemble d'actions se produisant simultanément, et non plus comme l'occurrence d'une seule action. Un tel événement sera appelé *multi-action*.

Nous proposons une algèbre de processus permettant d'intégrer la dépendance de simultanéité, que nous appelons Algèbre de Processus sur des Multi-Actions (MAPA: Multi-Action Process Algebra [Frą95]). La syntaxe de MAPA comprend seulement trois opérateurs : le *préfixage*, la *composition* et la *restriction*. Les principales caractéristiques de MAPA sont les suivantes : premièrement, un processus est préfixé par un multi-ensemble fini d'actions ; deuxièmement, à la composition de deux processus est associée une spécification explicite des restrictions de communication, formalisée par la notion de *canaux de communication*.

Un canal de communication est caractérisé par le type et le nombre d'actions qu'il peut véhiculer simultanément. Deux processus connectés par un canal peuvent seulement communiquer via des multi-actions permises par le canal. Nous définissons des opérations sur les canaux qui, par exemple, permettent de renverser la direction d'un canal, ou de grouper deux canaux en un seul.

Une sémantique opérationnelle de MAPA est proposée, de manière standard via des "règles de dérivation" à la Plotkin. Dans le cas de MAPA, où le "choix" en tant qu'opération primitive n'existe pas, il est possible de prendre en compte différents ensembles de règles de dérivation, chaque ensemble définissant une sémantique opérationnelle légèrement différente. Nous analysons quelques exemples de telles sémantiques et nous en choisissons celle qui, selon les critères proposés, est "la plus opérationnelle".

En plus des équivalences standards, telles que l'*isomorphisme de graphes de processus* ($\equiv$), l'*équivalence forte* ($\simeq$), l'*équivalence faible* ($\approx$) des processus, nous proposons l'*équivalence sémantique* ($\sim$).

L'absence du *choix* comme opérateur primitif garantit que l'équivalence faible est une congruence. Nous prouvons les inclusions strictes d'équivalences suivantes : $\equiv\ \subset\ \simeq\ \subset\ \approx\ \subset\ \sim$, et nous démontrons que, par rapport à $\approx$ (et donc à $\sim$), toutes les sémantiques opérationnelles analysées sont équivalentes.

Enfin, nous montrons que MAPA possède la "puissance expressive la plus générale" par rapport à au moins deux des trois critères proposés par Vaandrager dans [Vaa92].

Dans cette partie, nous préférons de ne pas utiliser les notions liées aux *structures de synchronisation* introduites dans la partie précédente, premièrement pour rendre le texte plus lisible et plus facilement abordable par des lecteurs déjà familiarisé avec les algèbres de processus CCS ou PBC. Ainsi, cette partie peut être lue indépendamment de la partie précédente. D'autre part, même si une présentation "à la structure de synchronisation" est possible, le fait que nous travaillons avec des primitives de communication bien précises[1], rendrait une telle présentation inutilement lourde. Une généralisation de toutes les notions proposées ici à une structure de synchronisation arbitraire serait forte intéressante, mais semble être difficile. Cela ne remet nullement en cause l'utilité de structures de synchronisation, dont les résultats sont indirectement utilisés.

[1] De fait, nous serons toujours dans le cadre de la structure de synchronisation $\mathcal{S}_M$ de l'Exemple II.1.

Chapter 8

Primitives de communication

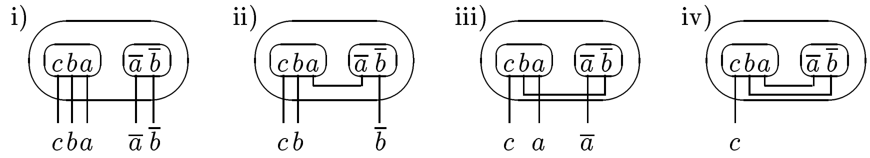
Dans ce chapitre, nous décrivons les primitives de communication, c'est-à-dire les primitives qui seront utilisées par les *agents*, définis par notre algèbre de processus, afin qu'ils puissent interagir entre eux. Nous reprenons l'idée simple et très intuitive proposée par Milner pour CCS [Mil80]. Dans cette approche, les *agents* peuvent communiquer par des "liens". Un agent dispose de deux actions primitives complémentaires pour un lien : l'envoi sur le lien et la réception sur le lien. Toutes les communications par les liens sont supposées être *synchrones* (c.-à-d. que ni l'envoi ni la réception ne peuvent se produire sans leur inverse) et *binaires* (c.-à-d. qu'à un envoi ne peut être associée qu'une seule réception, et vice versa).

Dans ce qui suit, nous allons considérer un ensemble non vide d'*actions atomiques*, $\mathtt{Act}$, muni d'une bijection $\bar{\ } : \mathtt{Act} \mapsto \mathtt{Act}$, appelée *conjugaison*, telle que $\overline{\overline{a}} = a$, pour chaque $a \in \mathtt{Act}$. L'action $\overline{\overline{a}}$ est le résultat de la double application de la bijection sur a, c.-à-d. que $\overline{\overline{a}}$ dénote en fait $\bar{\ }(\bar{\ }(a))$. Remarquons que nous n'imposons pas que toute action a soit différente de son inverse $\overline{a}$.

8.1 Multi-actions

Les actions atomiques peuvent être groupées en multi-ensembles appelés *multi-actions*. Comme dans la partie précédente, une multi-action est un multi-ensemble fini d'actions atomiques et l'ensemble de toutes les multi-actions sur $\mathtt{Act}$ est noté $\mu_f(\mathtt{Act})$. Intuitivement, une multi-action représente l'exécution simultanée de plusieurs actions (c.-à-d. de ses composantes), comme dans [BDH92] ou [BB93]. La définition proposée pour la multi-action diffère donc légèrement de celles adoptées dans Meije [AB84] ou SCCS [Mil83], où l'exécution simultanée de deux multi-actions mène toujours à leur synchronisation, donnant toujours comme résultat une seule multi-action.

Soient $\{c, b, a\}$ et $\{\overline{a}, \overline{b}\}$ deux multi-actions qui représentent respectivement une exécution simultanée des actions atomiques c, b, a, et une exécution simultanée de $\overline{a}$, $\overline{b}$. Intuitivement, une exécution simultanée de ces deux multi-actions peut être vue de l'extérieur comme :



i) la multi-action $\{c, b, a, \overline{a}, \overline{b}\}$, si les deux multi-actions ne communiquent pas ;

ii) $\{c, b, \overline{b}\}$, si les deux multi-actions se synchronisent sur la paire $(a, \overline{a})$, et ainsi a et $\overline{a}$ deviennent invisibles ;

iii) $\{c, a, \overline{a}\}$, si les deux multi-actions se synchronisent sur la paire $(b, \overline{b})$;

iv) $\{c\}$, si les deux multi-actions se synchronisent sur deux paires $(a, \overline{a})$ et $(b, \overline{b})$.

Il y a donc ici quatre résultats possibles.

La conjugaison sur les actions atomiques est étendue aux multi-actions $m \in \mu_f(\texttt{Act})$ en y associant $\overline{m}$ telle que :

$$\forall a \in \texttt{Act} \ \ \overline{m}(a) \stackrel{\text{def}}{=} m(\overline{a}) \ .$$

Remarquons que la conjugaison sur les multi-actions vérifie $\overline{\overline{m}} = m$.

La synchronisation de deux actions atomiques complémentaires donne toujours une action invisible (dans CCS dénotée par τ) qui ne se synchronise avec aucune autre action. Cela veut dire que les agents qui ne peuvent exécuter qu'une seule action atomique à la fois ne permettent pas de synchronisation multiple (synchronisation de plus de deux agents). En revanche, l'approche basée sur les multi-actions permettant aux agents d'exécuter simultanément plusieurs actions atomiques, permet par là-même la synchronisation multiple.

Le "schéma de synchronisation" entre les multi-actions peut être formalisé de la façon suivante :

"$n_1 + m$" peut se synchroniser avec "$n_2 + \overline{m}$", en donnant "$n_1 + n_2$".

Deux conséquences importantes de cette formalisation de la synchronisation entre les multi-actions sont d'une part que la simultanéité et la synchronisation ne peuvent être distinguées par observation (quand $m = \emptyset$), et d'autre part que la synchronisation de deux multi-actions donne un ensemble de multi-actions.

8.2 Canaux de communication

Considérons la construction $P \parallel_{(A,B)} Q$, où P, Q sont des processus, et (A, B) est une paire d'ensembles d'actions qui décrit les connexions entre P et Q.

L'ensemble A contient des actions permises du côté gauche de la connexion (le côté de P), et l'ensemble B contient des actions permises du côté droit de la connexion (le côté de Q). Evidemment, $P \parallel_{(A,B)} Q$ est équivalent à $Q \parallel_{(B,A)} P$, mais pas à $Q \parallel_{(A,B)} P$, à cause d'une asymétrie possible de la connexion. En général, les ensembles A et B peuvent être incohérents, dans le sens où il peut y avoir une action a dans A, telle que l'action inverse $\overline{a}$, qui pourrait se synchroniser avec a, n'est pas dans B. Ainsi, nous allons supposer que chaque connexion (A, B) est compatible, dans le sens où si une action a est dans A, alors l'action inverse $\overline{a}$ est dans B, et vice versa. Etant donné un ensemble A de la description de connexion (A, B), nous pouvons facilement recalculer l'ensemble B par $B = \overline{A} \stackrel{\text{def}}{=} \{\overline{a} \mid a \in A\}$. Ainsi, nous pouvons représenter une connexion par un seul ensemble d'actions A.

Pour le moment, nous avons seulement décrit quels types d'actions vont être permis par une connexion. Mais deux connexions peuvent également différer par le nombre maximal de communications atomiques (synchronisations entre les actions atomiques conjuguées) permises simultanément. Par exemple, nous voudrions distinguer une connexion qui permet seulement une seule synchronisation atomique à la fois d'une connexion qui permet deux synchronisations atomiques simultanément, ou plus encore.

Si nous gardons toujours l'intuition initiale pour les actions atomiques (comme envoi et réception de signaux), une connexion entre deux objets communiquant (par un *canal de communication*) peut être vue comme un ensemble de fils, chacun capable de transmettre un type donné de signaux. Prenons, par exemple, un canal de communication composé de trois fils : deux fils unidirectionnels, orientés dans la même direction, et capables de porter un signal dont l'envoi (ou respectivement la réception) est représenté(e) par l'action a (ou respectivement $\overline{a}$), et un fil bidirectionnel capable de porter un signal d'actions c et $\overline{c}$. Cela peut être illustré par la figure suivante :

$$\begin{array}{rcl} a & \rule[0.5ex]{8em}{0.4pt} & \overline{a} \\ a & \rule[0.5ex]{8em}{0.4pt} & \overline{a} \\ c/\overline{c} & \rule[0.5ex]{8em}{0.4pt} & c/\overline{c} \end{array}$$

Ce qui nous intéresse est de savoir quelles multi-actions peuvent être transmises par le canal. L'extrémité gauche du canal présenté ci-dessus permet la transmission des multi-actions suivantes : $\emptyset$, $\{a\}$, $\{c\}$, $\{\overline{c}\}$, $\{a, a\}$, $\{a, c\}$, $\{a, \overline{c}\}$, $\{a, a, c\}$ ou $\{a, a, \overline{c}\}$. L'extrémité droite permet, bien évidemment, les multi-actions obtenues par conjugaison des précédentes, c'est-à-dire : $\emptyset$, $\{\overline{a}\}$, $\{\overline{c}\}$, $\{c\}$, $\{\overline{a}, \overline{a}\}$, $\{\overline{a}, \overline{c}\}$, $\{\overline{a}, c\}$, $\{\overline{a}, \overline{a}, \overline{c}\}$ ou $\{\overline{a}, \overline{a}, c\}$.

Définition III.1 Un *canal*, ch, sur des actions atomiques $\mathtt{Act}$, est un ensemble de multi-actions, $ch \subseteq \mu_f(\mathtt{Act})$, qui vérifie :

1. $\emptyset \in ch$, (où $\emptyset$ représente la multi-action vide)

2. $\forall m, n \in \mu_f(\mathtt{Act}) \quad m \in ch, n \subseteq m \Rightarrow n \in ch.$

L'ensemble de tous les canaux sur $\texttt{Act}$ sera noté $\texttt{Chan(Act)}$, ou simplement $\texttt{Chan}$. □

Les deux propriétés d'un canal s'accordent parfaitement bien avec notre intuition : 1. chaque canal peut véhiculer "rien", ce qui est représenté par la multi-action vide $\emptyset$; 2. si un canal peut véhiculer une multi-action m, alors il peut véhiculer également n'importe quelle partie de m.

Nous étendons la conjugaison sur tout ensemble $\mathcal{L} \subseteq \mu_f(\texttt{Act})$ de multi-actions par : $\overline{\mathcal{L}} \stackrel{\text{def}}{=} \{\overline{m} \mid m \in \mathcal{L}\}$, et nous définissons la "somme" $(+)$ de deux ensembles de multi-actions par : $\mathcal{L}_1 + \mathcal{L}_2 \stackrel{\text{def}}{=} \{m + n \mid m \in \mathcal{L}_1, n \in \mathcal{L}_2\}$.

Proposition III.1 *Les canaux sont clos par union ($\cup$), intersection ($\cap$), somme ($+$), et conjugaison ($^-$), c'est-à-dire que :*

$$ch, ch' \in \texttt{Chan} \quad \Rightarrow \quad ch \cup ch', ch \cap ch', ch + ch', \overline{ch} \ \in \ \texttt{Chan} \ .$$

Preuve. Nous vérifions que :

1. $\emptyset$ est dans $ch \cup ch'$, $ch \cap ch'$, $ch + ch'$ car par définition du canal $\emptyset \in ch$, $\emptyset \in ch'$, et $\emptyset$ est dans $\overline{ch}$ car $\overline{\emptyset} = \emptyset$;
2. $\forall m, n \in \mu_f(\texttt{Act})$, si $n \subseteq m$ alors :
   (a) $m \in ch \cup ch' \Rightarrow n \in ch \cup ch'$:
   Si $m \in ch \cup ch'$ alors $m \in ch$ ou $m \in ch'$.
   Par définition du canal, $n \in ch$ ou $n \in ch'$, c.-à-d. $n \in ch \cup ch'$.
   (b) $m \in ch \cap ch' \Rightarrow n \in ch \cap ch'$:
   Si $m \in ch \cap ch'$ alors $m \in ch$ et $m \in ch'$.
   Par définition du canal, $n \in ch$ et $n \in ch'$, c.-à-d. $n \in ch \cap ch'$.
   (c) $m \in ch + ch' \Rightarrow n \in ch + ch'$:
   Si $m \in ch + ch'$ alors, par définition de la somme, $\exists\ m_1, m_2 \in \mu_f(\texttt{Act})$ telles que $m = m_1 + m_2$ et $m_1 \in ch$ et $m_2 \in ch'$. Comme toute sous-multi-action n de m peut être décomposée en $n = n_1 + n_2$ avec $n_1 \subseteq m_1$ et $n_2 \subseteq m_2$, on a $n = n_1 + n_2 \in ch + ch'$ car $n_1 \in ch$ et $n_2 \in ch'$ par définition du canal.
   (d) $m \in \overline{ch} \Rightarrow n \in \overline{ch}$:
   Or $n \subseteq m$ implique $\overline{n} \subseteq \overline{m}$. D'où $\overline{n} \in ch$ et par conjugaison, $n \in \overline{ch}$.

■

Ces quatre opérations ont une très forte signification intuitive. L'union de deux canaux ($ch \cup ch'$) représente le canal minimal capable de véhiculer ce que ch ou ch' peuvent véhiculer. L'intersection de deux canaux ($ch \cap ch'$) représente le canal qui peut véhiculer seulement ce que ch et ch' peuvent véhiculer. La somme de deux canaux ($ch + ch'$) met côte à côte les deux canaux, c'est-à-dire que le canal résultant véhicule tout ce que les deux canaux peuvent véhiculer simultanément. La conjugaison d'un canal $(\overline{ch})$ renverse la direction du canal.

Signalons tout de même, qu'une représentation d'un canal par un "ensemble de fils" n'est pas toujours possible (par exemple $\{\emptyset, a, aa, b\}$).

Chapter 9

Algèbre de Processus sur les Multi-Actions

9.1 Syntaxe

Dans cette section, nous exposons la syntaxe de notre Algèbre de Processus sur les Multi-Actions, que nous appelons MAPA, et qui est construite sur l'ensemble, $\mathtt{Act}$, d'actions atomiques comme défini dans le Chapitre III.8.

Le langage $\mathcal{P}$ des termes de MAPA est défini par :

$$\mathbf{P} ::= \mathbf{m;P} \;\mid\; \mathbf{P\,ch\,P} \;\mid\; \mathbf{P{:}ch} \;\mid\; \mathbf{g}$$

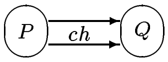
où :

- $\mathbf{m}$ est une multi-action, $\mathbf{m} \in \mu_f(\mathtt{Act})$,
- $\mathbf{ch}$ est un canal, $\mathbf{ch} \in \mathtt{Chan}(\mathtt{Act})$,
- $\mathbf{g}$ est un *identificateur d'agent* d'un ensemble dénombrable infini "*Agents*",
- pour chaque identificateur d'agent $\mathbf{g}$, il existe une et une seule équation définissante $\mathbf{g} \doteq \mathbf{P}$.

Préfixage. Un terme de la forme $(\mathbf{m;P})$ représente un processus capable d'exécuter une multi-action $\mathbf{m}$, et qui se comporte ensuite comme $\mathbf{P}$. Signalons que la construction $\emptyset$**;P** est syntaxiquement permise, mais nous allons voir qu'elle est peu intéressante car le terme $\emptyset$**;P** est sémantiquement équivalent à $\mathbf{P}$.

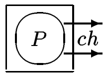
Composition. Un terme $(\mathbf{P\,ch\,Q})$ décrit un processus qui se comporte comme deux processus $\mathbf{P}$ et $\mathbf{Q}$ s'exécutant en parallèle et communiquant via un canal orienté **ch**.

P ch Q :



Le canal **ch** définit l'extrémité gauche de la connexion, c.-à-d. l'extrémité du côté de **P**.

Restriction. L'opération (**P:ch**) est appelée *restriction* (ou *interface*). Le processus (**P:ch**) se comporte comme **P**, mais ses capacités de communication sont limitées au canal **ch**, c'est-à-dire que seules les multi-actions de **P** permises par le canal **ch** pourront être vues de l'extérieur.

P:ch : 

Agent. En utilisant les *identificateurs d'agent*, on peut construire des processus avec des comportements infinis. Nous supposons que presque tous les agents **g** sont définis par l'équation *triviale* $\mathbf{g} \doteq \mathbf{g}$, et donc seul un nombre fini d'agents sont définis par un terme non trivial, c'est-à-dire que la partie droite de l'équation définissante, $\mathbf{g} \doteq P$, est différente de l'identificateur lui-même.

Notons que nous n'avons ni "choix" ni "renommage" comme opérations primitives.

9.2 Sémantique opérationnelle

La sémantique opérationnelle de notre langage $\mathcal{P}$ sera donnée en termes de *systèmes de transitions étiquetés* comme défini au Chapitre I.1. La sémantique opérationnelle du langage est le système de transitions $(\mathcal{P}, \mu_f(\texttt{Act}), \rightarrow)$, noté $\texttt{sem}(\mathcal{P})$, dont les états sont les éléments de $\mathcal{P}$ (c.-à-d. les termes de MAPA), les étiquettes sont des multi-actions, et les transitions sont définies par l'ensemble de règles, $\mathcal{R}$, suivant ([1]) :

$$
\begin{array}{ll}
\texttt{Inaction} & \dfrac{}{\mathbf{m;P} \xrightarrow{\emptyset} \mathbf{m;P}} \\[2ex]
\texttt{Prefix} & \dfrac{}{\mathbf{m;P} \xrightarrow{m} \mathbf{P}} \\[2ex]
\texttt{Restriction} & \dfrac{\mathbf{P} \xrightarrow{m} \mathbf{P'}}{\mathbf{P:ch} \xrightarrow{m} \mathbf{P':ch}}\ m \in \mathbf{ch} \\[2ex]
\texttt{Composition} & \dfrac{\mathbf{P} \xrightarrow{n_1+m} \mathbf{P'} \quad \mathbf{Q} \xrightarrow{n_2+\overline{m}} \mathbf{Q'}}{\mathbf{P\ ch\ Q} \xrightarrow{n_1+n_2} \mathbf{P'\ ch\ Q'}}\ m \in \mathbf{ch} \\[2ex]
\texttt{Call} & \dfrac{}{\mathbf{g} \xrightarrow{\emptyset} \mathbf{P}}\ \mathbf{g} \doteq \mathbf{P}
\end{array}
$$

où m, n_1, n_2 sont des multi-actions et $\emptyset$ est la multi-action vide.

Une transition $P \xrightarrow{m} Q$ dit qu'un agent décrit par le terme P est capable d'évoluer en un nouvel agent représenté par le terme Q en exécutant la multi-action m, ou, de manière équivalente, P est capable d'évoluer en Q si la multi-action $\overline{m}$ se produit dans son environnement. Il ne faut pas confondre une

[1] Plus précisément, $\rightarrow$ est défini comme le plus petit sous-ensemble de $\mathcal{P} \times \mu_f(\texttt{Act}) \times \mathcal{P}$, qui vérifie les cinq règles.

multi-action vide $\emptyset$ avec un pas invisible, ex. τ de CCS. Une multi-action vide signifie qu'aucune action n'est visible, donc qu'elle peut, mais ne doit pas, cacher le mouvement interne d'un agent, par contre une action τ cache toujours une action interne. Cela veut dire que nous ne pourrons pas distinguer (ni même définir) les phénomènes liés à la divergence [BR85, Nic87]. Intuitivement, tous les systèmes qui ne communiquent pas sont considérés comme équivalents indépendamment du fait de savoir s'ils divergent (évoluent de manière interne) ou s'ils restent bloqués.

Les règles présentées ci-dessus sont légèrement différentes de celles proposées par Milner pour CCS ou SCCS. Nous avons une nouvelle règle, `Inaction`, qui dit qu'un agent de la forme **m;P** peut rester inactif. De plus, la règle `Call` peut sembler surprenante si l'on confond "un pas invisible" avec une multi-action vide. La règle `Inaction` permet en fait à tout agent d'exécuter une multi-action vide, c'est-à-dire :

Lemme III.2 $\forall \mathbf{P} \in \mathcal{P}\ \exists \mathbf{Q} \in \mathcal{P}\quad \mathbf{P} \xrightarrow{\emptyset} \mathbf{Q}$.

Pour prouver ce lemme nous avons besoin d'une méthode d'énumération de tous les termes de MAPA. Pour cela, nous définissons la hauteur $\mathtt{high}(P)$ d'un terme P.

Définition III.2 La hauteur d'un terme de MAPA, `high`, est une application $\mathtt{high} : \mathcal{P} \mapsto \mathtt{Nat}$, définie par :

1. $\mathtt{high}(g) \stackrel{\text{def}}{=} 0$, pour tout identificateur d'agent g,
2. $\mathtt{high}(m; P) \stackrel{\text{def}}{=} \mathtt{high}(P) + 1$,
3. $\mathtt{high}(P : ch) \stackrel{\text{def}}{=} \mathtt{high}(P) + 1$,
4. $\mathtt{high}(P\ ch\ Q) \stackrel{\text{def}}{=} \max(\mathtt{high}(P), \mathtt{high}(Q)) + 1$. □

Preuve du Lemme III.2 Par induction sur la hauteur du terme **P** :

1. Pour tout terme P tel que $\mathtt{high}(P) = 0$, il existe un terme Q tel que $P \xrightarrow{\emptyset} Q$. P doit être un identificateur g auquel est associée une équation $g \doteq G$, et par la règle `Call`, $P \xrightarrow{\emptyset} G$.

2. Supposons, pour un $k \geq 0$, que $\forall P \in \mathcal{P}\ \ \mathtt{high}(P) \leq k \Rightarrow \exists Q \in \mathcal{P}\ \ P \xrightarrow{\emptyset} Q$. Soit P un terme de hauteur $\mathtt{high}(P) = k + 1$.

   Les trois cas suivants sont possibles :

   (a) $P = m; Q$. Alors, par la règle `Inaction` nous avons $P \xrightarrow{\emptyset} P$;

   (b) $P = Q : ch$, avec $\mathtt{high}(Q) = k$. Alors, par hypothèse, il existe un terme Q' tel que $Q \xrightarrow{\emptyset} Q'$, et puisque $\emptyset \in ch$ nous avons $Q : ch \xrightarrow{\emptyset} Q' : ch$, c.-à-d. $P \xrightarrow{\emptyset} Q : ch$;

(c) $P = Q\ ch\ R$, avec $\max(\mathtt{high}(Q), \mathtt{high}(R)) = k$. Alors, par hypothèse, il existe des termes Q' et R' tels que $Q \xrightarrow{\emptyset} Q'$, et $P \xrightarrow{\emptyset} P'$, et comme $\emptyset \in ch$, $\overline{\emptyset} = \emptyset$, et $\emptyset + \emptyset = \emptyset$, nous avons $P \xrightarrow{\emptyset} Q'\ ch\ R'$.

■

Dans le Chapitre III.10, nous allons considérer différents ensembles de "règles de dérivation" pour MAPA, à savoir les cinq ensembles suivants :

$\mathcal{R}_0 = \{\mathtt{Inaction}, \mathtt{Prefix}, \mathtt{Composition}, \mathtt{Restriction}, \mathtt{Call}\}$; $(\mathcal{R}_0 = \mathcal{R})$

$\mathcal{R}_1 = \{\mathtt{Inaction\text{-}bis}, \mathtt{Prefix}, \mathtt{Composition}, \mathtt{Restriction}, \mathtt{Call}\}$;

$\mathcal{R}_2 = \{\mathtt{CompLeft}, \mathtt{CompRight}, \mathtt{Prefix}, \mathtt{Composition}, \mathtt{Restriction}, \mathtt{Call}\}$;

$\mathcal{R}_3 = \{\mathtt{Inaction\text{-}bis}, \mathtt{Prefix}, \mathtt{Composition}, \mathtt{Restriction}, \mathtt{Call\text{-}bis}\}$;

$\mathcal{R}_4 = \{\mathtt{CompLeft}, \mathtt{CompRight}, \mathtt{Prefix}, \mathtt{Composition}, \mathtt{Restriction}, \mathtt{Call\text{-}bis}\}$;

où les règles alternatives sont définies par :

$$\mathtt{Inaction\text{-}bis} \qquad \frac{}{\mathbf{P} \xrightarrow{\emptyset} \mathbf{P}}$$

$$\mathtt{Call\text{-}bis} \qquad \frac{\mathbf{P} \xrightarrow{m} \mathbf{Q}}{\mathbf{g} \xrightarrow{m} \mathbf{Q}}\ \mathbf{g} \doteq \mathbf{P}$$

$$\mathtt{CompLeft} \qquad \frac{\mathbf{P} \xrightarrow{m} \mathbf{P'}}{\mathbf{P\ ch\ Q} \xrightarrow{m} \mathbf{P'\ ch\ Q}}$$

$$\mathtt{CompRight} \qquad \frac{\mathbf{Q} \xrightarrow{m} \mathbf{Q'}}{\mathbf{P\ ch\ Q} \xrightarrow{m} \mathbf{P\ ch\ Q'}}$$

Nous pouvons facilement imaginer cinq sémantiques opérationnelles différentes, $\mathtt{sem}_i(\mathcal{P})$, dont les relations de transition, $\rightarrow_i$, sont engendrées par l'ensemble des règles correspondantes, $\mathcal{R}_i$, pour chaque $i \in \{0, 1, 2, 3, 4\}$.

Dans le Chapitre 10, nous examinerons les différences entre ces cinq sémantiques opérationnelles, ainsi que les conditions sous lesquelles on peut les considérer comme équivalentes. Nous justifierons aussi notre choix pour la sémantique $\mathtt{sem}_0(\mathcal{P})$ qui, selon les critères proposés dans le même chapitre, est "la plus opérationnelle".

9.2.1 Conventions notationnelles

Pour simplifier les notations, une multi-action sera représentée par un mot simple, par exemple "$aa\overline{c}$" pour $\{a, a, \overline{c}\}$, ou "am" pour $\{a\} + m$, si m dénote une multi-action (comme dans la Section II.5.2.3).

Soit $\mathcal{L} \subseteq \mu_f(\mathtt{Act})$ un ensemble de multi-actions, on notera $[\mathcal{L}]$ le plus petit canal contenant les multi-actions de $\mathcal{L}$:

$$[\mathcal{L}] \stackrel{\text{def}}{=} \{n \in \mu_f(\mathtt{Act}) \mid \exists m \in \mathcal{L}\ n \subseteq m\} \cup \{\emptyset\}\ .$$

On notera “[ ]” le canal minimal, c’est-à-dire $\{\emptyset\}$. Les canaux finis seront représentés par les listes de leurs éléments maximaux (par rapport à $\subseteq$) entre crochets, ex. “$[aa, b]$” pour $\{\{a, a\}, \{a\}, \{b\}, \emptyset\}$. Pour les canaux infinis nous utiliserons la notation μ_f ; $\mu_f(A)$ représentera l’ensemble de toutes les multi-actions construites sur les actions d’un ensemble $A \subseteq \mathtt{Act}$, par exemple, $\mu_f(\mathtt{Act})$ représente l’ensemble de toutes les multi-actions, et $\mu_f(\{a, b\})$ (ou $\mu_f(ab)$), représente l’ensemble de toutes les multi-actions construites sur les actions a et b.

La Proposition III.1, nous permet de composer de tels canaux par les opérations : union ($\cup$), intersection ($\cap$), somme ($+$) et conjugaison ($^{-}$)

Dans l’ensemble “*Agents*” d’identificateur d’agents, il existe toujours une infinité d’agents g avec les équations définissantes triviales, $g \doteq g$. Nous en choisissons un que nous appelons “`stop`”. C’est-à-dire que l’équation définissante associée à l’agent `stop` est toujours : $\mathtt{stop} \doteq \mathtt{stop}$.

Exemple III.1 *Supposons que nous voulions modéliser un agent “Clock” dont le comportement est le suivant. L’agent attend un signal* c *et, après, il envoie un signal* s*. Ensuite, il attend le deuxième signal* c*, puis il envoie deux signaux* s *simultanément (c.-à-d. il exécute la multi-action* ss*) et ainsi de suite. Donc après avoir reçu le n-ème signal* c*, Clock exécute la multi-action* s^n*:*

$$\mathrm{c} \to \mathrm{s} \to \mathrm{c} \to \mathrm{ss} \to \mathrm{c} \to \mathrm{sss} \to \mathrm{c} \to \mathrm{ssss} \to \mathrm{c} \cdots .$$

Pour implémenter l’agent “Clock” dans MAPA, nous définissons d’abord les cinq agents suivants T, E, E_F, O, O_F tels que :

$$\begin{array}{rcl}
\mathrm{T} & \doteq & \mathrm{c}; \mathrm{e}; \mathrm{T} \\
\mathrm{E} & \doteq & \overline{\mathrm{e}}\mathrm{so}; \mathrm{E} \\
\mathrm{E}_F & \doteq & \overline{\mathrm{e}}\mathrm{s}; (\mathrm{E}\ \ [\mathrm{o}]\ \ (\mathrm{O}_F : \mu_f(\mathrm{s}) + [\overline{\mathrm{o}}])) \\
\mathrm{O} & \doteq & \overline{\mathrm{o}}\mathrm{se}; \mathrm{O} \\
\mathrm{O}_F & \doteq & \overline{\mathrm{o}}\mathrm{s}; (\mathrm{O}\ \ [\mathrm{e}]\ \ (\mathrm{E}_F : \mu_f(\mathrm{s}) + [\overline{\mathrm{e}}]))
\end{array}$$

Clock peut alors être défini par : $(\mathrm{T}\ \ [\mathrm{e}]\ \ \mathrm{E}_F) : [\mathrm{c}] \cup \mu_f(\mathrm{s})$.

La dérivation initiale de Clock est donnée ci-dessous. Les redexes développés sont soulignés et les boucles via $\emptyset$ sont signalées par $(*)$ sur le numéro du terme.

$$(\underline{\mathrm{T}}\ \ [\mathrm{e}]\ \ \underline{\mathrm{E}_F}) : [\mathrm{c}] \cup \mu_f(\mathrm{s}) \xrightarrow{\emptyset} \qquad : \ 1$$

$$(\underline{\mathrm{c}}; \mathrm{e}; \mathrm{T}\ \ [\mathrm{e}]\ \ \overline{\mathrm{e}}\mathrm{s}; (\mathrm{E}\ \ [\mathrm{o}]\ \ (\mathrm{O}_F : \mu_f(\mathrm{s}) + [\overline{\mathrm{o}}]))) : [\mathrm{c}] \cup \mu_f(\mathrm{s}) \xrightarrow{\mathrm{c}} \qquad : \ 2^*$$

$$(\underline{\mathrm{e}}; \mathrm{T}\ \ [\mathrm{e}]\ \ \underline{\overline{\mathrm{e}}\mathrm{s}}; (\mathrm{E}\ \ [\mathrm{o}]\ \ (\mathrm{O}_F : \mu_f(\mathrm{s}) + [\overline{\mathrm{o}}]))) : [\mathrm{c}] \cup \mu_f(\mathrm{s}) \xrightarrow{\mathrm{s}} \qquad : \ 3^*$$

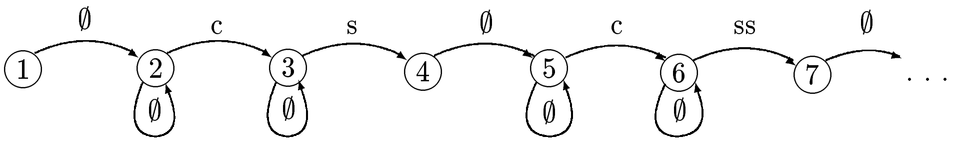
$$(\underline{\mathrm{T}}\ \ [\mathrm{e}]\ \ (\underline{\mathrm{E}}\ \ [\mathrm{o}]\ \ (\underline{\mathrm{O}_F} : \mu_f(\mathrm{s}) + [\overline{\mathrm{o}}]))) : [\mathrm{c}] \cup \mu_f(\mathrm{s}) \xrightarrow{\emptyset} \qquad : \ 4$$

$$(\underline{\mathrm{c}}; \mathrm{e}; \mathrm{T}\ [\mathrm{e}]\ (\overline{\mathrm{e}}\mathrm{so}; \mathrm{E}\ [\mathrm{o}]\ ((\overline{\mathrm{o}}\mathrm{s}; (\mathrm{O}\ [\mathrm{e}]\ (\mathrm{E}_F : \mu_f(\mathrm{s}) + [\overline{\mathrm{e}}]))) : \mu_f(\mathrm{s}) + [\overline{\mathrm{o}}]))) : [\mathrm{c}] \cup \mu_f(\mathrm{s}) \xrightarrow{\mathrm{c}} \qquad : \ 5^*$$

$$(\underline{\mathrm{e}}; \mathrm{T}\ [\mathrm{e}]\ (\underline{\overline{\mathrm{e}}\mathrm{so}}; \mathrm{E}\ [\mathrm{o}]\ ((\underline{\overline{\mathrm{o}}\mathrm{s}}; (\mathrm{O}\ [\mathrm{e}]\ (\mathrm{E}_F : \mu_f(\mathrm{s}) + [\overline{\mathrm{e}}]))) : \mu_f(\mathrm{s}) + [\overline{\mathrm{o}}]))) : [\mathrm{c}] \cup \mu_f(\mathrm{s}) \xrightarrow{\mathrm{ss}} \qquad : \ 6^*$$

$$(\mathrm{T}\ \ [\mathrm{e}]\ \ (\mathrm{E}\ \ [\mathrm{o}]\ \ ((\mathrm{O}\ \ [\mathrm{e}]\ \ (\mathrm{E}_F : \mu_f(\mathrm{s}) + [\overline{\mathrm{e}}])) : \mu_f(\mathrm{s}) + [\overline{\mathrm{o}}]))) : [\mathrm{c}] \cup \mu_f(\mathrm{s}) \xrightarrow{\emptyset} \quad \cdots \qquad : \ 7$$

ce qui peut être représenté graphiquement de la façon suivante :



Abstraction faite des multi-actions vides, nous obtenons bien exactement ce que nous voulions.

Essayons de prouver que la dérivation de l'agent Clock reste correcte aussi par la suite et non seulement pour la partie développée. Pour cela nous définissons les termes P_0, P'_0, P_1, P'_1, …, P_i, P'_i, …, de la manière suivante :

1. $P_0 \stackrel{\text{def}}{=} \mathrm{E}_F$

   $P'_0 \stackrel{\text{def}}{=} \overline{\mathrm{e}}\mathrm{s}; (\mathrm{E}\ \ [\mathrm{o}]\ \ (\mathrm{O}_F : ch_{\mathrm{O}}))$

2. $P_1 \stackrel{\text{def}}{=} \mathrm{E}\ \ [\mathrm{o}]\,(\mathrm{O}_F : ch_{\mathrm{O}})$

   $P'_1 \stackrel{\text{def}}{=} (\overline{\mathrm{e}}\mathrm{so}; \mathrm{E})\ \ [\mathrm{o}]\ \ ((\overline{\mathrm{o}}\mathrm{s}; (\mathrm{O}\ \ [\mathrm{e}]\ \ (\mathrm{E}_F : ch_{\mathrm{E}}))) : ch_{\mathrm{O}})$

3. Pour tout $i \geq 2$,

   $P_i \stackrel{\text{def}}{=} \mathrm{E}\ \ [\mathrm{o}]\ \ ((\mathrm{O}\ \ [\mathrm{e}]\ \ (P_{i-2} : ch_{\mathrm{E}})) : ch_{\mathrm{O}})$

   $P'_i \stackrel{\text{def}}{=} (\overline{\mathrm{e}}\mathrm{so}; \mathrm{E})\ \ [\mathrm{o}]\ \ \Big(\Big((\overline{\mathrm{o}}\mathrm{se}; \mathrm{O})\ \ [\mathrm{e}]\ \ \Big(P'_{i-2} : ch_{\mathrm{E}}\Big)\Big) : ch_{\mathrm{O}}\Big)$

où les canaux ch_{O} et ch_{E} sont définis par :

- $ch_{\mathrm{O}} \ \stackrel{\text{def}}{=}\ \mu_f(\mathrm{s}) + [\overline{\mathrm{o}}]$,
- $ch_{\mathrm{E}} \ \stackrel{\text{def}}{=}\ \mu_f(\mathrm{s}) + [\overline{\mathrm{e}}]$.

Lemme III.3 *Pour tout $i \geq 0$, nous avons :*

$$\begin{tikzcd} P_i \arrow[r, "\emptyset"] & P'_i \arrow[r, "\overline{\mathrm{e}}\mathrm{s}^{i+1}"] \arrow[loop below, "\emptyset"] & P_{i+1} \end{tikzcd}$$

(noté par : "$P_i \xrightarrow{\emptyset} (P'_i)^* \xrightarrow{\overline{\mathrm{e}}\mathrm{s}^{i+1}} P_{i+1}$")

et ce sont les seules transitions possibles à partir de P_i et P'_i dont les étiquettes (multi-actions) ne contiennent ni de $\overline{\mathrm{o}}$ ni de o.

Preuve du Lemme III.3

Par récurrence sur i.

1. On vérifie facilement que le lemme est vrai pour P_0 et P_1.

$$\underbrace{\mathrm{E}_F}_{P_0} \xrightarrow{\emptyset} \underbrace{\overline{\mathrm{e}}\mathrm{s}; (\mathrm{E}\ \ [\mathrm{o}]\ \ (\mathrm{O}_F : ch_{\mathrm{O}}))}_{(P'_0)^*} \xrightarrow{\overline{\mathrm{e}}\mathrm{s}} \underbrace{\mathrm{E}\ \ [\mathrm{o}]\ \ (\mathrm{O}_F : ch_{\mathrm{O}})}_{P_1}$$

$$\underbrace{\mathrm{E}\ \ [\mathrm{o}]\ \ (\mathrm{O}_F : ch_{\mathrm{O}})}_{P_1} \xrightarrow{\emptyset} \underbrace{(\overline{\mathrm{e}}\mathrm{so}; \mathrm{E})\ \ [\mathrm{o}]\ \ ((\overline{\mathrm{o}}\mathrm{s}; (\mathrm{O}\ \ [\mathrm{e}]\ \ (\mathrm{E}_F : ch_{\mathrm{E}}))) : ch_{\mathrm{O}})}_{(P'_1)^*}$$

$$\xrightarrow{\bar{\mathrm{e}}\mathrm{ss}} \underbrace{\mathrm{E}\ \ [\mathrm{o}]\ \ \left(\left(\mathrm{O}\ \ [\mathrm{e}]\ \ \left(\underbrace{\mathrm{E}_F}_{P_0} : ch_{\mathrm{E}}\right)\right) : ch_{\mathrm{O}}\right)}_{P_2}$$

D'autres transitions sont possibles, mais elles contiennent toutes un $\bar{\mathrm{o}}$ ou un o.

2. Supposons que pour un certain $k \geq 1$, si $i \leq k$ alors le lemme est vrai.

   Par définition :

$$P_{k+1} \stackrel{\mathrm{def}}{=} \mathrm{E}\ \ [\mathrm{o}]\ \ ((\mathrm{O}\ \ [\mathrm{e}]\ \ (P_{k-1} : ch_{\mathrm{E}})) : ch_{\mathrm{O}})$$

   Par hypothèse nous avons :

$$P_{k-1} \xrightarrow{\emptyset} (P'_{k-1})^* \xrightarrow{\bar{\mathrm{e}}\mathrm{s}^k} P_k$$

   D'autres transitions à partir de P_{k-1} ou P'_{k-1}, si elles existent, contiennent $\bar{\mathrm{o}}$ ou o (ou les deux). Cela veut dire que

$$P_{k-1} : ch_{\mathrm{E}} \xrightarrow{\emptyset} (P'_{k-1} : ch_{\mathrm{E}})^* \xrightarrow{\bar{\mathrm{e}}\mathrm{s}^k} P_k : ch_{\mathrm{E}}$$

   est la seule dérivation possible de $(P_{k-1} : ch_{\mathrm{E}})$ jusqu'à $(P_k : ch_{\mathrm{E}})$ (sans oublier la boucle via $\emptyset$ sur $(P'_{k-1} : ch_{\mathrm{E}})$ signalée par $(^*)$), car $\bar{\mathrm{o}}$ et o ne font pas partie de ch_{E}.

   Donc, nous avons directement :

$$P_{k+1} \xrightarrow{\emptyset} \left(\underbrace{(\bar{\mathrm{e}}\mathrm{so};\mathrm{E})\ \ [\mathrm{o}]\ \ (((\bar{\mathrm{o}}\mathrm{se};\mathrm{O})\ \ [\mathrm{e}]\ \ (P'_{k-1} : ch_{\mathrm{E}})) : ch_{\mathrm{O}})}_{P'_{k+1}}\right)^*$$

$$\xrightarrow{\bar{\mathrm{e}}\mathrm{sss}^k} \underbrace{\mathrm{E}\ \ [\mathrm{o}]\ \ ((\mathrm{O}\ \ [\mathrm{e}]\ \ (P_k : ch_{\mathrm{E}})) : ch_{\mathrm{O}})}_{P_{k+2}}$$

   D'autres transitions sont possibles, mais alors elles contiennent $\bar{\mathrm{o}}$ ou o.

■

En utilisant le Lemme III.3, il est facile de montrer que la dérivation de Clock est bien celle que nous espérons et qu'il n'y en a pas d'autres, c.-à-d. :

$$\begin{tikzcd}
K_0 \arrow[r, bend left, "\emptyset"] & K_1 \arrow[loop below, "\emptyset"] \arrow[r, bend left, "\mathrm{c}"] & K_2 \arrow[loop below, "\emptyset"] \arrow[r, bend left, "\mathrm{s}"] & K_3 \ \cdots\ K_{3i} \arrow[r, bend left, "\emptyset"] & K_{3i+1} \arrow[loop below, "\emptyset"] \arrow[r, bend left, "\mathrm{c}"] & K_{3i+2} \arrow[loop below, "\emptyset"] \arrow[r, bend left, "\mathrm{s}^i"] & K_{3(i+1)} \cdots
\end{tikzcd}$$

où l'état K_0 dénote "Clock", c.-à-d. le terme $(\mathrm{T}\ \ [\mathrm{e}]\ \ P_0) : [\mathrm{c}] \cup \mu_f(\mathrm{s})$, l'état K_1 dénote le terme $((\mathrm{c};\mathrm{e};\mathrm{T})\ \ [\mathrm{e}]\ \ P'_0) : [\mathrm{c}] \cup \mu_f(\mathrm{s})$, l'état K_2 dénote le terme $((\mathrm{e};\mathrm{T})\ \ [\mathrm{e}]\ \ P'_0) : [\mathrm{c}] \cup \mu_f(\mathrm{s})$ et l'état K_3 dénote le terme $(\mathrm{T}\ \ [\mathrm{e}]\ \ P_1) : [\mathrm{c}] \cup \mu_f(\mathrm{s})$, et ainsi de suite, c.-à-d. que pour tout $i \geq 0$, nous avons :

– l'état K_{3i} dénote le terme $(\mathrm{T}\ \ [\mathrm{e}]\ \ P_i):[\mathrm{c}] \cup \mu_f(\mathrm{s})$,

– l'état K_{3i+1} dénote le terme $((\mathrm{c};\mathrm{e};\mathrm{T})\ \ [\mathrm{e}]\ \ P_i'):[\mathrm{c}] \cup \mu_f(\mathrm{s})$,

– l'état K_{3i+2} dénote le terme $((\mathrm{e};\mathrm{T})\ \ [\mathrm{e}]\ \ P_i'):[\mathrm{c}] \cup \mu_f(\mathrm{s})$.

9.3 Sortes

Une *sorte* (ou *alphabet*, cf. [BBK87a]) d'un processus est un ensemble d'*actions* contenant toutes les *actions* qui peuvent être exécutées par le processus. Pour nous, "*actions*" signifiera ici *multi-actions*. La sorte d'un processus peut être très utile pour une analyse statique et syntaxique du processus, et peut être explorée pendant une implémentation. Clairement, chaque terme P possède une sorte minimale, que nous noterons $\mathtt{Sort}(P)$ (dans [BBK87a], noté $\alpha(P)$). C'est l'ensemble de toutes les étiquettes de transitions de son *graphe de processus* (voir Def. I.2). En général, la sorte minimale d'un terme P ne peut pas être calculée, vu que le problème est non-décidable (cf. [BBK87a]). Cependant, il est possible d'obtenir une approximation de la sorte minimale d'un terme via une analyse syntaxique. Comme Milner dans [Mil89], nous introduisons ici la notion de *sortes syntaxiques* $\mathcal{L} \subseteq \mu_f(\mathtt{Act})$ d'un terme P, notée $P :: \mathcal{L}$, que nous définissons par les règles suivantes :

1. $P :: \mathcal{L}_1 \wedge \mathcal{L}_1 \subseteq \mathcal{L}_2 \ \Rightarrow\ P :: \mathcal{L}_2$

2. $P :: \mathcal{L} \ \Rightarrow\ m; P :: (\mathcal{L} \cup \{\emptyset\} \cup \{m\})$

3. $P :: \mathcal{L}_1, Q :: \mathcal{L}_2 \Rightarrow P\ ch\ Q :: \{n_1 + n_2 \mid \exists m \in ch\ \ n_1 + m \in \mathcal{L}_1 \wedge n_2 + \overline{m} \in \mathcal{L}_2\}$

4. $P :: \mathcal{L} \ \Rightarrow\ P : ch :: (\mathcal{L} \cap ch)$

5. $\left(\left\{\begin{array}{c} g_1 :: \mathcal{L}_1 \cup [\,] \\ \cdots \\ g_n :: \mathcal{L}_n \cup [\,] \end{array}\right\} \Rightarrow \left\{\begin{array}{c} G_1 :: \mathcal{L}_1 \\ \cdots \\ G_n :: \mathcal{L}_n \end{array}\right\}\right) \Rightarrow \left\{\begin{array}{c} g_1 :: \mathcal{L}_1 \\ \cdots \\ g_n :: \mathcal{L}_n \end{array}\right\},\quad \text{où} \left\{\begin{array}{c} g_1 \doteq G_1 \\ \cdots \\ g_n \doteq G_n \end{array}\right\}$

La première règle nous permet d'augmenter la sorte d'un terme. Les trois règles suivantes correspondent aux trois opérations de notre langage : le *préfixage*, la *composition* et la *restriction*, et elles sont déduites directement des règles de la sémantique opérationnelle. La dernière règle permet de calculer les sortes des agents récursifs.

On peut remarquer la différence entre la définition de sorte proposée ici et celle proposée par Baeten et al. [BBK87a], où seules les actions "visibles" font partie d'une sorte, tandis qu'ici la multi-action vide $\emptyset$ appartient toujours à une sorte. Notre choix pour ne pas distinguer la multi-action vide a été dicté uniquement par des raisons techniques ([2]).

[2]Dans MAPA (avec l'ensemble de règles de dérivation $\mathcal{R}_0$) par le Lemme III.2, $\emptyset \in \mathtt{Sort}(P)$ pour tout terme P.

Proposition III.4 *Si, en utilisant les règles données ci-dessus, nous prouvons* $P :: \mathcal{L}$*, alors* $\mathcal{L}$ *est une sorte de* P ($\mathtt{Sort}(P) \subseteq \mathcal{L}$).

La preuve de cette proposition est basée sur les deux lemmes suivants.

Lemme III.5 $P :: \mathcal{L} \Rightarrow \emptyset \in \mathcal{L}$.

Preuve du Lemme III.5 Aucune règle n'élimine la multi-action $\emptyset$, et par le schéma de la Règle 5 pour la sorte d'un agent g, nous devons supposer $g :: \mathcal{L} \cup \{\emptyset\}$. ∎

Lemme III.6

(a) $P \xrightarrow{m} Q \;\Rightarrow\; (P :: \mathcal{L} \Rightarrow m \in \mathcal{L})$

(b) $P \xrightarrow{m} Q \;\Rightarrow\; (P :: \mathcal{L} \Rightarrow Q :: \mathcal{L})$

Preuve du Lemme III.6

Par induction sur la hauteur du terme P :

1. Si $\mathtt{high}(P) = 0$ alors P est un identificateur d'agent $g \doteq G$. Il y a une seule règle de dérivation pour un tel terme : $g \xrightarrow{\emptyset} G$.

   (a) Par le Lemme III.5, on obtient directement $g \xrightarrow{\emptyset} G \;\Rightarrow\; (g :: \mathcal{L} \Rightarrow \emptyset \in \mathcal{L})$.

   (b) Une preuve de $g :: \mathcal{L}$ doit commencer par le schéma de Règle 5, en donnant $g :: \mathcal{L}'$ avec $\mathcal{L}' \subseteq \mathcal{L}$. La sorte de g peut alors être augmentée via la Règle 1 :

$$
\begin{aligned}
g :: \mathcal{L} &\Rightarrow \exists \mathcal{L}' \; \overrightarrow{\exists g_i} \; \overrightarrow{\exists \mathcal{L}_i} (\mathcal{L}' \subseteq \mathcal{L} \wedge (g :: \mathcal{L}' \wedge \overrightarrow{g_i :: \mathcal{L}_i} \Rightarrow G :: \mathcal{L}' \wedge \overrightarrow{G_i :: \mathcal{L}_i})) \\
&\Rightarrow \exists \mathcal{L}' \; \overrightarrow{\exists g_i} \; \overrightarrow{\exists \mathcal{L}_i} (\mathcal{L}' \subseteq \mathcal{L} \wedge (g :: \mathcal{L}' \wedge \overrightarrow{g_i :: \mathcal{L}_i} \Rightarrow G :: \mathcal{L}' \wedge \overrightarrow{G_i :: \mathcal{L}_i}) \wedge \\
& \; g :: \mathcal{L}' \wedge \overrightarrow{g_i :: \mathcal{L}_i}) \\
&\Rightarrow \exists \mathcal{L}' \; (\mathcal{L}' \subseteq \mathcal{L} \wedge G :: \mathcal{L}') \\
&\Rightarrow G :: \mathcal{L}
\end{aligned}
$$

   où $\overrightarrow{\exists g_i}$ dénote $\exists g_1 \exists g_2 \ldots \exists g_n$ (de même pour $\overrightarrow{\exists \mathcal{L}_i}$) et $\overrightarrow{g_i :: \mathcal{L}_i}$ dénote $g_1 :: \mathcal{L}_1 \wedge g_2 :: \mathcal{L}_2 \wedge \ldots \wedge g_n :: \mathcal{L}_n$ (de même pour $\overrightarrow{G_i :: \mathcal{L}_i}$). Le $n+1$ dénote le nombre d'agents utilisés dans la Règle 5 (+ 1 pour l'agent en question g). Bien évidemment, $g_i \doteq G_i$ pour tout $i \leq n$, comme la règle l'exige.

2. Supposons que pour un $k \geq 0$ le Lemme III.6 est vrai pour tout terme P tel que $\mathtt{high}(P) \leq k$.

   Soit P un terme de hauteur $\mathtt{high}(P) = k + 1$.

   Les trois cas suivants sont possibles :

   $\underline{P = m; Q}$ Indépendamment de $\mathtt{high}(P)$, il n'y a que deux transitions à partir du terme $m; Q$: $m; Q \xrightarrow{\emptyset} m; Q$ et $m; Q \xrightarrow{m} Q$.

(a) Par le Lemme III.5 nous avons $P :: \mathcal{L} \Rightarrow \emptyset \in \mathcal{L}$. En plus, une preuve de $m; Q :: \mathcal{L}$ doit utiliser la Règle 2, formellement

$$\begin{aligned}
m; Q :: \mathcal{L} &\Rightarrow \exists \mathcal{L}' \ (\mathcal{L}' \subseteq \mathcal{L} \wedge \exists \mathcal{L}'' \ (Q :: \mathcal{L}'' \wedge (\mathcal{L}' = \mathcal{L}'' \cup \{m\}))) \\
&\Rightarrow \exists \mathcal{L}' \ (\mathcal{L}' \subseteq \mathcal{L} \wedge m \in \mathcal{L}' \wedge \exists \mathcal{L}'' \ (\mathcal{L}'' \subseteq \mathcal{L}' \wedge Q :: \mathcal{L}'')) \\
&\Rightarrow \exists \mathcal{L}' \ (\mathcal{L}' \subseteq \mathcal{L} \wedge m \in \mathcal{L}' \wedge Q :: \mathcal{L}') \\
&\Rightarrow m \in \mathcal{L}
\end{aligned}$$

(b) De l'avant-dernière implication nous pouvons conclure aussi $m; Q :: \mathcal{L} \Rightarrow Q :: \mathcal{L}$, et l'implication $m; Q :: \mathcal{L} \Rightarrow m; Q :: \mathcal{L}$ est toujours vraie.

$\underline{P = Q : ch}$ Comme $\texttt{high}(Q : ch) = k + 1$, alors $\texttt{high}(Q) = k$.

$Q : ch \xrightarrow{m} R$ si et seulement si $R = Q' : ch$, $m \in ch$ et $Q \xrightarrow{m} Q'$.

Une preuve de $Q : ch :: \mathcal{L}$ doit passer par la Règle 4 ; formellement

$$Q : ch :: \mathcal{L} \Rightarrow \exists \mathcal{L}' \ (\mathcal{L}' \subseteq \mathcal{L} \wedge \exists \mathcal{L}'' \ (Q :: \mathcal{L}'' \wedge (\mathcal{L}' = \mathcal{L}'' \cap ch)))$$

(a) Par hypothèse, $Q :: \mathcal{L}'' \Rightarrow m \in \mathcal{L}''$. Comme $m \in ch$ et $m \in \mathcal{L}''$, on a $Q : ch :: \mathcal{L} \Rightarrow m \in \mathcal{L}$.

(b) Par hypothèse nous avons aussi $Q :: \mathcal{L}'' \Rightarrow Q' :: \mathcal{L}''$. Alors

$$\begin{aligned}
Q : ch :: \mathcal{L} &\Rightarrow \exists \mathcal{L}' \ (\mathcal{L}' \subseteq \mathcal{L} \wedge \exists \mathcal{L}'' \ (Q' :: \mathcal{L}'' \wedge (\mathcal{L}' = \mathcal{L}'' \cap ch))) \\
&\Rightarrow \exists \mathcal{L}' \ (\mathcal{L}' \subseteq \mathcal{L} \wedge \exists \mathcal{L}'' \ (Q' : ch :: (\mathcal{L}'' \cap ch) \wedge (\mathcal{L}' = \mathcal{L}'' \cap ch))) \\
&\Rightarrow \exists \mathcal{L}' \ (\mathcal{L}' \subseteq \mathcal{L} \wedge Q' : ch :: \mathcal{L}') \\
&\Rightarrow Q' : ch :: \mathcal{L}
\end{aligned}$$

$\underline{P = Q \ ch \ R}$ $Q \ ch \ R \xrightarrow{m} Z$ si et seulement si $Z = Q' \ ch \ R'$ et il existe des multi-actions n, n_1, n_2 telles que $n \in ch$, $n_1 + n_2 = m$, $Q \xrightarrow{n+n_1} Q'$ et $R \xrightarrow{\overline{n}+n_2} R'$. En plus $\texttt{high}(Q) \leq k$ et $\texttt{high}(R) \leq k$, car $\texttt{high}(Q \ ch \ R) = k + 1$.

Une preuve de $Q \ ch \ R :: \mathcal{L}$ utilise sûrement la Règle 3 ; formellement

$$\begin{aligned}
Q \ ch \ R :: \mathcal{L} \Rightarrow \ &\exists \mathcal{L}' (\mathcal{L}' \subseteq \mathcal{L} \wedge \exists \mathcal{L}_Q, \mathcal{L}_R \ (Q :: \mathcal{L}_Q \wedge R :: \mathcal{L}_R \wedge \\
&\mathcal{L}' = \{n_1 + n_2 \mid \exists n \in ch \ \ n + n_1 \in \mathcal{L}_Q \wedge \overline{n} + n_2 \in \mathcal{L}_R\}))
\end{aligned}$$

(a) Par hypothèse, $Q :: \mathcal{L}_Q \Rightarrow (n + n_1) \in \mathcal{L}_Q$ et $R :: \mathcal{L}_R \Rightarrow (\overline{n} + n_2) \in \mathcal{L}_R$. Donc

$$\begin{aligned}
Q \ ch \ R :: \mathcal{L} &\Rightarrow \exists \mathcal{L}' (\mathcal{L}' \subseteq \mathcal{L} \wedge m \in \mathcal{L}') \\
&\Rightarrow m \in \mathcal{L}
\end{aligned}$$

(b) Par hypothèse, $Q :: \mathcal{L}_Q \Rightarrow Q' :: \mathcal{L}_Q$ et $R :: \mathcal{L}_R \Rightarrow R' :: \mathcal{L}_R$. Donc

$$\begin{aligned}
Q\ ch\ R :: \mathcal{L} \quad &\Rightarrow \quad \exists \mathcal{L}'(\mathcal{L}' \subseteq \mathcal{L} \wedge \exists \mathcal{L}_Q, \mathcal{L}_R\ (Q' :: \mathcal{L}_Q \wedge R' :: \mathcal{L}_R \wedge \\
&\qquad \mathcal{L}' = \{n_1 + n_2 \mid \exists n \in ch(n + n_1 \in \mathcal{L}_Q \wedge \overline{n} + n_2 \in \mathcal{L}_R)\})) \\
&\Rightarrow \quad \exists \mathcal{L}'(\mathcal{L}' \subseteq \mathcal{L} \wedge Q'\ ch\ R' :: \mathcal{L}') \\
&\Rightarrow \quad Q'\ ch\ R' :: \mathcal{L}
\end{aligned}$$

■

En utilisant le Lemme III.6, on peut maintenant prouver la Proposition III.4, c.-à-d. prouver que $P :: \mathcal{L} \Rightarrow \mathtt{Sort}(P) \subseteq \mathcal{L}$. En effet, si une multi-action m est dans $\mathtt{Sort}(P)$, alors il existe un dérivé Q de P, c.-à-d. une suite finie de transitions $P \stackrel{m_1}{\to} P_1 \stackrel{m_2}{\to} \ldots \stackrel{m_i}{\to} Q$, tel que $\exists Q'\ Q \stackrel{m}{\to} Q'$. Par itération du Lemme III.6(b), nous avons immédiatement $P :: \mathcal{L} \ \Rightarrow\ Q :: \mathcal{L}$, et par le Lemme III.6(a), nous obtenons $m \in \mathcal{L}$.

9.4 Relations d'équivalence

Dans cette section, nous rappelons et adaptons pour MAPA quelques notions de base de la *sémantique des processus*. Nous commençons par des équivalences sur les systèmes de transitions : l'*isomorphisme de graphes de processus* ($\equiv$), l'*équivalence forte* ($\simeq$) et l'*équivalence faible* ($\approx$), telles que définies dans la première partie, Section I.1.1. Ensuite, nous définissons une équivalence qui n'est pas directement basée sur le système de transitions $\mathtt{sem}(\mathcal{P})$, mais qui est définie par rapport aux opérations de MAPA. Pour chaque équivalence présentée, nous établissons plusieurs propriétés intéressantes de notre algèbre.

Dans la Section III.9.2, nous avons défini le système de transitions $\mathtt{sem}(\mathcal{P})$, dont les états étaient justement des termes de notre algèbre MAPA. Ainsi, le domaine sémantique de MAPA est constitué des états de $\mathtt{sem}(\mathcal{P})$; un terme P est alors interprété comme le graphe de processus $\mathtt{graph}(P)$ dans $\mathtt{sem}(\mathcal{P})$.

9.4.1 Isomorphisme de graphes de processus

On dit qu'un état s d'un système de transition $\mathtt{TS} = (\mathtt{St}, \mathtt{Lab}, \to)$ est à *branchement fini* (ang. *finitely branching*) si l'ensemble des transitions sortant de s, c.-à-d. $\{(\alpha, x) \in \mathtt{Lab} \times \mathtt{St} \mid s \stackrel{\alpha}{\to} x\}$, est fini. On dira alors qu'un système de transitions (ou graphe de processus) est à *branchement fini* si et seulement si chacun de ses états est à branchement fini. Cela signifie qu'un processus représenté par un tel graphe de processus a, à tout moment, un nombre fini de choix pour continuer.

Proposition III.7 *Le système de transitions* $\mathtt{sem}(\mathcal{P})$ *est à branchement fini.*

Preuve. Soit $\delta(P)$, le nombre de transitions sortant de l'état P dans $\texttt{sem}(\mathcal{P})$, c.-à-d. :

$$\delta(P) \stackrel{\text{def}}{=} \left|\{(m,Q) \in \mu_f(\texttt{Act}) \times \mathcal{P} \mid P \stackrel{m}{\rightarrow} Q\}\right| .$$

Par induction sur la hauteur du terme P :

1. Pour tout terme P de hauteur $\texttt{high}(P) = 0$, le nombre de transitions sortantes est fini.

   En effet, un tel terme P doit être un agent g, et donc il y a exactement une transition sortante : $g \stackrel{\emptyset}{\rightarrow} G$, où $g \doteq G$. Il s'ensuit que $\forall P \in \mathcal{P}$, $\texttt{high}(P) = 0$ implique $\delta(P) = 1$.

2. Supposons que pour un $k \geq 0$ et tout P de $\mathcal{P}$, $\texttt{high}(P) \leq k$ implique que $\delta(P)$ est fini.

   Soit P, un terme de hauteur $\texttt{high}(P) = k + 1$.

   Les trois cas suivants doivent être considérés :

   (a) $P = m; Q$, alors $\delta(P) = 2$;

   (b) $P = Q : ch$, alors $\delta(P) \leq \delta(Q)$;

   (c) $P = Q\ ch\ R$, avec $\max(\texttt{high}(Q), \texttt{high}(R)) = k$. Par hypothèse, $\delta(Q) = d_1$ et $\delta(R) = d_2$ sont finis. $\delta(P) \leq d_1 * d_2 * 2^{k_1+k_2}$, où k_1 est le plus grand des cardinaux des multi-actions qui étiquettent les transitions sortant de Q, et k_2 celui pour R, respectivement. Evidemment, $d_1 * d_2 * 2^{k_1+k_2}$ est un nombre fini.

■

Définition III.3 Soient **P** et **Q** deux termes de MAPA. Les termes sont équivalents par *isomorphisme de graphes de processus*, $\mathbf{P} \equiv \mathbf{Q}$, si et seulement si les graphes de processus $\texttt{graph}(\mathbf{P})$, $\texttt{graph}(\mathbf{Q})$ sont isomorphes. □

Proposition III.8 *Pour tous $P, Q, R \in \mathcal{P}$ et pour tous $ch, ch' \in \texttt{Chan}$, les six propriétés suivantes sont vérifiées :*

1. $P \equiv P\ ch\ \texttt{stop}$.
2. $P\ ch\ Q \equiv Q\ \overline{ch}\ P$.
3. $\texttt{Sort}(P) \subseteq ch \;\Rightarrow\; P \equiv P : ch$.
4. $(P : ch) : ch' \;\equiv\; P : ch \cap ch'$.
5. $ch + ch \subseteq ch \;\Rightarrow\; (P\ ch\ Q)\ ch\ R \;\equiv\; P\ ch\ (Q\ ch\ R)$.
6. $\equiv$ *est une congruence par rapport à la restriction et à la composition.*

Les propriétés 1 à 4 confirment notre intuition sur les opérations de MAPA. On peut toujours enlever ou ajouter, autrement dit, ignorer, les parties inactives, représentées ici par l'agent `stop`, sans changer le comportement global du système (Propriété 1). La composition est commutative si l'on n'oublie pas de "commuter" les connexions de manière adéquate (Propriété 2). Si un processus P n'utilise que des multi-actions d'un canal ch, alors la limitation à ch de la communication de P ne change rien au comportement de P (Propriété 3). La Propriété 4 dit que la restriction est commutative et idempotente. Il est intéressant de remarquer la ressemblance de ces deux résultats (Propriétés 3 et 4) avec ceux de Baeten et al. [BBK87a].

La Propriété 5 dit que la composition est associative et commutative si elle est effectuée via, par exemple, le canal minimal $[\,]$ qui vérifie la pré-condition $[\,] = [\,] + [\,]$. Le fait que $(P\ [\,]\ Q)\ [\,]\ R\ \equiv\ P\ [\,]\ (Q\ [\,]\ R)$ va être utilisé par la suite, dans notre implémentation de l'opérateur de choix.

L'équivalence par isomorphisme de graphes de processus n'est pas préservée par préfixage. Par la Propriété 1, nous avons $P \equiv P\ ch\ \mathtt{stop}$, donc en particulier $g \equiv g\ ch\ \mathtt{stop}$, pour l'agent $g \doteq m; g$. On vérifie que $m; g \not\equiv m; (g\ ch\ \mathtt{stop})$, puisque $\mathtt{graph}(m; g)$ a deux états alors que $\mathtt{graph}(m; (g\ ch\ \mathtt{stop}))$ en a trois.

Preuve de la Proposition III.8

1. Pour prouver que "$P \equiv P\ ch\ \mathtt{stop}$", pour tout terme P, il suffit de trouver une bijection φ entre les états de $\mathtt{graph}(P)$ et de $\mathtt{graph}(P\ ch\ \mathtt{stop})$, qui vérifie la Définition III.3. L'application $\varphi(x) \stackrel{\text{def}}{=} (x\ ch\ \mathtt{stop})$, pour tout terme x accessible à partir de P, vérifie la définition.

2. Comme pour (1), avec $\varphi(x\ ch\ y) \stackrel{\text{def}}{=} (y\ \overline{ch}\ x)$.

3. Comme pour (1), avec $\varphi(x) \stackrel{\text{def}}{=} (x : ch)$.

4. Comme pour (1), avec $\varphi((x : ch) : ch') \stackrel{\text{def}}{=} (x : ch \cap ch')$.

5. Comme pour (1), avec $\varphi((x\ ch\ y)\ ch\ z) \stackrel{\text{def}}{=} (x\ ch\ (y\ ch\ z))$. Seule la vérification que la bijection satisfait $p \stackrel{\alpha}{\to} q \Leftrightarrow \varphi(p) \stackrel{\alpha}{\to} \varphi(q)$ pour tous les états p, q accessibles de respectivement $((P\ ch\ Q)\ ch\ R)$ et $(P\ ch\ (Q\ ch\ R))$, n'est pas aussi évidente que dans les cas précédents.

   (a) $p \stackrel{\alpha}{\to} q \Rightarrow \varphi(p) \stackrel{\alpha}{\to} \varphi(q)$:
   La transition $((x\ ch\ y)\ ch\ z) \stackrel{m}{\to} p$ existe, si et seulement si,
   $\exists x', y', z' \in \mathcal{P}$, $\exists m_x, m'_x, n, m_y, m'_y, m_z \in \mu_f(\mathtt{Act})$, tels que :
   (i) $p = ((x'\ ch\ y')\ ch\ z')$,
   (ii) $x \xrightarrow{m_x + m'_x + n} x'$, $y \xrightarrow{m_y + m'_y + \overline{n}} y'$, $z \xrightarrow{m_z + \overline{m'_x + m'_y}} z'$,
   (iii) $n \in ch$ et $m'_x + m'_y \in ch$
   (iv) $m = m_x + m_y + m_z$.
   Par le fait que $ch + ch \subseteq ch$ et (iii) on a $n + m'_x + m'_y \in ch$.
   Donc $n + m'_x \in ch$ et $m'_y \in ch$.

Par conséquent, $(y\ ch\ z) \xrightarrow{m_z + \overline{m'_x} + m_y + \overline{n}} (y'\ ch\ z')$.

Donc aussi $(x\ ch\ (y\ ch\ z)) \xrightarrow{m_x + m_y + m_z} (x'\ ch\ (y'\ ch\ z'))$.

(b) $p \xrightarrow{\alpha} q \Leftarrow \varphi(p) \xrightarrow{\alpha} \varphi(q)$: Preuve similaire.

6. Nous devons prouver que si $P \equiv Q$, alors, pour tout canal ch et pour tout terme R, $P : ch \equiv Q : ch$ et $P\ ch\ R \equiv Q\ ch\ R$. Puisque $P \equiv Q$, il existe une bijection φ entre les états de $\mathtt{graph}(P)$ et de $\mathtt{graph}(Q)$ qui vérifie la Définition III.3. Nous devons seulement vérifier que $\varphi_1, \varphi_2 : \mathcal{P} \mapsto \mathcal{P}$ définis ci-dessous, sont des bijections qui vérifient respectivement $P : ch \equiv Q : ch$ et $P\ ch\ R \equiv Q\ ch\ R$, quand elles sont restreintes aux états accessibles, ce qui est immédiat.

$$\varphi_1(x) \stackrel{\text{def}}{=} \begin{cases} (\varphi(y) : ch) & \text{si } x = (y : ch) \\ \mathtt{stop} & \text{sinon} \end{cases}$$
$$\varphi_2(x) \stackrel{\text{def}}{=} \begin{cases} (\varphi(y)\ ch\ z) & \text{si } x = (y\ ch\ z) \\ \mathtt{stop} & \text{sinon} \end{cases}$$

■

9.4.2 Equivalence forte et équivalence faible

L'équivalence forte entre les termes de MAPA est définie de la même façon que l'équivalence forte entre les états correspondants de $\mathtt{sem}(\mathcal{P})$.

Définition III.4 Deux termes $\mathbf{P}$, $\mathbf{Q}$ sont *fortement équivalents* ssi $\mathbf{P} \simeq \mathbf{Q}$ dans $\mathtt{sem}(\mathcal{P})$.

□

Pour la définition de l'équivalence faible entre les termes de MAPA nous devons d'abord préciser la notion de transitions invisibles dans le système de transitions $\mathtt{sem}(\mathcal{P})$: ce seront les transitions étiquetées par la multi-action vide $\emptyset$. Cela signifie que dans les définitions de $\stackrel{*}{\rightsquigarrow}$, $\Longrightarrow$ et $\approx$ de la Section I.1.1, quand on les applique à $\mathtt{sem}(\mathcal{P})$, l'étiquette τ doit être remplacée par $\emptyset$.

Définition III.5 Deux termes $\mathbf{P}$, $\mathbf{Q}$ sont *faiblement équivalents* ssi $\mathbf{P} \approx \mathbf{Q}$ dans $\mathtt{sem}(\mathcal{P})$ (où toute transition étiquetée par la multi-action vide est considérée comme interne).

□

Proposition III.9 *Pour tous $P, G \in \mathcal{P}$ et pour tout identificateur d'agent g, les six propriétés suivantes sont vérifiées :*

1. $\equiv\ \subset\ \simeq\ \subset\ \approx$.

2. *Les trois formules suivantes sont équivalentes :*

   $P \simeq \mathtt{stop}$, $\quad P \approx \mathtt{stop}$, $\quad \mathtt{Sort}(P) = [\ ]$.

3. $P : [\,] \;\simeq\; \texttt{stop}$ *.*

4. $g \doteq G \;\Rightarrow\; g \approx G$ *.*

5. $\emptyset; P \approx P$ *.*

6. $\simeq$ *est une congruence dans MAPA.*

Preuve de la Proposition III.9

1. Directement par la Proposition I.2 nous avons $\equiv \;\subseteq\; \simeq \;\subseteq\; \approx$. Les inclusions sont strictes vue les contre-exemples suivants, où $g \doteq m; g$:

   (a) $m; g \not\equiv m; (g\ ch\ \texttt{stop})$ mais $m; g \simeq m; (g\ ch\ \texttt{stop})$,

   (b) $m; g \not\simeq g$ mais $m; g \approx g$.

2. Comme, par 1, on a que $P \simeq \texttt{stop} \Rightarrow P \approx \texttt{stop}$, il suffit de prouver les deux implications suivantes.

   $\texttt{Sort}(P) = [\,] \;\Rightarrow\; P \simeq \texttt{stop}$

   D'abord, nous montrons que $B \stackrel{\text{def}}{=} \{(P, Q) \mid \texttt{Sort}(P) = \texttt{Sort}(Q) = [\,]\}$ est une bisimulation. Pour cela, il suffit de prouver que pour tous les termes P et Q tels que $(P, Q) \in B$, si $P \xrightarrow{m} P'$ alors il existe Q' tel que $Q \xrightarrow{m} Q'$ et $\texttt{Sort}(P') = \texttt{Sort}(Q') = [\,]$. Puisque $\texttt{Sort}(P) = [\,]$, la multi-action m, si elle existe, est toujours égale à $\emptyset$. Or, par le Lemme III.2, il existe toujours des termes P' et Q' tels que $P \xrightarrow{\emptyset} P'$ et $Q \xrightarrow{\emptyset} Q'$. Bien évidemment, $\texttt{Sort}(P') = \texttt{Sort}(Q') = [\,]$, car $\texttt{Sort}(P') \subseteq \texttt{Sort}(P)$ et $\texttt{Sort}(Q') \subseteq \texttt{Sort}(Q)$.

   Après, nous vérifions que $\texttt{Sort}(\texttt{stop}) = [\,]$, en remarquant, par exemple, que le graphe de processus $\texttt{graph}(\texttt{stop})$ n'a qu'un état $\texttt{stop}$ et qu'une transition $\texttt{stop} \xrightarrow{\emptyset} \texttt{stop}$, ou en calculant la sorte syntaxique de $\texttt{stop}$, $\texttt{stop} :: [\,]$, par les règles de la Section III.9.3.

   En conséquence, si $\texttt{Sort}(P) = [\,]$, alors $(\texttt{stop}, P) \in B$.

   $P \approx \texttt{stop} \;\Rightarrow\; \texttt{Sort}(P) = [\,]$

   Par contraposée, nous montrons que $\texttt{Sort}(P) \neq [\,] \;\Rightarrow\; P \not\approx \texttt{stop}$.

   Effectivement, si $\texttt{Sort}(P) \neq [\,]$, alors il existe une suite de transitions $P \xrightarrow{\emptyset} P_1 \xrightarrow{\emptyset} \ldots \xrightarrow{m} Q$, telle que $m \neq \emptyset$, c.-à-d. $P \stackrel{m}{\Longrightarrow} Q$ selon la notation de la Section I.1.1. Or, $\texttt{stop} \stackrel{m}{\Longrightarrow} R$ pour un $m \neq \emptyset$ n'est jamais possible. Donc, $\texttt{stop} \not\approx P$.

3. $\texttt{Sort}(P : [\,]) = [\,]$, car $(P : [\,]) :: [\,]$ peut être prouvé par la Règle 4 introduite dans la Section III.9.3.

   En conséquence, $(P : [\,], \texttt{stop}) \in B$, où B est la bisimulation forte définie pour le point 2. Donc, $P : [\,] \simeq \texttt{stop}$.

4. Il suffit de vérifier que

$$B \stackrel{\text{def}}{=} \{(P,P) \mid P \in \mathcal{P}\} \cup \{(g,G),(G,g) \mid g \doteq G\}$$

est une bisimulation faible, c.-à-d. que si $(P,P') \in B$ et $P \stackrel{m}{\rightarrow}_1 Q$ alors il existe Q' tel que $P' \stackrel{m}{\Longrightarrow} Q'$ et $(Q,Q') \in B$. Un seul cas non trivial est quand $P = G$ et $P' = g$ avec $g \doteq G$. Si $G \stackrel{m}{\rightarrow} Q$, alors $g \stackrel{\emptyset}{\rightarrow} G \stackrel{m}{\rightarrow} Q$, donc $g \stackrel{m}{\Longrightarrow} Q$, c.-à-d. que $Q' = Q$.

5. Il suffit de vérifier que

$$B \stackrel{\text{def}}{=} \{(P,P),(\emptyset;P,P),(P,\emptyset;P) \mid P \in \mathcal{P}\}$$

est une bisimulation faible, c.-à-d. que si $(P,P') \in B$ et $P \stackrel{m}{\rightarrow}_1 Q$ alors il existe Q' tel que $P' \stackrel{m}{\Longrightarrow} Q'$ et $(Q,Q') \in B$. Trivial.

6. Nous devons vérifier que si $P \simeq Q$ alors

   (a) $m;P \simeq m;Q$

   (b) $P : ch \simeq Q : ch$

   (c) $P \ ch \ R \simeq Q \ ch \ R$

   (d) $R \ ch \ P \simeq R \ ch \ Q$

   pour toute multi-action m, tout canal ch et tout terme R.

   Pour cela, nous définissons les trois relations suivantes $B_1, B_2, B_3 \subseteq \mathcal{P} \times \mathcal{P}$:

   $B_1 \stackrel{\text{def}}{=} \ \simeq \ \cup \ \{(m;P, m;Q) \mid m \in \mu_f(\texttt{Act}), P \simeq Q\}$.

   $B_2 \stackrel{\text{def}}{=} \{(P : ch, Q : ch) \mid ch \in \texttt{Chan}(\texttt{Act}), P \simeq Q\}$.

   $B_3 \stackrel{\text{def}}{=} \{(P \ ch \ R, Q \ ch \ Z) \mid ch \in \texttt{Chan}(\texttt{Act}), P \simeq Q, R \simeq Z\}$. ([3])

   Nous vérifions que B_1, B_2, B_3 sont des (bi)simulations fortes, c.-à-d. que pour tous $(P,Q) \in B_i$ et $m \in \mu_f(\texttt{Act})$, si $P \stackrel{m}{\longrightarrow} P'$ alors il existe Q' tel que $Q \stackrel{m}{\rightarrow} Q'$, et $(P',Q') \in B_i$, pour $i \in \{1,2,3\}$.

   Nous donnons ici seulement la preuve pour B_3, les autres cas étant plus faciles.

   Nous avons : $(P \ ch \ R)$ et $(Q \ ch \ Z)$, avec $P \simeq Q$ et $R \simeq Z$.

   Or, $(P \ ch \ R) \stackrel{m}{\longrightarrow} (P' \ ch \ R')$ ssi il existe des multi-actions n, n_1, n_2 telles que :

   $n_1 + n_2 = m, \ \ n \in ch, \ \ P \xrightarrow{n_1+n} P'$ et $R \xrightarrow{n_2+\overline{n}} R'$.

   Puisque $P \simeq Q$ et $R \simeq Z$, il existe Q', Z' tels que $Q \xrightarrow{n_1+n} Q'$ avec $P' \simeq Q'$ et $Z \xrightarrow{n_2+\overline{n}} Z'$ avec $R' \simeq Z'$.

   Donc $(Q \ ch \ Z) \ \stackrel{m}{\longrightarrow} \ (Q' \ ch \ Z')$.

   Comme $P' \simeq Q'$ et $R' \simeq Z'$, on a $((P' \ ch \ R'),(Q' \ ch \ Z')) \in B_3$.

   D'où B_3 est une bisimulation forte. ■

[3]Remarque : B_3 inclut $\{(P \ ch \ R, Q \ ch \ R) \mid P \simeq Q\}$ et $\{(R \ ch \ P, R \ ch \ Q) \mid P \simeq Q\}$.

Dans le Chapitre III.10 nous verrons que l'équivalence faible $\approx$ est aussi une congruence. D'abord, nous montrerons que l'equivalence faible $\approx_1$ dans la sémantique $\mathtt{sem}_1(\mathcal{P})$ est une congruence (Lemme III.13), ensuite nous verrons, par le Théorème III.21, que $\approx_1$ coïncide avec $\approx$ (la relation $\approx$ est notée dans ce chapitre par $\approx_0$) comme d'ailleurs avec les équivalences faibles pour les autres sémantiques opérationnelles mentionnées dans la Section III.9.2.

9.4.3 Equivalence sémantique

MAPA est un langage conçu pour décrire les comportements d'*objets communicants* ; on aura donc envie de disposer d'une sémantique qui distingue seulement les processus qui diffèrent par rapport à leur comportement externe. Comme pour l'équivalence par échec, nous supposerons que la seule manière d'analyser les processus est d'interagir avec eux. Deux processus seront en *équivalence sémantique* s'ils "risquent" de se bloquer exactement dans les mêmes situations. L'équivalence par échec $\approx_f$ implique la supposition que l'environnement ne peut proposer au processus qu'un nombre fini des différentes actions à la fois. Dans le cas de MAPA, cette supposition ne s'accorde pas tout-à-fait avec l'hypothèse que les notions de "processus" et de "son environnement" peuvent être interchangées, puisque en MAPA nous pouvons définir des processus capables de proposer une infinité de (*multi-*)actions initiales différentes (voir Chapitre III.11). Bien évidement, un autre choix de classe d'*ensembles d'échec* pourrait être envisagé, comme cela a été discuté dans la Section I.1.1.6, pour rendre les notions de "processus" et de "son environnement" symétriques. Nous proposons ici une caractérisation d'identité de processus par rapport au comportement externe, en utilisant la notion de "contexte".

Un *contexte* C est un terme avec des "trous". Plus formellement :

- un identificateur 'g' est un contexte,
- $\{\ \}$ est un contexte (appelé *trou*),
- si C, C' sont des contextes, m est une multi-action, et ch un canal, alors $(m; C)$, $(C : ch)$ et $(C\, ch\, C')$ sont des contextes.

Si C est un contexte et $P \in \mathcal{P}$ est un terme, alors $C\{P\}$ dénote le résultat du placement de P dans les trous de C, c'est-à-dire en remplaçant chaque trou $\{\ \}$ par P (on peut vérifier que $C\{P\}$ est un terme). Par exemple, si $C = m; (\{\ \}\ ch\ \{\ \})$ et P est un terme, alors $C\{P\}$ dénote $m; (P\ ch\ P)$.

Intuitivement, un processus P peut se retrouver bloqué s'il existe une évolution interne de P qui l'amène dans un "cul-de-sac", c'est-à-dire dans un état Q, à partir duquel aucune action visible n'est plus possible, c.-à-d.

$$\exists Q \left(P \overset{*}{\rightsquigarrow} Q \wedge \mathtt{Sort}(Q) = [\ ] \right) .$$

Définition III.6 Deux termes $\mathbf{P}$, $\mathbf{Q}$ de MAPA sont *équivalents*, noté $\mathbf{P} \sim \mathbf{Q}$, lorsque, pour chaque contexte C, le processus $C\{\mathbf{P}\}$ peut dériver silencieusement en un processus de sorte vide si et seulement si $C\{\mathbf{Q}\}$ peut le faire aussi :

$$\exists P' \left(C\{\mathbf{P}\} \overset{*}{\leadsto} P' \wedge \mathtt{Sort}(P') = [\,] \right) \;\Leftrightarrow\; \exists Q' \left(C\{\mathbf{Q}\} \overset{*}{\leadsto} Q' \wedge \mathtt{Sort}(Q') = [\,] \right) .$$

□

Cette définition peut être appliquée avec succès pour prouver que deux processus sont non équivalents, mais si l'on veut prouver que deux processus sont équivalents, alors de toute évidence la définition ne peut pas être utilisée directement. Ainsi, la proposition suivante peut être très utile, car elle montre que l'on hérite de toutes les méthodes de preuve valides pour les équivalences précédentes.

Proposition III.10

1. $\forall P, Q \in \mathcal{P} \quad P \approx Q \;\Rightarrow\; P \sim Q.$
2. $\exists P, Q \in \mathcal{P} \quad P \sim Q \;\wedge\; P \not\approx Q.$

Preuve.

On prouve que $P \not\sim Q \;\Rightarrow\; P \not\approx Q$.

Puisque $P \not\sim Q$, il existe un contexte C tel que

$$\exists P' \left(C\{P\} \overset{*}{\leadsto} P' \wedge \mathtt{Sort}(P') = [\,] \right) \wedge \neg \left(\exists Q' \left(C\{Q\} \overset{*}{\leadsto} Q' \wedge \mathtt{Sort}(Q') = [\,] \right) \right)$$

(ou l'inverse). Cela signifie que $C\{P\} \not\approx C\{Q\}$. Nous savons (voir Lemme III.13 et Théorème III.21 dans le chapitre suivant) que $\approx$ est une congruence dans MAPA. Ainsi, on peut conclure que $P \not\approx Q$.

Pour prouver que l'implication inverse n'est pas toujours vraie, nous donnons le contre-exemple suivant : soient P et Q les deux termes suivants :

$$P \;=\; \left((ak; a; b; \mathtt{stop} \; [\,] \; ak; a; c; \mathtt{stop}) \;\; [k] \;\; \overline{k}; \mathtt{stop} \right) : [a, b, c]$$

$$Q \;=\; \left(a; (ak; b; \mathtt{stop} \; [\,] \; ak; c; \mathtt{stop}) \;\; [k] \;\; \overline{k}; \mathtt{stop} \right) : [a, b, c]$$

Les graphes de processus associés aux deux termes sont présentés dans la Figure 9.1. Les deux systèmes ne sont pas bisimulables, c.-à-d. $P \not\approx Q$, pendant que $P \sim Q$. ■

On peut aussi montrer que l'équivalence sémantique est strictement plus forte que l'équivalence par échec telle que définie dans la Section I.1.1, c'est-à-dire que : $\sim \;\; \subset \;\; \approx_f$.

Proposition III.11

1. $\forall P, Q \in \mathcal{P} \quad P \sim Q \;\Rightarrow\; \mathtt{graph}(P) \approx_f \mathtt{graph}(Q).$
2. $\exists P, Q \in \mathcal{P} \quad \mathtt{graph}(P) \approx_f \mathtt{graph}(Q) \;\wedge\; P \not\sim Q.$

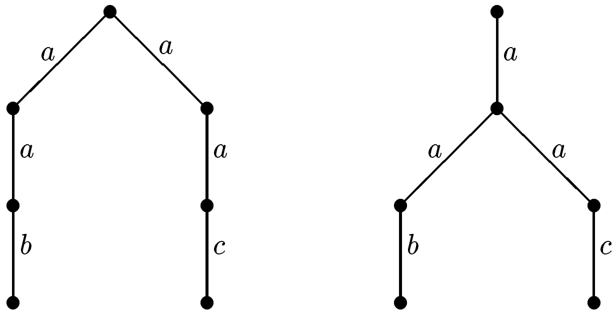


Figure 9.1: Graphes de processus $\mathtt{graph}(P)$ et $\mathtt{graph}(Q)$, où tout état dessiné par $\bullet$ (cercle plein) cache en plus une boucle via la multi-action vide.

Preuve. Soient P et Q deux termes de MAPA tels que $\mathtt{graph}(P) \not\approx_f \mathtt{graph}(Q)$. Cela veut dire qu'il existe une paire d'échec (ω, A) de $\mathtt{graph}(P)$ qui n'est pas une paire d'échec de $\mathtt{graph}(Q)$ (ou l'inverse) avec $A \neq \emptyset$. On prouve alors facilement qu'il existe un contexte C tel que

$$\exists P' \left(C\{P\} \overset{*}{\rightsquigarrow} P' \wedge \mathtt{Sort}(P') = [\,]\right) \wedge \neg \left(\exists Q' \left(C\{Q\} \overset{*}{\rightsquigarrow} Q' \wedge \mathtt{Sort}(Q') = [\,]\right)\right)$$

On construit un tel contexte de la manière suivante :

$$\begin{array}{lcl} C & = & (\{\ \}\ [m_1, m_2, \ldots, m_n]\ \mathtt{K}) : [k] \\ \mathtt{K} & \doteq & \sum \left\{k; \mathtt{stop},\ \overline{m}_1; \left(\cdots \sum \left\{k; \mathtt{stop},\ \overline{m}_n; \left(\sum\{\overline{m}; k; \mathtt{stop} \mid m \in A\}\right)\right\}\right)\right\} \end{array}$$

où $\omega = m_1 \cdots m_n$, l'opérateur $\sum$, qui va être introduit formellement dans le Chapitre III.11, intuitivement dénote le choix gardé, et k est une nouvelle action qui n'appartient à aucune multi-action de la trace ω et de l'ensemble A (ni à leur conjugués). Notons qu'un tel contexte existe toujours, car la trace ω et l'ensemble de refus A sont finis.

Pour prouver le point 2, c-à-d. qu'il existe des termes dont les graphes de processus sont équivalents par échec, $\approx_f$, mais qui ne sont pas équivalents sémantiquement, $\not\sim$, considérons les trois termes de MAPA G_1, G_2 et G_3 de la Figure 9.2. Les deux graphes $\mathtt{graph}(G_1)$ et $\mathtt{graph}(G_2)$ sont équivalents par $\approx_f$ puisqu'ils ont exactement le même ensemble E de paires d'échec, vu que les ensembles d'échec sont finis. En effet, l'ensemble E peut être décrit dans les deux cas par :

$$E = \{\langle \omega, X \rangle \mid \omega \in \mu_f(a) \cup \{\varepsilon\}, X \subset \mu_f(\mathtt{Act}) \setminus \{\emptyset\} \text{ et } X \text{ fini}\}$$

Par contre, le contexte $(G_3\ ch\ \{\ \}) : [b]$, où $ch = \mu_f(a)$, prouve que $G_1 \not\sim G_2$, car $(G_3\ ch\ G_1) : [b]$ peut bloquer si G_1 dérive dans la branche la plus à gauche (voir Fig. 9.2). Une telle situation n'est jamais possible pour $(G_3\ ch\ G_2) : [b]$ où un b pourra toujours être exécuté, indépendamment du choix d'une branche dans G_2.

Notons, que $\mathtt{graph}(G_1) \not\approx_f^\infty \mathtt{graph}(G_2)$, car la paire d'échec $\langle \varepsilon, \mu_f(\mathtt{Act}) \setminus \{\emptyset, a\}\rangle$ de $\mathtt{graph}(G_1)$ n'est pas une paire d'échec de $\mathtt{graph}(G_2)$.

Par le contre-exemple suivant nous pouvons voir que, même si les ensembles de refus sont infinis, nous avons toujours :

$$\exists P, Q \in \mathcal{P} \quad \mathtt{graph}(P) \approx_f^\infty \mathtt{graph}(Q) \ \wedge\ P \not\sim Q \ . \tag{9.1}$$

$$
\begin{aligned}
G_1 &= ((\overline{x};\mathtt{stop})\ [\,]\ \mathtt{P}_1) : \mu_f(a)\\
G_2 &= \Big(\Big(((k\overline{x};\mathtt{stop})\ [\,]\ (ka\overline{x};\mathtt{stop}))\ \ [k]\ \ (\overline{k};\mathtt{stop})\Big)\ \ [\overline{x}]\ \mathtt{P}_1\Big) : \mu_f(a)\\
G_3 &= ((\overline{ax};b;\mathtt{stop})\ [\overline{x}]\ \mathtt{R}_1) : \mu_f(\overline{a}) \cup [b]
\end{aligned}
$$

où :

$$
\begin{aligned}
\mathtt{P}_1 &\doteq \Big(((k;xa;\mathtt{stop})\ [\,]\ k;((xa\overline{y};\mathtt{stop})\ [\overline{y}]\ \mathtt{P}_2))\ \ [k]\ \ (\overline{k};\mathtt{stop})\Big) : \mu_f(a) \cup [x]\\
\mathtt{P}_2 &\doteq \Big(((k;ya;\mathtt{stop})\ [\,]\ k;((ya\overline{x};\mathtt{stop})\ [\overline{x}]\ \mathtt{P}_1))\ \ [k]\ \ (\overline{k};\mathtt{stop})\Big) : \mu_f(a) \cup [y]\\
\mathtt{R}_1 &\doteq \Big(((kx\overline{a};\mathtt{stop})\ [\,]\ \ ((kx\overline{ay};\mathtt{stop})\ [\overline{y}]\ \mathtt{R}_2))\ \ [k]\ \ (\overline{k};\mathtt{stop})\Big) : \mu_f(\overline{a}) + [x]\\
\mathtt{R}_2 &\doteq \Big(((ky\overline{a};\mathtt{stop})\ [\,]\ \ ((ky\overline{ax};\mathtt{stop})\ [\overline{x}]\ \mathtt{R}_2))\ \ [k]\ \ (\overline{k};\mathtt{stop})\Big) : \mu_f(\overline{a}) + [y]
\end{aligned}
$$

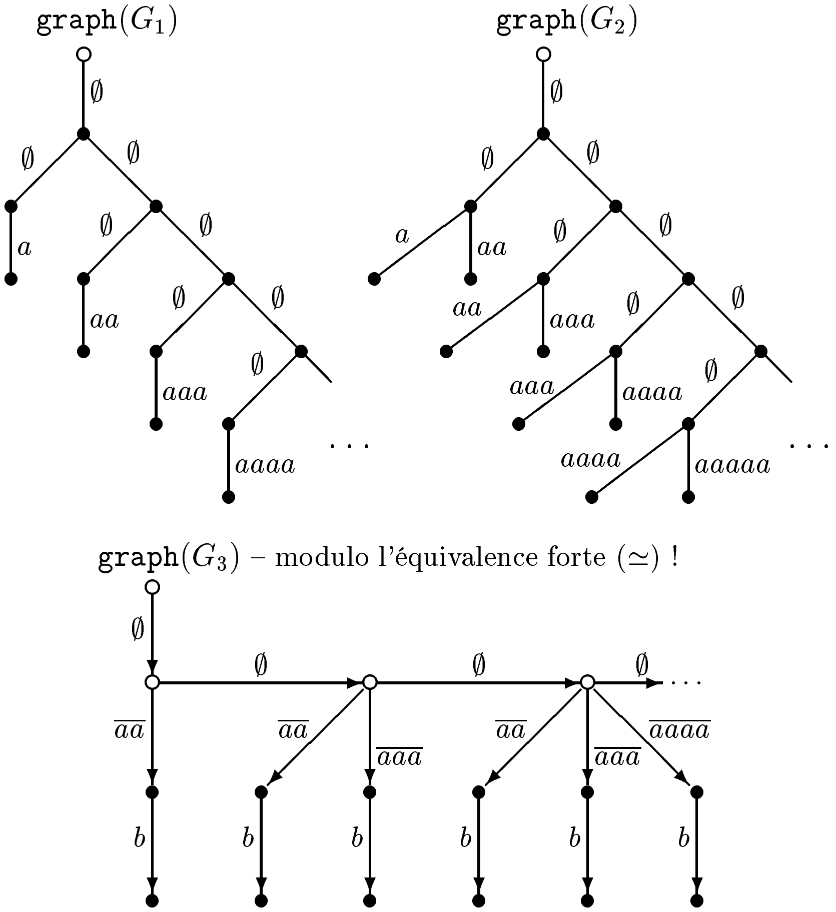


Figure 9.2: Termes G_1, G_2, G_3 avec leurs graphes de processus, où tout état dessiné par • (cercle plein) cache en plus une boucle via la multi-action vide.

Considérons les trois agents suivants :

$$\begin{array}{rcl} g & \doteq & ((ak; g\ [\ ]\ ak; b; \texttt{stop})\ [k]\ \overline{k}; \texttt{stop}) : [a, b] \\ h & \doteq & ((ak; g\ [\ ]\ (ak; b; \texttt{stop}\ [\ ]\ ak; p))\ [k]\ \overline{k}; \texttt{stop}) : [a, b] \\ p & \doteq & a; p \end{array}$$

On vérifie facilement que $g \not\sim h$, même si $\texttt{graph}(g) \approx_f^\infty \texttt{graph}(h)$. Les graphes de processus (modulo l'équivalence forte) associés à g et à h sont présentés dans la Figure 9.3. ■

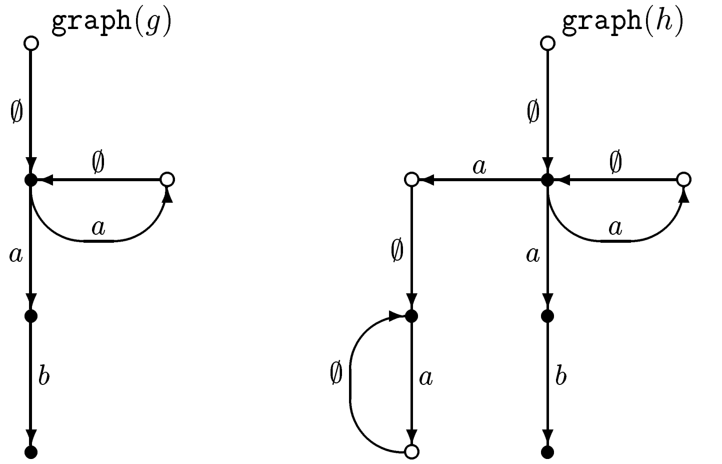


Figure 9.3: Graphes de processus $\texttt{graph}(g)$ et $\texttt{graph}(h)$ modulo l'équivalence forte, où tout état dessiné par $\bullet$ (cercle plein) cache en plus une boucle via la multi-action vide.

Chapter 10

Autres sémantiques opérationnelles

Dans la Section III.9.2, nous avons déjà mentionné que des formulations alternatives de la sémantique opérationnelle de MAPA peuvent être obtenues en choisissant un des quatre ensembles de règles $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, ou $\mathcal{R}_4$, au lieu de l'ensemble $\mathcal{R}_0$ $(= \mathcal{R})$. A présent, nous allons prouver que les cinq formulations sont équivalentes par rapport à l'équivalence faible, et nous allons justifier le choix de $\mathcal{R}_0$ comme formulation "la plus opérationnelle".

Afin de comparer de façon formelle ces cinq sémantiques opérationnelles, nous définissons le nouveau système de transition $\texttt{TS}^*$ comme l'union disjointe des cinq systèmes de transition, chacun engendré par l'ensemble de règles correspondant. Plus formellement, si $\texttt{sem}_i(\mathcal{P}) = (\mathcal{P}, \mu_f(\texttt{Act}), \rightarrow_i)$ est le système de transitions engendré par l'ensemble des règles correspondant à $\mathcal{R}_i$, pour $i \in \{0,1,2,3,4\}$, on prendra $\texttt{TS}^* \stackrel{\text{def}}{=} (\mathcal{P} \times \{0,1,2,3,4\}, \mu_f(\texttt{Act}), \rightarrow)$, tel que $\forall P, Q \in \mathcal{P}$, $\forall m \in \mu_f(\texttt{Act})$ et $\forall i, j \in \{0,1,2,3,4\}$:

$$\Big(\langle P, i\rangle \stackrel{m}{\rightarrow} \langle Q, j\rangle \;\stackrel{\text{def}}{\Longleftrightarrow}\; i = j \;\wedge\; P \stackrel{m}{\rightarrow}_i Q\Big)\ .$$

Théorème III.12 *$\langle P, i\rangle \approx \langle P, j\rangle$ dans $\texttt{TS}^*$, pour tout terme $P \in \mathcal{P}$ et pour tout $i, j \in \{0,1,2,3,4\}$.*

La preuve de ce théorème nécessite un certain nombre de résultats intermédiaires.

Lemme III.13 *L'équivalence faible $\approx_1$ (entre les états de $\texttt{sem}_1(\mathcal{P})$) est une congruence.*

Preuve du Lemme III.13

Nous devons vérifier que si $P \approx_1 Q$ alors

1. $m; P \approx_1 m; Q$
2. $P : ch \approx_1 Q : ch$
3. $P\ ch\ R \approx_1 Q\ ch\ R$

4. $R\ ch\ P \approx_1 R\ ch\ Q$

pour toute multi-action m, tout canal ch et tout terme R.

Pour cela, nous définissons les trois relations suivantes $B_1, B_2, B_3 \subseteq \mathcal{P} \times \mathcal{P}$:

$$B_1 \stackrel{\text{def}}{=} \ \approx_1 \ \cup \ \{(m;P, m;Q) \mid m \in \mu_f(\texttt{Act}), P \approx_1 Q\}.$$

$$B_2 \stackrel{\text{def}}{=} \{(P : ch, Q : ch) \mid ch \in \texttt{Chan(Act)}, P \approx_1 Q\}.$$

$$B_3 \stackrel{\text{def}}{=} \{(P\ ch\ R, Q\ ch\ Z) \mid ch \in \texttt{Chan(Act)}, P \approx_1 Q, R \approx_1 Z\}.$$

Nous vérifions que B_1, B_2, B_3 sont des (bi)simulations faibles par rapport à $\rightarrow_1$, c.-à-d. que pour tous $(P,Q) \in B_i$ et $m \in \mu_f(\texttt{Act})$, si $P \stackrel{m}{\longrightarrow}_1 P'$ alors il existe Q' tel que $Q \stackrel{m}{\Longrightarrow}_1 Q'$, et $(P',Q') \in B_i$, pour $i \in \{1,2,3\}$.

Nous donnons ici seulement la preuve pour B_3, les autres cas étant plus faciles.

Prenons $(P\ ch\ R)$ et $(Q\ ch\ Z)$, avec $P \approx_1 Q$ et $R \approx_1 Z$, c-à-d. $(P\ ch\ R, Q\ ch\ Z) \in B_3$. Or, $(P\ ch\ R) \stackrel{m}{\longrightarrow}_1 (P'\ ch\ R')$ si et seulement si il existe des multi-actions n, n_1, n_2 telles que : $n_1 + n_2 = m,\ \ n \in ch,\ \ P \xrightarrow{n_1+n}_1 P'$ et $R \xrightarrow{n_2+\overline{n}}_1 R'$.

Comme $P \approx_1 Q$ et $R \approx_1 Z$, il existe Q', Z' tels que $Q \stackrel{n_1+n}{\Longrightarrow}_1 Q'$ avec $P' \approx_1 Q'$ et $Z \stackrel{n_2+\overline{n}}{\Longrightarrow}_1 Z'$ avec $R' \approx_1 Z'$.

$Q \stackrel{n_1+n}{\Longrightarrow}_1 Q'$ signifie que $\exists i, i' \geq 0 \quad \exists Q_0, \ldots, Q_i,\ Q'_0, \ldots, Q'_{i'} \in \mathcal{P}$, tels que :

$$Q = Q_0 \stackrel{\emptyset}{\rightarrow}_1 Q_1 \stackrel{\emptyset}{\rightarrow}_1 \ldots \stackrel{\emptyset}{\rightarrow}_1 Q_i \xrightarrow{n_1+n}_1 Q'_0 \stackrel{\emptyset}{\rightarrow}_1 \ldots \stackrel{\emptyset}{\rightarrow}_1 Q'_{i'} = Q'$$

(Remarquons que même si $n_1 + n = \emptyset$ et $Q_i = Q'_0$, la transition $Q_i \xrightarrow{n_1+n}_1 Q'_0$ existe.)

De même pour $Z \stackrel{n_2+\overline{n}}{\Longrightarrow}_1 Z'$: $\quad \exists k, k' \geq 0 \quad \exists Z_0, \ldots, Z_k,\ Z'_0, \ldots Z'_{k'} \in \mathcal{P}$, tels que :

$$Z = Z_0 \stackrel{\emptyset}{\rightarrow}_1 Z_1 \stackrel{\emptyset}{\rightarrow}_1 \ldots \stackrel{\emptyset}{\rightarrow}_1 Z_k \xrightarrow{n_2+\overline{n}}_1 Z'_0 \stackrel{\emptyset}{\rightarrow}_1 \ldots \stackrel{\emptyset}{\rightarrow}_1 Z'_{k'} = Z'$$

Puisque dans la sémantique $\texttt{sem}_1(\mathcal{P})$ nous avons la règle `Inaction-bis`, nous pouvons supposer que $i = k$ et $i' = k'$. D'où immédiatement $(Q\ ch\ Z) \stackrel{m}{\Longrightarrow}_1 (Q'\ ch\ Z')$.

Comme $P' \approx_1 Q'$ et $R' \approx_1 Z'$, on a $((P'\ ch\ R'), (Q'\ ch\ Z')) \in B_3$.

B_3 est une bisimulation faible. ■

Lemme III.14 $g \doteq G \Rightarrow g \approx_1 G$.

Preuve du Lemme III.14 Ce résultat a déjà été établi pour $\approx_0$ (Prop. III.9(4)). Ici, la preuve est quasi identique. Nous prouvons que

$$B \stackrel{\text{def}}{=} \{(P,P) \mid P \in \mathcal{P}\} \cup \{(g,G),(G,g) \mid g \doteq G\}$$

est une bisimulation faible. Si $(P,P') \in B$ et $P \stackrel{m}{\rightarrow}_1 Q$ alors il existe Q' tel que $P' \stackrel{m}{\Longrightarrow}_1 Q'$ et $(Q,Q') \in B$. Le seul cas non trivial est quand $P = G$ et $P' = g$ avec $g \doteq G$. Si $G \stackrel{m}{\rightarrow}_1 Q$, alors $g \stackrel{\emptyset}{\rightarrow}_1 G \stackrel{m}{\rightarrow}_1 Q$, donc $g \stackrel{m}{\Longrightarrow}_1 Q$, c.-à-d. que $Q' = Q$. ■

Lemme III.15 *Si $P \stackrel{m}{\rightarrow}_0 Q$ alors $P \stackrel{m}{\rightarrow}_1 Q$.* *(c.-à-d. $\rightarrow_0 \ \subseteq \ \rightarrow_1$)*

Preuve. La règle `Inaction` est dérivable de la règle `Inaction-bis`. ■

Lemme III.16 *Si* $P \xrightarrow{m}_1 Q \xrightarrow{n}_0 R$ *alors* $\exists Q'$ *tel que* $P \xrightarrow{m}_0 Q' \xrightarrow{n}_1 R$ *et* $Q \approx_1 Q'$.

Preuve. Par induction sur la hauteur du terme P, on a :

1. Si $\texttt{high}(P) = 0$ alors P est un identificateur d'agent $g \doteq G$, et il existe deux cas possibles :

   (a) $g \xrightarrow{\emptyset}_1 g \xrightarrow{\emptyset}_0 G$.
   Dans ce cas $g \xrightarrow{\emptyset}_0 G \xrightarrow{\emptyset}_1 G$ fait l'affaire, car par le Lemme III.14, $g \approx_1 G$.

   (b) $g \xrightarrow{\emptyset}_1 G \xrightarrow{\emptyset}_0 X$.
   Dans ce cas on a $g \xrightarrow{\emptyset}_0 G \xrightarrow{\emptyset}_1 X$, puisque $\rightarrow_0 \;\subseteq\; \rightarrow_1$ (par Lemme III.15)

2. Supposons, que pour un $k \geq 0$, le lemme est vrai pour tous les termes X tels que $\texttt{high}(X) \leq k$.

   Soit P un terme tel que $\texttt{high}(P) = k + 1$.

   Trois cas sont possibles :

   (a) $P = m; X$. Deux cas sont à analyser :

      i. $m; X \xrightarrow{\emptyset}_1 m; X \xrightarrow{m}_0 X'$.
      On a directement $m; X \xrightarrow{\emptyset}_0 m; X \xrightarrow{m}_1 X'$.

      ii. $m; X \xrightarrow{m}_1 X \xrightarrow{n}_0 X''$.
      On a directement $m; X \xrightarrow{m}_0 X \xrightarrow{n}_1 X''$.

   (b) $P = X : ch$, où $\texttt{high}(X) = k$.
   $X : ch \xrightarrow{m}_1 X' : ch \xrightarrow{n}_0 X'' : ch$.
   C.-à-d. $X \xrightarrow{m}_1 X' \xrightarrow{n}_0 X''$ et $m, n \in ch$, même si la transition $X : ch \xrightarrow{m}_1 X' : ch$ est obtenue par la règle `Inaction-bis`, car dans ce cas $X = X'$ et $m = \emptyset \in ch$, et donc $X \xrightarrow{m}_1 X'$ existe.
   Par hypothèse, il existe X''' tel que $X \xrightarrow{m}_0 X''' \xrightarrow{n}_1 X''$, et $X''' \approx_1 X'$.
   D'où $X : ch \xrightarrow{m}_0 X''' : ch \xrightarrow{n}_1 X'' : ch$,
   avec $X''' : ch \approx_1 X' : ch$, car $\approx_1$ est une congruence (Lemme III.13).

   (c) $P = X \; ch \; Y$, avec $\texttt{high}(X), \texttt{high}(Y) \leq k$.
   Si $X \; ch \; Y \xrightarrow{m}_1 X' \; ch \; Y' \xrightarrow{n}_0 X'' \; ch \; Y''$, alors
   $X \xrightarrow{m_1+\alpha}_1 X' \xrightarrow{n_1+\beta}_0 X''$ et $Y \xrightarrow{m_2+\overline{\alpha}}_1 Y' \xrightarrow{n_2+\overline{\beta}}_0 Y''$,
   avec $m_1 + m_2 = m$, $n_1 + n_2 = n$, $\alpha, \beta \in ch$ (même si la transition $X \; ch \; Y \xrightarrow{m}_1 X' \; ch \; Y'$ est obtenue par la règle `Inaction-bis`).
   Par hypothèse, il existe X''' et Y''' tels que :
   $X \xrightarrow{m_1+\alpha}_0 X''' \xrightarrow{n_1+\beta}_1 X''$ et $Y \xrightarrow{m_2+\overline{\alpha}}_0 Y''' \xrightarrow{n_2+\overline{\beta}}_1 Y''$,
   avec $X''' \approx_1 X'$, $Y''' \approx_1 Y'$,
   et donc $X \; ch \; Y \xrightarrow{m}_0 X''' \; ch \; Y''' \xrightarrow{n}_1 X'' \; ch \; Y''$.
   $X''' \; ch \; Y''' \approx_1 X' \; ch \; Y'$, car $\approx_1$ est une congruence. ■

Lemme III.17

1. *Si* $P \xrightarrow{m}_1 Q$ *alors* $P \xRightarrow{m}_2 Q$. (*c.-à-d.* $\rightarrow_1 \subseteq \Longrightarrow_2$)
2. *Si* $P \xrightarrow{m}_2 Q$ *alors* $P \xRightarrow{m}_1 Q$. (*c.-à-d.* $\rightarrow_2 \subseteq \Longrightarrow_1$)

Preuve. Par induction sur la hauteur du terme P :

$\underline{\rightarrow_1 \subseteq \Longrightarrow_2}$

1. Si $\texttt{high}(P) = 0$, c.-à-d. P est un identificateur d'agent. Alors, on a directement :
   $P \xrightarrow{\emptyset}_1 Q \Rightarrow P \xrightarrow{\emptyset}_2 Q \vee P = Q$, c.-à-d. que $P \xrightarrow{\emptyset}_1 Q \;\Rightarrow\; P \xRightarrow{\emptyset}_2 Q$.
2. Supposons que pour un $k \geq 0$ le lemme est vrai pour tous les termes X tels que $\texttt{high}(X) \leq k$.
   Soit P un terme tel que $\texttt{high}(P) = k+1$. Trois cas sont possibles :
   (a) $P = m; X$
   On peut avoir deux évolutions de P dans $\texttt{sem}_1$:
      i. $m; X \xrightarrow{\emptyset}_1 m; X$. Comme $\overset{*}{\rightsquigarrow}_2$ est réflexive, on a $m; X \xRightarrow{\emptyset}_2 m; X$.
      ii. $m; X \xrightarrow{m}_1 X$. On a la même transition dans $\texttt{sem}_2$, donc $m; X \xRightarrow{m}_2 X$.
   (b) $P = X : ch$, où $\texttt{high}(X) = k$.
   Si $X : ch \xrightarrow{m}_1 X' : ch$ alors $X \xrightarrow{m}_1 X'$ et $m \in ch$, même si la transition $X : ch \xrightarrow{m}_1 X' : ch$ est obtenue par la règle `Inaction-bis`.
   Par hypothèse, $X \xRightarrow{m}_2 X'$.
   Donc, $X : ch \xRightarrow{m}_2 X' : ch$ car m et $\emptyset$ sont dans ch.
   (c) $P = X \; ch \; Y$, avec $\texttt{high}(X), \texttt{high}(Y) \leq k$.
   Si $X \; ch \; Y \xrightarrow{m}_1 X' \; ch \; Y'$ alors $X \xrightarrow{n_1+n}_1 X'$, $Y \xrightarrow{n_2+\overline{n}}_1 Y'$, $m = n_1 + n_2$ et $n \in ch$, même si `Inaction-bis` est utilisée.
   Par hypothèse, $X \xRightarrow{n_1+n}_2 X'$ et $Y \xRightarrow{n_2+\overline{n}}_2 Y'$.
   Comme dans la sémantique $\texttt{sem}_2(\mathcal{P})$ nous avons les règles `CompLeft` et `CompRight`, $X \; ch \; Y \xRightarrow{m}_2 X' \; ch \; Y'$.

$\underline{\rightarrow_2 \subseteq \Longrightarrow_1}$

1. Si $\texttt{high}(P) = 0$, c.-à-d. $P = g \doteq G$. Dans $\texttt{sem}_2$ il n'y a qu'une transition sortante de P : $g \xrightarrow{\emptyset}_2 G$. La même transition existe dans $\texttt{sem}_1$, donc $g \xRightarrow{\emptyset}_1 G$.
2. Supposons que pour un $k \geq 0$ le lemme est vrai pour tous les termes X tels que $\texttt{high}(X) \leq k$.
   Soit P un terme tel que $\texttt{high}(P) = k+1$. Trois cas sont possibles :
   (a) $P = m; X$ Dans $\texttt{sem}_2$ le terme $m; X$ a une seule évolution possible qui existe aussi dans $\texttt{sem}_1$.

(b) $P = X : ch$, où $\texttt{high}(X) = k$.
Si $X : ch \xrightarrow{m}_2 X' : ch$ alors $X \xrightarrow{m}_2 X'$ et $m \in ch$.
Par hypothèse, $X \xRightarrow{m}_1 X'$.
Donc, $X : ch \xRightarrow{m}_1 X' : ch$, car m et $\emptyset$ sont dans ch.

(c) $P = X\ ch\ Y$, avec $\texttt{high}(X), \texttt{high}(Y) \leq k$.
Si $X\ ch\ Y \xrightarrow{m}_2 X'\ ch\ Y$ alors un des trois cas suivants est possible :

i. $X \xrightarrow{m}_2 X'$ et $Y = Y'$. Par hypothèse, il existe un terme R tel que $X \xRightarrow{m}_1 X'$. De l'autre côté la règle $Y \xRightarrow{\emptyset}_1 Y$ est toujours possible (par la Règle `Inaction-bis`), donc $X\ ch\ Y \xRightarrow{m}_1 X'\ ch\ Y'$ (vue la preuve du Lemme III.13).

ii. $Y \xrightarrow{m}_2 Y'$ et $X = X'$. Comme pour le point précédent.

iii. $X \xrightarrow{n_1+n}_2 X'$, $Y \xrightarrow{n_2+\overline{n}}_2 Y'$, $m = n_1 + n_2$ et $n \in ch$.
Par hypothèse, $X \xRightarrow{n_1+n}_1 X'$, $Y \xRightarrow{n_2+\overline{n}}_1 Y'$, donc $X\ ch\ Y \xRightarrow{m}_1 X\ ch\ Y$ (vue la preuve du Lemme III.13). ■

Lemme III.18

1. Si $P \xrightarrow{m}_1 Q$ alors $P \xRightarrow{m}_3 Q$. *(c.-à-d. $\rightarrow_1\ \subseteq\ \Longrightarrow_3$)*

2. Si $P \xrightarrow{m}_3 Q$ alors $P \xRightarrow{m}_1 Q$. *(c.-à-d. $\rightarrow_3\ \subseteq\ \Longrightarrow_1$)*

Preuve.

$\rightarrow_1\ \subseteq\ \Longrightarrow_3$

On peut même prouver mieux : $\rightarrow_1\ \subseteq\ \rightarrow_3$, car l'ensemble de règles $\mathcal{R}_3$ implique la règle `Call`. En effet, comme `Inaction-bis` ($P \xrightarrow{\emptyset}_3 P$) est dans $\mathcal{R}_3$, par `Call-bis` on a $g \xrightarrow{\emptyset}_3 P$ si $g \doteq P$.

Ainsi toute preuve de $P \xrightarrow{m}_1 Q$ donne aussi une preuve de $P \xrightarrow{m}_3 Q$.

$\rightarrow_3\ \subseteq\ \Longrightarrow_1$

On montre que toutes les règles qui définissent $\rightarrow_3$, c.-à-d. `Inaction-bis`, `Prefix`, `Composition`, `Restriction` et `Call-bis`, sont aussi valables pour $\Longrightarrow_1$. Les règles : `Inaction-bis`, `Prefix` et `Restriction` sont trivialement valables pour $\Longrightarrow_1$. La règle `Composition` est aussi valable pour $\Longrightarrow_1$, ce qui a été montré dans la preuve du Lemme III.13.

Il nous reste donc à prouver que la règle `Call-bis`, c.-à-d. :

$$\frac{P \xRightarrow{m}_1 Q}{g \xRightarrow{m}_1 Q}\ g \doteq P$$

est aussi valide pour $\Longrightarrow_1$. Trivial, car si $P \xRightarrow{m}_1 Q$ et $g \doteq P$, alors on a directement $g \xrightarrow{\emptyset}_1 P \xRightarrow{m}_1 Q$, et donc $g \xRightarrow{m}_1 Q$.

En conséquence une preuve de $P \xrightarrow{m}_3 Q$ donne aussi une preuve de $P \xRightarrow{m}_1 Q$.

■

Lemme III.19 *Soit $R \subseteq \mathcal{P} \times \mathcal{P}$ la relation suivante sur les termes de MAPA :*

$$R \stackrel{def}{=} \bigcup_{k \geq 0} R^k$$

avec :

$$\begin{array}{rcl} R^0 & \stackrel{def}{=} & \{(P,P) \mid P \in \mathcal{P}\} \quad \textit{(identité)} \\ R^{k+1} & \stackrel{def}{=} & \{(g,Y) \mid g \doteq X, (X,Y) \in R^k\} \cup \{(X : ch, Y : ch) \mid ch \in \mathtt{Chan}, (X,Y) \in R^k\} \\ & & \cup \{(X \ ch \ X', Y \ ch \ Y') \mid ch \in \mathtt{Chan}, (X,Y) \in R^k, (X',Y') \in R^k\} \ \cup \ R^k \end{array}$$

1. *Si $(P,Q) \in R$ alors $P \approx_1 Q$* *(c.-à-d. $R \subseteq \approx_1$)*
2. *Si $(P,Q) \in R$ et $Q \stackrel{m}{\rightarrow}_1 Q'$ alors :*
   (a) *soit $(P,Q') \in R$ et $m = \emptyset$,*
   (b) *soit $\exists P'$ tel que $(P',Q') \in R$ et $P \stackrel{m}{\rightarrow}_4 P'$.*

Preuve.

1. $R \subseteq \approx_1$.
   Immédiat : Grâce au Lemme III.14, au fait que $\approx_1$ est réflexive et transitive et que c'est une congruence, on a par induction sur k que $\forall k \ R^k \subseteq \approx_1$.
2. D'abord nous prouvons, par récurrence sur la hauteur du terme Q, que si $Q \stackrel{m}{\rightarrow}_1 Q'$ alors : *Propriété (*)*
   - soit $(Q,Q') \in R$ et $m = \emptyset$,
   - soit $\exists P'$ tel que $(P',Q') \in R$ et $Q \stackrel{m}{\rightarrow}_4 P'$.

   (a) Si $\mathtt{high}(Q) = 0$, c.-à-d. Q est un identificateur d'agent $g \doteq G$. On a directement : $g \stackrel{\emptyset}{\rightarrow}_1 g$ ou $g \stackrel{\emptyset}{\rightarrow}_1 G$ avec $(g,g) \in R^0$ et $(g,G) \in R^1$.
   Remarquons que, indépendamment du terme Q, si la règle `Inaction-bis` est utilisée dans l'évolution de Q, la propriété (*) est trivialement vérifiée.

   (b) Supposons que pour un $h \geq 0$ la propriété (*) est vraie pour tous les termes X tels que $\mathtt{high}(X) \leq h$.
   Soit Q un terme tel que $\mathtt{high}(Q) = k+1$. Trois cas sont possibles :

      i. $P = m; X$
      On peut avoir deux évolutions de P dans $\mathtt{sem}_1$:
         A. $m; X \stackrel{\emptyset}{\rightarrow}_1 m; X$. Cas particulier du cas général de l'aplication de la règle `Inaction-bis` : déjà traité.
         B. $m; X \stackrel{m}{\rightarrow}_1 X$. On a la même transition dans $\mathtt{sem}_4$, c.-à-d. $m; X \stackrel{m}{\rightarrow}_4 X$.

      ii. $Q = X : ch$, où $\mathtt{high}(X) = k$.
      Si $X : ch \stackrel{m}{\rightarrow}_1 X' : ch$ alors $X \stackrel{m}{\rightarrow}_1 X'$ et $m \in ch$.
      Par hypothèse :

- soit $(X, X') \in R$ et $m = \emptyset$; par définition de R, $(X : ch, X' : ch) \in R$.
- soit $\exists Y$ tel que $(Y, X') \in R$ et $X \xrightarrow{m}_4 Y$; par la règle `Restriction` nous avons directement $X : ch \xrightarrow{m}_4 Y : ch$. Par définition de R, si $(Y, X') \in R$ alors $(Y : ch, X' : ch) \in R$.

iii. $Q = X \ ch \ X'$, où $\mathtt{high}(X), \mathtt{high}(X') \leq k$.
Si $X \ ch \ X' \xrightarrow{m}_1 X_1 \ ch \ X_1'$ alors $X \xrightarrow{n_1+n}_1 X_1$, $X' \xrightarrow{n_2+\overline{n}}_1 X_1'$, $n_1+n_2 = m$ et $n \in ch$. Par hypothèse, les quatre cas suivants sont possibles :
- Soit $(X, X_1) \in R$, $n_1 + n = \emptyset$, $(X', X_1') \in R$ et $n_2 + \overline{n} = \emptyset$; Par définition de R, $(X \ ch \ X', X_1 \ ch \ X_1') \in R$ et $m = \emptyset$.
- Soit $(X, X_1) \in R$ et $\exists Y_1'$ tel que $(Y_1', X_1') \in R$ et $X' \xrightarrow{m}_4 Y_1'$; Par la règle `CompRight` nous avons $X \ ch \ X' \xrightarrow{m}_4 X \ ch \ Y_1'$, et par la définition de R, $(X \ ch \ Y_1', X_1 \ ch \ X_1') \in R$.
- Soit $\exists Y_1$ tel que $(Y_1, X_1) \in R$ et $X \xrightarrow{m}_4 Y_1$ et $(X', X_1') \in R$; Cas symétrique, avec `CompLeft`.
- Soit $\exists Y_1$ tel que $(Y_1, X_1) \in R$, $X \xrightarrow{n_1+n}_4 Y_1$ et $\exists Y_1'$ tel que $(Y_1', X_1') \in R$, $X' \xrightarrow{n_2+\overline{n}}_4 Y_1'$;
Par la règle `Composition`, $X \ ch \ X' \xrightarrow{m}_4 Y_1 \ ch \ Y_1'$, et par définition de R, $(Y_1 \ ch \ Y_1', X_1 \ ch \ X_1') \in R$.

La propriété (*) montre que le Lemme III.19 est vrai pour $(P, Q) \in R^0$.

3. Supposons maintenant que, pour certain $k \geq 0$, si $(P, Q) \in R^i$ pour un $i \leq k$ et $Q \xrightarrow{m}_1 Q'$ alors :

   - soit $(P, Q') \in R$ et $m = \emptyset$,
   - soit $\exists P'$ tel que $(P', Q') \in R$ et $P \xrightarrow{m}_4 P'$.

   Soit $(P, Q) \in R^{k+1}$. Remarquons tout de suite que si Q évolue via la règle `Inaction-bis`, c.-à-d. $Q \xrightarrow{\emptyset}_1 Q$, alors le lemme est trivialement vérifié. Sinon, trois cas sont possibles :

   (a) $(P, Q) \in \{(g, Y) \mid g \doteq X, (X, Y) \in R^k\}$, c.-à-d. $P = g$, $Q = Y$, et $g \doteq X$.
   Si $Y \xrightarrow{m}_1 Y'$, alors, par hypothèse :
   - soit $m = \emptyset$ et $(X, Y') \in R$; par définition de R, on a $(g, Y') \in R$.
   - soit $\exists X'$ tel que $(X', Y') \in R$ et $X \xrightarrow{m}_4 X'$; par la règle `Call-bis` nous avons directement $g \xrightarrow{m}_4 X'$.

   (b) $(P, Q) \in \{(X : ch, Y : ch) \mid ch \in \mathtt{Chan}, (X, Y) \in R^k\}$.
   Si $Y : ch \xrightarrow{m}_1 Y' : ch$ alors $Y \xrightarrow{m}_1 Y'$ et $m \in ch$, et dans ce cas, par hypothèse :
   - soit $(X, Y') \in R$ et $m = \emptyset$; dans ce cas, par définition de R, $(X : ch, Y' : ch) \in R$.
   - soit $\exists X'$ tel que $(X', Y') \in R$ et $X \xrightarrow{m}_4 X'$; par la règle `Restriction` nous avons directement $X : ch \xrightarrow{m}_4 X' : ch$. Par définition de R, si $(X', Y') \in R$ alors $(X' : ch, Y' : ch) \in R$.

(c) $(P,Q) \in \{(X\ ch\ X', Y\ ch\ Y') \mid ch \in \texttt{Chan}, (X,Y) \in R^k, (X',Y') \in R^k\}$.

Si $Y\ ch\ Y' \xrightarrow{m}_1 Y_1\ ch\ Y_1'$ alors $Y \xrightarrow{n_1+n}_1 Y_1$, $Y' \xrightarrow{n_2+\overline{n}}_1 Y_1'$, $n_1 + n_2 = m$ et $n \in ch$.

Par hypothèse, les quatre cas suivants sont possibles :

- Soit $(X, Y_1) \in R$, $n_1 + n = \emptyset$, $(X', Y_1') \in R$ et $n_2 + \overline{n} = \emptyset$; Dans ce cas, par définition de R, $(X\ ch\ X', Y_1\ ch\ Y_1') \in R$ et $m = \emptyset$.
- Soit $(X, Y_1) \in R$ et $\exists X_1'$ tel que $(X_1', Y_1') \in R$ et $X' \xrightarrow{m}_4 X_1'$; Par la règle `CompRight` nous avons $X\ ch\ X' \xrightarrow{m}_4 X\ ch\ X_1'$, et par la définition de R, $(X\ ch\ X_1', Y_1\ ch\ Y_1') \in R$.
- Soit $\exists X_1$ tel que $(X_1, Y_1) \in R$ et $X \xrightarrow{m}_4 X_1$ et $(X', Y_1') \in R$;
Cas symétrique, avec `CompLeft`.
- Soit $\exists X_1$ tel que $(X_1, Y_1) \in R$ et $X \xrightarrow{m}_4 X_1$ et
$\exists X_1'$ tel que $(X_1', Y_1') \in R$ et $X' \xrightarrow{m}_4 X_1'$;
Par la règle `Composition`, $X\ ch\ X' \xrightarrow{m}_4 X_1\ ch\ X_1'$, et par définition de R, $(X_1\ ch\ X_1', Y_1\ ch\ Y_1') \in R$.

■

Lemme III.20 *Si $P \xrightarrow{m}_4 Q$ alors $P \stackrel{m}{\Longrightarrow}_1 Q$.* *(c.-à-d. $\rightarrow_4 \subseteq \Longrightarrow_1$)*

Preuve. Similaire à la preuve du Lemme III.18(2). ■

Preuve du Théorème III.12

Nous allons prouver que $\langle P, i\rangle \approx \langle P, j\rangle$, pour tout terme $P \in \mathcal{P}$ et tout $i, j \in \{0, 1, 2, 3, 4\}$ dans $\texttt{TS}^*$. Comme $\approx$ est une relation d'équivalence, il suffit alors de prouver : $\langle P, 0\rangle \approx \langle P, 1\rangle$, $\langle P, 1\rangle \approx \langle P, 2\rangle$, $\langle P, 1\rangle \approx \langle P, 3\rangle$ et $\langle P, 1\rangle \approx \langle P, 4\rangle$. Nous avons choisi ces associations (nous comparons successivement la sémantique $\texttt{sem}_1$ avec les autres sémantiques), car nous savons, par le Lemme III.13, que l'équivalence faible dans $\texttt{sem}_1(\mathcal{P})$ est une congruence, ce qui facilite considérablement les preuves.

<u>$\langle P, 0\rangle \approx \langle P, 1\rangle$</u> :

Il suffit de prouver que $B \stackrel{\text{def}}{=} \{(\langle P, 0\rangle, \langle Q, 1\rangle), (\langle Q, 1\rangle, \langle P, 0\rangle) \mid P \approx_1 Q\}$ est une bisimulation faible dans $\texttt{TS}^*$.

Soit $(\langle P, 0\rangle, \langle Q, 1\rangle) \in B$:

1. Si $\langle P, 0\rangle \xrightarrow{m} \langle P', 0\rangle$ alors, par Lemme III.15, $\langle P, 1\rangle \xrightarrow{m} \langle P', 1\rangle$.
Comme $P \approx_1 Q$, il existe un terme Q' tel que $P' \approx_1 Q'$ et
$\langle Q, 1\rangle \stackrel{m}{\Longrightarrow} \langle Q', 1\rangle$.
2. Comme $P \approx_1 Q$, si $\langle Q, 1\rangle \xrightarrow{m} \langle Q', 1\rangle$, alors il existe P' tel que $P' \approx_1 Q'$ et $P \stackrel{m}{\Longrightarrow}_1 P'$. Par itération du Lemme III.16, il existe P'' tel que $P'' \approx_1 P'$ et $\langle P, 0\rangle \stackrel{m}{\Longrightarrow} \langle P'', 0\rangle$ (voir Figure 10.1, où P_0^0 correspond à P, P_i^0 correspond à P' et P_0^i à P''). Enfin, on a $P'' \approx_1 Q'$, car $\approx_1$ est une relation d'équivalence.

$$
\begin{tikzcd}
P_0^0 \arrow[r, "m_1"] \arrow[d, "m_1"'] & P_1^0 \arrow[r, "m_2"] \arrow[d, "m_2"'] & P_2^0 & \cdots\cdots & P_{i-2}^0 \arrow[r, "m_{i-1}"] \arrow[d, "m_{i-1}"'] & P_{i-1}^0 \arrow[r, "m_i"] \arrow[d, "m_i"'] & P_i^0 \arrow[d, "\emptyset"] \\
P_0^1 \arrow[r, "m_2"] \arrow[d, "m_2"'] & P_1^1 & & & P_{i-2}^1 \arrow[r, "m_i"] \arrow[d, "m_i"'] & P_{i-1}^1 \arrow[r, "\emptyset"] \arrow[d, "\emptyset"] & Q^1 \\
P_0^2 & & & & P_{i-2}^2 \arrow[r, "\emptyset"] & Q^2 & \\
\vdots & & & & & & \\
P_0^{i-2} \arrow[r, "m_{i-1}"] \arrow[d, "m_{i-1}"'] & P_1^{i-2} \arrow[r, "m_i"] \arrow[d, "m_i"'] & P_2^{i-2} \arrow[d, "\emptyset"] & & & & \\
P_0^{i-1} \arrow[r, "m_i"] \arrow[d, "m_i"'] & P_1^{i-1} \arrow[r, "\emptyset"] \arrow[d, "\emptyset"] & Q^{i-1} & & & & \\
P_0^i \arrow[r, "\emptyset"] & Q^i & & & & &
\end{tikzcd}
$$

L'itération commence par le terme P_i^0. Par le Lemme III.2, il existe un terme Q^1 tel que $P_i^0 \xrightarrow{\emptyset}_0 Q^1$. En appliquant le Lemme III.16 aux transitions $P_{i-1}^0 \xrightarrow{m_i}_1 P_i^0 \xrightarrow{\emptyset}_0 Q^1$, on obtient qu'il existe un terme $P_{i-1}^1 \approx_1 P_i^0$ tel que $P_{i-1}^0 \xrightarrow{m_i}_0 P_{i-1}^1 \xrightarrow{\emptyset}_1 Q^1$; on continue ensuite ainsi de droite à gauche et de haut en bas.

Figure 10.1: Etant donnée une dérivation $P_0^0 \xrightarrow{m_1}_1 P_1^0 \cdots \xrightarrow{m_i}_1 P_i^0$, on peut trouver une dérivation $P_0^0 \xrightarrow{m_1}_0 P_0^1 \cdots \xrightarrow{m_i}_0 P_0^i$, telle que $P_0^i \approx_1 P_i^0$, par itération du Lemme III.16. Les flèches horizontales dénotent des transitions dans $\mathtt{sem}_1$ et les flèches verticales dénotent des transitions dans $\mathtt{sem}_0$. Les termes P_x^y et P_k^l sont équivalents par $(\approx_1)$ si $x + y = k + l$ et les termes Q^k existent toujours (par Lemme III.2).

$\underline{\langle P,1\rangle \approx \langle P,2\rangle}$: On prouve que $B \stackrel{\text{def}}{=} \{(\langle P,1\rangle,\langle Q,2\rangle),(\langle Q,2\rangle,\langle P,1\rangle) \mid P \approx_1 Q\}$ est une bisimulation faible dans $\mathtt{TS}^*$.

Soit $(\langle P,1\rangle,\langle Q,2\rangle) \in B$.

1. Si $\langle P,1\rangle \xrightarrow{m} \langle P',1\rangle$, alors il existe Q', tel que $Q \stackrel{m}{\Longrightarrow}_1 Q'$ et $P' \approx_1 Q'$.
   Par itération du Lemme III.17(1), $Q \stackrel{m}{\Longrightarrow}_2 Q'$, c.-à-d. que $\langle Q,2\rangle \stackrel{m}{\Longrightarrow} \langle Q',2\rangle$ et $(\langle P',1\rangle,\langle Q',2\rangle) \in B$.
2. Si $\langle Q,2\rangle \xrightarrow{m} \langle Q',2\rangle$, alors, par le Lemme III.17(2), $Q \stackrel{m}{\Longrightarrow}_1 Q'$.
   Comme $P \approx_1 Q$, il existe P' tel que $P \stackrel{m}{\Longrightarrow}_1 P'$ et $P' \approx_1 Q'$.
   C.-à-d. que $\langle P,1\rangle \stackrel{m}{\Longrightarrow} \langle P',1\rangle$ et $(\langle P',1\rangle,\langle Q',2\rangle) \in B$.

$\underline{\langle P,1\rangle \approx \langle P,3\rangle}$: Preuve similaire, en utilisant le Lemme III.18.

$\underline{\langle P,1\rangle \approx \langle P,4\rangle}$: On prouve que $B \stackrel{\text{def}}{=} \{(\langle P,1\rangle,\langle Q,4\rangle),(\langle Q,4\rangle,\langle P,1\rangle) \mid P \approx_1 Q\}$ est une bisimulation faible dans $\mathtt{TS}^*$.

Soit $(\langle P,1\rangle,\langle Q,4\rangle) \in B$.

1. Si $\langle P,1\rangle \xrightarrow{m} \langle P',1\rangle$, alors il existe Q', tel que $Q \stackrel{m}{\Longrightarrow}_1 Q'$ et $P' \approx_1 Q'$.
   Par itération du Lemme III.19, il existe un terme Q'' tel que $Q \stackrel{m}{\Longrightarrow}_4 Q''$ et $Q' \approx_1 Q''$. Plus précisément, si $Q \xrightarrow{m_1}_1 Q'_1$ alors par le lemme nous savons qu'il existe un Q''_1 tel que $Q \stackrel{m_1}{\Longrightarrow}_4 Q''_1$ et (Q''_1, Q'_1) sont dans la relation R définie dans le lemme. Si maintenant $Q'_1 \xrightarrow{m_2}_1 Q'_2$, par le même lemme nous savons qu'il existe toujours un Q''_2 tel que $Q''_1 \stackrel{m_2}{\Longrightarrow}_4 Q''_2$ et $(Q''_2, Q'_2) \in R$, et ainsi de suite. C.-à-d. que, étant donnée la dérivation $Q \stackrel{m}{\Longrightarrow}_1 Q'$ ($Q \xrightarrow{m_1}_1 Q'_1 \xrightarrow{m_2}_1 Q'_2 \cdots \xrightarrow{m_n}_1 Q'$), nous pouvons toujours construire une dérivation $Q \stackrel{m}{\Longrightarrow}_4 Q''$ avec $(Q'', Q') \in R$. Comme $R \subseteq\ \approx_1$, on a $Q'' \approx_1 Q'$.
   C.-à-d. que $\langle Q,4\rangle \stackrel{m}{\Longrightarrow} \langle Q'',4\rangle$ et $(\langle P',1\rangle,\langle Q'',4\rangle) \in B$.
2. Si $\langle Q,4\rangle \xrightarrow{m} \langle Q',4\rangle$, alors, par le Lemme III.20, $Q \stackrel{m}{\Longrightarrow}_1 Q'$.
   Comme $P \approx_1 Q$, il existe P' tel que $P \stackrel{m}{\Longrightarrow}_1 P'$ et $P' \approx_1 Q'$.

■

Le Théorème III.12 nous permet de prouver que les cinq sémantiques engendrées par les ensembles de règles $\mathcal{R}_0$, $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$ et $\mathcal{R}_4$, sont toutes équivalentes par rapport à l'équivalence faible ou, formulé autrement, les équivalences faibles[1] $\approx_0, \approx_1, \ldots, \approx_4$, sont en fait la même relation sur les termes de MAPA.

Théorème III.21 *Pour tous $P, Q \in \mathcal{P}$ et $i, j \in \{0,1,2,3,4\}$ nous avons :*

$$P \approx_i Q \Leftrightarrow P \approx_j Q\ .$$

[1] Par $\approx_i$, pour $i \in \{0,1,2,3,4\}$ nous notons l'équivalence faible dans la sémantique $\mathtt{sem}_i(\mathcal{P})$.

Preuve.

Par la construction du système de transition $\mathtt{TS}^*$, $P \approx_i Q$ ssi $\langle P, i\rangle \approx \langle Q, i\rangle$.
Par le Théorème III.12 nous avons $\langle P, i\rangle \approx \langle P, j\rangle$ et $\langle Q, i\rangle \approx \langle Q, j\rangle$.
Comme l'équivalence faible est une relation d'équivalence, $\approx$ est transitive.
Nous concluons alors, que si $P \approx_i Q$ alors $\langle P, j\rangle \approx \langle Q, j\rangle$.
Donc, $P \approx_j Q$. ■

Le théorème nous permet de conclure que l'équivalence faible dans MAPA est une congruence, indépendamment du choix d'ensemble de règles pour la sémantique opérationnelle, car $\approx_1$ l'est (cf. Lemme III.13).

Il reste à justifier pourquoi nous avons choisi $\mathcal{R}_0$ plutôt qu'un autre ensemble de règles de dérivation. Nous pouvons d'abord noter que si la règle `Call-bis` participe aux règles de dérivation, alors la Proposition III.7 n'est pas valide. Prenons, par exemple, la définition d'agent suivante : "$g \doteq g$ [ ] c; g". On peut voir que $\mathtt{graph}(\langle g, i\rangle)$ est "de branchement infini" pour tout $\mathcal{R}_i$ avec la règle `Call-bis`, c'est-à-dire pour $\mathcal{R}_3$ et $\mathcal{R}_4$. Effectivement, dans les deux sémantiques opérationnelles on a la transition suivante :

$$(g\ [\ ]\ \mathrm{c}; g) \xrightarrow{\mathrm{c}} (g\ [\ ]\ g)\ ,$$

d'où, par la règle `Call-bis`, on a aussi

$$\frac{(g\ [\ ]\ \mathrm{c}; g) \xrightarrow{\mathrm{c}} (g\ [\ ]\ g)}{g \xrightarrow{\mathrm{c}} (g\ [\ ]\ g)}\ g \doteq g\ [\ ]\ \mathrm{c}; g\ .$$

En ayant cette transition et en appliquant successivement les règles `Prefix`, `Composition` et `Call-bis`, on peut prouver aussi :

$$\dfrac{\dfrac{g \xrightarrow{\mathrm{c}} g\ [\ ]\ g \qquad \overline{\mathrm{c}; g \xrightarrow{\mathrm{c}} g}}{g\ [\ ]\ \mathrm{c}; g \xrightarrow{\mathrm{cc}} (g\ [\ ]\ g)\ [\ ]\ g}}{g \xrightarrow{\mathrm{cc}} (g\ [\ ]\ g)\ [\ ]\ g} \qquad \dfrac{\dfrac{g \xrightarrow{\mathrm{cc}} (g\ [\ ]\ g)\ [\ ]\ g \qquad \overline{\mathrm{c}; g \xrightarrow{\mathrm{c}} g}}{g\ [\ ]\ \mathrm{c}; g \xrightarrow{\mathrm{ccc}} ((g\ [\ ]\ g)\ [\ ]\ g)\ [\ ]\ g}}{g \xrightarrow{\mathrm{ccc}} ((g\ [\ ]\ g)\ [\ ]\ g)\ [\ ]\ g} \quad \ldots$$

D'où, pour tout $i > 0$, il existe la transition suivante :

$$g \xrightarrow{c^i} (\ldots((g\ \underbrace{[\ ]\ g)\ [\ ]\ g)\ldots)[\ ]\ g}_{i-fois}\ .$$

Cela veut dire qu'il y a une infinité de transitions sortant de l'état g, et même de genre différent de celles que nous avons calculées. Dans le cas général (avec la règle `Call-bis`), l'ensemble de transitions qui sortent d'un état, peut ne pas être récursif (cf. [dS84, Vaa92]). Dans [Vaa92], l'auteur appelle ce type de sémantique SOS (c.-à-d. les formulations via des règles de dérivation qui engendrent des systèmes de transitions avec des états dont l'ensemble de transitions sortantes n'est pas récursif) *"non-effectives"*, ou "peu opérationnelles" – dans le sens que, étant donné un état (terme), le calcul de toutes les transitions accessibles à partir de cet état peut être infini.

Nous avons donc décidé de retirer les formulations $\mathcal{R}_3$ et $\mathcal{R}_4$ comme "peu opérationnelles".

Un autre critère pour comparer les sémantiques opérationnelles est la taille des graphes de processus engendrés.

Proposition III.22 *Notons $\rightarrow_i$, pour $i \in \{0,1,2\}$, les transitions de $\mathtt{sem}_i(\mathcal{P})$ et notons $\stackrel{*}{\Longrightarrow}_i$ la relation d'accessibilité, c.-à-d. $\{(P,Q) \in \mathcal{P}\times\mathcal{P} \mid \exists\alpha\ P\stackrel{\alpha}{\Longrightarrow}_i Q\}$. On a alors :*

1. $\rightarrow_0 \ \subset \ \rightarrow_1$ *et* $\stackrel{*}{\Longrightarrow}_0 \ \subset \ \stackrel{*}{\Longrightarrow}_1$
2. $\rightarrow_2 \ \not\subseteq \ \rightarrow_0$ *et* $\rightarrow_0 \ \not\subseteq \ \rightarrow_2$, *mais* $\stackrel{*}{\Longrightarrow}_0 \ \subset \ \stackrel{*}{\Longrightarrow}_2$

Preuve.

1.
   (a) $\rightarrow_0 \ \subseteq \ \rightarrow_1$: Par le Lemme III.15.

   (b) $\rightarrow_0 \ \not\supseteq \ \rightarrow_1$: Par le contre-exemple suivant. Considérons $(g \doteq a; \mathtt{stop})$. Alors $(g, \emptyset, g)$ est dans $\rightarrow_1$ mais pas dans $\rightarrow_0$.

   (c) $\stackrel{*}{\Longrightarrow}_0 \ \subseteq \ \stackrel{*}{\Longrightarrow}_1$: Evident car $\rightarrow_0 \ \subseteq \ \rightarrow_1$.

   (d) $\stackrel{*}{\Longrightarrow}_0 \ \not\supseteq \ \stackrel{*}{\Longrightarrow}_1$: Par le contre-exemple suivant. Considérons $(g \doteq a; \mathtt{stop})$. Le couple d'états $((g\ [\,]\ g), (a; \mathtt{stop}\ [\,]\ g))$ est dans $\stackrel{*}{\Longrightarrow}_1$ mais pas dans $\stackrel{*}{\Longrightarrow}_0$.

2.
   (a) $\rightarrow_0 \ \not\subseteq \ \rightarrow_2$: Par le contre-exemple suivant.
   Le triplet $((a; \mathtt{stop}), \emptyset, (a; \mathtt{stop}))$ est dans $\rightarrow_0$ mais pas dans $\rightarrow_2$.

   (b) $\rightarrow_0 \ \not\supseteq \ \rightarrow_2$: Par le contre-exemple suivant. Considérons $(g \doteq a; \mathtt{stop})$. Alors $((b; g\ [\,]\ g), b, (g\ [\,]\ g))$ est dans $\rightarrow_2$ mais pas dans $\rightarrow_0$.

   (c) $\stackrel{*}{\Longrightarrow}_0 \ \subseteq \ \stackrel{*}{\Longrightarrow}_2$:
   Par les Lemme III.15 et III.17(1) nous avons $\rightarrow_0 \subseteq \rightarrow_1 \subseteq \Longrightarrow_2$,
   d'où directement $\stackrel{*}{\Longrightarrow}_0 \ \subseteq \ \stackrel{*}{\Longrightarrow}_2$.

   (d) $\stackrel{*}{\Longrightarrow}_0 \ \not\supseteq \ \stackrel{*}{\Longrightarrow}_2$: Par le contre-exemple suivant. Considérons "$g \doteq a; \mathtt{stop}$". Alors $((g\ [\,]\ g), (a; \mathtt{stop}\ [\,]\ g))$ est dans $\stackrel{*}{\Longrightarrow}_2$ mais pas dans $\stackrel{*}{\Longrightarrow}_0$. ∎

La relation $\stackrel{*}{\Longrightarrow}_i$ est la relation d'accessibilité via la relation de transition $\rightarrow_i$. Donc, pour tout état P, l'ensemble des états du graphe de processus $\mathtt{graph}_i(P)$, par rapport à $\rightarrow_i$, est égal à $\{Q \mid P\stackrel{*}{\Longrightarrow}_i Q\}$. Autrement dit, l'inclusion $\stackrel{*}{\Longrightarrow}_i \subseteq \stackrel{*}{\Longrightarrow}_j$ signifie que pour tout état P, l'ensemble des états de $\mathtt{graph}_i(P)$ est inclus dans l'ensemble des états de $\mathtt{graph}_j(P)$.

En résumé, la sémantique $\mathtt{sem}_0(\mathcal{P})$ $(= \mathtt{sem}(\mathcal{P}))$ est telle que pour tout terme $P \in \mathcal{P}$, le graphe de processus $\mathtt{graph}(P)$ associé est toujours "de branchement fini" (ce qui n'est pas le cas pour $\mathtt{sem}_3(\mathcal{P})$ et $\mathtt{sem}_4(\mathcal{P})$) et minimal pour l'inclusion des états par rapport à $\mathtt{sem}_1(\mathcal{P})$ et $\mathtt{sem}_2(\mathcal{P})$.

Chapter 11

Puissance expressive de MAPA

Trois critères différents selon lesquels une algèbre de processus peut avoir la "puissance expressive la plus générale" sont proposés dans [Vaa92] :

1. *Chaque machine de Turing peut être simulée pas à pas.* Cela veut dire que pour toute *description instantanée*[1], δ, d'une machine de Turing TM, nous pouvons construire un terme $P(TM, \delta)$, dont le graphe de processus simule pas par pas l'évolution de TM.

2. *Chaque graphe de processus "effectif"*[2] *peut être spécifié modulo une notion d'équivalence comportementale.*

3. *Chaque opération dans la classe d'opérations "naturelles"est réalisable en termes d'opérations dans le langage modulo une notion d'équivalence comportementale.*

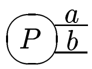
Nous allons montrer comment les opérations de *renommage* et de *choix* peuvent parfois être implémentées dans MAPA. Cependant, le point 3 reste ouvert et dépend fortement de la signification donnée au terme "naturel". Ensuite, nous formulerons notre résultat principal selon lequel tout "graphe de processus récursivement énumérable" peut être spécifié dans MAPA modulo l'*équivalence faible*($\approx$). C'est-à-dire que MAPA a la "puissance d'expression la plus générale" par rapport au point 2. En le prouvant, on montre en même temps que MAPA satisfait aussi le premier critère, car le point 2 englobe le point 1.

[1] Une *description instantanée* d'une machine de Turing décrit un état global de calcul de la machine, c.-à-d. l'état local de la machine, le contenu du ruban (ou des rubans, s'il y en a plusieurs), et la position de la tête sur le ruban.

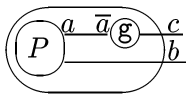
[2] Dans la suite, nous utiliserons le terme "récursivement énumérable" à la place du terme "effectif", pour rester conforme à la terminologie de Vaandrager, dans laquelle un état de "graphe de processus effectif" est un état qui a un ensemble récursif de transitions sortantes (c.f. [Vaa92]).

11.1 Renommage

Soit P un processus tel que $\texttt{Sort}(P) \subseteq [ab]$, c'est-à-dire tel que le processus communique par les actions a et b, éventuellement simultanément. Cela peut être représenté par la figure suivante, dans laquelle le processus P communique via deux canaux indépendants $[a]$ et $[b]$ (puisque $[a]+[b]=[ab]$) :



Pour renommer une action du processus P, par exemple a par c, nous pouvons envisager la configuration suivante :



où l'agent $\mathtt{g}$ possède l'équation définissante suivante ($\mathtt{g} \doteq \overline{a}c; \mathtt{g}$). Dans la syntaxe de MAPA nous notons cela : $((P\,[a]\,\mathtt{g}) : [cb])$.

Même si en général le renommage d'actions atomiques d'un processus arbitraire n'est pas évident, nous pouvons noter que si la sorte d'un processus est connue, nous sommes en mesure de faire beaucoup plus qu'un simple renommage d'actions atomiques, par exemple des duplications, des synchronisations multiples, ou d'autres opérations non triviales sur les actions de processus.

11.2 Choix

Considérons l'exemple suivant. Soit $t = ((\overline{\mathrm{c}}; \texttt{stop}\ [\overline{\mathrm{c}}]\ \ (\mathrm{c}; P\,[\,]\,\mathrm{c}; Q)) : ch)$ un terme. Supposons que le canal ch ne porte ni c ni $\overline{\mathrm{c}}$ (c'est-à-dire, $\{\mathrm{c}\}, \{\overline{\mathrm{c}}\} \notin ch$). Dans ce cas, le processus peut uniquement exécuter la multi-action vide d'une des deux synchronisations possibles, devenant alors :

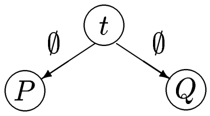
- soit $t \xrightarrow{\emptyset} ((\texttt{stop}\ [\overline{\mathrm{c}}]\ \ (P\,[\,]\,\mathrm{c}; Q)) : ch) \quad (\equiv\ \ (P : ch))$,
- soit $t \xrightarrow{\emptyset} ((\texttt{stop}\ [\overline{\mathrm{c}}]\ \ (\mathrm{c}; P\,[\,]\,Q)) : ch) \quad (\equiv\ \ (Q : ch))$.

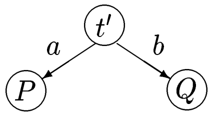
Si de plus $\texttt{Sort}(P) \subseteq ch$ et $\texttt{Sort}(Q) \subseteq ch$, alors le processus t est en fait un choix “interne” entre P et Q, car dans ce cas $(P : ch) \equiv P$ et $(Q : ch) \equiv Q$.

Les multi-actions nous permettent de modifier légèrement le processus t afin que le choix devienne “externe”, c'est-à-dire que c'est l'environnement qui décide quel sous-processus P ou Q sera choisi. Prenons, par exemple :

$$t' = ((\overline{\mathrm{c}}; \texttt{stop}\ [\overline{\mathrm{c}}]\ \ (\mathrm{c}a; P\,[\,]\,\mathrm{c}b; Q)) : ch)\,,$$

avec $\{a\}, \{b\} \in ch$. Intuitivement, le processus t' décrit un choix “gardé” entre P et Q. L'environnement peut choisir P ou Q, en explicitant ce choix via l'action a pour P et b pour Q.





Notons que la composition parallèle par un canal vide $[\ ]$, qui vérifie $\overline{[\ ]} = [\ ]$ et $[\ ] + [\ ] \subseteq [\ ]$, est commutative d'après la Proposition III.8(2) et associative d'après la Proposition III.8(5). La commutativité et l'associativité de $[\ ]$ nous permettent d'écrire $P\ [\ ]\ Q\ [\ ]\ R$ sans parenthèses, ou, de façon plus générale, $[\ ]_{i \in I}\ P_i$, où I est un ensemble fini d'indices, sans ambiguïté.

Soit $\{m_i; P_i \mid i \in I\}$ un ensemble fini de termes et $k \in \texttt{Act}$ tel que pour tout $i \in I$, $\{k\}, \{\overline{k}\} \notin [\texttt{Sort}(m_i; P_i)]$.

Nous définissons une "macro" $\sum\{m_i; P_i \mid i \in I\}$ par

- $\Big(([\ ]_{i \in I}\ km_i; P_i\ )\ \ [k]\ \ \overline{k}; \texttt{stop} \Big) : \mu_f(\texttt{Act} \setminus \{k, \overline{k}\})$ — $I \neq \emptyset$,
- $\texttt{stop}$ — si $I = \emptyset$.

Cette macro est en fait l'implémentation générale de l'opération du choix gardé (ang. *guarded choice*), si importante dans d'autres algèbres de processus. Pour montrer que la macro implémente vraiment l'opérateur de choix gardé, supposons un opérateur n-aire $\Box_n$ $(n \geq 2)$ ayant les règles de dérivation standard suivantes :

$$\frac{\mathbf{P}_i \xrightarrow{m} \mathbf{Q}}{\Box_n(\mathbf{P}_1, .., \mathbf{P}_i, .., \mathbf{P}_n) \xrightarrow{m} \mathbf{Q}}\ m \neq \emptyset \qquad \frac{\mathbf{P}_i \xrightarrow{\emptyset} \mathbf{Q}}{\Box_n(\mathbf{P}_1, .., \mathbf{P}_i, .., \mathbf{P}_n) \xrightarrow{\emptyset} \Box_n(\mathbf{P}_1, .., \mathbf{Q}, .., \mathbf{P}_n)}$$

pour tout $i \in \{1, \ldots, n\}$.

Théorème III.23 *Dans MAPA enrichie par l'opération choix* $\Box_n$*, dont l'ensemble de termes va être noté* $\mathcal{P}^{\Box}$*, nous avons :*

$$\Box_n(m_1; P_1, \ldots, m_n; P_n)\ \simeq\ \sum\{m_1; P_1, \ldots, m_n; P_n\}$$

quelles que soient[3] *les multi-actions non vides* $m_i \in \mu_f(\texttt{Act}) \setminus \{\emptyset\}$ *et les termes* $P_i \in \mathcal{P}^{\Box}$*, pour* $i \in \{1, \ldots, n\}$.

Dans la preuve de ce théorème, nous avons besoin du lemme suivant, dont la preuve, omise ici, est semblable à celle de la Proposition III.8(1–3).

Lemme III.24 $\forall m \in \mu_f(\texttt{Act})$, $\forall ch \in \texttt{Chan}$, $\forall P, Q \in \mathcal{P}^{\Box}$ *:*

$$m \notin ch\ \ \Rightarrow\ \ (P\ [\ ]\ m; Q) : ch\ \equiv\ P : ch\ .$$

[3] Puisque l'ensemble $D \subset \texttt{Act}$ d'actions sur lequel un processus construit ses multi-actions est fini, alors il existe toujours une action $k \in \texttt{Act}$ qui n'est pas dans D. D'où, nous pouvons toujours choisir une action $k \in \texttt{Act}$ qui vérifie les conditions.

Preuve du Théorème III.23

Grâce au Lemme III.24, nous pouvons prouver que $B = R \cup R^{-1} \cup \equiv$, où $\equiv$ dénote l'équivalence par isomorphisme de graphe de processus des termes de MAPA enrichie par l'opération $\Box_n$, et

$$R = \{ (\Box_n(m_1; P_1, \ldots, m_n; P_n), \sum\{m_1; P_1, \ldots, m_n; P_n\}) \mid m_1, \ldots, m_n \in \mu_f(\texttt{Act}) \setminus \{\emptyset\} \text{ et } P_1, \ldots, P_n \in \mathcal{P}\ \Box\},$$

est une bisimulation forte, c'est-à-dire que :

1. Si $\Box_n(m_1; P_1, \ldots, m_n; P_n) \xrightarrow{m} P$, alors il existe Q tel que $(P, Q) \in B$ et $\sum\{m_1; P_1, \ldots, m_n; P_n\} \xrightarrow{m} Q$;
2. Si $\sum\{m_1; P_1, \ldots, m_n; P_n\} \xrightarrow{m} Q$, alors il existe P tel que $(P, Q) \in B$ et $\Box_n(m_1; P_1, \ldots, m_n; P_n) \xrightarrow{m} P$.

Ad. 1.

(a) $m = \emptyset$.
Par définition, $m_1, \ldots, m_n$ ne sont pas des multi-actions vides.
Donc $P = \Box_n(m_1; P_1, \ldots, m_n; P_n)$.
Dans ce cas, $Q = \sum\{m_1; P_1, \ldots, m_n; P_n\}$ fait l'affaire,
car $\sum\{m_1; P_1, \ldots, m_n; P_n\} \xrightarrow{\emptyset} \sum\{m_1; P_1, \ldots, m_n; P_n\}$
et, par hypothèse, $(P, Q) \in B$.

(b) Si $m \neq \emptyset$ alors $m = m_i$ pour un $i \in \{1, \ldots, n\}$, et dans ce cas $P = P_i$.
D'autre part, $\sum\{m_1; P_1, \ldots, m_i; P_i, \ldots, m_n; P_n\} \xrightarrow{m_i}$
$((km_1; P_1 \,[\,]\ldots[\,]\, P_i \,[\,]\ldots[\,]\, km_n; P_n)\ [k]\ \texttt{stop}) : \mu_f(\texttt{Act} \setminus \{k, \overline{k}\})$.
D'après la Prop. III.8(1) et le Lemme III.24, le terme cible de la transition est équivalent par $\equiv$ à $P_i : \mu_f(\texttt{Act} \setminus \{k, \overline{k}\})$.
Comme $\texttt{Sort}(P_i) \subseteq \mu_f(\texttt{Act} \setminus \{k, \overline{k}\})$, par la Prop. III.8(3), nous avons $P_i : \mu_f(\texttt{Act} \setminus \{k, \overline{k}\}) \equiv P_i$.

Ad. 2. Le terme $\sum\{m_1; P_1, \ldots, m_n; P_n\}$ peut évoluer seulement selon les deux types de règles déjà présentées.

(a) $\sum\{m_1; P_1, \ldots, m_n; P_n\} \xrightarrow{\emptyset} \sum\{m_1; P_1, \ldots, m_n; P_n\}$.
Cas facile, car $\Box_n(m_1; P_1, \ldots, m_n; P_n) \xrightarrow{\emptyset} \Box_n(m_1; P_1, \ldots, m_n; P_n)$.

(b) $\sum\{m_1; P_1, \ldots, m_i; P_i, \ldots, m_n; P_n\} \xrightarrow{m_i}$
$((km_1; P_1 \,[\,]\ldots[\,]\, P_i \,[\,]\ldots[\,]\, km_n; P_n)\ [k]\ \texttt{stop}) : \mu_f(\texttt{Act} \setminus \{k, \overline{k}\})$.
D'après le même raisonnement que pour le point précédant, le terme cible de la transition est équivalent par $\equiv$ à P_i.
De l'autre côté, nous avons directement
$\Box_n(m_1; P_1, \ldots, m_i; P_i, \ldots, m_n; P_n) \xrightarrow{m_i} P_i$. ■

11.3 Graphes de processus récursivement énumérables

La puissance expressive des algèbres de processus dans la description des graphes de processus a déjà été analysée [dS84, BBK87b, Vaa92]. Par exemple, Vaandrager dans [Vaa92] prouve que l'on ne peut pas espérer trouver une algèbre de processus avec une sémantique SOS "effective", capable de décrire chaque graphe de processus récursif, modulo l'équivalence par traces ($\simeq_t$). D'autre part, presque toutes les algèbres de processus habituelles (ex. CCS, CSP, SCCS, ACP$_\tau$ [BBK87b, BW90], PC [Vaa92]) sont suffisamment puissantes pour décrire tout graphe de processus "effectif" modulo l'équivalence faible.

Définition III.7 Un graphe de processus $(\mathtt{St}, \mathtt{Lab}, \rightarrow, s)$ est *récursivement énumérable* s'il existe une machine de Turing qui, pour chaque état $p \in \mathtt{St}$ (considéré en tant qu'entrée), calcule toutes ses transitions sortantes, c.-à-d. énumère (une représentation de) l'ensemble $\{(\alpha, q) \mid p \xrightarrow{\alpha} q\}$. □

Le théorème suivant montre que notre algèbre de processus très simple (munie seulement de trois combinateurs) est suffisamment puissante pour décrire tout "graphe de processus récursivement énumérable" modulo l'équivalence faible $\approx$.

Théorème III.25 *Tout graphe de processus récursivement énumérable étiqueté par des multi-ensembles finis sur un alphabet fini, peut être décrit (modulo $\approx$) par un terme de MAPA.*

Nous avons considéré ici les graphes de processus dont les transitions sont étiquetées par des multi-ensembles finis sur un ensemble fini L_f. Ainsi, nous pouvons supposer, sans perte de généralité, que tout élément de L_f est codé par un simple caractère de ruban dans une représentation appropriée pour une machine de Turing, et qu'une multi-action est codée comme un mot sur de tels caractères.

Notons toutefois que nous travaillons toujours avec un domaine infini d'étiquettes de transition d'un graphe de processus. Même si L_f est fini (non vide), l'ensemble des multi-ensembles finis, $\mu_f(L_f)$, n'est pas fini.

11.3.1 Esquisse de la preuve du Théorème III.25

Si $G = (\mathtt{St}, \mu_f(L_f), \rightarrow, s)$ est un graphe de processus récursivement énumérable sur $\mu_f(L_f)$, alors, par définition, il existe une machine de Turing qui pour chaque état $p \in \mathtt{St}$ calcule l'ensemble des transitions sortantes. Nous supposerons, sans perte de généralité, que :

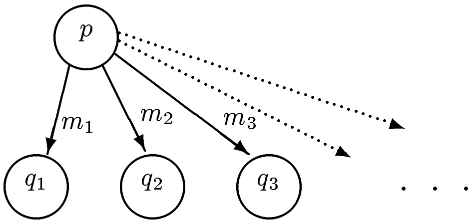
1. Une telle machine de Turing (TM) travaille sur trois rubans : un ruban d'entrée (en lecture seule), un ruban de travail (en lecture-écriture) et un ruban de sortie (en écriture seule) (voir Figure 11.1).

2. Les états du graphe de processus (éléments de St) sont codés dans TM par des mots sur un alphabet fini $\mathcal{V}$, et l'état p se trouve codé sur le ruban d'entrée.

3. Les étiquettes du graphe de processus (multi-ensembles finis sur L_f) sont simplement des mots sur l'alphabet L_f (on suppose que $L_f \cap \mathcal{V} = \emptyset$).

4. Le résultat du calcul de TM, c'est-à-dire l'ensemble $\{(\alpha, q) \in \texttt{Lab} \times \texttt{St} \mid (p, \alpha, q) \in \rightarrow\}$, est écrit sur le ruban de sortie comme une liste de paires *(bag,state)* séparées par un symbole spécial #.

   Par exemple, le résultat $m_1 q_1 \# m_2 q_2 \# m_3 q_3 \# \ldots$ sur le ruban de sortie, représente les transitions sortantes de l'état p du graphe de processus suivant :



   L'ensemble des transitions sortantes n'est pas forcément fini. Dans ce cas, la machine ne s'arrête jamais ; en revanche toute transition va être écrite sur le ruban de sortie au bout d'un temps fini.

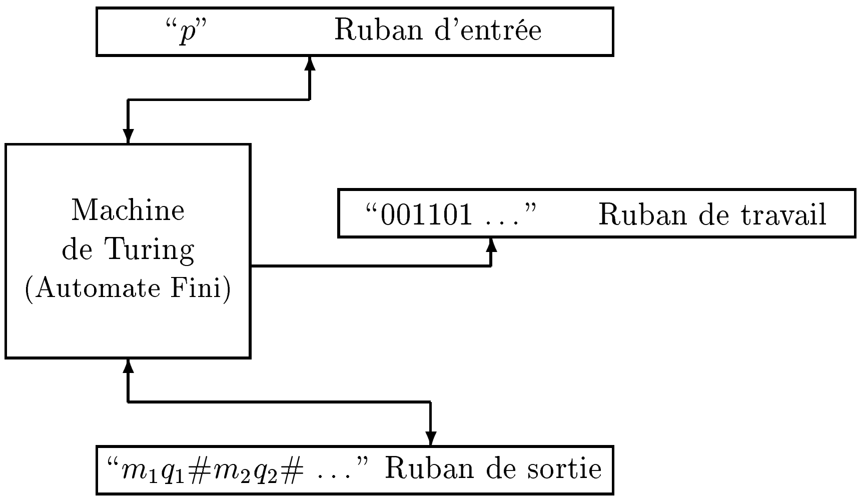


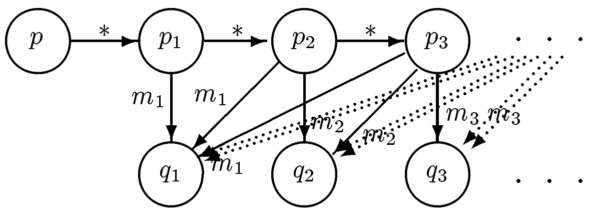
Figure 11.1: Schéma de la machine de Turing.

Formellement, notre machine de Turing sera un 4–uplet :

$$\texttt{TM} = (\mathcal{Q}, Q_0, \Gamma, f)$$

où $\mathcal{Q} = \{Q_0, Q_1, \ldots, Q_k\}$ est l'ensemble fini des états de la machine avec Q_0 comme état de départ (initial), $\Gamma = \mathcal{V}_I \cup \mathcal{V}_T \cup (\mathcal{V}_O \cup \{\#\} \cup L_f)$ est l'ensemble des symboles sur lesquels la machine travaille (nous supposerons que les ensembles $\mathcal{V}_I$, $\mathcal{V}_T$, $\mathcal{V}_O$, $\{\#\}$, L_f, sont deux à deux disjoints et que le ruban d'entrée n'utilise que les symboles de $\mathcal{V}_I$, le ruban de travail n'utilise que les symboles de $\mathcal{V}_T$, et le ruban de sortie n'utilise que les symboles de $\mathcal{V}_O \cup \{\#\} \cup L_f$), et $f : \mathcal{Q} \times \Gamma \mapsto \mathcal{Q} \times \Gamma \times \{\mathtt{r}_I, \mathtt{l}_I, \mathtt{r}_T, \mathtt{l}_T, \mathtt{r}_O, \mathtt{l}_O\}$ est la fonction de transition permettant à la machine de transformer son état, de modifier le symbole courant sur un de ses rubans et de déplacer à gauche ($\mathtt{l}$) ou à droite ($\mathtt{r}$) la tête d'un de ses rubans. Les indices sont utilisés pour préciser de quel ruban il s'agit. Une machine de Turing est munie aussi d'un ensemble d'états finaux qui ne sont pas considérés ici. De toute façon, nous devons prendre en compte aussi les cas où la machine ne s'arrête jamais, c.-à-d. quand elle n'atteint jamais un état final.

Etant donnée la description d'une telle machine de Turing `TM`, avec le code de l'état initial du graphe de processus modélisé sur le ruban d'entrée, nous sommes capables de construire un terme MAPA dont le graphe de processus est équivalent (par rapport à $\approx$) à celui encodé par `TM`. En fait, un tel terme décrira un graphe de processus de la forme suivante :



Les chemins silencieux (notés $*$) représentent des calculs internes de `TM`. L'état p_1 est atteint dès que le premier "#" est écrit sur le ruban de sortie, ce qui signifie que la première paire (m_1, q_1) a été calculée. Après cela, le graphe de processus peut continuer à simuler la machine de Turing (c.-à-d. à calculer les autres transitions) ou peut exécuter la multi-action m_1 et, dans ce cas, la simulation de la machine de Turing avec p sur le ruban d'entrée est interrompue. Quand l'état p_2 est atteint, à la fois les transitions (m_1, q_1) et (m_2, q_2) sont calculées, et donc, une d'entre elles peut être franchie. Dès qu'une des transitions calculées (m_i, q_i) est franchie, la simulation de la machine de Turing recommence, mais, cette fois, avec q_i sur le ruban d'entrée.

Pour décrire un tel graphe de processus dans MAPA, nous devons principalement simuler trois choses :

1. un automate fini qui implémente la fonction de transition f de la machine de Turing ;

2. un ruban (infini à droite) de machine de Turing ;

3. un "ramasseur" de signaux (actions atomiques) qui, après avoir recueilli des signaux (multi-actions singletons) un par un, est capable d'engendrer la multi-action composée de tous ces signaux.

Automate fini pour la fonction de transition

Sans perte de généralité, nous supposerons que la fonction de transition f de la machine TM est décomposable en trois fonctions :

1. $f_I : \mathcal{Q} \times \mathcal{V}_I \mapsto \mathcal{Q} \times \{\mathtt{r}_I, \mathtt{l}_I\}$,

2. $f_T : \mathcal{Q} \times \mathcal{V}_T \mapsto \mathcal{Q} \times \mathcal{V}_T \times \{\mathtt{r}_T, \mathtt{l}_T\}$,

3. $f_O : \mathcal{Q} \mapsto \mathcal{Q} \times (\mathcal{V}_O \cup \{\#\} \cup L_f)$.

Une telle présentation garantit qu'il n'y a effectivement pas d'écriture sur le ruban d'entrée, ni de lecture sur le ruban de sortie.

Comme le codage d'un état du graphe de processus modélisé est le même sur le ruban de sortie que sur le ruban d'entrée, nous distinguerons les symboles de $\mathcal{V}$, respectivement, par l'indexage par I ou O. C'est-à-dire, que $\mathcal{V}_I$ dénotera l'ensemble $\mathcal{V}$ mais utilisé par le ruban d'entrée, tandis que $\mathcal{V}_I$ dénotera le même ensemble mais utilisé par le ruban de sortie.

Pour tout Q_i de $\mathcal{Q}$ nous définissons un agent de la manière suivante :

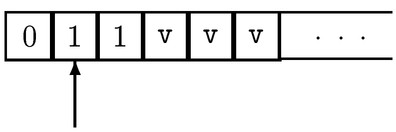
$$Q_i \doteq \sum \left(\begin{array}{l} \{(x\overline{xz};Q_j) \mid f_I(Q_i,x)=(Q_j,z),\ x\in\mathcal{V}_I,\ z\in\{\mathtt{r}_I,\mathtt{l}_I\}\}\cup \\ \{(x\overline{yz};Q_j) \mid f_T(Q_i,x)=(Q_j,y,z),\ x,y\in\mathcal{V}_T,\ z\in\{\mathtt{r}_T,\mathtt{l}_T\}\}\cup \\ \{(\overline{y};Q_j) \mid f_O(Q_i)=(Q_j,y),\ y\in\mathcal{V}_O\cup L_f\}\cup \\ \left\{(\overline{\#\#\#};Q_j) \mid f_O(Q_i)=(Q_j,\#)\right\}\cup \\ \{(\mathrm{w};\mathtt{stop})\} \end{array} \right)$$

Intuitivement, l'automate fini se trouvant à l'état Q_i est représenté par le terme Q_i. Son évolution vers un nouvel état se fait par le franchissement d'une des multi-actions disponibles (sous $\sum$) qui encode un accès à un des rubans. L'action $\mathrm{w} \notin \Gamma$ permet d'arrêter l'exécution de la machine à tout moment.

Bien entendu, d'autres choix sont possibles qui, par exemple, permettent un accès simultané sur plusieurs rubans.

Simulation d'un ruban de machine de Turing

Un ruban est une liste infinie de cases, chacune étant capable de mémoriser une valeur x appartenant à un ensemble fini $\mathcal{V}$. A un instant donné, on peut lire et écrire dans une case, et on peut se déplacer vers la case voisine qui est à gauche ou à droite de cette case. Seule une partie finie du ruban peut prendre des valeurs différentes d'une valeur spéciale, notée ici $\mathtt{v}$. Par exemple, si $\mathcal{V} = \{0, 1, \mathtt{v}\}$, alors le ruban peut prendre la forme suivante :



Un ruban peut être considéré comme un agent $\textsc{Tape}(\mathcal{V},\mathtt{l},\mathtt{r})$, qui interagit avec son environnement par des multi-actions de la forme $\{\overline{x}yz\}$, où $x \in \mathcal{V}$ représente l'ancienne valeur de la case courante, $y \in \mathcal{V}$ représente la nouvelle valeur de la case courante, et $z \in \{\mathtt{l},\mathtt{r}\}$ le choix de la case suivante : $\mathtt{l}$ - la case voisine gauche, et $\mathtt{r}$ - la case voisine droite. Ainsi, la sorte de l'agent $\textsc{Tape}(\mathcal{V},\mathtt{l},\mathtt{r})$ sera

$$ch_T = [\{\overline{x}yz \mid x,y \in \mathcal{V}, z \in \{\mathtt{l},\mathtt{r}\}\}] \ .$$

Le ruban vide dont les cases peuvent mémoriser les valeurs de $\mathcal{V}$ est défini par :

$$\textsc{Tape}(\mathcal{V},\mathtt{l},\mathtt{r}) \doteq \mathbf{F}_{\mathrm{O}} : [ch_T]$$

où :
- $\mathbf{F}_{\mathrm{O}} \doteq \sum \{\overline{\mathtt{v}}y\mathtt{l}; (\overline{\mathrm{o}}; \overline{\mathrm{o}}; \mathbf{C}^y_{\mathrm{O}} \ \ [\mathrm{e}] \ \ \overline{\mathrm{e}}; \mathbf{F}_{\mathrm{E}}) \ , \overline{\mathtt{v}}y\mathtt{r}; (\mathrm{e}; \mathbf{C}^y_{\mathrm{O}} \ \ [\mathrm{e}] \ \ \mathbf{F}_{\mathrm{E}}) \ \mid \ y \in \mathcal{V}\} : ch_T + [\overline{\mathrm{o}}]$
- $\mathbf{F}_{\mathrm{E}} \doteq \sum \{\overline{\mathtt{v}}y\mathtt{l}; (\overline{\mathrm{e}}; \overline{\mathrm{e}}; \mathbf{C}^y_{\mathrm{E}} \ \ [\mathrm{o}] \ \ \overline{\mathrm{o}}; \mathbf{F}_{\mathrm{O}}) \ , \overline{\mathtt{v}}y\mathtt{r}; (\mathrm{o}; \mathbf{C}^y_{\mathrm{E}} \ \ [\mathrm{o}] \ \ \mathbf{F}_{\mathrm{O}}) \ \mid \ y \in \mathcal{V}\} : ch_T + [\overline{\mathrm{e}}]$

et pour chaque x de $\mathcal{V}$:
- $\mathbf{C}^x_{\mathrm{O}} \doteq \sum \{\overline{x}y\mathtt{l}; \overline{\mathrm{o}}; \overline{\mathrm{o}}; \mathbf{C}^y_{\mathrm{O}} \, , \overline{x}y\mathtt{r}; \mathrm{e}; \mathrm{e}; \mathbf{C}^y_{\mathrm{O}} \ \mid \ y \in \mathcal{V}\}$
- $\mathbf{C}^x_{\mathrm{E}} \doteq \sum \{\overline{x}y\mathtt{l}; \overline{\mathrm{e}}; \overline{\mathrm{e}}; \mathbf{C}^y_{\mathrm{E}} \, , \overline{x}y\mathtt{r}; \mathrm{o}; \mathrm{o}; \mathbf{C}^y_{\mathrm{E}} \ \mid \ y \in \mathcal{V}\}$

L'agent $\mathbf{C}^x_{\mathrm{O}}$ représente une case impaire dans laquelle on a déjà lu et écrit la valeur x, $\mathbf{C}^x_{\mathrm{E}}$ représente une case paire, et $\mathbf{F}_{\mathrm{E}}$ (resp. $\mathbf{F}_{\mathrm{O}}$) représente la première case non visitée, dans le cas où c'est une case impaire (resp. paire).

Exemple III.2 *Soit $\mathcal{V} = \{0,1,\mathtt{v}\}$:*

$\textsc{Tape}(\mathcal{V},\mathtt{l},\mathtt{r}) \ \xrightarrow{\emptyset} \ (\mathbf{F}_{\mathrm{O}}) : ch_T \ \xrightarrow{\emptyset}$ *(Instanciation de $\mathbf{F}_{\mathrm{O}}$ par sa définition (`Call`))*

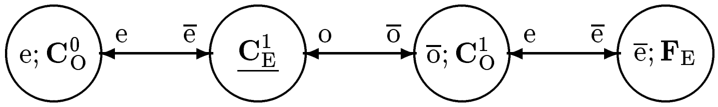
$\xrightarrow{\overline{\mathtt{v}}0\mathtt{r}} \ \big((\mathrm{e}; \mathbf{C}^0_{\mathrm{O}} \ \ [\mathrm{e}] \ \ \mathbf{F}_{\mathrm{E}}) : ch_T + [\overline{\mathrm{o}}]\big) : ch_T \ \xrightarrow{\emptyset}$ *(idem pour $\mathbf{F}_{\mathrm{E}}$)*

$\xrightarrow{\overline{\mathtt{v}}1\mathtt{r}} \ \big((\mathrm{e}; \mathbf{C}^0_{\mathrm{O}} \ \ [\mathrm{e}] \ \ (\mathrm{o}; \mathbf{C}^1_{\mathrm{E}} \ \ [\mathrm{o}] \ \ \mathbf{F}_{\mathrm{O}}) : ch_T + [\overline{\mathrm{e}}]\big) : ch_T + [\overline{\mathrm{o}}]\big) : ch_T \ \xrightarrow{\emptyset}$ *(idem pour $\mathbf{F}_{\mathrm{O}}$)*

$\xrightarrow{\overline{\mathtt{v}}1\mathtt{l}} \ \big((\mathrm{e}; \mathbf{C}^0_{\mathrm{O}} \ \ [\mathrm{e}] \ \ (\mathrm{o}; \mathbf{C}^1_{\mathrm{E}} \ \ [\mathrm{o}] \ \ (\overline{\mathrm{o}}; \overline{\mathrm{o}}; \mathbf{C}^1_{\mathrm{O}} \ \ [\mathrm{e}] \ \ \overline{\mathrm{e}}; \mathbf{F}_{\mathrm{E}}) : ch_T + [\overline{\mathrm{o}}]\big) : ch_T + [\overline{\mathrm{e}}]\big) : ch_T + [\overline{\mathrm{o}}]\big) : ch_T$

$\xrightarrow{\emptyset} \ \big((\mathrm{e}; \mathbf{C}^0_{\mathrm{O}} \ \ [\mathrm{e}] \ \ (\mathbf{C}^1_{\mathrm{E}} \ \ [\mathrm{o}] \ \ (\overline{\mathrm{o}}; \mathbf{C}^1_{\mathrm{O}} \ \ [\mathrm{e}] \ \ \overline{\mathrm{e}}; \mathbf{F}_{\mathrm{E}}) : ch_T + [\overline{\mathrm{o}}]\big) : ch_T + [\overline{\mathrm{e}}]\big) : ch_T + [\overline{\mathrm{o}}]\big) : ch_T$

La figure suivante représente schématiquement le résultat de la dérivation. Seule la seconde case (soulignée) est directement accessible, car les autres sont gardées (préfixées) par une action interne.



Un "ramasseur" d'actions atomiques

Un "ramasseur" est un agent qui recueille séquentiellement les "caractères" (multi-actions singletons $\{\alpha\}$) d'un ensemble fini L_f, jusqu'à ce qu'un caractère spécial $\{\#\}$ ($\# \not\in L_f$) arrive. Ensuite, l'agent est prêt à exécuter la multi-action qui est la somme multi-ensembliste de tous les caractères recueillis. Nous ajouterons à cette multi-action la multi-action $\overline{\mathrm{ww}}$, qui va être utilisée pour informer les autres composants du système global que la multi-action est en phase d'exécution.

La sorte de "ramasseur" sera alors : $ch_R = [L_f] \cup [\#] \cup (\overline{\mu_f(L_f)} + [\overline{\mathrm{ww}}])$.

$$\text{RAMASSEUR} \doteq \left(\sum \{\#; \overline{\mathrm{ww}}; \mathtt{stop}\} \cup \{x; (\overline{x}\mathrm{e}; \mathtt{stop}\ \ [\mathrm{e}]\ \ \mathrm{G_E}) \ \ | \ \ x \in L_f\}\right) : ch_R$$

où :

- $\mathrm{G_E} \doteq \left(\sum \{\#; \overline{\mathrm{wwe}}; \mathtt{stop}\} \cup \{x; (\overline{x}\mathrm{o}\overline{\mathrm{e}}; \mathtt{stop}\ \ [\mathrm{o}]\ \ \mathrm{G_O}) \ \ | \ \ x \in L_f\}\right) : ch_R + [\overline{\mathrm{e}}]$
- $\mathrm{G_O} \doteq \left(\sum \{\#; \overline{\mathrm{wwo}}; \mathtt{stop}\} \cup \{x; (\overline{x}\mathrm{e}\overline{\mathrm{o}}; \mathtt{stop}\ \ [\mathrm{e}]\ \ \mathrm{G_E}) \ \ | \ \ x \in L_f\}\right) : ch_R + [\overline{\mathrm{o}}]$

Exemple III.3 *Soit $L_f = \{\alpha, \beta, \ldots\}$:*

$\text{RAMASSEUR} \ \stackrel{\emptyset}{\to}\ \left(\sum \{\#; \overline{\mathrm{ww}}; \mathtt{stop}\} \cup \{x; (\overline{x}\mathrm{e}; \mathtt{stop}\ \ [\mathrm{e}]\ \ \mathrm{G_E}) \ \ | \ \ x \in L_f\}\right) : ch_R$

$\xrightarrow{\alpha}\ (\overline{\alpha}\mathrm{e}; \mathtt{stop}\ \ [\mathrm{e}]\ \ \mathrm{G_E}) : ch_R \ \stackrel{\emptyset}{\to}$ *(Instanciation de* $\mathrm{G_E}$ *par sa définition)*

$\xrightarrow{\beta}\ \left(\overline{\alpha}\mathrm{e}; \mathtt{stop}\ \ [\mathrm{e}]\ \ (\overline{\beta}\mathrm{o}\overline{\mathrm{e}}; \mathtt{stop}\ \ [\mathrm{o}]\ \ \mathrm{G_O}) : ch_R + [\overline{\mathrm{e}}]\right) : ch_R \ \stackrel{\emptyset}{\to}$ *(idem pour* $\mathrm{G_O}$*)*

$\xrightarrow{\#}\ \left(\overline{\alpha}\mathrm{e}; \mathtt{stop}\ \ [\mathrm{e}]\ \ (\overline{\beta}\mathrm{o}\overline{\mathrm{e}}; \mathtt{stop}\ \ [\mathrm{o}]\ \ (\overline{\mathrm{o}}; \mathtt{stop}) : ch_R + [\overline{\mathrm{o}}]) : ch_R + [\overline{\mathrm{e}}]\right) : ch_R$

$\xrightarrow{\overline{\alpha}\overline{\beta}\overline{\mathrm{ww}}}\ \mathtt{stop}$

La construction du terme final

Soit L_f un ensemble fini non vide. Soit $\mathcal{G} = (\mathtt{St}, \mu_f(L_f), \to, s)$ un graphe de processus récursivement énumérable étiqueté par des multi-ensembles finis sur L_f. Par définition, il existe une machine de Turing, $\mathtt{TM} = (\mathcal{Q}, Q_0, \Gamma, f)$, telle que nous l'avons déjà décrite, qui, étant donné un état p de $\mathtt{St}$ encodé sur le ruban d'entrée, calcule l'ensemble des transitions sortantes de p, en énumérant sur son ruban de sortie les paires *(bag,state)*, séparées par #.

Nous allons maintenant construire un terme de MAPA qui décrit le graphe de processus $\mathcal{G}$ modulo l'équivalence faible ($\approx$). Pour le faire, nous utiliserons les termes déjà définis, à savoir :

1. $\mathtt{AF} = Q_0$ - Le terme décrit l'*Automate Fini* pour la fonction de transition de la machine de Turing $\mathtt{TM}$.

2. $\mathtt{T}_I = \text{TAPE}(\mathcal{V}_I, \mathtt{r}_I, \mathtt{l}_I)$ - Le terme décrit un ruban vide capable de mémoriser dans ses cases les éléments de l'ensemble $\mathcal{V}_I$. Ce terme sera utilisé comme ruban d'entrée de la machine de Turing, mais après avoir été initialisé par le code d'un état p de $\mathcal{G}$, ce qui sera noté par $\mathtt{T}_I^p$.

3. $\mathtt{T}_T = \text{TAPE}(\mathcal{V}_T, \mathtt{r}_T, \mathtt{l}_T)$ - Le terme décrit un ruban vide capable de mémoriser dans ses cases les éléments de l'ensemble $\mathcal{V}_T$. Ce terme sera utilisé pour modéliser le ruban de travail de la machine de Turing $\mathtt{TM}$.

4. $\mathtt{R} = \text{RAMASSEUR}$ - Le terme décrit le "ramasseur d'actions atomiques" de L_f.

Pour faciliter la notation nous pré-définissons quelques canaux :

1. $ch_I = [\{\overline{x}yz \mid x, y \in \mathcal{V}_I, z \in \{\mathtt{r}_I, \mathtt{l}_I\}\}]$ - le canal utilisé par le ruban d'entrée.
2. $ch_T = [\{\overline{x}yz \mid x, y \in \mathcal{V}_T, z \in \{\mathtt{r}_T, \mathtt{l}_T\}\}]$ - le canal utilisé par le ruban de travail.
3. $ch_O = [\mathcal{V}_O] \cup [L_f] \cup [\#\#\#] \cup \left([\overline{\mathrm{w}}] + \overline{\mu_f(L_f)}\right)$ - le canal utilisé par le terme $\mathtt{O}$, qui implémente le ruban de sortie.
4. $ch = \overline{ch_I} \cup \overline{ch_T} \cup \overline{ch_O}$ - le canal de l'automate fini.

Le terme dont le graphe de processus est faiblement équivalent à $\mathcal{G}$ a la forme suivante :

$$(\mathtt{AF}\ ch\ \ (\mathtt{T}_I^s\ [\ ]\ \mathtt{T}_T\ [\ ]\ \ (\mathtt{O} : ch_O))) : \overline{\mu_f(L_f)}$$

L'agent $\mathtt{AF}$ décrit l'automate fini dans l'état initial Q_0. Cet agent est composé, via le canal ch, à un ruban d'entrée $\mathtt{T}_I^s$ (sur lequel est encodé l'état initial s du graphe de processus $\mathcal{G}$), d'un ruban de travail initialement vide $\mathtt{T}_T$ et d'un agent $\mathtt{O}$ (qui est sous la restriction via le canal ch_O, et qui a pour but de traiter tout ce qui est écrit sur le ruban de sortie). Le terme entier est sous la restriction $\overline{\mu_f(L_f)}$, ce qui signifie que seules les multi-actions sur $\overline{\mu_f(L_f)}$ pourront être visibles de l'extérieur.

L'agent $\mathtt{O}$ est défini par :

$$\begin{array}{rcl}
\mathtt{O} & \doteq & \left(\mathtt{R}\ [\overline{\mathrm{w}}]\ \ \left(\mathtt{F}\ \overline{ch_I}\ \ \left(\mathtt{M} : ch_I \cup ([\overline{\mathrm{w}}] + \overline{\mu_f(L_f)})\right)\right)\right)\ [\ ]\ (\#; \mathtt{O}) \\
\mathtt{F} & \doteq & \left(\sum\{(y_O \mathrm{v}_I \overline{y_I \mathtt{r}_I}; \mathtt{F}) \mid y_O \in \mathcal{V}_O\} \cup \{\#; \mathtt{stop}\}\right) \\
\mathtt{M} & \doteq & (\mathrm{w}; \mathtt{AF})\ ch\ \ (\mathtt{T}_I\ [\ ]\ \mathtt{T}_T\ [\ ]\ \ (\mathtt{O} : ch_O))
\end{array}$$

Au départ, l'agent $\mathtt{O}$ est composé de trois parties :

1. $\mathtt{R}$ - un RAMASSEUR prêt à recueillir les "caractères" de L_f, qui est en communication par $\overline{\mathrm{w}}$ avec le terme $\left(\mathtt{F}\ \overline{ch_I}\ \ \left(\mathtt{M} : ch_I \cup ([\overline{\mathrm{w}}] + \overline{\mu_f(L_f)})\right)\right)$;
2. $\left(\mathtt{F}\ \overline{ch_I}\ \ \left(\mathtt{M} : ch_I \cup ([\overline{\mathrm{w}}] + \overline{\mu_f(L_f)})\right)\right)$ mémorise l'état qui sera atteint si la multi-action ramassée par $\mathtt{R}$ est exécutée.
3. $\#; \mathtt{O}$. Ce terme reste inactif jusqu'à l'arrivée de l'action $\overline{\#}$.

Dès que la première transition (m, p) (où m est l'étiquette, et p est le code de l'état cible), est calculée, c.-à-d. dès que la multi-action $\overline{\#\#\#}$ a été engagée, le ramasseur est prêt à exécuter la multi-action $m + \overline{\mathrm{ww}}$. La deuxième partie va contenir le terme qui correspond au terme global d'implémentation de la machine de Turing, avec p sur son ruban d'entrée et avec la partie $\mathtt{AF}$ gardée par l'action w. Enfin, le terme $(\#; \mathtt{O})$ sera transformé en $\mathtt{O}$, c.-à-d. qu'une autre transition (m', p') pourra être calculée.

Dès qu'une des multi-actions m d'une des transitions (m, p) calculées est demandée par l'environnement du terme global, elle est exécutée. Plus précisément, la multi-action $m + \overline{\mathrm{w}}\overline{\mathrm{w}}$ est exécutée par un des "ramasseurs", cela bloque automatiquement le `AF` (par un $\overline{\mathrm{w}}$) en cours d'évolution, en activant en même temps (par l'autre $\overline{\mathrm{w}}$) la partie `AF` du terme d'implémentation de la machine de Turing associé, et donc avec p sur le ruban d'entrée.

Chapter 12

Conclusions

Dans cette partie, nous avons proposé une nouvelle algèbre de processus, appelée MAPA. La syntaxe de MAPA comprend seulement trois opérateurs : le préfixage d'un processus par une multi-action, la composition parallèle de deux processus et la restriction d'un processus dans la communication avec son environnement.

La composition parallèle de deux processus est accompagnée d'une spécification explicite des canaux de communication entre les deux processus. De même, la restriction d'un processus est accompagnée d'une spécification explicite des canaux de communication entre le processus et son environnement.

Un canal caractérise le type des communications (des informations) qui peuvent être transportées, aussi bien d'un point de vue qualitatif que quantitatif. Il s'ensuit que MAPA permet de prendre en compte l'encodage de l'information circulant via les canaux de communications. Cet aspect n'a pas été développé ici et reste un point intéressant pour la recherche future.

En résumé, MAPA est un langage de description des systèmes réactifs à événements discrets simple et puissant (voir le Théorème III.25).

En ne considérant que les opérations "essentielles" et en évitant les opérations plus abstraites, comme le *choix distribué* ou le *pilotage*[1], nous obtenons un langage dans lequel même l'équivalence faible est une congruence.

A l'occasion de l'élaboration de la "vraie" sémantique de notre algèbre, nous avons proposé un nouveau schéma pour la définition de la relation d'équivalence entre les termes d'une algèbre de processus quelconque, en se basant sur l'idée de Hoare, mais en la formulant différemment.

[1] *Pilotage* est un des opérateurs de MEIJE, qui permet de synchroniser tout le comportement d'un agent, sans en préjuger de la nature.

Part IV

Conclusions et Perspectives

De façon générale, cette thèse traite de la spécification et de la modélisation de systèmes réactifs. Elle s'organise autour de deux thèmes principaux : les *primitives de communication*, à l'aide desquelles les systèmes interagissent, et les *opérations de base*, utilisées pour la composition de systèmes réactifs.

Ces deux "thèmes" recouvrent donc les deux axes de la spécification ou de la modélisation de systèmes réactifs, axes que nous avons appelés "vertical" et "horizontal". Nous avons considéré qu'une prise en compte de ces deux axes était nécessaire dans le cadre de la spécification ou de la modélisation de systèmes. Cette démarche se distingue des autres approches qui privilégient seulement l'une de ces deux lignes d'analyse possibles et complémentaires, à savoir les opérations de base.

Traditionnellement, les chercheurs se sont consacrés à l'étude des opérations de base en fixant dès le départ des primitives de communications de manière très intuitive, certes, mais aussi un peu ad-hoc. C'est pour cela, par exemple, que les mêmes symboles d'opérateurs de CCS et CSP, à savoir la composition parallèle ($|$) et la restriction/abstraction ($\backslash$), intuitivement très proches, ont des sémantiques très différentes dans les deux approches. A cause de cette différence dans l'intuition des actions primitives (dont la sémantique est volontairement ignorée), la comparaison entre les deux algèbres est très difficile.

A notre connaissance, c'est Glynn Winskel qui a proposé pour la première fois de séparer les primitives de communications et les opérateurs pour la composition des systèmes, dans le but de définir une sémantique des algèbres "CCS–like" (c'est-à-dire les algèbres de processus construites sur les opérateurs de CCS, mais dont les sémantiques respectives ont été rendues "paramétrables" par le choix de primitives de communications, formalisées par une "algèbre de synchronisation") en termes de *structures d'événements* [Win83, Win86]. Nous nous sommes beaucoup inspiré de ces travaux dans l'élaboration de ce que nous appelons "structure de synchronisation". A partir d'une structure de synchronisation qui décrivait (ou plutôt modélisait) des primitives de communication disponibles à un niveau d'abstraction du monde réel, nous avons élaboré une théorie, dite de *multi-actions*, où on peut construire (encoder) de nouvelles primitives de communication en combinant par une exécution simultanée des primitives (signaux) déjà existantes (voir Chapitre II.4 de la deuxième partie).

Une application des structures de synchronisation et de leurs propriétés algébriques est proposée dans le contexte des systèmes de transitions (Chapitre II.5 de la deuxième partie), où nous analysons comment on peut composer de tels systèmes, dont les transitions sont étiquetées par des éléments d'une structure de synchronisation ou par des ensembles de tels éléments, appelés *multi-actions*, et qui représentent l'exécution simultanée de tous ses éléments.

Notre objectif à court terme a été de rendre paramétrables les opérations de synchronisation et de restriction dans l'Algèbre des Petri-Box (PBC) par les primitives de communication, ce qui à son tour a permis de modéliser, par exemple, la lecture simultanée dans une variable partagée, ce qui jusqu'à présent dépasser le cadre de cette algèbre. Notre étude nous a amené à étendre les schémas de la synchronisation et de la restriction dans PBC (Chapitre II.6 de la deuxième partie).

Dans la dernière partie de la thèse, nous avons proposé une nouvelle algèbre de processus (appelée MAPA, en ang. *Multi-Action Process Algebra*), dont la syntaxe inclut seulement trois opérateurs : le *préfixage*, la *composition parallèle* et la *restriction*, et dont les actions sont des *multi-actions*. L'idée d'utiliser des *multi-actions*, c.-à-d. des actions composées d'un ensemble d'envois et/ou de réceptions simultanées de signaux simples, n'est pas nouvelle. Notre but était de montrer que l'impact de l'utilisation des multi-actions est considérable. Premièrement, nous avons montré que plusieurs résultats intéressants, comme la congruence de l'équivalence faible, peuvent être établis. Nous avons aussi montré qu'une telle algèbre, munie d'un nombre d'opérateurs réduit, possède pourtant la "puissance expressive la plus générale", dans le sens où tout *graphe de processus récursivement énumérable* peut y être décrit, modulo l'équivalence faible.

Outre de nombreux résultats, notre étude a aussi soulevé plusieurs questions qui pourraient faire l'objet de travaux ultérieurs. Dans cette perspective, nous pouvons mentionner les directions de recherche suivantes :

1. Exploration de la possibilité du raffinement des actions en utilisant les structures de synchronisation.

   En effet, rien n'empêche de considérer une structure de synchronisation dont les actions représentent une classe de processus. La relation de synchronisation d'une telle structure dénoterait les résultats possibles, eux-mêmes étant des processus, d'exécutions coopérantes des processus considérés.

2. Intégration des structures de synchronisation dans MAPA.

   Dans la même optique, il serait très intéressant d'intégrer les structures de synchronisation dans une algèbre de processus à la MAPA, avec les mêmes opérations, c.-à-d. le *préfixage*, la *composition parallèle* et la *restriction*. Un des avantages majeurs serait d'ajouter la possibilité d'intégrer le raffinement d'actions. Bien entendu, à cause du préfixage, les actions devraient être interprétées comme des processus de type très restreint et à comportement fini.

3. Elaboration d'une sémantique "concurrente" de MAPA en termes de systèmes de transitions localisés.

   Ce travail consisterait à appliquer les résultats généraux présentés dans [Fan95] à notre algèbre de processus MAPA.

4. Développement de l'approche de l'algèbre des relations dans la spécification de systèmes concurrents.

Dans la phase de rédaction de cette thèse, nous avons remarqué qu'une présentation en termes d'algèbre des relations, aussi bien pour la partie consacrée à la structure de synchronisation que pour la partie consacrée à MAPA, pourrait être envisagée. Même si une telle présentation n'implique pas forcément de résultats supplémentaires, elle semble, néanmoins, être intéressante, car elle rendrait les définitions plus compactes et plus uniformes.

Appendix A

Multi-ensembles

Soit Q un ensemble. Un multi-ensemble (ang. *bag, multi-set*) sur Q est une application totale de Q dans les entiers naturels `Nat`. Un multi-ensemble m est *fini* si pour presque tout élément x de Q on a $m(x) = 0$ (c.-à-d. que l'on a $m(x) = 0$, sauf pour une partie finie de Q). L'ensemble de tous les multi-ensembles sur Q est noté $\mu(Q)$ et sa partie finitaire par $\mu_f(Q)$.

Nous utiliserons la même notation pour les ensembles et les multi-ensembles, en considérant les ensembles comme des multi-ensembles, car tout sous-ensemble A de Q peut être vu comme un multi-ensemble sur Q :

$$A(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{si } x \text{ est dans } A \\ 0 & \text{sinon} \end{cases}$$

La notation habituelle pour les ensembles peut être utilisée et étendue. Par exemple $\{a, a, b\}$ dénote le multi-ensemble m avec $m(a) = 2$, $m(b) = 1$ et $m(x) = 0$ pour tout autre x de Q. Le multi-ensemble *vide*, dénoté par $\emptyset$ (ou $\{\}$), est défini par :

$$\emptyset(x) \stackrel{\text{def}}{=} 0, \text{ pour tout } x \text{ dans } Q$$

Bien évidemment la notation $\{x \in Q \mid \dots\}$ dénote toujours un sous-ensemble de Q.

Nous utilisons les opérations suivantes sur les multi-ensembles :

$$+ \ : \ (m + n)(x) \stackrel{\text{def}}{=} m(x) + n(x) \qquad \text{(multi-somme)}$$

$$\cup \ : \ (m \cup n)(x) \stackrel{\text{def}}{=} \max(m(x), n(x)) \qquad \text{(multi-union)}$$

$$\cap \ : \ (m \cap n)(x) \stackrel{\text{def}}{=} \min(m(x), n(x)) \qquad \text{(multi-intersection)}$$

$$\setminus \ : \ (m \setminus n)(x) \stackrel{\text{def}}{=} \max(m(x) - n(x), 0) \qquad \text{(multi-différence)}$$

avec en plus la relation d'inclusion et d'appartenance :

$$\in \ : \ x \in m \ \text{ ssi } m(x) > 0,$$

$$\subseteq \ : \ m \subseteq n \ \text{ ssi pour tout } x \text{ dans } Q,\ m(x) \leq n(x)$$

Soient A, B deux sous-ensembles de $\mu_f(Q)^k$, c.-à-d. deux ensembles de k-uplets de multi-ensembles finis sur Q. Nous définissons $A{+_k}B$ comme l'ensemble suivant de k-uplets d'éléments de $\mu_f(Q)$:

$$A +_k B \stackrel{\text{def}}{=} \{(m_1 + n_1, .., m_k + n_k) \mid (m_1, .., m_k) \in A, (n_1, .., n_k) \in B\}$$

Remarquons quelques propriétés utiles, bien qu'évidentes, de ces opérations. Soient A, B, C trois ensembles de k-uplets de multi-ensembles finis sur Q ($A, B, C \subseteq \mu_f(Q)^k$).

1. $A +_k B = B +_k A$ commutativité
2. $(A +_k B) +_k C = A +_k (B +_k C)$ associativité
3. $(A \cup B) +_k C = (A +_k C) \cup (B +_k C)$ distributivité par rapport à l'union

Bibliography

[AB84] Didier Austry and Gérard Boudol. Algèbre de processus et synchronisation. *Theoretical Computer Science*, 30:91–131, 1984.

[AD93] André Arnold and A. Dicky. Equivalences and preorders of transition systems. In *MFCS'93*, volume 711 of *LNCS*. Springer-Verlag, 1993.

[Arn92] André Arnold. *Systèmes de transitions finis et sémantique des processus communicants*. Masson, 1992.

[BB90] Gérard Berry and Gérard Boudol. The Chemical Abstract Machine. In *POPL*, pages 81–94. ACM, 1990.

[BB93] Jos C.M. Baeten and Jan A. Bergstra. Non interleaving process algebra. In *CONCUR'93*, volume 715 of *LNCS*. Springer-Verlag, 1993.

[BBK87a] Jos C.M. Baeten, Jan A. Bergstra, and Jan W. Klop. Conditional axioms and α/β–calculus in process algebra. In M. Wirsing, editor, *IFIP, Conference on Formal Description of Programming Concepts – III*, Ebberup 1986, 1987. North–Holland.

[BBK87b] Jos C.M. Baeten, Jan A. Bergstra, and Jan W. Klop. On the consistency of Koomen's Fair Abstraction Rule. *Theoretical Computer Science*, 51:129–176, 1987.

[BBP94] Jan A. Bergstra, Inge Bethke, and Alban Ponse. Process algebra with iteration and nesting. *The Computer Journal*, 37(4), 1994.

[BC87] Gérard Boudol and Ilaria Castellani. On the semantics of concurrency: Partial orders and transition systems. In *TAPSOFT'87*, volume 249 of *LNCS*. Springer-Verlag, 1987.

[BC88] Gérard Boudol and Ilaria Castellani. A non-interleaving semantics for CCS based on proved transitions. Rapport de Recherche 919, INRIA – Sophia Antipolis, France, 1988.

[BD87] Eike Best and Raymond Devillers. Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*, 55:87–136, 1987.

[BDE93] Eike Best, Raymond Devillers, and Javier Esparza. General refinement and recursion operators for the Petri Box Calculus. In *STACS'93*, volume 665 of *LNCS*. Springer-Verlag, 1993.

[BDG95] Marco Bernardo, Lorenzo Donatiello, and Roberto Gorrieri. MPA: a stochastic process algebra. Technical Report UBLCS-94-10, Università di Bologna, Italy, 1995.

[BDH92] Eike Best, Raymond Devillers, and Jon Hall. The Box Calculus: a new causal algebra with multi-label communication. In *Advances in Petri Nets 1992*, volume 609 of *LNCS*. Springer-Verlag, 1992.

[Bed88] Marek A. Bednarczyk. *Categories of Asynchronous Transition Systems*. PhD thesis, University of Sussex, 1988.

[Ber86] Gérard Berthelot. Transformations and decompositions of nets. In *Petri Nets: Central Models and Their Properties*, volume 254 of *LNCS*. Springer-Verlag, 1986.

[Bes86] Eike Best. COSY: Its relation to nets and to CSP. In *Petri Nets: Applications and Relationships to Other Models od Concurrency*, volume 255. Springer-Verlag, 1986.

[BF95] Eike Best and Hans Fleischhack, editors. *PEP: Programming Environment Based on Petri Nets*, volume 14/95 of *Hildesheimer Informatik–Berichte*. Universität Hildesheim, 1995. Proceedings of the PEP Workshop, October 11–12, 1994, Humbold–Universität zu Berlin.

[BFF+95a] Eike Best, Hans Fleischhack, Wojciech Frączak, Richard Hopkins, Hanna Klaudel, and Elisabeth Pelz. A class of composable high level Petri nets. In *International Conference on Application and Theory of Petri Nets*, volume 935 of *LNCS*. Springer-Verlag, 1995.

[BFF+95b] Eike Best, Hans Fleischhack, Wojciech Frączak, Richard Hopkins, Hanna Klaudel, and Elisabeth Pelz. A high level Petri net semantics of $B(PN)^2$. In *STRICT'95*, Workshop in Computing. Springer-Verlag, 1995.

[BH93] Eike Best and Richard Hopkins. $B(PN)^2$ – a Basic Petri Net Programming Notation. In *PARLE'93*, volume 694 of *LNCS*. Springer-Verlag, 1993.

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

[BIM88] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced: Preliminary report. In *POPL*, pages 229–239. ACM, 1988.

[BK84] Jan A. Bergstra and Jan W. Klop. Process algebra for synchronous communication. *Information and Control*, 60:109–137, 1984.

[BR85] Stephen D. Brookes and A.W. Roscoe. An improved failures model for communicating processes. In *Seminar on Concurrency*, volume 197 of *LNCS*. Springer-Verlag, 1985.

[Bra83] G.W. Brams. *Réseaux de Petri : Théorie et Analyse*. MASSON, 1983. Tome 1 et 2.

[BW90] Jos C.M. Baeten and W.P. Weijland. *Process Algebra*. Tracts in Theoretical Computer Science. Cambrige University Press, 1990.

[DD90] Philippe Darondeau and Pierpaolo Degano. Causal trees: Interleaving + causality. In *Semantics of Systems of Concurrent Processes*, volume 469 of *LNCS*. Springer-Verlag, 1990.

[Des90] Jörg Desel. Reduction and design of well-behaved concurrent systems. In *CONCUR'90*, volume 458 of *LNCS*. Springer-Verlag, 1990.

[Dev93] Raymond Devillers. On a more liberal synchronisation operator for the Petri Box Calculus. Technical Report LIT-281, Université Libre de Bruxelles, 1993.

[Dev94] Raymond Devillers. The synchronisation operator revisited for Petri Box Calculus. Technical Report LIT-290, Université Libre de Bruxelles, 1994.

[DK95] Raymond Devillers and Hanna Klaudel. Refinement and recursion in a high level Petri Box Calculus. In *STRICT'95*, Workshop in Computing. Springer-Verlag, 1995.

[dS84] Robert de Simone. *Calculabilité et Expressivité dans l'Algèbre de Processus Parallèles* MEIJE. Thèse de doctorat, Université Paris VII, 1984.

[Fan95] Jean Fanchon. Algebras of located and directed processes. Rapport de Recherche 987, Laboratoire de Recherche en Informatique, Orsay, France, 1995.

[FK94] Wojciech Frączak and Hanna Klaudel. A multi-action synchronization schema and its application to the PBC. In *Methodologies, techniques, and tools for design development*, volume 5 of *ESDA'94*. ASME, 1994.

[Frą95] Wojciech Frączak. Multi-action process algebra. In *Algorithms, Concurrency and Knowledge*, volume 1023 of *LNCS*, pages 126–140. Springer-Verlag, 1995.

[Gen91] H.J. Genrich. Predicate/transition nets. In Kurt Jensen and Grzegorz Rozenberg, editors, *High-level Petri Nets*. Springer-Verlag, 1991.

[GV92] Jan F. Groote and Frits W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100:202–260, 1992.

[Hen88] Matthew Hennessy. *Algebraic Theory of Processes*. The MIT Press, 1988.

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

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

[HP80] Matthew Hennessy and Gordon D. Plotkin. A term model for CCS. In *MFCS'80*, volume 88 of *LNCS*. Springer-Verlag, 1980.

[Jen86] Kurt Jensen. Colored Petri nets. In *Petri Nets: Central Models and Their Properties*, volume 254 of *LNCS*. Springer-Verlag, 1986.

[KEB94] Maciej Koutny, Havier Esparza, and Eike Best. Operational semantics for Petri Box Calculus. In *CONCUR'94*, volume 836 of *LNCS*. Springer-Verlag, 1994.

[Kla95] Hanna Klaudel. *Modèles algébriques, basés sur les réseaux de Petri, pour la sémantique des langages de programmation concurrents*. Thèse de doctorat, Université de Paris–Sud, 1995.

[Kou94] Maciej Koutny. Partial order semantics of Box Expressions. In *Application and Theory of Petri Nets*, volume 815 of *LNCS*. Springer-Verlag, 1994.

[KP95] Hanna Klaudel and Elisabeth Pelz. Communication as unification in the Petri Box Calculus. In *FCT'95*, volume 965 of *LNCS*. Springer-Verlag, 1995.

[Kri91] Padmanabhan Krishnan. Distributed CCS. In *CONCUR'91*, volume 527 of *LNCS*. Springer-Verlag, 1991.

[LTS79] P.E. Lauer, P.R. Torrigiani, and Mike W. Shields. COSY – A system specification language based on paths and processes. *Acta Informatica*, 12:109–158, 1979.

[Mil78] Robin Milner. Synthesis of communicating behaviour. In J. Winkowski, editor, *MFCS'78*, volume 64 of *LNCS*. Springer-Verlag, 1978.

[Mil80] Robin Milner. A Calculus of Communicating Systems. volume 92 of *LNCS*. Springer-Verlag, 1980.

[Mil83] Robin Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil85] George J. Milne. CIRCAL and the representation of communication, concurrency, and time. *ACM Transations on Programming Languages and Systems*, 7(2):270–298, 1985.

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

[Mil91] Robin Milner. The polyadic π-calculus: a tutorial. Technical report, Laboratory for Foundation of Computer Science, Computer Science Department, University of Edinburgh, UK, 1991.

[NH84] Rocco De Nicola and Matthew Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–134, 1984.

[Nic87] Rocco De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24:211–237, 1987.

[Par90] Joachim Parrow. The expressive power of parallelism. *Future Generation Computer Systems*, 6:271–285, 1990.

[Pel89] Elisabeth Pelz. About the concurrent behaviour of EN systems: Definabilty and closure results. In *Advances in Petri Nets*, volume 425 of *LNCS*. Springer-Verlag, 1989.

[Pet62] Carl Adam Petri. Kommunikation mit automaten. Schriften des IIM 2, Institut für Instrumentelle Mathematik, Bonn, 1962.

[Plo81] Gordon D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Pom86] Lucia Pomello. Some equivalence notions for concurrent systems - an overview. In *Advances in Petri Nets*, volume 222 of *LNCS*. Springer-Verlag, 1986.

[PRS92] Lucia Pomello, Grzegorz Rozenberg, and Carla Simone. A survey of equivalence notions for net based systems. In *Advances in Petri Nets*, volume 609 of *LNCS*. Springer-Verlag, 1992.

[Rei85] Wonfgang Reisig. *Petri nets*, volume 3 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.

[RT86] Grzegorz Rozenberg and P.S. Thiagarajan. Petri nets: basic notions, structure, behaviour. In *Current Trends in Concurrency*, volume 224 of *LNCS*. Springer-Verlag, 1986.

[Ruq93] Lu Ruqian. A true concurrency model of CCS semantics. *Theoretical Computer Science*, 113:231–258, 1993.

[RV83] Grzegorz Rozenberg and R. Verraedt. Subset languages of Petri nets II: Closure properties. *Theoretical Computer Science*, 27, 1983.

[Sta86] Eugene W. Stark. The computation category of a Concurrent Transition System. Technical Report 86/08, State University of New York at Stony Brook Computer Science Dept., 1986.

[Sta87] Eugene W. Stark. Concurrent transition system semantics of process networks. In *POPL*, pages 199–210. ACM, 1987.

[Tau89] Dirk Taubner. Finite representations of CCS and TCSP programs by automata and Petri nets. volume 369 of *LNCS*. Springer-Verlag, 1989.

[TV89] Dirk Taubner and Walter Vogler. Step failures semantics and a complete proof system. *Acta Informatica*, 27, 1989.

[Vaa92] Frits W. Vaandrager. Expressiveness results for process algebras. In *Semantics: Foundations and Applications*, volume 666 of *LNCS*. Springer-Verlag, 1992.

[Vau85] Jacques Vautherin. *Un modèle algébrique, basé sur les réseaux de Petri, pour l'étude des systèmes parallèles.* Thèse de doctorat, Université de Paris–Sud, 1985.

[Vau86] Jacques Vautherin. Parallel systems specification with colored Petri nets and algebraic abstract data types. In *Proceedings of the 7th European Workshop on Aplications and Theory of Petri Nets*, pages 5–25, Oxford, England, 1986.

[vG90] Rob J. van Glabbeek. The linear time – branching time spectrum. In *CONCUR'90*, volume 458 of *LNCS*. Springer-Verlag, 1990.

[vGV87] Rob J. van Glabbeek and Frits W. Vaandrager. Petri Net Models for Algebraic Theories of Concurrency. In *PARLE'87*, volume 259 of *LNCS*. Springer-Verlag, 1987.

[VNCG92] Guy Vidal-Naquet and Annie Choquet-Geniet. *Réseaux de Petri et systèmes parallèles.* Armand Colin, 1992.

[Vog91] Walter Vogler. Executions : A new partial order semantics of Petri nets. *Theoretical Computer Science*, 91:205–238, 1991.

[Vog92] Walter Vogler. Modular construction and partial order semantics of Petri nets. volume 625 of *LNCS*. Springer-Verlag, 1992.

[Win83] Glynn Winskel. Event structure semantics for CCS and related languages. Technical report, Computer Science Department, University of Aarhus, Denmark, 1983.

[Win86] Glynn Winskel. Event structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *LNCS*. Springer-Verlag, 1986.