Année 2002

THÈSE

présentée à

L'UNIVERSITÉ D'AIX-MARSEILLE I

au sein du Laboratoire d'Informatique Fondamentale de Marseille

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ D'AIX-MARSEILLE I

Spécialité : **Informatique**

Réceptivité, Mobilité et π-Calcul

par **Cédric LHOUSSAINE**

Soutenue le 28 juin 2002

Devant le jury composé de

Roberto AMADIO	Professeur à l'Université de Provence	(directeur de thèse)
Gérard BOUDOL	Directeur de Recherche à l'INRIA Sophia-Antipolis	(directeur de thèse)
Ilaria CASTELLANI	Chargée de Recherche à l'INRIA Sophia-Antipolis	(Examinateur)
Matthew HENNESSY	Professeur à l'Université de Sussex	(Rapporteur)
Jean-Jacques LÉVY	Directeur de Recherche à l'INRIA Rocquencourt	(Examinateur)
Vasco VASCONCELOS	Professeur Associé à l'Université de Lisbonne	(Rapporteur)

Table des matières

Table des figures

Remerciements

VOICI LA TRADITIONNELLE PAGE DES REMERCIEMENTS. Il y aurait beaucoup de monde à remercier mais n'étant pas très à l'aise dans cet exercice ils seront brefs !

En premier lieu je remercie Gérard et Roberto, mes directeurs de thèse sans lesquels j'en serais encore probablement au premier chapitre de ce manuscrit ! Merci à Matthew et Vasco pour avoir accepté d'être mes rapporteurs et à Jean-Jacques et Ilaria de participer à mon jury.

Merci à mes collègues et amis : Cécile, Éric, Fred, Gilles et Laurent pour avoir rendu agréables mes heures passées au laboratoire et pour leur grande sympathie.

Je tiens particulièrement à remercier mes parents pour m'avoir soutenu durant ces longues années d'études et pour la confiance qu'ils m'ont accordée.

Enfin, je remercie Sophie pour m'avoir supporté mais surtout pour m'avoir énormément soutenu pendant la période difficile qu'a été pour moi la rédaction de cette thèse.

CHAPITRE 1

Introduction

CES DERNIÈRES ANNÉES ONT ÉTÉ MARQUÉES PAR UNE RÉVOLUTION technologique : l'invention de la micro-électronique. L'ère qui s'en suivit, qualifiée par certains de « *l'ère numérique* », a vu l'émergence, dans toutes les sphères de la société, de l'informatique. Non seulement outil de production et de travail, mais également outil de création et de distraction, l'informatique est devenue un élément indispensable à énormément de secteurs d'activités, et le nombre de ses champs d'applications et de ses utilisateurs croît à une vitesse effrénée. Cette vitesse de développement est due à la conjonction de plusieurs facteurs. Un des plus significatifs est probablement les avancées purement technologiques réalisées récemment dans le domaine de la micro-électronique, et qui introduisent des machines toujours plus performantes. Mais un facteur certainement encore plus déterminant est de nature commerciale et industrielle. L'informatique permet de surenchérir sur de nouvelles sources de profits et un nouvel accroissement de la productivité.

Plus récemment, nous sommes entrés dans une nouvelle ère, « *l'ère de l'information* », avec l'essor et, dans une certaine mesure, la démocratisation de la *communication*. En effet, si on connecte et échange des informations entre plusieurs ordinateurs depuis les années 60, ce n'était possible que pour une minorité privilégiée d'universitaires et de militaires. Aujourd'hui, il existe non seulement des réseaux privés – ou locaux – ne comportant que quelques dizaines d'ordinateurs, mais aussi un réseau planétaire – *l'internet* – auquel a accès une multitude de protagonistes. L'internet, le réseau des réseaux, ne répond alors plus du tout aux mêmes besoins que dans les années 60. Il est le théâtre non seulement de coopérations mais aussi de rivalités et de conflits. Des données de tout type ainsi que des programmes naviguent d'un ordinateur à un autre parfois de manière totalement transparente aux utilisateurs. Il s'est donc rapidement révélé nécessaire de comprendre la complexité de ces échanges afin de mieux les contrôler et pouvoir vérifier que les logiciels réalisés pour les réseaux d'ordinateurs répondent aux attentes de leurs concepteurs. Divers modèles d'analyse ont été proposés comme l'évaluation de performances, l'algorithmique distribuée ou la fouille de données. Parmi celles-ci figurent des méthodes formelles, c'est-à-dire bien définies mathématiquement. Il devient alors possible de mo-

déliser, dans un langage non équivoque, les systèmes dits *concurrents* c'est-à-dire dans lesquels divers acteurs évoluent et inter-agissent. La description des spécifications des systèmes – de plus en plus préliminaire à leur réalisation – grâce à ces méthodes formelles, nous donne les moyens d'énoncer des propriétés et de les prouver.

Cette thèse tente d'apporter quelques contributions à ces recherches en s'intéressant à ses deux aspects : la modélisation des systèmes d'une part, et la vérification de propriétés grâce à ces modèles d'autre part.

Les modèles qui nous intéressent nous donnent la capacité de décrire l'évolution et le comportement des intéractions réalisables au sein de systèmes concurrents. On les appelle *calculs* – ou *algèbres* – *de processus* en ce sens qu'ils représentent des langages de programmation réduits à leur plus simple expression tout comme le λ-calcul représente le noyau de langages séquentiels tels que ML. Un des premiers calculs proposés fût CCS[1] [Mil80] dont l'opération atomique est la synchronisation, pour représenter la communication, et l'opérateur principal est la composition parallèle, pour permettre l'exécution concurrente. Aujourd'hui, un calcul nommé le *π-calcul*, une émanation directe de CCS, semble avoir fait l'unanimité dans la communauté scientifique et sert très souvent de référence en termes d'expressivité ou pour la spécification et la vérification de systèmes.

Le π-calcul, tout comme le λ-calcul « pur », permet d'exprimer un grand nombre de comportements y compris certains indésirables ! Pour les langages séquentiels, souvent un « mauvais » comportement est par exemple celui où un programme ne termine pas. Dans tous les langages de programmation, un mauvais comportement est celui où un programme utilise des opérateurs sans se conformer à leur sémantique ou de manière non cohérente. Dans ce dernier cas, l'introduction du concept de *type* a permis de restreindre des langages à ses « bons » programmes, ce sont les programmes bien typés. Pour les calculs de processus, le typage est également utilisé pour vérifier que la sémantique des opérateurs élémentaires est respectée. Cependant, les types ont trouvé de nouvelles fonctions et permettent aujourd'hui de restreindre par exemple le π-calcul à des fragments satisfaisant des comportements attendus. Par exemple, si l'émission de plusieurs messages à l'attention d'un seul récepteur semble naturelle, l'inverse, conduisant parfois à des résultats indéterminés, doit être rejeté. Un système de types, proposé dans [Ama00], permet de vérifier qu'un système n'aura jamais ce mauvais comportement.

Le π-calcul permet de rendre compte des interactions mais aussi d'une forme de *mobilité*. On entend ici par mobilité la capacité d'un système à reconfigurer dynamiquement la topologie de son réseau d'interconnexions. Celui-ci est représenté par les noms de canaux de communications et par les processus qui en ont connaissance. La mobilité s'exprime alors par l'échange de ces noms entre processus et donc par la création de nouvelles connexions. Cependant, cette forme de mobilité ne permet pas de rendre compte de l'existence de machines et de la migration de programmes entre celles-ci tels que dans les réseaux réels. Plusieurs raisons peuvent motiver la modélisation de machines et plus généralement de domaines où coexistent des processus. Par exemple, un domaine peut être une unité d'exécution (une machine physique) de sorte que sa défaillance entraîne la défaillance de tous les

[1] Calculus of Communicating Systems

processus qu'il héberge. Ou encore, un domaine peut être une unité de migration, c'est-à-dire un espace qu'un processus quitte ou investit pour, par exemple, accéder à des ressources. Là encore des modèles ont très récemment été proposés mais aucun d'eux ne semble encore avoir fait l'unanimité. L'un d'entre eux, le Dπ [HR98a], est directement issu du π-calcul et offre des constructions exprimant la migration et une communication purement locale : deux processus doivent être colocalisés pour pouvoir communiquer.

Un problème connu des systèmes concurrents est celui de l'*inter-blocage*. Il s'agit d'une situation dans laquelle un processus P détient une ressource r et attend l'accès à une ressource r' détenue par un processus P' se trouvant dans la situation inverse. Évidemment, aucun des deux processus ne pourra jamais continuer son exécution. Pour nombre d'algorithmes parallèles, l'absence d'inter-blocages a été prouvée, mais il n'existe pas de méthode formelle permettant de vérifier systématiquement cette propriété. La principale contribution de cette thèse est la proposition d'un outil d'analyse statique décidable qui vérifie une forme plus faible d'absence d'inter-blocage que nous appelons « *livrabilité des messages* ». On entend par là, que tout processus qui émet un message sur un canal a la garantie que ce message pourra être reçu. Ainsi, dans l'énoncé de l'inter-blocage donné plus haut, le processus P est assuré que la requête pour l'accès à la ressource r' aura bien été reçue par P'. Cependant, nous employons bien le terme « reçu » et non le terme « traité », dans le sens où P n'est pas pour autant garanti que P' libérera la ressource r'. En effet, ce dernier problème est très souvent intrinsèquement lié à l'application, et fournir un outil d'analyse statique répondant à cette question est très difficile, voir probablement impossible. Notre objectif est plus modeste, il s'agit de fournir au programmeur un outil l'avertissant qu'un code est susceptible de mener à une situation où un message ne sera pas reçu ce qui peut s'avérer ne pas toujours être évident. Ainsi, on force une discipline de programmation pour prendre en compte toutes les requêtes éventuelles, à charge au programmeur de les traiter « correctement ».

Cette thèse est organisée en trois parties. Dans la première, nous nous intéressons au problème de livrabilité des messages dans le cadre des systèmes concurrents en prenant pour modèle le π-calcul asynchrone. Nous commençons par dresser un panorama de l'état de la recherche dans le domaine des systèmes concurrents et des outils aujourd'hui disponibles pour leur étude. Ensuite, nous présentons le *π-calcul réceptif* et montrons qu'il satisfait la propriété de livrabilité des messages. Dans le quatrième chapitre, nous développons des techniques de preuves d'équivalence de processus pour prouver, dans le dernier chapitre, que le π-calcul réceptif préserve l'expressivité du π-calcul. Cette propriété est obtenue par un codage *complètement adéquat* de π_1 – le π-calcul avec récepteurs uniques évoqué plus haut – dans le π-calcul réceptif lui même avec récepteurs uniques.

Dans la deuxième partie nous étudions la livrabilité des messages dans le cadre étendu aux systèmes répartis où le concept de domaine est formalisé. Nous tentons d'analyser et de classifier quelques modèles de la distribution dans un premier temps. Ensuite, nous choisissons Dπ comme modèle de la distribution pour l'étude du problème de la livrabilité des messages. Nous en donnons une version réceptive et montrons que la propriété attendue est obtenue. Enfin, nous montrons, par une série d'exemples, un « style » de programmation réceptive que l'on peut adopter pour implanter des protocoles ou programmes à priori non réceptifs.

La dernière partie de cette thèse est consacrée à l'inférence de types pour le système de types décrit dans la partie précédente. En effet, la propriété de livrabilité des messages y est obtenue par la conjonction d'un système d'analyse statique simple clairement décidable, et d'un système de types dans lequel les types sont dits *dépendants*. Nous montrons qu'en présence de tels types l'inférence n'est pas triviale et nous introduisons le concept de *relations de liaison* pour le résoudre. L'algorithme que nous proposons consiste dans un premier temps en la génération de contraintes à partir du terme pour nous lequel désirons inférer un typage. Dans un deuxième temps, un algorithme d'unification résout ces contraintes (si celles-ci ont une solution). L'algorithme d'inférence de types obtenu donne un typage principal – c'est-à-dire un représentant de tous les typages possibles – pour tout terme typable.

Première partie

Le π-calcul réceptif

CHAPITRE 2

Algèbres de processus et typage

CE PREMIER CHAPITRE a pour objectif de donner une introduction aux algèbres de processus auxquels nous ferons référence dans cette thèse. La première section est naturellement dédiée à la présentation du calcul dont nous étudierons des propriétés et que nous étendrons par la suite : il s'agit du π-calcul asynchrone. Nous en donnerons brièvement la syntaxe et deux façons de décrire sa sémantique opérationnelle. Nous aborderons également un premier système de types simple ainsi que les notions d'équivalence de processus. Dans la seconde section, nous mentionnerons d'autres calculs de processus. Nous nous limiterons à donner leurs spécificités vis-à-vis du π-calcul asynchrone. Enfin, la dernière section est consacrée à un bref survol de quelques travaux effectués dans le domaine de l'analyse statique de processus. Plus particulièrement, nous nous intéresserons à la capture de propriétés par typage.

2.1 Le π-calcul asynchrone

Dans les calculs de processus l'opération de base la plus couramment rencontrée est la **communication**. Par opération de base nous attendons « pas de calcul élémentaire ou atomique ». On peut cependant distinguer dans la communication deux actions successives :

- la synchronisation de deux processus ;
- la transmission de données.

La première action consiste à mettre en « contact » deux protagonistes généralement par l'intermédiaire d'un canal. Un premier calcul à l'origine de celui qui nous intéresse dans cette section, a permis d'étudier les processus où la communication est réduite à cette étape de synchronisation, il s'agit de CCS ([Mil80, Mil89]). Dans celui-ci les canaux sont nommés (notés $a, b, \ldots$) et un processus faisant une requête de synchronisation sur un canal a s'écrit $\overline{a}.P$; après synchronisation ce processus déclenche la continuation P. Réciproquement, l'acceptation d'une synchronisation s'écrit $a.Q$, où Q est la continuation déclenchée après la synchronisation. Dans un tel calcul les actions de requête ($\overline{a}$) et

d'acceptation (a) sont totalement symétriques en ce sens que remplacer les unes par les autres et vice versa n'affecte pas significativement le comportement d'un programme clos – *i.e.* sans interaction avec un environnement potentiel.

Dans CCS, la transmission de données n'est pas nécessaire. En effet, R. MILNER a montré qu'il est possible de réduire un CCS enrichi du **passage de valeurs** (ou *de noms*) sur les canaux au calcul avec simple synchronisation. Cette réduction consiste essentiellement à coder chaque réception d'une valeur $a(x).P$ en une somme $\sum_{v\in\mathcal{N}} a_v.[\![[v/x]P]\!]$ où $\mathcal{N}$ est l'ensemble des valeurs du calcul, $a_v \in \mathcal{N}$ et $[\![[v/x]P]\!]$ est le codage de P où la valeur v est substituée à la variable x. La somme $\sum_{i\in I} a_i.P_i$, aussi appelée *choix non-déterministe*, déclenche la continuation P_i sur synchronisation avec l'action $\overline{a}_i$. Il est clair que ce codage n'est réalisable dans CCS qu'avec sommes infinies puisque l'ensemble des noms est lui même infini. Avec sommes finies, c'est la combinaison du passage de valeurs avec un opérateur de création dynamique de noms qui a permis d'étendre significativement l'expressivité du calcul et a donné naissance au π-calcul. Une première preuve de cette expressivité est le codage du λ-calcul dans le π-calcul ([Mil90]). La transmission d'un nom de canal nouvellement créé permet au processus récepteur d'établir éventuellement une nouvelle communication sur ce canal et ainsi de modifier dynamiquement la topologie du réseau. C'est pourquoi on qualifie souvent le π-calcul de modèle pour les systèmes mobiles et communiquant ([MPW92, Mil99]). Dans ce modèle les données ne sont pas structurées, elles sont atomiques et se réduisent à des noms. Néanmoins, il peut s'avérer commode de pouvoir transmettre plusieurs données lors d'une même communication. Les canaux transmettent alors des tuples de noms ; cette extension porte le nom de **π-calcul polyadique** ([Mil93]).

Afin de pouvoir exprimer l'exécution concurrente de processus il est nécessaire d'avoir dans le langage un opérateurs de composition parallèle. D'autre part, la récursion et le branchement conditionnel permettent d'obtenir toute l'expressivité désirée pour un modèle de langage de programmation.

Nous donnons dans la section suivante la syntaxe et la sémantique opérationnelle du **π-calcul polyadique asynchrone** ([Bou92, HT91]). L'asynchronisme est simplement exprimé par le fait que l'envoie d'une donnée sur un canal (aussi appelé **message**) n'est pas bloquante pour le processus émetteur, ce qui intuitivement, implique qu'on ne peut pas contrôler le moment où un message sera effectivement consommé. Il s'agit en fait d'une simplification du π-calcul qui n'enlève rien à son expressivité en l'absence de sommes de processus gardés par des messages (voir [Pal97, Bou92, HT92, NP96]).

Syntaxe et sémantique opérationnelle

La syntaxe du langage est définie par la grammaire donnée dans la figure 2.1. On note $\mathcal{N}$ l'ensemble des noms de canaux (notés $a, b, c \ldots$) et $\mathcal{P}$ l'ensemble des identificateurs de processus (notés $A, B, \ldots$).

Dans les termes de ce langage, il y a trois constructions liantes pour les noms. La première concerne les récepteurs $a(b_1, \ldots, b_n).P$ où la portée des noms b_i (pour $1 \leq i \leq n$) est P. La restriction $(\nu a)P$ lie le nom a au terme P. Enfin, la déclaration de processus récursifs paramétrés $(\mathsf{rec}\, A(a_1, \ldots, a_n).P)$ lie l'identificateur de processus A et les noms a_i au processus P. On note $\mathit{bn}(P)$ l'ensemble des noms liés dans P, $\mathit{fn}(P)$ l'ensemble de ses noms libres et $\mathit{nm}(P)$ l'ensemble de tous ses noms (c'est-à-dire $\mathit{bn}(P) \cup \mathit{fn}(P)$). Les termes du langage sont identifiés modulo α-conversion.

Le processus $\mathbf{0}$ désigne le processus terminé. Le branchement conditionnel $[a = b]P, Q$ peut

$P, Q, R \ldots$	::=	$\overline{a}(b_1, \ldots, b_n)$	*messages*
	|	$a(b_1, \ldots, b_n).P$	*récepteurs*
	|	$P \mid Q$	*composition parallèle*
	|	$[a = b]P, Q$	*branchement conditionnel*
	|	$(\nu a)P$	*restriction*
	|	$T(a_1, \ldots, a_n)$	*instanciation de processus paramétré*
	|	$\mathbf{0}$	*inaction*
T	::=	A	*identificateur de processus*
	|	$(\mathsf{rec}\, A(a_1, \ldots, a_n).P)$	*processus récursif paramétré*

FIG. 2.1: Syntaxe du π-calcul polyadique asynchrone

se lire `if` $a = b$ `then` P `else` Q. Dans la déclaration des processus récursifs on suppose que les occurrences de A dans P sont gardées pour une réception. Intuitivement, le processus

$$(\mathsf{rec}\, A(a_1, \ldots, a_n).P)(b_1, \ldots, b_n)$$

exécute P en y remplaçant les occurrences des a_i par les b_i. Une occurrence d'un terme $A(c_1, \ldots, c_n)$ dans P est interprétée comme un appel récursif au processus paramétré cette fois par les c_i. Par exemple, le processus

$$(\mathsf{rec}\, A(a).(\nu r)(\overline{a}(r) \mid r().A(a)))(c)$$

envoie un nom r nouvellement créé et l'émet sur le canal c. Puis, un accusé de réception est attendu sur r avant de pouvoir émettre un nouveau nom sur c, et ainsi de suite.

La sémantique opérationnelle peut se décrire de deux façons : soit à l'aide d'un **système de transitions étiquetées** (utile pour la définition de relations de bisimulation), soit dans le style de la **machine chimique abstraite** ([BB90, Bou93]) qui offre une approche plus simple et plus intuitive du calcul. Nous ne donnons ici qu'une esquisse de la sémantique opérationnelle dans les deux styles. Nous utilisons la notation vectorielle ($\vec{a}$) pour les tuples de noms.

Dans le système de transitions étiquetées, les processus réalisent des **actions** pour se transformer en un nouveau processus, ce qui est noté $P \xrightarrow{\alpha} P'$. L'ensemble des actions α est donné par la grammaire suivante :

$$\alpha ::= \tau \mid a\vec{b} \mid (\nu\vec{c})\overline{a}\vec{b}$$

L'action τ exprime une communication interne à P, $a\vec{b}$ est la réception par P des noms $\vec{b}$ sur le canal a, et $(\nu\vec{c})\overline{a}\vec{b}$ l'émission par P sur le canal a des noms $\vec{b}$, certains pouvant être privés à P et spécifiés par $\vec{c}$. Pour cette dernière action, on suppose que $a \notin \{\vec{c}\} \subseteq \{\vec{b}\}$. On dénote par $\mathit{nm}(\alpha)$ (resp. $\mathit{fn}(\alpha)$ at $\mathit{bn}(\alpha)$) l'ensemble des noms (resp. libres et liés) de l'action α. Par exemple, $\mathit{fn}((\nu\vec{c})\overline{a}\vec{b}) = \{a, \vec{b}\} - \{\vec{c}\}$. Dans les actions de réception et d'émission, a est dit *sujet* de α (noté $\mathit{subj}(\alpha)$).

$$(out)\quad \frac{}{\overline{a}(\vec{b}) \xrightarrow{\overline{a}\vec{b}} \mathbf{0}} \qquad (in)\quad \frac{}{a(\vec{b}).P \xrightarrow{a\vec{c}} [\vec{c}/\vec{b}]P}$$

$$(ext)\quad \frac{P \xrightarrow{\alpha} P'}{(\nu a)P \xrightarrow{(\nu a)\alpha} P'} \; a \neq \mathit{subj}(\alpha) \qquad (\nu)\quad \frac{P \xrightarrow{\alpha} P'}{(\nu a)P \xrightarrow{\alpha} (\nu a)P'} \; a \notin \mathit{nm}(\alpha)$$

$$(cm)\quad \frac{P \xrightarrow{(\nu\vec{c})\overline{a}\vec{b}} P' \;,\; Q \xrightarrow{a\vec{b}} Q'}{P \mid Q \xrightarrow{\tau} (\nu\vec{c})(P' \mid Q')} \; \{\vec{c}\} \cap \mathit{fn}(Q) = \emptyset$$

FIG. 2.2: Quelques règles du système de transitions étiquetées

La figure 2.2 donne quelques règles du système de transitions étiquetées dans le style de la liaison précoce – ou encore dit *early* (voir par exemple [Qua99] pour les styles dits *late* et *open*). La règle (cm) indique que si un terme P peut envoyer un message (par une action $(\nu\vec{c})\overline{a}\vec{b}$) et Q peut le recevoir (par l'action duale $a\vec{b}$) alors la composition parallèle de P et Q effectue une action interne (action τ). Puisque P a communiqué des noms privés $\vec{c}$, le processus résultant de la réception (Q') est dans la confidence de ces noms, ce qui explique la restriction dans $(\nu\vec{c})(P' \mid Q')$. Une des règles non reportées dans la figure est le symétrique de (cm), c'est-à-dire quand c'est Q qui émet et P qui reçoit.

Exemple 2.1.1 Les règles données sont suffisantes pour montrer que la transition

$$((\nu b)\overline{a}(b) \mid a(c).\overline{c}) \xrightarrow{\tau} (\nu b)(\mathbf{0} \mid \overline{b})$$

est correcte. Il suffit pour cela de montrer que $(\nu b)\overline{a}(b) \xrightarrow{(\nu b)\overline{a}b} \mathbf{0}$ par les règles (*ext*) et (*out*), et que $a(c).\overline{c} \xrightarrow{ab} \overline{b}$ par la règle (*in*). □

La sémantique opérationnelle dans le style de la machine chimique abstraite (ou CHAM) est décrite à l'aide d'une relation de réduction (non étiquetée) et d'une relation d'équivalence structurelle. Cette dernière par exemple, établie la commutativité, l'associativité et l'élément neutre $\mathbf{0}$ pour la composition parallèle. L'équivalence la plus importante est certainement celle permettant l'extension de la portée d'une restriction :

$$((\nu a)P \mid Q) \;\equiv\; (\nu a)(P \mid Q) \qquad \textit{extension de portée}$$

sous la condition que a ne soit pas libre dans Q. La relation de réduction notée $\rightarrow$ est définie par les axiomes de la figure 2.3, modulo équivalence structurelle et éventuellement sous contexte :

$$\frac{P \equiv P' \;,\; P' \rightarrow Q' \;,\; Q' \equiv Q}{P \rightarrow Q} \qquad \frac{P \rightarrow Q}{(\nu a)P \rightarrow (\nu a)Q} \qquad \frac{P \rightarrow Q}{P \mid R \rightarrow Q \mid R}$$

Exemple 2.1.2 La transition de l'exemple précédent peut être réalisée par la réduction suivante :

$$((\nu b)\overline{a}(b) \mid a(c).\overline{c}) \rightarrow (\nu b)(\overline{b})$$

$(\overline{a}(\vec{b}) \mid a(\vec{c}).P)$	$\rightarrow$	$[\vec{c}/\vec{b}]P$	*communication*
$(\mathsf{rec}\, A(\vec{a}).P)(\vec{b})$	$\rightarrow$	$[(\mathsf{rec}\, A(\vec{a}).P)/A][\vec{b}/\vec{c}]P$	*dépliage*
$[a = a]P, Q$	$\rightarrow$	P	*branchement-vrai*
$[a = b]P, Q$	$\rightarrow$	$Q \quad a \neq b$	*branchement-faux*

FIG. 2.3: Règle de réduction CHAM

puisque $((\nu b)\overline{a}(b) \mid a(c).\overline{c}) \equiv (\nu b)(\overline{a}(b) \mid a(c).\overline{c})$ (par extension de portée) et $(\overline{a}(b) \mid a(c).\overline{c}) \rightarrow (\nu b)(\overline{b})$ (par l'axiome de communication). □

Sorting

Dans le π-calcul polyadique il est possible d'avoir des «erreurs» de communication. Ces erreurs ont lieu lorsqu'une transmission ne peut être réalisée à cause d'une mésentente sur l'arité du canal de communication entre le message et le processus récepteur (par exemple : $\overline{a}(b) \mid a(c_1, c_2).P$ où a est unaire pour le message et binaire pour la réception). Il est possible de vérifier statiquement qu'une telle erreur ne se produira jamais au cours de l'exécution d'un processus. Cette analyse statique est le premier « **système de type** » pour le π-calcul (encore appelé *sorting*). Il s'agit d'attribuer à chaque nom de canal a d'un processus une **sorte** comportant l'arité du canal et la sorte des noms transmis sur celui-ci. On peut définir les sortes par la grammaire suivante :

$$s ::= \langle s_1, \ldots, s_n \rangle \mid \langle\rangle$$

Il est ensuite possible de vérifier statiquement qu'un processus utilise les canaux en respectant leurs sortes. Une telle analyse peut se décrire au moyen par un système d'inférence simple dans lequel on prouve des jugements $S \vdash P$ où S (appelé *sortage*) attribue à chaque nom libre de P une sorte et signifiant « P est bien sorté avec le sortage S ». Des exemples de règles d'inférence sont :

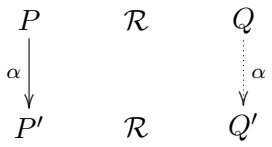
$$\frac{S(a) = \langle s_1, \ldots, s_n \rangle \ , \ S(b_i) = s_i}{S \vdash \overline{a}(\vec{b})} \qquad \frac{S \vdash P \ , \ S \vdash Q}{S \vdash P \mid Q} \qquad \frac{S \uplus \{b_i \mapsto s_i\} \vdash P \quad S(a) = \langle s_1, \ldots, s_n \rangle}{S \vdash a(\vec{b}).P}$$

Ces règles suffisent à montrer que par exemple le terme $(b.\mathbf{0} \mid \overline{a}(b, c) \mid a(d, e).\overline{d})$ est bien sorté avec le sortage $\{b \mapsto \langle\rangle, c \mapsto s, a \mapsto \langle\langle\rangle, s\rangle\}$ où s est une sorte quelconque.

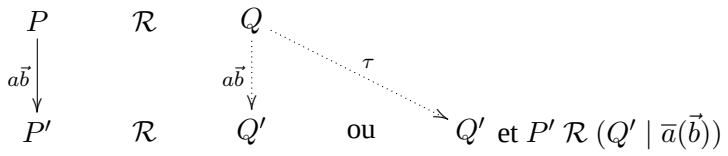
Équivalence

Nous avons déjà rencontré une équivalence de processus lors de la définition de la sémantique opérationnelle dans le style CHAM : l'équivalence structurelle. Cependant, celle-ci est une équivalence trop forte pour pouvoir, par exemple, identifier les processus $\mathbf{0}$ et $(\nu b)\overline{b}$. En effet, l'équivalence structurelle permet essentiellement de réécrire un terme afin d'y faire apparaître des redex. Intuitivement, une équivalence de processus doit identifier des programmes qui ne peuvent être distingués par aucun contexte. En d'autres termes, des processus sont équivalents s'ils sont capables de se simuler l'un l'autre, c'est-à-dire de pouvoir effectuer, du point de vue de l'environnement, les mêmes actions

au même moment. Ainsi énoncée, il est naturelle que l'équivalence doit identifier les processus **0** et $(\nu b)\overline{b}$ puisque ceux-ci ne peuvent effectuer aucune action. C'est pourquoi la **bisimulation** est la notion d'équivalence qui a été choisie pour les calculs de processus. Une relation de bisimulation $\mathcal{R}$ peut se résumer par la commutation du diagramme suivant :



qui doit être lu ainsi : si P et Q sont en relation et P effectue une action et devient P', alors il existe Q' tel que Q fait la même action que P et devient Q'. De plus, P' et Q' restent en relation. Ici, la bisimulation est dite *forte*. Dans le cas *faible*, on ne prend pas compte les actions τ précédant ou succédant à une action de réception ou d'émission, c'est-à-dire qu'on n'observe pas les actions internes aux processus pour les distinguer. Si cette bisimulation est satisfaisante pour le π-calcul synchrone, elle ne l'est plus dans le cas asynchrone car elle ne respecte pas l'intuition selon laquelle la réception d'un message peut se produire bien après son émission et qu'on ne peut observer le moment où ce message sera effectivement consommé. Alors que dans la bisimulation décrite précédemment, on requiert lors de l'observation d'une réception sur P qu'un processus équivalent Q doit réaliser au même moment la même réception. La bisimulation asynchrone est donc une adaptation de la précédente où la réception d'un message par P peut être différée pour Q, ce dernier réalisant à la place une action interne. Le diagramme suivant illustre la bisimulation asynchrone pour l'action de réception ; pour les autres actions le diagramme reste identique au précédent :



Plus formellement, la définition de bisimulation asynchrone est la suivante :

Définition 2.1.3 *Une relation $\mathcal{R}$ sur les processus est une relation de* bisimulation asynchrone *si elle est symétrique, et si $P\mathcal{R}Q$ implique :*

1. *si $P \xrightarrow{\tau} P'$ alors il existe Q' tel que $Q \xrightarrow{\tau} Q'$ et $P'\mathcal{R}Q'$;*
2. *si $P \xrightarrow{(\nu\vec{c})\overline{a}\vec{b}} P'$ et $\{\vec{c}\} \cap fn(Q) = \emptyset$ alors il existe Q' tel que $Q \xrightarrow{(\nu\vec{c})\overline{a}\vec{b}} Q'$ et $P'\mathcal{R}Q'$;*
3. *si $P \xrightarrow{a\vec{b}} P'$ alors il existe Q' tel que soit $Q \xrightarrow{a\vec{b}} Q'$ et $P'\mathcal{R}Q'$, soit $Q \xrightarrow{\tau} Q'$ et $P'\mathcal{R}(Q' \mid \overline{a}(\vec{b}))$*

On note $\sim$ la plus grande relation de bisimulation asynchrone.

Il existe d'autres définitions équivalentes de bisimulations asynchrones et étudiées par exemples dans [ACS98]. Dans le chapitre 4 nous irons plus loin dans l'étude de la bisimulation asynchrone en nous penchant sur les techniques de preuves.

2.2 D'autres calculs

Dans cette section, nous présentons brièvement deux autres calculs de processus auxquels nous faisons référence dans la suite de cette thèse. Le premier est le **Join-calcul** ([FG96, Fou98]), le second est le **calcul bleu** ([Bou97a, Dal99]). Tous deux différent syntaxiquement du π-calcul mais aussi sémantiquement, et ce notamment dans la gestion du processus de communication. Nous donnons quelques éléments syntaxiques pour chacun de ces langages ainsi que les principales caractéristiques les différenciant du π-calcul.

Le Join-calcul

Le Join-calcul ([FG96, Fou98]) a été conçu, selon leurs auteurs, dans l'optique d'une implantation et d'une extension répartie. Cette dernière prend en compte les notions d'agent, de localisation et de mobilité. Nous reviendrons en début de deuxième partie sur cette extension.

Comme le π-calcul, le Join-calcul est un calcul du premier ordre : seuls des noms peuvent être communiqués. Les messages $a\langle\vec{b}\rangle$ sont asynchrones et les canaux polyadiques. Dans la construction syntaxique $\mathsf{def}\ D\ \mathsf{in}\ P$, D est un ensemble de *définitions* de récepteurs et P le processus autorisé à les solliciter. Un récepteur est une *règle de réaction* $J \triangleright P$ où P est déclenché si le *filtre* J est sollicité. Un filtre est une composition de messages requis ($a_1\langle\vec{b_1}\rangle \mid \ldots \mid a_n\langle\vec{b_n}\rangle$) pour l'activation d'une continuation. Une première différence avec le π-calcul est qu'il n'y a dans le Join-calcul qu'une seule construction liante : la définition. Elle réunie l'abstraction et la restriction du π-calcul. Par exemple dans le terme :

$$P \mid (\mathsf{def}\ a\langle b\rangle \triangleright b\langle\rangle\ \mathsf{in}\ Q)$$

le processus P ne peut pas invoquer le récepteur $a\langle b\rangle \triangleright b\langle\rangle$ car la définition a pour effet de créer un nouveau nom a qui est donc lié dans Q. On a alors une définition statique de tous les récepteurs. Cette caractéristique syntaxique implique également que la communication d'un nom ne fait que communiquer la capacité d'émettre sur ce nom à la différence du π-calcul où un nom reçu peut être utilisé pour définir un nouveau récepteur. D'autre part, la définition est persistante au cours du calcul, c'est-à-dire qu'après invocation d'un récepteur, celui-ci est répliqué pour à nouveau être éventuellement sollicité. Un autre effet de ces définitions est que le Join-calcul ne calcule jamais sur des noms libres. Enfin, alors que le π-calcul n'autorise la réception que d'un message à la fois, l'utilisation de filtres dans le Join-calcul permet la requête de plusieurs messages avant le déclenchement d'une continuation.

Le calcul bleu

Alors que dans le π-calcul ([Bou97a, Dal99]) et dans le Join-calcul la communication est une opération atomique, le calcul bleu la décompose en ses deux actions effectives : la synchronisation et la transmission. Nous retrouvons les deux types d'acteurs des calculs asynchrones : les récepteurs (ou ressources notées $\langle a \Leftarrow P\rangle$) et les messages (notés $a\vec{b}$). À cela s'ajoutent des constructions fonctionnelles qui font du calcul bleu un calcul d'ordre supérieur : l'*abstraction* $\lambda a.P$ et l'*application* Pa. La synchronisation est réalisée par la réduction suivante :

$$\langle a \Leftarrow P\rangle \mid a\vec{b} \quad \rightarrow \quad P\vec{b}$$

où, de l'interaction de la ressource P, nommée a, et du message pour a comportant les données $\vec{b}$, résulte l'application de P aux données $\vec{b}$. La transmission est la β-réduction usuelle du λ-calcul mais réduite à l'application à des noms uniquement :

$$(\lambda a.P)b \quad \rightarrow \quad [b/a]P$$

Le terme du π-calcul $a(c).\bar{c} \mid \overline{a}(b)$ se réduisant en $\overline{b}$ en une seule étape de réduction, peut s'écrire dans le calcul bleu $\langle a \Leftarrow \lambda c.c\rangle \mid ab$ et se réduire en b par les deux réductions successives :

$$\langle a \Leftarrow \lambda c.c\rangle \mid ab \rightarrow (\lambda c.c)b \rightarrow b$$

La stratégie d'évaluation des λ-termes est, par construction syntaxique, l'appel par nom. Les deux lieurs du π-calcul sont présents : l'abstraction (par λ-abstraction) et la restriction (également notée $(\nu a)P$). La réplication des ressources $\langle a = P\rangle$ existe dans le langage et est interprétée comme la composition parallèle d'un nombre infini de ressources $\langle a \Leftarrow P\rangle$.

2.3 Propriétés

Nous avons vu que le Join-calcul possède une syntaxe suffisamment contraignante pour assurer par exemple l'unicité des ressources nommées. Cependant, les calculs de processus présentés par la simple définition de leur syntaxe et de leur sémantique opérationnelle n'ont généralement pas pour objectif de capturer des propriétés attendues ou désirées pour la programmation de systèmes concurrents. En effet, il faut ce que ces calculs restent suffisamment expressifs et souples pour permettre l'étude du plus nombre de propriétés. Néanmoins, il est souvent possible de restreindre le langage par une analyse statique qui détermine si un terme possède la propriété désirée sans pour autant restreindre l'expressivité de ce langage. Par exemple, nous avons vu que le π-calcul polyadique possède un système de sortage (qui plus est décidable) assurant que tout terme bien sorté n'aura pas d'erreur d'arité en cours d'exécution. Une telle analyse statique peut tout de même parfois requérir un enrichissement syntaxique du langage en particulier s'il s'agit de typage. Nous rappelons ici quelques systèmes de types élaborés pour les calculs de processus. Tous ont pour objectif un contrôle plus fin de l'utilisation des canaux de communication.

Contrôle de la polarité

Dans [PS93], B. PIERCE et D. SANGIORGI ont introduit le concept de « direction dans les canaux de communication ». Il s'agit d'indiquer explicitement dans les termes du langage si un nom doit être utilisé uniquement en écriture (*i.e.* pour envoyer des messages), uniquement en lecture (*i.e.* pour recevoir des messages) ou les deux. Ils enrichissent pour cela les sortes du π-calcul polyadique avec les étiquettes w (écriture), r (lecture), et b (les deux). De plus, dans toute construction liante, un nom abstrait est agrémenté de sa sorte. Ainsi dans le terme

$$a(b\colon \langle\rangle^{\mathsf{w}}).P \mid \overline{a}(c)$$

P reçoit un canal qu'il ne peut utiliser qu'en écriture. Ce terme devrait être bien sorté avec le sortage

$$a\colon \langle\langle\rangle^{\mathsf{w}}\rangle^{\mathsf{b}}, c\colon \langle\rangle^{\mathsf{w}}$$

puisque a est utilisé à la fois en écriture et en lecture et qu'il communique un canal utilisé en lecture. Cependant, il serait souhaitable que ce terme soit également bien sorté avec le sortage

$$a\colon \langle\langle\rangle^{\mathsf{w}}\rangle^{\mathsf{b}}, c\colon \langle\rangle^{\mathsf{b}}$$

du moment que P n'utilise c qu'en écriture. Ceci est permis par la définition d'une relation de sous-typage. Cette relation sur les sortes (notée $\leq$) stipule qu'un message $\overline{a}(b)$ est bien sorté si $s_b \leq s_a$ où s_b est la sorte de b et a a une sorte $\langle s_a\rangle^I$ (où I est b ou w). De manière similaire, une réception $a(b\colon s)$ est bien sortée si $s_a \leq s$ où a a une sorte $\langle s_a\rangle^I$ (où I est b ou r). Intuitivement, si $s \leq s'$ alors un nom de sorte s peut librement être utilisé dans un contexte où on attend un nom de sorte s'. Sur les étiquettes cette intuition se traduit par les relations $\mathsf{b} \leq \mathsf{w}$ et $\mathsf{b} \leq \mathsf{r}$.

Contrôle de la multiplicité

Aux indications de polarité des canaux, il est possible d'ajouter une information supplémentaire concernant la multiplicité de leur usage. C'est ce qui est fait par exemple dans [KPT96]. Le type d'un nom a permet alors de savoir s'il est utilisé une seule fois (1) ou un nombre illimité de fois (ω) et dans quelle direction. Ainsi, par exemple, $\langle\langle\rangle_1^{\mathsf{w}}\rangle_\omega^{\mathsf{b}}$ est le type des canaux pouvant être utilisés en lecture et en écriture plusieurs fois pour communiquer des noms utilisables une seule fois en écriture. Ce type peut donc être attribué au nom c dans le terme

$$(\mathsf{rec}\, A(a).a(b).\overline{b} \mid A(a))(c) \mid \overline{c}(d)$$

si d a le type $\langle\rangle_1^{\mathsf{w}}$. Dans [IK00], les auteurs raffinent ce système de type avec une relation de sous-typage permettant ainsi de donner, dans l'exemple précédent, le type $\langle\rangle_w^{\mathsf{b}}$ à d avec la relation $\omega \leq 1$.

Contrôle de la séquentialité

Des systèmes de types plus élaborés permettent de rendre compte de l'ordonnancement de la consommation des ressources. Dans [Bou97b], pour ce faire G. BOUDOL utilise des types dépendants (des noms de ressources) et s'inspire de la Logique Unifiée ([Gir93]) pour attribuer des types aux processus du calcul bleu. Ainsi, si P est un terme fonctionnel de type $\sigma \to \tau$, la ressource $\langle r \Leftarrow P\rangle$ a le type $\langle r\rangle\sigma \to \tau$ pour indiquer que la synchronisation avec la ressource r précède la disponibilité d'une fonction de type $\sigma \to \tau$. D'après l'auteur ce système de types garantirait l'absence de requêtes à des ressources inexistantes.

Dans [Kob98], N. KOBAYASHI rend encore plus explicite l'ordonnancement de l'utilisation des canaux en agrémentant les sortes avec des étiquettes partiellement ordonnées. Par exemple, pour le processus $y().\overline{x}$, y a le type $\langle\rangle^{t_y}$ et x le type $\langle\rangle^{t_x}$ avec l'ordonnancement $t_y \leq t_x$ indiquant que les canaux dont le type est étiqueté par t_y sont utilisés avant ceux dont le type est étiqueté par t_x.

La réceptivité uniforme

D. SANGIORGI dans [San99] (voir aussi le chapitre 8.2 de [SW01]) introduit la notion de *réceptivité* des canaux de communication. Intuitivement, la réceptivité d'un canal (en entrée) est sa capacité

à être immédiatement disponible pour une réception sur ce canal. Ainsi, tout message sur un canal réceptif doit pouvoir être directement consommé par un processus. Cette propriété est indécidable mais peut néanmoins être capturée par des systèmes de types. D. SANGIORGI en étudie deux pour deux paradigmes de réceptivité.

Le premier paradigme est celui de la *linéarité*. Un canal linéairement réceptif n'est exploitable qu'une seule fois en tant que sujet d'une communication. Le système de type assurant la réceptivité linéaire vérifie qu'un nom nouvellement créé est disponible immédiatement pour une réception et qu'il est utilisé au plus une fois comme sujet d'une émission. D'autre part, il impose que la continuation P d'un processus gardé $a(b).P$ ne contient pas de réception sur un canal réceptif ; et, si b est réceptif alors dans P ce canal b ne peut être utilisé que pour émettre un message. Cette dernière caractéristique implique la nécessité d'avoir un seul processus recevant des messages sur un canal réceptif donné (caractéristique inhérente au Join-calcul).

Réciproquement, le deuxième système de types assure ce que D. SANGIORGI appelle la ω-*réceptivité*. Ici, les canaux dits ω-réceptifs sont *persistants* et *uniformes*. La persistance garantie que toutes les émissions de messages sur un canal ω-réceptif seront immédiatement et toujours consommées quelque soit le nombre de ces messages. L'uniformité quant à elle, garantie que toutes les continuations des processus gardés par un même canal ω-réceptif sont syntaxiquement identiques. Ces deux caractéristiques sont principalement obtenues en imposant que la seule occurrence d'un processus gardé par une entrée sur un canal ω-réceptif soit répliquée.

CHAPITRE 3

Le π-calcul réceptif

DANS CE CHAPITRE nous présentons le calcul de processus qui nous servira de base tout au long de cette thèse. Les termes de ce langage sont essentiellement des termes du π-calcul polyadique asynchrone ayant la propriété de **réceptivité**. C'est pourquoi nous appellerons ce langage π^r. Celui-ci est obtenu par une analyse statique des termes du π-calcul décrite par la conjonction d'un système d'inférence de types et d'un système de bonne formation des termes. Nous verrons également qu'il est facile de capturer un sous-ensemble de π^r comportant uniquement les processus ayant la propriété d'**unicité des récepteurs** ; ce dernier calcul sera appelé π_1^r.

Ce chapitre est organisé de la manière suivante. La première section donne la syntaxe du langage et la sémantique opérationnelle décrite à l'aide d'un système de transitions étiquetées. La seconde section présente le système d'inférence de types, la troisième formalise la notion de réceptivité et donne le système de bonne formation des termes. Enfin, la dernière section établit les résultats de réceptivité et de **livrabilité des messages**.

3.1 Syntaxe et sémantique opérationnelle

La syntaxe de π^r diffère légèrement de celle du π-calcul asynchrone présentée dans le deuxième chapitre par l'utilisation de valeurs, c'est-à-dire de noms utilisés exclusivement pour les tests d'égalité (ou branchements conditionnels)[1]. Pour se faire, la restriction doit préciser si le nom créé est une valeur ou un canal. On considère un ensemble infini dénombrable $\mathcal{N}$ de noms (notés $a, b, c \ldots, x, y, z, \ldots$), et un ensemble infini dénombrable $\mathcal{P}$ d'identificateurs de processus (notés $A, B \ldots$). Ces derniers sont paramétrés par des noms et un appel récursif est noté $A(\vec{a})$.

[1] Notons que le test d'égalité est nécessaire au codage de π_1 et que le test sur des valeurs est suffisant. L'extension aux tests sur les noms n'aurait pas posé de problème.

(*out*) $\dfrac{}{\overline{a}(\vec{b}) \xrightarrow{\overline{a}\vec{b}} \mathbf{0}}$ (*in*) $\dfrac{}{a(\vec{b}).P \xrightarrow{a\vec{c}} [\vec{c}/\vec{b}]P}$

(*ext*) $\dfrac{P \xrightarrow{\alpha} P'}{(\nu w)P \xrightarrow{(\nu w)\alpha} P'}\ \mathit{subj}(w) \neq \mathit{subj}(\alpha)$ (ν) $\dfrac{P \xrightarrow{\alpha} P'}{(\nu w)P \xrightarrow{\alpha} (\nu w)P'}\ \mathit{subj}(w) \notin \mathit{nm}(\alpha)$

(*cm*) $\dfrac{P \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} P' \ ,\ Q \xrightarrow{a\vec{b}} Q'}{P \mid Q \xrightarrow{\tau} (\nu\vec{w})(P' \mid Q')}\ \mathit{subj}(\vec{w}) \cap \mathit{fn}(Q) = \emptyset$ (*cm'*) *(symétrique)*

(*cp*) $\dfrac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}\ \mathit{bn}(\alpha) \cap \mathit{fn}(Q) = \emptyset$ (*cp'*) *(symétrique)*

(m_t) $\dfrac{P \xrightarrow{\alpha} P'}{[a = a]P, Q \xrightarrow{\alpha} P'}$ (m_f) $\dfrac{Q \xrightarrow{\alpha} Q'}{[a = b]P, Q \xrightarrow{\alpha} Q'}\ a \neq b$

(*rec*) $\dfrac{[(\mathsf{rec}\, A(\vec{b}).P)/A][\vec{c}/\vec{b}]P \xrightarrow{\alpha} P'}{(\mathsf{rec}\, A(\vec{b}).P)(\vec{c}) \xrightarrow{\alpha} P'}$ (*id*) $\dfrac{P =_\alpha P' \ ,\ P' \xrightarrow{\alpha} Q' \ ,\ Q' =_\alpha Q}{P \xrightarrow{\alpha} Q}$

FIG. 3.1: Règles du système de transitions étiqueté avec liaison précoce pour π^r.

La syntaxe du calcul est la suivante :

$$\begin{array}{rcl} w & ::= & a \mid a\mathbf{:}\mathit{val} \\ P, Q, R \ldots & ::= & \mathbf{0} \mid \overline{a}(\vec{b}) \mid a(\vec{b}).P \mid (P \mid Q) \mid (\nu w)P \mid [a = b]P, Q \mid T(\vec{a}) \\ T & ::= & A \mid (\mathsf{rec}\, A(\vec{a}).P) \end{array}$$

On ajoute deux conditions sur les termes pour qu'ils soient légaux :

- dans les processus récepteur $a(\vec{b}).P$ et récursifs $(\mathsf{rec}\, A(\vec{b}).P)$, tous les noms de $\vec{b}$ sont supposés distincts, et différents de a dans le cas de la réception ;
- on fait l'hypothèse standard que la récursion est gardée, c'est-à-dire que dans $(\mathsf{rec}\, A(\vec{a}).P)(\vec{b})$ tous les appels récursifs à A dans P apparaissent sous un préfixe de réception.

On utilisera la notation $\mathit{subj}(w)$ pour désigner le nom a quand $w = a$ ou $w = a\mathbf{:}\mathit{val}$. On dira qu'un terme est *clos* s'il ne contient pas d'identificateur de processus libre.

Une construction souvent employée est la réception répliquée. Il s'agit d'un processus recevant un message sur un canal qui déclenche une continuation et régénère le récepteur. Nous utiliserons l'abréviation suivante pour ce type de processus :

$$a^*(\vec{b}).P \stackrel{\mathsf{def}}{=} \mathsf{rec}\, A(a).a(\vec{b}).(P \mid A(a))$$

où on écrit simplement $\mathsf{rec}\, A(\vec{a}).P$ pour $(\mathsf{rec}\, A(\vec{a}).P)(\vec{a})$.

La sémantique opérationnelle du calcul est donnée figure 3.1. Les actions sont de la forme :

$$\alpha \quad ::= \quad \tau \mid (\nu\vec{w})\overline{a}\vec{b} \mid a\vec{b}$$

On note $\mathit{nm}(\alpha) = \mathit{fn}(\alpha) \cup \mathit{bn}(\alpha)$ où :

$$\begin{array}{lll} \mathit{fn}(\tau) = \emptyset & \mathit{fn}(a(\vec{b})) = \{a\} \cup \{\vec{b}\} & \mathit{fn}((\nu\vec{w})\overline{a}\vec{b}) = \{a, \vec{b}\} \backslash \mathit{subj}(\vec{w}) \\ \mathit{bn}(\tau) = \emptyset & \mathit{bn}(a(\vec{b})) = \emptyset & \mathit{bn}((\nu\vec{w})\overline{a}\vec{b}) = \mathit{subj}(\vec{w}) \end{array}$$

On définit aussi $\mathit{bcn}(\alpha)$, l'ensemble des noms de canaux liés dans l'action α par :

$$\begin{array}{rcl} \mathit{bcn}(\tau) & = & \mathit{bcn}(a(\vec{b})) = \mathit{bcn}(\overline{a}\vec{b}) = \emptyset \\ \mathit{bcn}((\nu a)(\nu\vec{w})\overline{a}\vec{b}) & = & \{a\} \cup \mathit{bcn}((\nu\vec{w})\overline{a}\vec{b}) \\ \mathit{bcn}((\nu a : \mathit{val})(\nu\vec{w})\overline{a}\vec{b}) & = & \mathit{bcn}((\nu\vec{w})\overline{a}\vec{b}) \end{array}$$

En reprenant notre abréviation pour la réception répliquée, il est facile de voir que

$$a^*(\vec{b}).P \xrightarrow{a(\vec{c})} [\vec{c}/\vec{b}]P \mid a^*(\vec{b}).P$$

À titre d'exemple nous donnons l'implantation dans notre calcul du protocole d'*appel de procédure distante* ou RPC[2].

Exemple 3.1.1 Le protocole RPC consiste en un serveur sollicité par un client pour exécuter une procédure $f(\vec{x})$ et retourner le résultat à ce client. Le client envoie au serveur des données $\vec{d}$ et un nom de canal qu'il vient de créer pour recevoir le résultat. En supposant que le serveur attend des données sur un canal a, le serveur S et le client C s'écrivent :

$$\begin{array}{rcl} S & \stackrel{\mathsf{def}}{=} & a^*(\vec{x}, y).\overline{y}(f(\vec{x})) \\ C & \stackrel{\mathsf{def}}{=} & (\nu r)(\overline{a}(\vec{d}, r) \mid r(\vec{b}).P) \end{array}$$

Nous pouvons montrer que les deux réductions successives

$$(S \mid C) \xrightarrow{\tau} (\nu r)(S \mid \overline{r}(f(\vec{d})) \mid r(\vec{b}).P) \xrightarrow{\tau} (\nu r)(R \mid [f(\vec{d})/\vec{b}]P)$$

sont valides. Nous ne donnons que la preuve de la première réduction :

$$\dfrac{\dfrac{\vdots}{a^*(\vec{x}, y).\overline{y}(f(\vec{x})) \xrightarrow{a(\vec{d},r)} S \mid \overline{r}(f(\vec{d}))} \qquad \dfrac{\dfrac{}{\overline{a}(\vec{d}, r) \xrightarrow{\overline{a}(\vec{d},r)} \mathbf{0}}}{(\overline{a}(\vec{d}, r) \mid r(\vec{b}).P) \xrightarrow{\overline{a}(\vec{d},r)} r(\vec{b}).P}}{(\nu r)(\overline{a}(\vec{d}, r) \mid r(\vec{b}).P) \xrightarrow{(\nu r)\overline{a}(\vec{d},r)} r(\vec{b}).P}$$

$$(S \mid C) \xrightarrow{\tau} (\nu r)(S \mid \overline{r}(f(\vec{d})) \mid r(\vec{b}).P)$$

□

[2]Remote Procedure Call

$$\frac{}{A\!:\!\tau \vdash A\!:\!\tau} \qquad \frac{}{a\!:\!\tau \vdash a\!:\!\tau} \qquad \frac{\Gamma \vdash \overrightarrow{a\!:\!\tau} \quad , \quad \Delta \vdash \overrightarrow{b\!:\!\sigma}}{\Gamma\,,\,\Delta \vdash \overrightarrow{a\!:\!\tau}\,,\,\overrightarrow{b\!:\!\sigma}}$$

$$\frac{}{\vdash \mathbf{0}} \qquad \frac{\Gamma \vdash \overrightarrow{b\!:\!\tau}}{a\!:\!Ch(\vec{\tau})\,,\,\Gamma \vdash \overline{a}(\vec{b})} \qquad \frac{a\!:\!Ch(\vec{\tau})\,,\,\Gamma\,,\,\Delta \vdash P \quad , \quad \Delta \vdash \overrightarrow{b\!:\!\tau}}{a\!:\!Ch(\vec{\tau})\,,\,\Gamma \vdash a(\vec{b}).P}$$

$$\frac{\Gamma \vdash P \quad , \quad \Gamma \vdash Q}{\Gamma \vdash P \mid Q} \qquad \frac{a,b\!:\!val\,,\,\Gamma \vdash P \quad , \quad a,b\!:\!val\,,\,\Gamma \vdash Q}{a,b\!:\!val\,,\,\Gamma \vdash [a=b]P,Q}$$

$$\frac{a\!:\!val\,,\,\Gamma \vdash P}{\Gamma \vdash (\nu a\!:\!val)P} \qquad \frac{a\!:\!Ch(\vec{\tau})\,,\,\Gamma \vdash P}{\Gamma \vdash (\nu a)P} \qquad \frac{\Gamma \vdash P}{\Gamma\,,\,\Delta \vdash P}$$

$$\frac{A\!:\!Ch(\vec{\tau})\,,\,\Gamma\,,\,\Delta \vdash P \quad , \quad \Delta \vdash \overrightarrow{a\!:\!\tau}}{\Gamma \vdash (\mathsf{rec}\,A(\vec{a}).P)\!:\!Ch(\vec{\tau})} \qquad \frac{\Gamma \vdash T\!:\!Ch(\vec{\tau}) \quad , \quad \Delta \vdash \overrightarrow{a\!:\!\tau}}{\Gamma\,,\,\Delta \vdash T(\vec{a})}$$

FIG. 3.2: Système de types pour π^r

3.2 Le système de types

Le système de types que nous considérons dans cette partie n'est qu'une variante du système de sortes présenté dans le chapitre précédent. Nous ne faisons ici qu'étendre ce dernier avec le type *val*. Les types sont donnés par la grammaire suivante[3] :

$$\tau,\sigma \ldots \quad ::= \quad Ch(\tau_1,\ldots,\tau_n) \mid val$$

Le type $Ch(\tau_1,\ldots,\tau_n)$ (où éventuellement $n=0$) est le type des canaux communicant des n-uplets de noms de types respectivement $\tau_1,\ldots\tau_n$. Le système d'inférence permet de prouver des jugements de la forme $\Gamma \vdash P$ où Γ est un *contexte de typage*, P un terme du calc61 10.909 Tf 15.15

)45(dans)-245(91038(n)]TJ0.0067Td()TJF6h)T899Tf12.121Td()TJF4.99Tf9.39.697Td58Td(r)TJF..99Tf151.637Td(n)T45(calc

paramétré par n arguments de types respectifs $\tau_1, \ldots, \tau_n$ ». Le système de types pour les noms n'est pas indispensable mais nous l'adoptons ici en prévision de la partie suivante où il nous sera très utile. Dans les constructions liantes (réception, restrictions et récursion) on suppose comme d'habitude que les noms liés n'apparaissent pas dans le contexte résultant.

Une première propriété simple pour le typage des noms est l'unicité du contexte de typage.

Lemme 3.2.1 *Si* $\Gamma \vdash \overrightarrow{a : \tau}$ *et* $\Delta \vdash \overrightarrow{a : \tau}$ *alors* $\Gamma = \Delta$.

Une propriété importante pour les calculs typés est la préservation du typage en cours de réduction. En effet, si cette propriété n'était pas garantie, il faudrait vérifier dynamiquement que le programme reste typable à chaque pas d'exécution ce qui contredirait le principe d'analyse statique.

Théorème 3.2.2 (Préservation du typage) *Si* $\Gamma \vdash P$ *et* $P \xrightarrow{\alpha} P'$ *alors,*

1. *si* $\alpha = a\vec{b}$ *alors* $\Gamma = a : Ch(\vec{\tau}), \Delta$ *et* $\Gamma, \Delta' \vdash P'$ *où* $\Delta' \vdash \overrightarrow{b : \tau}$;
2. *si* $\alpha = (\nu \vec{w})\overline{a}\vec{b}$ *alors* $\Gamma = a : Ch(\vec{\tau}), \Delta, \Delta'$ *et il existe* Γ' *tel que* $\Gamma', \Delta' \vdash \overrightarrow{b : \tau}$ *et* $\Gamma, \Gamma' \vdash P'$;
3. *si* $\alpha = \tau$ *alors* $\Gamma \vdash P'$.

Ce théorème se démontre simplement par induction sur la dernière règle utilisée pour inférer $\Gamma \vdash P$. On a cependant besoin des trois lemmes suivants.

Lemme 3.2.3 (Substitution) *Si* $\Gamma \vdash P$ *et* $\Gamma(\vec{a}) = \Gamma(\vec{b})$, *alors* $\Gamma \vdash [\vec{b}/\vec{a}]P$.

Lemme 3.2.4 (Renforcement) *Si* $\Gamma, a : \tau \vdash P$ *et* $a \notin fn(P)$ *alors* $\Gamma \vdash P$.

Lemme 3.2.5 (Dépliage) *Si* $\Gamma \vdash (\mathsf{rec}\, A(\vec{a}).P)(\vec{b})$ *alors* $\Gamma \vdash [(\mathsf{rec}\, A(\vec{a}).P)/A, \vec{b}/\vec{a}]P$.

Nous omettons ici les preuves de ces lemmes qui sont classiques. Dans la suite nous ne considérons que des termes typables.

3.3 Le système de bonne formation

Le système de types donné dans la section précédente a pour unique objectif de vérifier qu'un processus utilise des noms conformément à la spécification donnée par leurs types. Ces derniers ne donnent qu'une information du type « canal de telle arité » ou « valeur ». De plus, le théorème de préservation du typage assure qu'en cours de réduction les noms seront toujours utilisés en respectant leurs types.

Nous avons évoqué dans le chapitre précédent la réceptivité uniforme. La réceptivité que nous nous proposons d'étudier ici rentre dans le cadre de la ω-réceptivité de [San99] dans le sens où les processus récepteurs (qu'on appellera aussi dès lors **ressources**) sont *persistants*. Les deux principales différences résident dans le fait qu'on ne distingue pas des canaux non-réceptifs [4] et qu'on n'impose pas l'uniformité des récepteurs, mais tout en préservant des contraintes fortes déjà utilisées dans [San99]. Pour obtenir ces propriétés nous utilisons un système d'inférence permettant de prouver

[4] plus exactement, dès qu'un canal est utilisé pour une réception, alors celui-ci est réceptif.

$$\frac{}{\Vdash \mathbf{0}} \qquad \frac{}{\Vdash \overline{a}(\vec{b})} \qquad \frac{a \Vdash P}{a \Vdash a(\vec{b}).P} \qquad \frac{I \Vdash P \quad , \quad I' \Vdash Q}{I, I' \Vdash P \mid Q} \qquad \frac{I \Vdash P \quad , \quad I \Vdash Q}{I \Vdash [a = b]P, Q}$$

$$\frac{a, I \Vdash P}{I \Vdash (\nu a)P} \qquad \frac{I \Vdash P}{I \Vdash (\nu a : \mathit{val})P} \qquad \frac{}{\Vdash A} \qquad \frac{a \Vdash P}{\Vdash (\mathsf{rec}\, A(a, \vec{b}).P)} \qquad \frac{\Vdash T}{a \Vdash T(a, \vec{b})}$$

FIG. 3.3: Système de bonne formation pour π^r.

la **bonne formation** des termes. Ce système vérifie essentiellement que les termes du calcul respectent deux critères :

- il n'y a pas de réceptions imbriquées sur des noms différents ;
- toutes les réceptions apparaissent dans un processus récursif ;

Le premier critère garantie qu'un récepteur sur un nom donné est directement exploitable. Par exemple, dans le terme $(\overline{b} \mid a.b.P)$ le message sur le canal b ne peut être reçu tant qu'il n'y aura pas eu de message sur le canal a. Vérifier statiquement qu'il y aura effectivement un tel message est un problème difficile. Pour cela, il faut faire appel à des systèmes de types complexes[5] comme celui de [Kob98]. Nous verrons plus loin que si la contrainte que nous avons choisie est très forte (au sens ou elle élimine beaucoup de termes), celle-ci demeure néanmoins raisonnable. De plus, elle est également présente dans le système de types garantissant la ω-réceptivité de [San99].

La deuxième contrainte assure la persistance des ressources. Les faire apparaître dans des processus récursifs permet de les régénérer dès qu'elles ont été consommées éventuellement dans un état différent relâchant ainsi la contrainte d'uniformité.

Le système d'inférence est donné dans la figure 3.3. Il permet de prouver des jugements de la forme $I \Vdash P$, où I est un ensemble fini de noms appelé **interface**, qui signifie que « P est bien formé avec l'interface I ». Intuitivement, l'interface d'un processus P contient les noms de canaux sur lesquels P peut recevoir (*toujours et immédiatement*) des messages. L'union d'interfaces I et I' est l'union ensembliste notée I, I'. Enfin, dans les règles pour les constructions liantes on fait l'hypothèse standard que les noms liés n'apparaissent pas dans l'interface résultante.

Nous commentons à présent ces règles. Dans la règle de réception $a(\vec{b}).P$, l'interface de la continuation P est supposée être $\{a\}$, c'est-à-dire que P peut réaliser une réception sur le canal a et *uniquement* sur celui-ci. On réalise ici la première contrainte mentionnée plus haut mais aussi la contrainte selon laquelle un nom reçu ne peut être utilisé comme canal de réception. Cette dernière caractérise le π-calcul (polyadique, asynchrone) **local** de [MS98] où, typiquement, $a(c).c(\vec{d}).P$ n'est pas bien formé. Supposer la « localité » dans notre cas est important. En effet, dans le cas contraire l'environnement d'un processus tel que $(\nu a)(\overline{a}(\vec{u}) \mid \overline{b}(a))$ pourrait créer un récepteur sur a et vérifier que le message sur a sera effectivement reçu deviendrait beaucoup plus difficile. Notons d'ailleurs que la règle de restriction sur les canaux contraint le corps de la restriction à contenir un récepteur sur le nom restreint. Dans les règles concernant les processus paramétrés, le premier argument est le

[5] il faut entendre par là que les types sont très peu lisibles et donc, par exemple, difficilement exploitables par un programmeur.

nom sur lequel un récepteur est défini dans le corps de la récursion. On remarque que c'est la seule façon d'introduire un nom dans l'interface. Enfin, la règle concernant la composition parallèle peut être agrémentée de la condition $I \cap I' = \emptyset$. Celle-ci a pour effet de garantir l'unicité des récepteurs, c'est-à-dire que pour un nom donné il existe au plus un processus capable de recevoir sur ce nom. L'unicité des récepteurs est une caractéristique syntaxique du Join-calcul qui peut aussi être capturée par typage dans le π-calcul (il s'agit du π_1-calcul de R. AMADIO, [Ama97, Ama00]). Nous conservons cette condition car l'unicité des récepteurs semble être une hypothèse sensible tant d'un point de vue de spécification que d'implantation (*cf.* modèles de programmation objet, [FG96]). D'autre part, elle nous sera utile pour montrer que les contraintes ne sont pas imposées au détriment de l'expressivité du calcul. Nous nommons ce calcul π_1^r.

Avant de revenir sur l'exemple du RPC, nous donnons quelques constructions dérivées. L'agent *identité* est défini par :

$$Id_a \stackrel{\text{def}}{=} \mathsf{rec}\, A(a).a(\vec{b}).(\overline{a}(\vec{b}) \mid A(a)) = a^*(\vec{b}).\overline{a}(\vec{b})$$

Ce processus, qui attend toujours un message sur a et le renvoie sur ce même canal, est bien formé avec l'interface $\{a\}$ et possède le typage $a : \mathit{Ch}(\vec{\tau}) \vdash Id_a$. Nous verrons plus loin que ce terme est équivalent au processus qui attend toujours des messages sur a et les absorbe (comme un « *trou noir* ») :

$$T_a \stackrel{\text{def}}{=} \mathsf{rec}\, A(a).a(\vec{b}).A(a)$$

Cet agent a le même typage et la même interface que Id_a. Nous rappelons aussi la construction de *réception linéaire*[6] de [Ama97], donnée par :

$$a(\vec{b}){:}P \stackrel{\text{def}}{=} a(\vec{b}).(P \mid T_a)$$

Il est facile de voir que ce terme a le même typage que $a(\vec{b}).P$, et que $a \Vdash a(\vec{b}){:}P$ si $\Vdash P$.

Exemple 3.3.1 Dans le protocole RPC, le processus client $(\nu r)(\overline{a}(\vec{d}, r) \mid r(\vec{b}).P)$ est bien formé seulement si r est dans l'interface de P. En utilisant la construction de réception linéaire il est possible de transformer ce client en processus bien formé, à condition que la continuation P soit bien formée avec une interface vide :

$$C \stackrel{\text{def}}{=} (\nu r)(\overline{a}(\vec{d}, r) \mid r(\vec{b}){:}P)$$

et qui a le même typage que précédemment $a : \mathit{Ch}(\vec{\tau}, \mathit{Ch}(\vec{\sigma})), \Gamma \vdash C$. □

Tout comme le typage, la bonne formation est préservée par réduction. Pour le montrer il nous faut tout d'abord établir quelques lemmes.

Lemme 3.3.2 (Substitution) *Soit $I \Vdash P$ et S une substitution injective sur I. Alors $\mathsf{S}(I \Vdash P)$, et si $I \Vdash [\mathsf{S}]P$ alors $\mathsf{S}^{-1}(I) \Vdash P$.*

[6] aussi appelée *input once*.

Preuve: Simple induction sur l'inférence de $I \Vdash P$. □

Le lemme suivant établit que si le corps d'un processus récursif est bien formé, alors tout dépliage du processus récursif est bien formé.

Lemme 3.3.3 *Si $I \Vdash P$ et $\Vdash T$, et $fn(T) \cap bn(P) = \emptyset$ alors $I \Vdash [T/A]P$.*

Preuve: Si A n'est pas libre dans P la preuve est terminée. Sinon, dans l'inférence de $I \Vdash P$ il suffit de remplacer les axiomes $\Vdash A$ par des preuves de $\Vdash T$. □

Lemme 3.3.4 *Si $I \Vdash P$ et $P =_\alpha Q$ alors $I \Vdash Q$.*

Preuve: La preuve se fait simplement en remarquant que si $I \Vdash P$ alors $I \subseteq fn(P)$. □

Le lemme suivant établit l'unicité de l'interface pour les processus bien formés.

Lemme 3.3.5 *Si $I \Vdash P$ et $J \Vdash P$ alors $I = J$.*

Preuve: Par induction sur la preuve de la bonne formation de P. □

Proposition 3.3.6 (Préservation de la bonne formation) *Si $I \Vdash P$ est valide et $P \xrightarrow{\alpha} P'$ alors $I, bcn(\alpha) \Vdash P'$.*

Preuve: Par induction sur l'inférence de $P \xrightarrow{\alpha} P'$. Nous ne considérons que la réduction (*rec*), les autres cas se traitent trivialement en utilisant les lemmes précédents. Soit $P = (\mathsf{rec}\, A(a,\vec{b}).Q)(c,\vec{d})$, d'après (*rec*) on a $[\mathsf{rec}\, A(a,\vec{b}).Q/A, c/a, \vec{d}/\vec{b}]Q \xrightarrow{\alpha} P'$ $(*)$. D'autre part, on a $I = \{c\}$ et l'inférence de $\{c\} \Vdash (\mathsf{rec}\, A(a,\vec{b}).P)(c,\vec{d})$ a la forme suivante :

$$\dfrac{\dfrac{\dfrac{\vdots}{\{a\} \Vdash Q}}{\Vdash \mathsf{rec}\, A(a,\vec{b}).Q}}{\{c\} \Vdash (\mathsf{rec}\, A(a,\vec{b}).Q)(c,\vec{d})}$$

De $\{a\} \Vdash Q$ et des lemmes 3.3.2 et 3.3.3 on déduit que $\{c\} \Vdash [\mathsf{rec}\, A(a,\vec{b}).Q/A, c/a, \vec{d}/\vec{b}]Q$ est valide. Par ce jugement, par $(*)$ et par hypothèse d'induction, on conclut que $\{c\}, bcn(\alpha) \Vdash P'$. □

3.4 Livrabilité des messages

Nous montrons dans cette section la **livrabilité des messages**. Cette propriété peut s'énoncer ainsi : si un message est envoyé sur un canal donné dans un processus bien formé, alors ce dernier contient un récepteur pour ce message. Cette propriété est une conséquence directe de la propriété de réceptivité, c'est-à-dire : un processus bien formé avec une interface I possède un récepteur pour chaque nom de I et ces récepteurs sont persistants. Pour l'établir formellement nous avons besoin de définir le prédicat $P \downarrow a$, signifiant que « P peut recevoir sur le canal a », c'est-à-dire :

$$P \downarrow a \overset{\text{def}}{\Leftrightarrow} \exists \vec{b} \exists P'.\, P \xrightarrow{a\vec{b}} P'$$

Proposition 3.4.1 (Réceptivité) *Soit P un processus clos bien formé. Alors :*

1. *si $I \Vdash P$ alors $P \downarrow a$ si, et seulement si $a \in I$,*
2. *si $P \downarrow a$ et $P \xrightarrow{\alpha} P'$, alors $P' \downarrow a$.*

Preuve: (1) ($\Rightarrow$) Par induction sur la dérivation de l'action $P \xrightarrow{a\vec{c}} P'$. Si la dernière action est (*in*) alors $P = a(\vec{b}).Q$ et nécessairement $a \Vdash a(\vec{b}).Q$. Les autres cas se traitent facilement en appliquant l'hypothèse d'induction et éventuellement les lemmes 3.3.2 et 3.3.3.
($\Leftarrow$) On procède par induction sur l'inférence de $I \Vdash P$ en supposant que P peut éventuellement contenir des identificateurs de processus libres mais gardés par une réception. Si la dernière règle utilisée est celle pour la réception on conclue directement. Dans le cas des règles pour la composition parallèle, le matching et la restriction, on utilise simplement l'hypothèse d'induction. La dernière règle envisageable est celle des processus paramétrés. D'après l'hypothèse émise, P ne peut pas être un identificateur de processus puisque celui-ci n'est pas gardé. Par conséquent, on a $P = (\mathsf{rec}\, A(c, \vec{d}).Q)(a, \vec{b})$ et l'inférence suivante :

$$\dfrac{\dfrac{\dfrac{\vdots}{c \Vdash Q}}{\Vdash (\mathsf{rec}\, A(c, \vec{d}).Q)}}{a \Vdash (\mathsf{rec}\, A(c, \vec{d}).Q)(a, \vec{b})}$$

Par hypothèse d'induction on a donc $Q \downarrow c$ et par substitution $[(\mathsf{rec}\, A(c, \vec{d}).Q)/A, a/c, \vec{b}/\vec{d}]Q \downarrow a$, et par la règle (*rec*) on en déduit $P \downarrow a$.
(2) On utilise simplement le point précédent et la proposition 3.3.6. □

La propriété de réceptivité est à la fois une propriété de sûreté forte et une contrainte très lourde. En effet, si un processus reçoit un message sur un canal donné il faut s'assurer qu'il est immédiatement capable de recevoir à nouveau des messages sur ce canal. On peut donc se demander comment programmer des agents non réceptifs tels qu'un buffer à une place[7], qui attend des données sur un canal et les renvoie sur un autre.

Exemple 3.4.2 Dans le π-calcul synchrone, un buffer à une place peut s'écrire :

$$(\mathsf{rec}\, B(a, b).a(x).\overline{b}(x).B(a, b))$$

On ne peut clairement pas l'écrire aussi facilement dans π_1^r puisque après réception d'un message sur a, ce canal n'est pas immédiatement disponible pour une nouvelle réception. Il faut donc adopter un « style de programmation réceptif ». Par exemple, plutôt que de ne pas répondre aux messages sur a, on peut les traiter à l'aide de l'agent identité Id_a donné précédemment. On peut aussi utiliser des « clefs » passées dans les messages, puis filtrées par une construction conditionnelle. On pourrait par exemple écrire le buffer ainsi : d'abord il reçoit sur a un message demandant la sauvegarde d'une donnée dans le buffer (si tel n'est pas le cas le message est renvoyé), puis sur le même canal il reçoit

[7] aussi appelé « one-slot buffer ».

une requête d'extraction de la donnée qui est alors délivrée sur un canal de retour privé (encore une fois, si tel n'est pas le cas, le message est renvoyé). En utilisant la notation if $a \neq b$ then P else Q pour $[a = b]Q, P$, et en rappelant que rec $A(\vec{a}).P$ est une abréviation pour $(\text{rec } A(\vec{a}).P)(\vec{a})$, ce buffer peut s'écrire ainsi :

$$\begin{aligned} \mathit{Buff}_a \stackrel{\text{def}}{=} \ & \text{rec } B(a).a(k_1, x, y).\text{if } k_1 \neq \text{write} \\ & \quad \text{then } (\overline{a}(k_1, x, y) \mid B(a)) \\ & \quad \text{else rec } B_1(a, x).a(k_2, z, y).\text{if } k_2 \neq \text{read} \\ & \qquad \text{then } (\overline{a}(k_2, z, y) \mid B_1(a, x)) \\ & \qquad \text{else } (\overline{y}(x) \mid B(a)) \end{aligned}$$

Les requêtes d'écriture et de lecture sont respectivement $\overline{a}(\text{write}, b, \_)$ et $(\nu c)(\overline{a}(\text{read}, \_, c) \mid c(x){:}P)$ où « _ » représente n'importe quel nom. Comme on peut le voir, le buffer est maintenant une sorte d'*agent,* c'est-à-dire un processus qui est invoqué par son nom (a) et qui réagit suivant un certain protocole interne. Par exemple :

$$\begin{aligned} \overline{a}(\text{write}, b, \_) \mid (\nu c)(\overline{a}(\text{read}, \_, c) \mid c(x){:}P) \mid \mathit{Buff}_a \ & \xrightarrow{\tau} \ (\nu c)(\overline{a}(\text{read}, \_, c) \mid c(x){:}P) \mid \\ & \qquad \text{if write} \neq \text{write then } \ldots \text{ else } B_1(a, b) \\ & \xrightarrow{\tau} \ (\nu c)(c(x){:}P \mid \\ & \qquad \text{if read} \neq \text{read then } \ldots \text{ else } (\overline{c}(b) \mid \mathit{Buff}_a)) \\ & \xrightarrow{\tau} \ (\nu c)([b/x]P \mid \mathit{Buff}_a) \end{aligned}$$

En revanche, s'il n'y a pas de requête de lecture, mais deux requêtes d'émission, le buffer en acceptera une première et, à la réception de la seconde, celle-ci sera systèmatiquement réexpédiée :

$$\begin{aligned} \overline{a}(\text{write}, b, \_) \mid \overline{a}(\text{write}, c, \_) \mid \mathit{Buff}_a \ & \xrightarrow{\tau} \ \overline{a}(\text{write}, c, \_) \mid \text{if write} \neq \text{write then } \ldots \text{ else } B_1(a, b) \\ & \xrightarrow{\tau} \ \text{if write} \neq \text{read then } (\overline{a}(\text{write}, c, \_) \mid B_1(a, b)) \text{ else } \ldots \\ & \ \vdots \end{aligned}$$

□

π_1^r étant un sous-calcul du π-calcul asynchrone, il hérite de la notion de **bisimulation asynchrone** dont nous rappelons la définition.

Définition 3.4.3 (Bisimulation) *Une relation symétrique $\mathcal{S}$ sur les processus bien formés est une bisimulation si $P\mathcal{S}Q$ implique :*

1. *si $P \xrightarrow{\tau} P'$ alors $Q \xrightarrow{\tau} Q'$ et $P'\mathcal{S}Q'$;*
2. *si $P \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} P'$ et $\mathit{subj}(\vec{w}) \cap \mathit{fn}(Q) = \emptyset$, alors $Q \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} Q'$ et $P'\mathcal{S}Q'$;*
3. *si $P \xrightarrow{a\vec{b}} P'$ alors soit $Q \xrightarrow{a\vec{b}} Q'$ et $P'\mathcal{S}Q'$, soit $Q \xrightarrow{\tau} Q'$ et $P'\mathcal{S}(Q' \mid \overline{a}(\vec{b}))$.*

On note $\sim$ la plus grande bisimulation.

On note $P \downarrow \overline{a}$ le fait que P peut réaliser une émission sur a, et on définit le préordre $\gtrsim$ sur les processus typables et bien formés :

$$P \gtrsim Q \stackrel{\text{def}}{\Leftrightarrow} \begin{cases} P \sim Q & \text{and} \\ \forall \Gamma.\, \Gamma \vdash P \Rightarrow \Gamma \vdash Q & \text{and} \\ \forall I.\, I \Vdash P \Leftrightarrow I \Vdash Q \end{cases}$$

On note $\simeq$ l'équivalence induite par $\gtrsim$. D'après le corollaire 3.4 de [ACS98], $\sim$ est une congruence. Il est donc facile de voir que $\simeq$ est également une congruence. Les propriétés énoncées dans le lemme suivant sont utiles pour montrer la livrabilité des messages.

Lemme 3.4.4 *On a les relations suivantes :*

1. $(\nu w)P \mid Q \simeq (\nu w)(P \mid Q)$ *si* $\mathit{subj}(w) \notin \mathit{fn}(Q)$;
2. $[a = a]P, Q \gtrsim P$;
3. $[a = b]P, Q \gtrsim Q$;
4. $(\mathsf{rec}\, A(\vec{b}).P)(\vec{c}) \gtrsim [(\mathsf{rec}\, A(\vec{b}).P)/_A, \vec{c}/\vec{b}]P$.

Preuve: Nous ne montrons pas ici la validité des équivalences de bisimilation car s'agit de résultats bien connus.
(1) Montrons que $\forall \Gamma.\, \Gamma \vdash (\nu w)P \mid Q \Rightarrow (\nu w)(P \mid Q)$. La preuve de $\Gamma \vdash (\nu w)P \mid Q$ a la forme suivante :

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{a : \tau, \Delta' \vdash P}}{\Delta' \vdash (\nu w)P} \quad \Delta' \subseteq \Delta \qquad \dfrac{\vdots}{\Delta \vdash Q}}{\Delta \vdash (\nu w)P \mid Q}}{\Gamma \vdash (\nu w)P \mid Q}$$

où $\Delta \subseteq \Gamma$ et $\tau = \mathit{val}$ si $w = a : \mathit{val}$. Puisqu'il n'y a pas d'occurrence de a dans Δ, le contexte $a : \tau, \Delta$ est bien défini et par affaiblissement on a $a : \tau, \Delta \vdash Q$, et donc $\Gamma \vdash (\nu w)(P \mid Q)$.

Réciproquement, la preuve de $\Gamma \vdash (\nu w)(P \mid Q)$ a la forme suivante :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Delta \vdash P} \quad \dfrac{\vdots}{\Delta \vdash Q}}{\Delta \vdash P \mid Q}}{\vdots}}{a\!:\!\tau, \Gamma' \vdash P \mid Q}}{\Gamma' \vdash (\nu w)(P \mid Q)}}{\vdots}}{\Gamma \vdash (\nu w)(P \mid Q)}$$

où $\Delta \subseteq a\!:\!\tau, \Gamma'$ et $\Gamma' \subseteq \Gamma$. Si $\Delta = a\!:\!\tau, \Delta'$, alors par le lemme 3.2.4, on a $\Delta' \vdash Q$ et $\Delta' \vdash (\nu w)P$, d'où $\Gamma \vdash (\nu w)P \mid Q$. Sinon, $\Delta \subseteq \Gamma$, et par affaiblissement $a\!:\!\tau, \Delta \vdash P$. De ceci on conclue $\Delta \vdash (\nu w)P$ et $\Gamma \vdash (\nu w)P \mid Q$. La preuve que $\forall I.\, I \Vdash ((\nu w)P \mid Q) \Leftrightarrow I \Vdash (\nu w)(P \mid Q)$ est triviale.

(2) Il suffit de montrer que $I \Vdash P \Rightarrow I \vdash [a = a]P, Q$. Par hypothèse, $[a = a]P, Q$ étant bien formé, il existe une interface J telle que $J \vdash [a = a]P, Q$, et par la règle correspondante pour inférer ce jugement on a $J \Vdash P$. Par le lemme 3.3.5, on conclue $I = J$. La preuve du point (3) est identique.

(4) Il suffit de montrer que $I \Vdash [(\mathsf{rec}\, A(\vec{b}).P)/_A, \vec{c}/\vec{b}]P \Rightarrow I \Vdash (\mathsf{rec}\, A(\vec{b}).P)(\vec{c})$ en utilisant les lemmes 3.3.2, 3.3.3 et 3.3.5. □

En résultat préliminaire, nous montrons qu'un processus qui peut envoyer un message est équivalent à un autre où apparaît explicitement ce message.

Lemme 3.4.5 *$P \downarrow \overline{a}$ si et seulement si $P \gtrsim (\nu \vec{w})(\overline{a}(\vec{b}) \mid R)$ avec $a \notin$ subj$(\vec{w})$.*

Preuve: ($\Rightarrow$) Par induction sur la dernière règle utilisée pour inférer $P \xrightarrow{\alpha} P'$ où α est une action d'émission de sujet a.

- (*out*) Alors $P = \overline{a}(\vec{b})$ est déjà de la forme recherchée.
- (*ext*) Alors $P = (\nu w')Q$, $\alpha = (\nu w')\alpha'$ avec α' action d'émission de sujet a différent de *subj*(w'), et $Q \xrightarrow{\alpha'} P'$. D'où $Q \downarrow \overline{a}$ et par hypothèse d'induction, $Q \simeq (\nu \vec{w})(\overline{a}(\vec{b}) \mid R)$. Comme $\simeq$ est une congruence on en déduit que $P \simeq (\nu w', \vec{w})(\overline{a}(\vec{b}) \mid R)$.
- (*cp*) Alors $P = Q_1 \mid Q_2$, $P' = Q_1' \mid Q_2$, et $Q_1 \xrightarrow{\alpha} Q_1'$. Par hypothèse d'induction on a $Q_1 \simeq (\nu \vec{w})(\overline{a}(\vec{b}) \mid R)$. $\simeq$ contenant la relation d'α-conversion, on peut avoir choisi $\vec{w}$ tel que $\{\vec{w}\} \cap$ *fn*$(Q_2) = \emptyset$. Ainsi, par le lemme 3.4.4(1), on conclut que $P \simeq (\nu \vec{w})(\overline{a}(\vec{b}) \mid R \mid Q_2)$ qui est de la forme recherchée.
- (m_t) Alors $P = [a = a]P_1, P_2$ et $P_1 \xrightarrow{\alpha} P'$. Par hypothèse d'induction, on a donc $P_1 \gtrsim (\nu \vec{w})(\overline{a}(\vec{b}) \mid R)$. D'après le lemme 3.4.4(2) on a $P \gtrsim P_1$, donc $P \gtrsim (\nu \vec{w})(\overline{a}(\vec{b}) \mid R)$.

Les autres cas se traitent de manière similaire.

($\Leftarrow$) Il est évident, par définition de $\sim$, que si $P \gtrsim Q$ et $Q \downarrow \overline{a}$ alors $P \downarrow \overline{a}$. D'autre part, puisque

$a \notin \mathit{subj}(\vec{w})$, on a $(\nu\vec{w})(\overline{a}(\vec{b}) \mid R) \downarrow \overline{a}$, d'où $P \downarrow \overline{a}$. □

Nous pouvons alors prouver le résultat principal de ce chapitre, où $\rightarrow$ est utilisé pour $\xrightarrow{\tau}$.

Théorème 3.4.6 (Livrabilité des messages) *Soit P un processus typé avec un contexte Γ et bien formé avec une interface I. Si $P \xrightarrow{*} P'$ et $P' \gtrsim (\nu\vec{w})(\overline{a}(\vec{b}) \mid R)$ avec $a \in I \cup \mathit{subj}(\vec{w})$ alors $R \downarrow a$.*

Preuve: Par les théorème 3.2.2(3) et 3.3.6 on a $\Gamma \vdash P'$ et $I \Vdash P'$. D'autre part, on a $I, J \Vdash \overline{a}(\vec{b}) \mid R$ avec $J \subseteq \mathit{subj}(\vec{w})$, d'où $I, J \Vdash R$; et $\Gamma, \Delta \vdash \overline{a}(\vec{b}) \mid R$. De ce typage on déduit que a n'est pas de type *val*, et donc si $a \in \mathit{subj}(\vec{w})$ alors $a \in J$. Par la proposition 3.4.1 on conclue que $R \downarrow a$. □

CHAPITRE 4

Bisimulation asynchrone «up-to»

AVANT DE MONTRER QUE π_1^r CONSERVE TOUTE L'EXPRESSIVITÉ DU π-CALCUL, dans ce chapitre nous développons des techniques de preuve de bisimulation dites **« up-to »**. Celles-ci ont été précédemment développées dans le cadre de la bisimulation synchrone ([Mil89, SM92, San95]). Nous montrons ici que nous pouvons définir des techniques « up-to » pour la bisimulation asynchrone (faible) et plus particulièrement pour la bisimulation de π_1, calcul très proche de π_1^r, développé par R. AMADIO dans [Ama97, Ama00].

Nous rappelons dans la première section de ce chapitre la syntaxe et la sémantique opérationnelle de π_1 dans une forme qui en fait un « sur-calcul » de π_1^r et dans la seconde section nous développons les techniques « up-to ».

4.1 π_1 : syntaxe et sémantique opérationnelle

Les termes de π_1 sont les mêmes que ceux de π_1^r à l'exception des processus paramétrés. Les identificateurs de processus sont des paires formées d'un identificateur A et d'un entier naturel k :

$$T \quad ::= \quad A^k \mid (\mathsf{rec}\, A^k(\vec{b}).P)$$

Les termes de π_1 sont sujets aux même contraintes que ceux de π_1^r. On note que A^k et A^n sont des identificateurs distincts dès que $k \neq n$, et la substitution $[T/A^k]P$ satisfait :

$$[T/A^k]B^n = \begin{cases} T & \text{si } A^k = B^n \\ B^n & \text{sinon} \end{cases}$$

Le système d'inférence permettant de vérifier la bonne formation des termes est essentiellement celui de π_1^r, à l'exception des cas concernant la réception et les processus paramétrés. Pour ces derniers, on utilise des jugements $\Vdash T : k$ indiquant leur arité. Les règles modifiées sont données dans la figure 4.1.

$$\frac{a, I \Vdash P}{a, I \Vdash a(\vec{b}).P}$$

$$\frac{}{\Vdash A^k : k} \qquad \frac{a_1, \ldots, a_k \Vdash P}{\Vdash (\mathsf{rec}\, A^k(a_1, \ldots, a_k, \vec{b}).P) : k} \qquad \frac{\Vdash T : k}{a_1, \ldots, a_k \Vdash T(a_1, \ldots, a_k, \vec{b})}$$

FIG. 4.1: Système de bonne formation pour π_1

Comme on peut le constater sur les règles, l'arité d'un processus paramétré est le nombre de canaux sur lesquels celui-ci peut faire des réceptions. Aussi, ce système accepte des termes qui ne sont pas réceptifs tel que, par exemple, $a().b().(T_a \mid T_b)$. Dès lors, nous utiliserons la notation $I \Vdash^r P$ pour faire référence aux jugements du système permettant de vérifier la bonne formation des termes dans π_1^r ; et nous garderons la notation $I \Vdash P$ pour les jugements du système de π_1. Il est facile de vérifier que dans π_1 l'interface d'un processus bien formé est unique.

Lemme 4.1.1 *Si $I \Vdash P$ et $I' \Vdash P$ alors $I = I'$.*

Le lemme suivant, dont la preuve est triviale, montre que π_1 est effectivement un « sur-calcul » de π_1^r.

Lemme 4.1.2 *Si $I \Vdash^r P$ et Q est le terme de π_1 obtenu à partir de P en associant l'arité 1 à tous ses identificateurs de processus, alors $I \Vdash Q$.*

La sémantique opérationnelle de π_1 est la même que celle de π_1^r (figure 3.1) où la règle (*rec*) prend en compte l'arité des identificateurs de processus :

$$(\mathrm{rec})\ \frac{[(\mathsf{rec}\, A^k(\vec{b}).P)/A^k][\vec{c}/\vec{b}]P \xrightarrow{\alpha} P'}{(\mathsf{rec}\, A^k(\vec{b}).P)(\vec{c}) \xrightarrow{\alpha} P'}$$

4.2 Bisimulation asynchrone

Nous nous intéressons ici essentiellement à la bisimulation asynchrone faible. Il nous faut définir la notion d'*actions faibles* : des actions éventuellement précédées et suivies d'une ou plusieurs actions internes. Plus formellement, elle sont définies de la façon suivante :

$$\begin{array}{rcl} \Rightarrow & \stackrel{\mathrm{def}}{=} & \xrightarrow{\tau}{}^{*} \\ \stackrel{a\vec{b}}{\Rightarrow} & \stackrel{\mathrm{def}}{=} & \Rightarrow \xrightarrow{a\vec{b}} \Rightarrow \\ \stackrel{(\nu\vec{w})\overline{a}\vec{b}}{\Rightarrow} & \stackrel{\mathrm{def}}{=} & \Rightarrow \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} \Rightarrow \end{array}$$

On définit la bisimulation par co-induction à l'aide des fonctions sur les relations $\mathcal{B}$ et $\mathcal{F}$.

Définition 4.2.1 *Soit $S \subseteq \mathcal{P} \times \mathcal{P}$, alors $(P, Q) \in \mathcal{B}($*

2. *si* $P \xrightarrow{\tau} P'$ *alors* $\exists Q'.\ Q \xrightarrow{\tau} Q'$ *et* $(P', Q') \in S$;
3. *si* $P \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} P'$, $a \notin I$ *et subj*$(\vec{w}) \cap$ *fn*$(Q) = \emptyset$, *alors* $\exists Q'.\ Q \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} Q'$ *et* $(P', Q') \in S$;
4. *si* $P \xrightarrow{a\vec{b}} P'$, *alors soit* $\exists Q'.\ Q \xrightarrow{a\vec{b}} Q'$ *et* $(P', Q') \in S$, *soit* $\exists Q'.\ Q \xrightarrow{\tau} Q'$ *et* $(P', Q' \mid \overline{a}(\vec{b})) \in S$.

$\mathcal{F}$ *est définie de la même manière en remplaçant toutes les actions par des actions faibles.*

Définition 4.2.2 (Bisimulation) $S \subseteq \mathcal{P} \times \mathcal{P}$ *est une bisimulation asynchrone (ou* bisimulation*) si* $S \subseteq \mathcal{B}(S)$. S *est une bisimulation asynchrone faible (ou* bisimulation faible*) si* $S \subseteq \mathcal{F}(S)$. *On note* $\sim = \bigcup\{S \mid S \text{ est une bisimulation}\}$ *et* $\approx = \bigcup\{S \mid S \text{ est une bisimulation faible}\}$.

Cette définition de la bisimulation diffère de celle usuellement donnée (voir par exemple [ACS98]) par la contrainte $a \notin I$ (condition (3) de la définition 4.2.1). Cette contrainte signifie qu'on autorise l'observation d'une émission sur un canal a seulement si le processus considéré ne contient pas déjà un récepteur sur ce canal. Intuitivement, ceci est justifié par le fait qu'observer une émission sur a revient à encapsuler le processus dans un environnement contenant un récepteur sur a. Et si le processus contient déjà un tel récepteur, le terme qui en résulte n'est pas bien formé. Il s'en suit, que deux processus ayant des interfaces différentes peuvent toujours être distingués en les mettant en parallèle avec un message sur un canal apparaissant dans une seule des deux interfaces. C'est pourquoi, on requiert également dans la définition 4.2.1 (condition (1)) que deux processus en relation de bisimulation doivent avoir la même interface. Plus fondamentalement, sans la condition $a \notin I$, le codage donné dans le chapitre suivant ne serait pas complétement adéquat (*cf.* remarque 5.1.8).

$\mathcal{B}$ (resp. $\mathcal{F}$) étant monotone, par le théorème du point fixe de TARSKI-KNASTER [Tar55], $\sim$ (resp. $\approx$) est la plus grande bisimulation (resp. faible) et c'est le plus grand point fixe de $\mathcal{B}$ (resp. $\mathcal{F}$). Par conséquent, pour établir $P \sim Q$, il suffit de trouver une bisimulation S telle que PSQ.

La preuve de la proposition suivante est essentiellement donnée dans [ACS98].

Proposition 4.2.3

1. $(P \approx Q) \Rightarrow (P \mid \overline{a}(\vec{b})) \approx (Q \mid \overline{a}(\vec{b}))$;
2. $\approx$ *est une relation d'équivalence.*

Le lemme B.9 de [ACS98] reste également valide dans notre cas : on obtient une définition équivalente à celle de $\mathcal{F}$ quand, dans cette dernière, on fait correspondre une action faible à une action forte. On préférera d'ailleurs souvent cette dernière définition pour faire nos preuves dans la suite.

Exemple 4.2.4 Montrons que la relation

$$S \stackrel{\text{def}}{=} \{(T_a, (P_1 \mid \ldots \mid P_n \mid Id_a)) \mid P_i \in M\}$$

où M est l'ensemble composé de tous les messages sur a et de $\mathbf{0}$, est une bisimulation faible. On a toujours $a \Vdash T_a$ et $a \Vdash (P_1 \mid \ldots \mid P_n \mid Id_a)$. On montre que $S \subseteq \mathcal{F}(S)$.

1. Si T_a fait une action, la seule qu'il puisse réaliser est une action de réception de $T_a \xrightarrow{a\vec{b}} T_a$. Alors, puisque $Id_a \xrightarrow{a\vec{b}} (\overline{a}(\vec{b}) \mid Id_a)$ on a $(P_1 \mid \ldots \mid P_n \mid Id_a) \xrightarrow{a\vec{b}} (P_1 \mid \ldots \mid P_n \mid \overline{a}(\vec{b}) \mid Id_a)$ et, par construction de S, $(T_a, (P_1 \mid \ldots \mid P_n \mid \overline{a}(\vec{b}) \mid Id_a)) \in S$.

2. Si $(P_1 \mid \ldots \mid P_n \mid Id_a)$ fait une action interne, c'est qu'il existe un $P_i = \overline{a}(\vec{b})$ tel que

$$(P_1 \mid \ldots \mid \overline{a}(\vec{b}) \mid \ldots \mid P_n \mid Id_a) \xrightarrow{\tau} (P_1 \mid \ldots \mid \mathbf{0} \mid \ldots \mid P_n \mid \overline{a}(\vec{b}) \mid Id_a)$$

   En correspondance, T_a n'effectue aucune action interne et $(T_a, (P_1 \mid \ldots \mid \mathbf{0} \mid \ldots \mid P_n \mid \overline{a}(\vec{b}) \mid Id_a)) \in S$.

3. On ne peut observer aucune action d'émission pour $(P_1 \mid \ldots \mid P_n \mid Id_a)$ car la seule possible serait sur le canal a mais celui-ci appartient à l'interface du processus. Il reste donc l'action de réception qui se traite comme dans le premier cas.

S est donc une relation de bisimulation faible et en particulier $(T_a, Id_a) \in S$. Par conséquent $T_a \approx Id_a$. □

L'exemple 4.2.4 met en évidence une première technique permettant de prouver $P \approx Q$, elle consiste à trouver une relation de bisimulation faible S telle que PSQ. Dans ce cas on essaie de choisir la plus petite relation possible de sorte qu'on ait le minimum de couples (P, Q) pour lesquels il faut vérifier les conditions de la définition 4.2.1. Malheureusement, une telle relation existe rarement. La technique de preuve « up-to » permet, lorsque cette relation S est « trop grande » pour pouvoir traiter tous les cas, de ne vérifier les conditions que pour une partie S' des couples de S. En d'autres termes, on vérifie que $S' \subseteq \mathcal{F}(S)$. On conclue que, si S et S' sont reliées par une certaine condition, alors tous les couples de termes de S' sont bisimilaires. On aura évidemment choisi un ensemble S' qui contient les couples de termes dont on souhaitait prouver l'équivalence !

Il convient dans un premier temps de définir une relation d'ordre partiel $\leq_{\mathcal{F}}$ sur les relations binaires. L'ordre $S_1 \leq_{\mathcal{F}} S_2$ formalise l'idée selon laquelle les conditions de la définition de $\mathcal{F}(S_2)$ sont vérifiées au moins pour la partie S_1 des couples de S_2.

Définition 4.2.5 *On définit l'ordre partiel $\leq_{\mathcal{F}}$ sur les relations par :*

$$S_1 \leq_{\mathcal{F}} S_2 \;\Leftrightarrow\; S_1 \subseteq S_2 \text{ et } S_1 \subseteq \mathcal{F}(S_2)$$

D'après cette définition, S est une bisimulation faible si et seulement si $S \leq_{\mathcal{F}} S$.

Définition 4.2.6 (Bisimulation faible up-to) *Soit H une fonction sur les relations ; on dit que S est une* bisimulation faible up-to *H si H préserve $\leq_{\mathcal{F}}$[1] et si $S \subseteq \mathcal{F}(H(S))$.*

Lemme 4.2.7 *Soient $R = \bigcup_{i \in I} R_i$ et $S = \bigcup_{j \in J} S_j$,*

1. *si $\forall i \in I, \exists j \in J$ tel que $R_i \leq_{\mathcal{F}} S_j$, alors $R \leq_{\mathcal{F}} S$;*
2. *si $\forall i \in I, \exists i' \in I$ tel que $R_i \leq_{\mathcal{F}} R_j$, alors R est une bisimulation asynchrone faible.*

Preuve: (1) De l'hypothèse, on a $R_i \subseteq S$ pour tout i et donc $R \subseteq S$. Il reste à montrer que $R \subseteq \mathcal{F}(S)$. Par monotonie de $\mathcal{F}$ on a $\mathcal{F}(S_j) \subseteq \mathcal{F}(S)$ pour tout j et donc $R_i \subseteq \mathcal{F}(S)$ pour tout i ce qui permet de conclure.
(2) Avec $I = J$ et $S_i = R_i$ pour tout i, de (1) on déduit $R \leq_{\mathcal{F}} R$ et donc $S \subseteq \mathcal{F}(S)$. □

[1] Dans [San95], un tel opérateur est dit *respectfull*.

On définit l'union, la composition et le chaînage d'opérateurs sur les relations comme suit :

$$\begin{array}{rcll} (\bigcup_{i\in I} H_i)(S) & \stackrel{\text{def}}{=} & \bigcup_{i\in I}(H_i(S)) & \textit{union} \\ (H \circ H')(S) & \stackrel{\text{def}}{=} & H(H'(S)) & \textit{composition} \\ (H^\frown H')(S) & \stackrel{\text{def}}{=} & H(S) \circ H'(S) & \textit{chaînage} \end{array}$$

La preuve de la proposition suivante est donnée dans [San95]. Nous la redémontrons tout de même car nos définitions initiales sont légèrement différentes et la preuve obtenue plus simple.

Proposition 4.2.8

1. *Les opérateurs qui préservent $\leq_{\mathcal{F}}$ sont clos par union et composition ;*
2. *Si H préserve $\leq_{\mathcal{F}}$ et S est une bisimulation faible up-to H, alors $S \subseteq \approx$.*

Preuve: La preuve de (1) est directe. Pour montrer (2), on définit l'opérateur $T = \lambda R.(R \cup H(R))$. Par (1), T préserve $\leq_{\mathcal{F}}$. Soit l'ensemble $\{S_n \mid n \geq 0\}$ défini inductivement par

$$\begin{array}{rcl} S_0 & = & S \\ S_{n+1} & = & T(S_n) \end{array}$$

Puisque, H étant monotone, on a $S_0 \leq_{\mathcal{F}} S_0 \cup H(S_0)$ et que T préserve $\leq_{\mathcal{F}}$, alors pour tout $n \geq 0$ on a $S_n \leq_{\mathcal{F}} S_{n+1}$. Par le lemme 4.2.7(2), $\bigcup_{n\geq 0} S_n$ est une bisimulation. Comme $S \subseteq \bigcup_{n\geq 0} S_n$, on conclue $S \subseteq \approx$. □

L'union et la composition d'opérateurs qui préservent $\leq_{\mathcal{F}}$ est donc un opérateur qui préserve aussi $\leq_{\mathcal{F}}$. Dans [San95], D. SANGIORGI montre que dans le cas de la bisimulation synchrone, cette propriété est encore vraie pour le chaînage de tels d'opérateurs. Nous donnons un exemple qui montre que ça n'est plus le cas pour la bisimulation asynchrone. Considérons que notre langage soit étendue avec le choix $+$ et le préfixe τ, et soit la relation $S = \{(\tau,\tau),(\mathbf{0},\mathbf{0})\}$ et l'opérateur $H'_1(S) = \sim \circ S \circ \sim$. Clairement, S est une bisimulation et donc $S \leq_{\mathcal{F}} S$ et on essaie de montrer que $H'_1(S) \leq_{\mathcal{F}} H'_1(S)$. D'après la définition de bisimulation forte, on a $a.\overline{a}+\tau \sim \tau$ donc $(a.\overline{a}+\tau,\tau) \in H'_1(S)$, or $a.\overline{a}+\tau \stackrel{a}{\rightarrow} \overline{a}$ et $\tau \stackrel{\tau}{\rightarrow} \mathbf{0}$ mais $(\overline{a},\mathbf{0} \mid \overline{a}) \notin H'_1(S)$, d'où $H'_1(S) \not\leq_{\mathcal{F}} H'_1(S)$. Donc le chaînage d'opérateurs préservant $\leq_{\mathcal{F}}$ n'est pas toujours un opérateur qui préserve $\leq_{\mathcal{F}}$. On peut néanmoins retrouver une propriété similaire mais il est nécessaire de préalablement appliquer un opérateur de clôture (E) par des messages. Cet opérateur est défini de la manière suivante :

$$E(S) \stackrel{\text{def}}{=} \{(P,Q) \mid P \equiv_P (P_1 \mid M), Q \equiv_P (Q_1 \mid M), (P_1,Q_1) \in S \;\&\; M = \Pi_{i\in I}\overline{a_i}(\vec{b_i})\}$$

où $\equiv_P$ est la congruence induite par associativité et commutativité de la composition parallèle et par $(P \mid \mathbf{0}) \equiv_P P$. Clairement, $\equiv_P$ est une bisimulation et E est un opérateur idempotent.

Lemme 4.2.9 *L'opérateur E préserve $\leq_{\mathcal{F}}$.*

Preuve: E étant clairement un opérateur monotone, nous n'avons qu'à montrer que $S_1 \leq_{\mathcal{F}} S_2$ implique $E(S_1) \subseteq \mathcal{F}(E(S_2))$. Étant donné que $\equiv_P$ est une bisimulation on peut se contenter de ne considérer que les couples de termes $(P \mid M, Q \mid M) \in E(S_1)$ avec PS_1Q et $M = \Pi_{i\in I}\overline{a_i}(\vec{b_i})$. On

montre que ce couple de termes satisfait les conditions de la définition 4.2.1. Puisque $S_1 \subseteq \mathcal{F}(S_2)$ et PS_1Q, on a $(I \Vdash P \Leftrightarrow I \Vdash Q)$ et donc $(I \Vdash (P \mid M) \Leftrightarrow I \Vdash (Q \mid M))$. Inspectons les actions possibles pour $(P \mid M)$.

- $(P \mid M) \xrightarrow{\tau} (P' \mid M)$ avec $P \xrightarrow{\tau} P'$. Puisque $S_1 \subseteq \mathcal{F}(S_2)$, il existe Q' tel que $Q \Rightarrow Q'$ et $(P', Q') \in S_2$. On en déduit que $(Q \mid M) \Rightarrow (Q' \mid M)$ et $(P' \mid M, Q' \mid M) \in E(S_2)$.
- $(P \mid M) \xrightarrow{\tau} (P' \mid M')$ avec $P \xrightarrow{a\vec{b}} P'$ et $M \xrightarrow{\overline{a}\vec{b}} M'$. Alors, il existe Q' tel que, soit $Q \stackrel{a\vec{b}}{\Rightarrow} Q'$ et $(P', Q') \in S_2$, d'où $(Q \mid M) \Rightarrow (Q' \mid M')$ et $(P' \mid M', Q' \mid M') \in E(S_2)$. Soit, $Q \Rightarrow Q'$ et $(P', Q' \mid \overline{a}\vec{b})$, et donc $(Q \mid M) \Rightarrow (Q' \mid M)$ et $(P' \mid M', Q' \mid M' \mid \overline{a}(\vec{b})) \in E(S_2)$ car $M \equiv_P (M' \mid \overline{a}(\vec{b}))$.
- $(P \mid M) \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} (P' \mid M)$ avec $P \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} P'$. Le raisonnement est le même que dans le premier cas.
- $(P \mid M) \xrightarrow{\overline{a}\vec{b}} (P \mid M')$ avec $M \xrightarrow{\overline{a}\vec{b}} M'$. On a alors $Q \mid M \xrightarrow{\overline{a}\vec{b}} Q \mid M'$. D'autre part, de $S_1 \subseteq S_2$ on déduit que $(P, Q) \in S_2$ d'où $(P \mid M', Q \mid M') \in E(S_2)$.
- $(P \mid M) \xrightarrow{a\vec{b}} (P' \mid M)$ avec $P \xrightarrow{a\vec{b}} P'$. Alors, il existe Q' tel que soit $Q \stackrel{a\vec{b}}{\Rightarrow} Q'$ et $(P', Q') \in S_2$ et donc $(Q \mid M) \stackrel{a\vec{b}}{\Rightarrow} (Q' \mid M)$ et $(P' \mid M, Q' \mid M) \in E(S_2)$. Soit $Q \Rightarrow Q'$ et $(P', Q' \mid \overline{a}(\vec{b})) \in S_2$ et donc $(Q \mid M) \Rightarrow (Q' \mid M)$ et $(P' \mid M, Q' \mid M \mid \overline{a}(\vec{b})) \in E(S_2)$. □

Deux propriétés s'en suivent immédiatement.

Lemme 4.2.10

1. $E(\approx) = \approx$;
2. $E(R) \subseteq \approx \Leftrightarrow R \subseteq \approx$.

Proposition 4.2.11 *Si H et H' préservent $\leq_{\mathcal{F}}$, alors $(E \circ H)^{\frown}(E \circ H')$ préserve $\leq_{\mathcal{F}}$.*

Preuve: Soient S_1 et S_2 tels que $S_1 \leq_{\mathcal{F}} S_2$. Par le lemme 4.2.9 et la proposition 4.2.8 (1), on a $E(H(S_1)) \leq_{\mathcal{F}} E(H(S_2))$ et $E(H'(S_1)) \leq_{\mathcal{F}} E(H'(S_2))$, et on peut facilement en déduire que $E(H(S_1)) \circ E(H'(S_1)) \subseteq E(H(S_2)) \circ E(H'(S_2))$. Il reste à montrer que $E(H(S_1)) \circ E(H'(S_1)) \subseteq \mathcal{F}(E(H(S_2)) \circ E(H'(S_2)))$. Soit $(P, Q) \in E(H(S_1)) \circ E(H'(S_1))$, et R tel que $(P, R) \in E(H(S_1))$ et $(R, Q) \in E(H'(S_1))$. Il est clair que $I \Vdash P \Leftrightarrow I \Vdash Q$.

- Si $P \xrightarrow{\alpha} P'$ et α est une action interne ou d'émission, puisque $E(H(S_1)) \leq_{\mathcal{F}} E(H(S_2))$, il existe R' tel que $R \stackrel{\alpha}{\Rightarrow} R'$ $(*)$ et $(P', R') \in E(H(S_2))$. D'autre part de $(*)$ et $E(H'(S_1)) \leq_{\mathcal{F}} E(H'(S_2))$, il existe Q' tel que $Q \stackrel{\alpha}{\Rightarrow} Q'$ et $(R', Q') \in E(H'(S_2))$. Finalement, $(P', Q') \in (E(H(S_2)) \circ E(H'(S_2)))$.
- Si $P \xrightarrow{a\vec{b}} P'$, il existe R' tel que soit (1) $R \stackrel{a\vec{b}}{\Rightarrow} R'$ et $(P', R') \in E(H(S_2))$, soit (2) $R \Rightarrow R'$ et $(P', (R' \mid \overline{a}(\vec{b})) \in E(H(S_2))$.
  1. Il existe Q' tel que soit $Q \stackrel{a\vec{b}}{\Rightarrow} Q'$ et $(R', Q') \in E(H'(S_2))$, et on conclue directement. Soit $Q \Rightarrow Q'$ et $(R', (Q' \mid \overline{a}(\vec{b}))) \in E(H'(S_2))$, d'où $(P', (Q' \mid \overline{a}(\vec{b}))) \in E(H(S_2)) \circ E(H'(S_2))$.
  2. Il existe Q' tel que $Q \Rightarrow Q'$ et $(R', Q') \in E(H'(S_2))$. E étant idempotent, on en déduit $((R' \mid \overline{a}(\vec{b})), (Q' \mid \overline{a}(\vec{b}))) \in E(H'(S_2))$, ce qui nous permet de conclure. □

Avant de donner d'autres bisimulations up-to, nous définissons deux relations sur les processus que nous allons montrer être une bisimulation forte pour la première et faible pour la seconde. Un contexte d'évaluation est un terme de la grammaire suivante :

$$\mathbf{E} \quad ::= \quad [] \mid (\mathbf{E} \mid P) \mid (\nu w)\mathbf{E}$$

Définition 4.2.12 *On définit la relation $\cong$ sur les processus comme étant la plus petite équivalence induite par α-conversion, associativité et commutativité de la composition parallèle et par :*

(a)	$(\mathbf{0} \mid P) \cong P$	*(b)*	$((\nu w)P \mid Q) \cong (\nu w)(P \mid Q)$	*si* $subj(w) \notin fn(Q)$
(c)	$(\nu a)T_a \cong \mathbf{0}$	*(d)*	$(\nu a : val)P \cong P$	*si* $a \notin fn(P)$
(e)	$(P \cong Q) \Rightarrow (\mathbf{E}[P] \cong \mathbf{E}[Q])$	*(f)*	$(\nu w w')P \cong (\nu w' w)P$	

Lemme 4.2.13 *$\cong$ est une bisimulation.*

Preuve: Il suffit de montrer que si $P \cong Q$, alors pour toute action α telle que $P \xrightarrow{\alpha} P'$ il existe Q' tel que $Q \xrightarrow{\alpha} Q'$ et $P' \cong Q'$. En d'autres termes, on montre que $\cong$ est une bisimulation synchrone. □

On introduit la notion de *réduction déterministe* : $P >_d P'$ si

$$P \cong \mathbf{E}[(\nu a)(\overline{a}(\vec{b}) \mid a(\vec{c}).Q)] \xrightarrow{\tau} \mathbf{E}[(\mathbf{0} \mid ([\vec{b}/\vec{c}]Q \mid Id_a))] \cong P' \cong \mathbf{E}[[\vec{b}/\vec{c}]Q]$$

où $a \notin fn(Q)$ et $\mathbf{E}$ est un contexte d'évaluation. On écrit aussi $\geq_d$ pour $>_d \cup \cong$, $<_d$ et $\leq_d$ pour respectivement $(>_d)^{-1}$ et $(\geq_d)^{-1}$. Le lemme suivant justifie la dénomination « déterministe ».

Lemme 4.2.14 *Si $P \xrightarrow{\alpha} P'$ et $P >_d Q$ alors, soit $P' \cong Q$, soit il existe Q' tel que $Q \xrightarrow{\alpha} Q'$ et $P' >_d Q'$.*

Lemme 4.2.15 *$\leq_d$ et $\geq_d$ sont des bisimulations faibles.*

Preuve: Il suffit de faire la preuve pour $\geq_d$, c'est-à-dire montrer que $\geq_d \subseteq \mathcal{F}(\geq_d)$. Soient P et Q tels que $P \geq_d Q$. Il est facile de voir que I, $I \Vdash P$ ssi $I \Vdash Q$. D'autre part, si $P \xrightarrow{\alpha} P'$, par le lemme 4.2.14, soit $Q \cong P'$ et donc $Q \geq_d P'$, soit il existe Q' tel que $Q \xrightarrow{\alpha} Q'$ et $P' \geq_d Q'$. On en déduit que (P, Q) satisfont les clauses de $\mathcal{F}(\geq_{\mathcal{F}})$ et donc $(P, Q) \in \mathcal{F}(\geq_{\mathcal{F}})$.
Il reste à montrer que si $P \geq_{\mathcal{F}} Q$ alors $(Q, P) \in \mathcal{F}(\geq_{\mathcal{F}})$. Supposons que $Q \xrightarrow{\alpha} Q'(*)$, montrons qu'il existe P' tel que $P \stackrel{\alpha}{\Rightarrow} P'$ et $Q' \geq_{\mathcal{F}} P'$. De $P \geq_{\mathcal{F}} Q$, et du fait que $\cong$ est une bisimulation forte on en déduit qu'il existe P'' tel que $P \xrightarrow{\tau} P''$ et $P'' \cong Q$. Et à nouveau par le lemme 4.2.13 et (*), il existe P' tel que $P'' \xrightarrow{\alpha} P'$ et $P' \cong Q'$, d'où $Q' \geq_{\mathcal{F}} P'$. □

On définit les opérateurs suivants sur les relations :

$$\begin{aligned}
H_1(S) &\stackrel{\text{def}}{=} \sim \circ E(S) \circ \sim \\
H_2(S) &\stackrel{\text{def}}{=} \geq_d \circ S \circ \leq_d \\
H_3(S) &\stackrel{\text{def}}{=} \{(P, Q) \mid P \cong (\nu \vec{w})(P_1 \mid T_a), Q \cong (\nu \vec{w})(Q_1 \mid T_a) \;\&\; (P_1, Q_1) \in S\}
\end{aligned}$$

Lemme 4.2.16 *1. L'identité et les opérateurs constants associant une relation S à $\sim$, $\geq_d$ et $\leq_d$ respectivement, préservent $\leq_{\mathcal{F}}$.*

2. Les opérateurs E, H_1, H_2 et H_3 préservent $\leq_{\mathcal{F}}$.

Preuve: (1) $\sim$ est aussi une bisimulation faible. Pour $\geq_d$ et $\leq_d$ le lemme 4.2.15 nous permet de conclure directement.
(2) Tous les opérateurs étant monotones, nous n'avons qu'à montrer que $S_1 \leq_{\mathcal{F}} S_2$ implique $H(S_1) \subseteq \mathcal{F}(H(S_2))$.

($H_{1,2}$) Direct à partir de (1) et de la proposition 4.2.8(1).

(H_3) Encore une fois étant donné que $\cong$ est une bisimulation on peut se contenter de ne considérer que les couples de termes $((\nu\vec{w})(P \mid T_a), (\nu\vec{w})(Q \mid T_a)) \in H_3(S_1)$ avec PS_1Q. Les cas où $(\nu\vec{w})(P \mid T_a)$ réalise une action interne ou une action d'émission se traitent facilement. Considérons le cas où $(\nu\vec{w})(P \mid T_a) \xrightarrow{b\vec{c}} (\nu\vec{w})(P' \mid T_a)$ avec $P \xrightarrow{b\vec{c}} P'$. Puisque a n'est pas dans l'interface de P (car $(P \mid T_a)$ doit être bien formé et que $a \Vdash T_a$), par la proposition 3.4.1 on a $b \neq a$. D'autre part, de $(P, Q) \in E(S_2)$, on a soit $Q \stackrel{b\vec{c}}{\Rightarrow} Q'$ avec $P'S_2Q'$ et on conclue directement. Soit, $Q \Rightarrow Q'$ avec $P'S_2(Q' \mid \bar{b}(\vec{c}))$ et donc $(\nu\vec{w})(Q \mid T_a) \Rightarrow (\nu\vec{w})(Q' \mid T_a)$ et $((\nu\vec{w})(P' \mid T_a), (\nu\vec{w})(Q \mid T_a \mid \bar{b}(\vec{c}))) \in H_3(S_2)$. De plus, étant donné que $\mathit{subj}(\vec{w}) \cap \mathit{fn}(\bar{b}(\vec{c})) = \emptyset$, on a $(\nu\vec{w})(Q \mid T_a \mid \bar{b}(\vec{c})) \cong ((\nu\vec{w})(Q \mid T_a) \mid \bar{b}(\vec{c}))$ et donc finalement $((\nu\vec{w})(P' \mid T_a), ((\nu\vec{w})(Q \mid T_a) \mid \bar{b}(\vec{c}))) \in H_3(S_2)$. □

CHAPITRE 5

Expressivité de π_1^r

POUR MONTRER QUE π_1^r CONSERVE TOUTE L'EXPRESSIVITÉ DU π-CALCUL, il est suffisant de coder π_1 dans π_1^r. En effet, dans [Ama00], R.AMADIO donne un codage de la réception du Join-calcul dans π_1 à bisimulation asynchrone faible près. D'autre part, dans [FG96] les auteurs montrent qu'il y a une traduction *fully-abstract* du π-calcul asynchrone dans le Join-calcul. Pour coder π_1 dans π_1^r nous procédons en deux temps. Tout d'abord nous codons les termes de π_1 dans π_1^r dont la syntaxe est étendue avec des gestionnaires de canaux et la sémantique opérationnelle est enrichie de règles de transitions décrivant le comportement de ces gestionnaires. Pour vérifier la correction de ce codage nous montrons que deux processus de π_1 sont faiblement bisimilaires si et seulement leurs traductions le sont aussi. Dans un deuxième temps, nous montrons que les gestionnaires de canaux peuvent être implantés dans π_1^r (non étendu) et que la substitution des gestionnaires de canaux par leur implantation n'affecte pas la bisimilarité. Ces deux étapes du codage sont les deux sections de ce chapitre.

5.1 Codage de π_1^r

Dans le codage présenté ici, nous allons utiliser quelques abréviations. La construction conditionnelle $\mathsf{if}\ C\ \mathsf{then}\ P\ \mathsf{else}\ Q$, où C est une combinaison booléenne d'égalités de noms, est définie par :

$$
\begin{aligned}
\mathsf{if}\ [a=b]\ \mathsf{then}\ P\ \mathsf{else}\ Q &\stackrel{\mathsf{def}}{=} [a=b]P, Q \\
\mathsf{if}\ [a\neq b]\ \mathsf{then}\ P\ \mathsf{else}\ Q &\stackrel{\mathsf{def}}{=} [a=b]Q, P \\
\mathsf{if}\ \neg C\ \mathsf{then}\ P\ \mathsf{else}\ Q &\stackrel{\mathsf{def}}{=} \mathsf{if}\ C\ \mathsf{then}\ Q\ \mathsf{else}\ P \\
\mathsf{if}\ C\vee C'\ \mathsf{then}\ P\ \mathsf{else}\ Q &\stackrel{\mathsf{def}}{=} \mathsf{if}\ C\ \mathsf{then}\ P\ \mathsf{else}\ \mathsf{if}\ C'\ \mathsf{then}\ P\ \mathsf{else}\ Q \\
\mathsf{if}\ C\wedge C'\ \mathsf{then}\ P\ \mathsf{else}\ Q &\stackrel{\mathsf{def}}{=} \mathsf{if}\ C\ \mathsf{then}\ (\mathsf{if}\ C'\ \mathsf{then}\ P\ \mathsf{else}\ Q)\ \mathsf{else}\ Q
\end{aligned}
$$

Dans π_1^r (où on omet les arités des processus paramétrés), nous utilisons la notation « _ » pour des canaux par défaut dans différents contextes :

- dans une émission $\overline{a}(\vec{b}, \_, \vec{b'})$ représente $(\nu c)(\overline{a}(\vec{b}, c, \vec{b'}) \mid T_c)$ ou $(\nu c : \mathit{val})(\overline{a}(\vec{b}, c, \vec{b'})$ suivant le type de a ;
- dans une réception $a(\vec{b}, \_, \vec{b'}).P$ représente $a(\vec{b}, c, \vec{b'}).P$ où $c \notin \mathit{fn}(P)$;
- dans l'instanciation d'un processus paramétré $T(\_, \vec{b})$ représente $(\nu a)T(a, \vec{b})$, qui est bien formé si $\Vdash T$;
- dans les définitions récursive $(\mathsf{rec}\, A(\_, \vec{b}).P)$ représente $(\mathsf{rec}\, A(a, \vec{b}).(T_a \mid P))$, qui est bien formé si $\Vdash P$.

Dans les preuves on utilise aussi la notation suivante sur les actions :

$$P \xrightarrow{(\nu \vec{c})\overline{a}\vec{b}, \_, \vec{b'}} Q \quad \stackrel{\mathsf{def}}{=} \quad P \xrightarrow{(\nu \vec{c}, c:val)\overline{a}\vec{b}, c, \vec{b'}} Q$$

où $c \notin \mathit{fn}(P) \cup \{\vec{c}\}$.

L'idée du codage est de convertir tout message sur un canal a en une requête au gestionnaire $CM(a)$ du canal a en lui envoyant les arguments du message accompagnés d'une clef *out*. Symétriquement, on convertit une réception sur a en une requête à $CM(a)$ en lui envoyant une clef *in* et un canal de retour privé sur lequel on va effectivement recevoir un message. Le gestionnaire de canal filtre les messages en fonction des clefs et réagit en conséquence. Cette technique est similaire à celle utilisée pour l'implantation des communications dans des langages semblables au π-calcul (par exemple dans [Car85, Tur96] où, cependant, on utilise des structures de données plus élaborées pour gérer les communications de manière plus réaliste). Malheureusement ce codage souffre d'un inconvénient : l'environnement peut faire des requêtes de réception aux gestionnaires de canaux ce qui compromet l'abstraction. Pour y palier nous authentifions toutes les requêtes de réception en introduisant des clefs privées in_a pour chaque gestionnaire de canal et qui ne sont connues que par les processus qui peuvent effectivement recevoir sur ce canal.

Formellement, pour chaque nom a on suppose un nom frais in_a, c'est-à-dire qui n'a pas d'occurrence dans le processus traduit. Le nom in_a a le type *val* et est utilisé comme clef du canal a. On traduit alors un terme P de π_1 avec interface I, où $I = \{a_1, \ldots, a_n\}$, dans le processus suivant :

$$(\!|I, P|\!) = (\nu \mathit{in}_{a_1} : \mathit{val}, \ldots, \mathit{in}_{a_n} : \mathit{val})(CM(a_1, \mathit{in}_{a_1}) \mid \ldots \mid CM(a_n, \mathit{in}_{a_n}) \mid [\![P]\!])$$

où $[\![P]\!]$ est défini plus bas. $(\!|I, P|\!)$ devient alors un processus bien formé de π_1^r avec une interface vide. Les types sont traduits de la manière suivante :

$$\begin{aligned} [\![\mathit{val}]\!] &= \mathit{val} \\ [\![\mathit{Ch}(\tau_1, \ldots, \tau_n)]\!] &= \mathit{Ch}(\mathit{val}, [\![\tau_1]\!], \ldots, [\![\tau_n]\!], \mathit{Ch}([\![\tau_1]\!], \ldots, [\![\tau_n]\!]), \mathit{val}, \mathit{val}) \end{aligned}$$

où le premier argument est le type (*val*) de la clef d'émission/réception, ensuite viennent les types des arguments du message à délivrer, suivi du type du canal de retour sur lequel ils doivent effectivement être envoyés, et enfin les types (*val*) de deux clefs pour un choix interne. Les règles de transition suivantes décrivent le comportement d'un gestionnaire de canal (on suppose $j_1 \neq \mathit{in}_a$) :

$$(CM_\tau) \quad (CM(a, in_a) \mid \overline{a}(j_1, \vec{b_1}, r_1, c_1, c_1') \mid \overline{a}(in_a, \vec{b_2}, r_2, c_2, c_2')) \xrightarrow{\tau} (CM(a, in_a) \mid \overline{r_2}(\vec{b_1}))$$
$$(CM_{in}) \quad (CM(a, in_a) \mid \overline{a}(in_a, \vec{b_2}, r_2, c_2, c_2')) \xrightarrow{a(j_1, \vec{b_1}, r_1, c_1, c_1')} (CM(a, in_a) \mid \overline{r_2}(\vec{b_1}))$$

Dans cette spécification, le gestionnaire de canal fait correspondre une requête de réception à une requête d'émission, cette dernière pouvant être faite par l'environnement. Dans un premier temps, pour simplifier les preuves, nous raisonnons à partir de cette spécification axiomatique. En d'autres termes, on étend π_1^r avec une nouvelle constante $CM(a, in_a)$ (où les deux paramètres sont libres), dont le comportement est décrit par les règles (CM_τ) et (CM_{in}) et telle que $a \Vdash^r CM(a, in_a)$. Dans la deuxième section nous verrons comment implanter le gestionnaire de canal dans π_1^r à bisimulation asynchrone faible près. Afin de pouvoir utiliser la spécification de CM on doit ajouter une règle de transition permettant les manipulations structurelles :

$$P \xrightarrow{\alpha} P' \,\&\, Q \equiv P \;\Rightarrow\; Q \xrightarrow{\alpha} P$$

L'équivalence structurelle $\equiv$ est essentiellement définie comme $\cong$ (définition 4.2.12).

Définition 5.1.1 (Équivalence Structurelle) *L'équivalence structurelle est la plus petite relation d'équivalence induite par associativié et commutativité de la composition parallèle et par les axiomes suivants :*

(a) $(\mathbf{0} \mid P) \equiv P$ *(b)* $((\nu w)P \mid Q) \equiv (\nu w)(P \mid Q)$ *si* $subj(w) \notin fn(Q)$.
(c) $(\nu w w')P \equiv (\nu w' w)P$ *(d)* $(\mathsf{rec}\, A(\vec{b}).P)(\vec{c}) \equiv [(\mathsf{rec}\, A(\vec{b}).P)/_A][\vec{c}/\vec{b}]P$
(e) $[a = a]P, Q \equiv P$ *(f)* $[a = b]P, Q \equiv Q$ *si* $a \neq b$
(g) $P \equiv Q \Rightarrow (\mathbf{E}[P] \equiv \mathbf{E}[Q])$

Le dépliage de la récursion et du branchement conditionnel ont été ajoutés à l'équivalence structurelle, telle qu'elle est traditionnellement définie, pour permettre l'application des transitions CM_τ et CM_{in} lorsque qu'une requête de réception émane d'un processus « encapsulé » dans une récursion ou un branchement conditionnel. Ces dépliages ne sont évidemment possible qu'au niveau « haut », c'est-à-dire qu'il ne s'appliquent pas sous une réception. On peut montrer que, comme le lemme 4.2.13, l'équivalence structurelle reste une bisimulation forte.

La traduction $[\![P]\!]$ des processus est définie comme suit :

$$\begin{aligned}
[\![\mathbf{0}]\!] &= \mathbf{0} \\
[\![\overline{a}(\vec{b})]\!] &= \overline{a}(_, \vec{b}, _, _, _) \\
[\![a(\vec{b}).P]\!] &= (\nu r)(\overline{a}(in_a, _, r, _, _) \mid r(\vec{b}){:}[\![P]\!]) \\
[\![P \mid Q]\!] &= ([\![P]\!] \mid [\![Q]\!]) \\
[\![[a = b]P, Q]\!] &= [a = b][\![P]\!], [\![Q]\!] \\
[\![(\nu a)P]\!] &= (\nu a)(\nu in_a : val)(CM(a, in_a) \mid [\![P]\!]) \\
[\![(\nu a : val)P]\!] &= (\nu a : val)[\![P]\!]
\end{aligned}$$

Pour la traduction des processus paramétrés on suppose l'existence d'une fonction injective qui associe à chaque identificateur avec arité A^k de π_1, un identificateur de π_1^r (qu'on notera A_k). De plus,

on suppose, dans les clauses suivantes, que T possède l'arité k, c'est-à-dire que T est soit A^k soit $(\mathsf{rec}\, A^k(\vec{b}).P)$, et $\vec{b} = \vec{a}, \vec{c}$ où $\vec{a}$ est de longueur k si la longueur de $\vec{b}$ est supérieur ou égale à k, et $\vec{a} = \vec{b}$ (et $\vec{c}$ est vide) sinon.

$$\begin{aligned} [\![T(\vec{b})]\!] &= [\![T]\!](\_, \vec{a}, \vec{in_a}, \vec{c}) \\ [\![A^k]\!] &= A_k \\ [\![(\mathsf{rec}\, A^k(\vec{b}).P)]\!] &= (\mathsf{rec}\, A_k(\_, \vec{a}, \vec{in_a}, \vec{c}).[\![P]\!]) \end{aligned}$$

On note que les seuls récepteurs dans cette traduction sont les gestionnaires de canaux et les réceptions linéaires sur canaux de retour $r(\vec{b})$:$[\![P]\!]$. Cette traduction préserve la bonne formation :

Lemme 5.1.2 *Si $I \Vdash P$ alors $\Vdash^r [\![P]\!]$ et $I \Vdash (\!|I, P|\!)$.*

Preuve: Par induction sur la preuve de $I \Vdash P$. □

Lemme 5.1.3 *Supposons que $I \Vdash P$ et soit S une substitution telle que $[\mathsf{S}]P$ est défini, et S injective sur I. Alors $[\mathsf{S}][\![P]\!] = [\![[\mathsf{S}]P]\!]$ et $[\mathsf{S}](\!|I, P|\!) = (\!|\mathsf{S}(I), [\mathsf{S}]P|\!)$.*

On montre maintenant que cette traduction est fully-abstract vis-à-vis de la bisimulation asynchrone. Le lemme suivant établit que $(\!|I, P|\!)$ « simule » P.

Lemme 5.1.4 *Supposons $I \Vdash P$. Alors,*

1. *si $P \xrightarrow{\tau} P'$ alors $(\!|I, P|\!) \xrightarrow{\tau} Q >_d (\!|I, P'|\!)$;*
2. *si $P \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} P'$, $I' \Vdash P'$ et $a \notin I$ alors $(\!|I, P|\!) \xrightarrow{(\nu\vec{w}, r)\overline{a}\_,\vec{b},r,\_,\_} ((\!|I', P'|\!) \mid T_r)$;*
3. *si $P \xrightarrow{a\vec{b}} P'$ alors $(\!|I, P|\!) \xrightarrow{a(j,\vec{b},r,c,c')} Q >_d (\!|I, P'|\!)$.*

Preuve: On procède par induction sur la preuve de $P \xrightarrow{\alpha} P'$. On ne traite que les trois transitions les plus significatives (*out*), (*in*) et (*cm*).

(*out*) Soit $\overline{a}(\vec{b}) \xrightarrow{\overline{a}\vec{b}} \mathbf{0}$ et $\Vdash \overline{a}(\vec{b})$. On a $(\!|\emptyset, \overline{a}(\vec{b})|\!) = (\nu r)(\overline{a}(\_, \vec{b}, r, \_, \_) \mid T_r)$ et par (*ext*), (*cp*) et (*out*) on peut inférer $(\nu r)(\overline{a}(\_, \vec{b}, r, \_, \_) \mid T_r) \xrightarrow{(\nu r)\overline{a}\_,\vec{b},r,\_,\_} (\mathbf{0} \mid T_r)$ ce qui montre le cas de base de (2).

(*in*) Soit $a(\vec{b}).P \xrightarrow{a\vec{c}} [\vec{c}/\vec{b}]P$ et $a \Vdash P$. Par la traduction de la réception on a

$$(\!|\{a\}, a(\vec{b}).P|\!) \equiv (\nu \mathit{in}_a : \mathit{val}, r)((CM(a, \mathit{in}_a) \mid (\overline{a}(\mathit{in}_a, \_, r, \_, \_)) \mid r(\vec{b}):[\![P]\!])$$

Par (ν), (*cp*) et (CM_{in}) on a donc la réduction

$$(\!|\{a\}, a(\vec{b}).P|\!) \xrightarrow{a(j,\vec{c},r',c,c')} (\nu \mathit{in}_a : \mathit{val}, r)(CM(a, \mathit{in}_a) \mid \overline{r}(\vec{c}) \mid r(\vec{b}):[\![P]\!]) = Q$$

De plus, $Q \equiv (\nu \mathit{in}_a : \mathit{val})(CM(a, \mathit{in}_a) \mid (\nu r)(\overline{r}(\vec{c}) \mid r(\vec{b}):[\![P]\!]))$ d'où

$$Q >_d (\nu \mathit{in}_a : \mathit{val})(CM(a, \mathit{in}_a) \mid [\vec{c}/\vec{b}][\![P]\!])$$

Enfin, par le lemme 5.1.3, on a $(\nu \mathit{in}_a : \mathit{val})(CM(a, \mathit{in}_a) \mid [\vec{c}/\vec{b}][\![P]\!]) = (\!|\{a\}, [\vec{c}/\vec{b}]P|\!)$ ce qui montre le cas de base de (3).

(*cm*) Soit $(P \mid Q) \xrightarrow{\tau} (\nu\vec{w})(P' \mid Q')$ avec $P \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} P'$ et $Q \xrightarrow{a\vec{b}} Q'$; et soient $I = I_1, I_2$ tels que $I_1 \Vdash P$ et $I_2 \Vdash Q$ avec $I_1 \cap I_2 = \emptyset$. Puisque $I_1 \cap I_2 = \emptyset$, on a $(\!|I, (P \mid Q)|\!) \equiv (\!|I_1, P|\!) \mid (\!|I_2, Q|\!)$. Nécessairement $a \in I_2$, donc $a \notin I_1$ et donc, par hypothèse d'induction, on a également

$$(\!|I_1, P|\!) \xrightarrow{(\nu\vec{w}, r, \vec{w'})\overline{a}j,\vec{b},r,c,c'} ((\!|I_1', P'|\!) \mid T_r) \quad \text{et} \quad (\!|I_2, Q|\!) \xrightarrow{a(j,\vec{b},r,c,c')} R >_d (\!|I_2, Q'|\!)$$

où $I' \Vdash P'$ et $\vec{w'} = (j : \mathit{val}, c : \mathit{val}, c' : \mathit{val})$ avec $\{j, c, c'\} \cap \mathit{fn}((\!|I_1, P|\!) \mid (\!|I_2, Q|\!)) = \emptyset$. On en déduit

$$(\!|I_1, P|\!) \mid (\!|I_2, Q|\!) \xrightarrow{\tau} (\nu\vec{w}, r, \vec{w'})((\!|I_1', P'|\!) \mid T_r \mid R) >_d (\nu\vec{w}, r)((\!|I_1', P'|\!) \mid T_r \mid (\!|I_2, Q'|\!))$$

Il reste à montrer que $(\nu\vec{w}, r)((\!|I_1', P'|\!) \mid T_r \mid (\!|I_2, Q'|\!)) \cong (\!|I, (\nu\vec{w})(P' \mid Q')|\!)$ $(*)$. En remarquant que $r \notin \mathit{fn}((\!|I_1', P'|\!) \mid (\!|I_2, Q'|\!))$ (voir les deux cas précédents), on a

$$\begin{aligned}(\nu\vec{w}, r)((\!|I_1', P'|\!) \mid T_r \mid (\!|I_2, Q'|\!)) &\cong (\nu\vec{w})((\!|I_1', P'|\!) \mid (\nu r)T_r \mid (\!|I_2, Q'|\!)) \\ &\cong (\nu\vec{w})((\!|I_1', P'|\!) \mid (\!|I_2, Q'|\!))\end{aligned}$$

Puisque $I_1' \Vdash P'$, par le lemme 3.3.6 on a $I_1' = I_1, J$ avec $J = \mathit{bcn}((\nu\vec{w})\overline{a}\vec{b})$. Par la définition de la traduction il est facile de montrer que

$$(\!|I, (\nu\vec{w})(P' \mid Q')|\!) \equiv (\nu\vec{w})(\!|I \cup J, (P' \mid Q')|\!) \equiv (\nu\vec{w})((\!|I_1', P'|\!) \mid (\!|I_2, Q'|\!))$$

ce qui montre $(*)$. □

Dans le cas de l'émission (2), on remarque l'introduction du processus absorbant T_r agissant sur un nom de canal frais r. On montre que ces processus superflus peuvent être factorisés. En particulier on observe la propriété suivante.

Lemme 5.1.5 *Supposons $r \notin \mathit{fn}(P \mid Q)$. Alors $P \approx Q$ si et seulement si $(P \mid T_r) \approx (Q \mid T_r)$.*

Preuve: $(\Rightarrow)$ On observe que r ne peut pas être dans l'interface de P et Q. De plus, $\approx$ est conservée par composition parallèle qui préserve la bonne formation.
$(\Leftarrow)$ On montre que la relation $S = \{(P, Q) \mid (P \mid T_r) \approx (Q \mid T_r)\ \&\ r \notin \mathit{fn}(P \mid Q)\}$ est une bisimulation faible. □

Puisque la clef in_a est gardée secrète, un message $\overline{a}(j, \vec{b}, r, c, c')$ reçu par un processus $(\!|I, P|\!)$ est toujours interprété comme une requête d'émission. Par conséquent, les champs j, r, c et c' sont hors de propos. Cette remarque est formalisée par le lemme suivant.

Lemme 5.1.6 *Soient $I \Vdash P$ et $a \in I$. Alors pour tout j, r, c, c' on a*

$$(\!|I, (P \mid \overline{a}(\vec{b}))|\!) \sim ((\!|I, P|\!) \mid \overline{a}(j, \vec{b}, r, c, c'))$$

Preuve: Soit J tel que $I = \{a\} \cup J$ et $a \notin J$, on a

$$\begin{aligned}(\!|I, (P \mid \overline{a}(\vec{b}))|\!) &\equiv (\nu \mathit{in}_a : \mathit{val})(CM(a, \mathit{in}_a) \mid \overline{a}(\_, \vec{b}, \_, \_, \_) \mid (\!|J, P|\!)) \\ ((\!|I, P|\!) \mid \overline{a}(j, \vec{b}, r, c, c')) &\equiv (\nu \mathit{in}_a : \mathit{val})(CM(a, \mathit{in}_a) \mid \overline{a}(j, \vec{b}, r, c, c') \mid (\!|J, P|\!))\end{aligned}$$

avec $j \neq in_a$. Soit $M = \Pi_{i\in I}\overline{a_i}(j_i, \vec{b_i}, r_i, c_i, c'_i)$. On observe que

$$(CM(a, in_a) \mid \overline{a}(\_, \vec{b}, \_, \_, \_) \mid M) \quad \sim \quad (CM(a, in_a) \mid \overline{a}(j, \vec{b}, r, c, c') \mid M)$$

pourvu que $j \neq in_a$. □

Vis-à-vis des transitions réalisables par $(\!|I, P|\!)$, on a les propriétés suivantes :

Lemme 5.1.7 *Supposons que $I \Vdash P$. Alors*

1. *si $(\!|I, P|\!) \xrightarrow{\tau} Q$ alors $P \xrightarrow{\tau} P'$ et $Q >_d (\!|I, P'|\!)$;*
2. *si $(\!|I, P|\!)$ réalise une action d'émission sur un canal $a \notin I$ alors $(\!|I, P|\!) \xrightarrow{(\nu\vec{w}, r)\overline{a}\_, \vec{b}, r, \_, \_} (Q \mid T_r)$ et $P \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} P'$ pour I' et P' tels que $I' \Vdash P'$ et $Q \cong (\!|I', P'|\!)$;*
3. *si $(\!|I, P|\!) \xrightarrow{a(j,\vec{b},r,c,c')} Q'$ alors $P \xrightarrow{a\vec{b}} P'$ pour P' tel que $Q' >_d (\!|I, P'|\!)$.*

Preuve: Supposons que $I = \{a_1, \ldots, a_n\}$, on a alors

$$(\!|I, P|\!) = (\nu in_{a_1} : val, \ldots, in_{a_n} : val)(CM(a_1, in_{a_1}) \mid \ldots \mid CM(a_n, in_{a_n}) \mid [\![P]\!])$$

1. Les seuls récepteurs prêts à recevoir sont les gestionnaires de canaux. Par conséquent, la seule transition possible sur une action τ est (CM_τ). Donc, il existe un canal a tel que

$$\begin{array}{rcl} (\!|I, P|\!) & \equiv & \mathbf{E}[CM(a, in_a) \mid \overline{a}(j_1, \vec{b_1}, r_1, c_1, c'_1) \mid \overline{a}(in_a, \vec{b_2}, r_2, c_2, c'_2))] \\ & \xrightarrow{\tau} & (CM(a, in_a) \mid \overline{r_2}(\vec{b_1})) \equiv Q \end{array}$$

   On procède par induction sur la structure de P et de $\mathbf{E}$ et sur la dérivation de $(\!|I, P|\!) \xrightarrow{\tau} Q$ pour montrer que $P \xrightarrow{\tau} P'$ et $Q \geq_d (\!|I, P'|\!)$.

2. Un message qui émane de $(\!|I, P|\!)$ sur un canal $a \notin I$ est nécessairement réalisé par $[\![P]\!]$ et, le type de a étant $Ch(val, \tau_1, \ldots, \tau_n, Ch(\tau_1, \ldots, \tau_n), val, val)$, l'action réalisée est de la forme $(\nu\vec{w'})\overline{a}(j, \vec{b}, r, c, c')$. De plus, pour tout $a_i \in I$, $j \neq in_{a_i}$. On a donc

$$(\!|I, P|\!) \xrightarrow{(\nu\vec{w'})\overline{a}(j,\vec{b},r,c,c')} R \quad \text{et} \quad R \equiv (CM(a_1, in_{a_1}) \mid \ldots \mid CM(a_n, in_{a_n}) \mid R')$$

   avec $[\![P]\!] \xrightarrow{(\nu\vec{w'})\overline{a}(j,\vec{b},r,c,c')} R'$ $(*)$. On termine la preuve par induction sur la structure de P et sur la dérivation de $(*)$ en montrant que $P \xrightarrow{\overline{a}\vec{b}} P'$ et $R' \equiv [\![P']\!] \mid T_r$. Regardons juste le cas où P est le message $\overline{a}(\vec{b})$, alors

$$[\![\overline{a}(\vec{b})]\!] = (\nu r)(\overline{a}(\_, \vec{b}, r, \_, \_)) \mid T_r) \xrightarrow{(\nu r)\overline{a}(\_,\vec{b},r,\_,\_)} (\mathbf{0} \mid T_r)$$

   On a bien $(\!|\emptyset, \overline{a}(\vec{b})|\!) \xrightarrow{(\nu r)\overline{a}(\_,\vec{b},r,\_,\_)} \mathbf{0} \mid T_r$ et $\overline{a}(\vec{b}) \xrightarrow{\overline{a}(\vec{b})} \mathbf{0}$.

3. Si $(\!|I,P|\!) \xrightarrow{a(j,\vec{b},r,c,c')} Q'$, nécessairement c'est par la règle CM_{in} que cette action est réalisée. Par conséquent, $(\!|I,P|\!) \equiv \mathbf{E}[CM(a,in_a) \mid \overline{a}(in_a,\vec{b'},r',c'',c''')]$. On termine la preuve la induction sur P et sur $\mathbf{E}$ pour montrer que $P \xrightarrow{a(\vec{b})} P'$ et $Q \geq_d (\!|I,P'|\!)$. Regardons juste le cas où $P = a(\vec{b'}).Q$. On a alors $\{a\} \Vdash P$ et $P \xrightarrow{a(\vec{b})} [\vec{b}/\vec{b'}]Q$. De plus,

$$\begin{array}{rcll}
(\!|\{a\},P|\!) & \equiv & (\nu in_a : val, r')(CM(a,in_a) \mid \overline{a}(in_a,_,r',_,_) \mid r'(\vec{b'}){:}[\![Q]\!]) & \\
& \xrightarrow{a(j,\vec{b},r,c,c')} & (\nu in_a : val, r)(CM(a,in_a) \mid \overline{r'}(\vec{b}) \mid r'(\vec{b'}){:}[\![Q]\!]) & \\
& \geq_d & (\nu in_a : val, r)(CM(a,in_a) \mid [\vec{b}/\vec{b'}][\![Q]\!]) & \\
& = & (\nu in_a : val, r)(CM(a,in_a) \mid [\![[\vec{b}/\vec{b'}]Q]\!]) & \text{par le lemme } 5.1.3 \\
& \equiv & (\!|\{a\},[\vec{b}/\vec{b'}]Q|\!) &
\end{array}$$

□

Remarque 5.1.8 Notons ici l'importance de la clause $a \notin I$ dans la définition 4.2.1 de la bisimulation. S'il était possible d'observer des messages sur des canaux appartenant à l'interface d'un processus, alors le point 2 du lemme précédent serait faux. En effet, on pourrait en particulier observer des requêtes de réception aux gestionnaires de canaux qui ne correspondent à rien du point de vue du processus codé.

On peut maintenant montrer le résultat principal de cette section : la traduction de π_1 dans π_1^r étendu avec les constantes $CM(a,in_a)$ est fully-abstract vis-à-vis de la bisimulation asynchrone faible.

Théorème 5.1.9 *Supposons que $I \Vdash P_1$ et $I \Vdash P_2$ dans π_1. Alors*

$$P_1 \approx P_2 \quad \Leftrightarrow \quad (\!|I,P_1|\!) \approx (\!|I,P_2|\!)$$

Preuve: ($\Rightarrow$) On définit la relation

$$S = \{((\!|I,P_1|\!),(\!|I,P_2|\!)) \mid P_1 \approx P_2\}$$

On montre que S est une bisimulation up to $H_3 \circ H_2 \circ H_1$ où H_1, H_2 et H_3 sont définis à la fin du chapitre précédent.

(τ) Supposons que $(\!|I,P_1|\!) \xrightarrow{\tau} Q_1$. Alors :

$$\begin{array}{lll}
P_1 \xrightarrow{\tau} P_1' \quad \& \quad Q_1 >_d (\!|I,P_1'|\!) & \text{par le lemme 5.1.7 (1),} \\
P_2 \stackrel{\tau}{\Rightarrow} P_2' \quad \& \quad P_1' \approx P_2' & \text{puisque } P_1 \approx P_2, \\
(\!|I,P_2|\!) \stackrel{\tau}{\Rightarrow} (\!|I,P_2'|\!) & \text{par le lemme 5.1.4 (1)}
\end{array}$$

Donc $Q_1 >_d (\!|I,P_1'|\!)\ S\ (\!|I,P_2'|\!)$, car $P_1' \approx P_2'$.

(*out*) Supposons que $(\!|I,P_1|\!) \xrightarrow{(\nu\vec{w},r)\overline{a}\_,\vec{b},r,\_,\_} ((\!|I',P_1'|\!) \mid T_r)$ suivant le lemme 5.1.7 (2). Alors :

$$\begin{array}{lll}
P_1 \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} P_1' & & \text{et} \\
P_2 \stackrel{(\nu\vec{w})\overline{a}\vec{b}}{\Rightarrow} P_2' \quad \& \quad P_1' \approx P_2' & & \text{puisque } P_1 \approx P_2, \\
(\!|I,P_2|\!) \stackrel{(\nu\vec{w},r)\overline{a}_,\vec{b},r,_,_}{\Rightarrow} ((\!|I',P_2'|\!) \mid T_r) & & \text{par le lemme 5.1.4 (2).}
\end{array}$$

Ainsi, $(\!|I', P_1'|\!) \mid T_r)\ H_3(S)\ ((\!|I', P_2'|\!) \mid T_r)$.

(*in*) Supposons que $(\!|I, P_1|\!) \xrightarrow{a(j,\vec{b},r,c,c')} Q_1$. Alors :

$P_1 \xrightarrow{a\vec{b}} P_1' \quad \& \quad Q_1 >_d (\!|I, P_1'|\!)$	par le lemme 5.1.7 (3). Alors soit
$P_2 \stackrel{a\vec{b}}{\Rightarrow} P_2' \quad \& \quad P_1' \approx P_2'$	puisque $P_1 \approx P_2$, et ainsi
$(\!|I, P_2|\!) \stackrel{a(j,\vec{b},r,c,c')}{\Rightarrow} (\!|I, P_2'|\!)$	par le lemme 5.1.4 ; soit
$P_2 \stackrel{\tau}{\Rightarrow} P_2' \quad \& \quad P_1' \approx (P_2' \mid \overline{a}(\vec{b}))$	puisque $P_1 \approx P_2$,
$(\!|I, P_2|\!) \stackrel{\tau}{\Rightarrow} (\!|I, P_2'|\!)$	par le lemme 5.1.4 (1).

Par le lemme 5.1.6, $(\!|I, P_2' \mid \overline{a}(\vec{b})|\!) \sim ((\!|I, P_2'|\!) \mid \overline{a}(j, \vec{b}, r, c, c'))$. D'où

$$Q_1 >_d (\!|I, P_1'|\!)\ S\ (\!|I, P_2' \mid \overline{a}(\vec{b})|\!)\ \sim\ ((\!|I, P_2'|\!) \mid \overline{a}(j, \vec{b}, r, c, c'))$$

($\Leftarrow$) On définit la relation

$$S = \{(P_1, P_2) \mid I \Vdash P_1, I \Vdash P_2\ \&\ (\!|I, P_1|\!) \approx (\!|I, P_2|\!)\}$$

On montre que S est une bisimulation. Ainsi

$$(\!|I, P_1|\!) \approx (\!|I, P_2|\!)\ \Rightarrow\ P_1\ S\ P_2\ \Rightarrow\ P_1 \approx P_2$$

(τ) Supposons que $P_1 \xrightarrow{\tau} P_1'$. Alors,

$(\!|I, P_1|\!) \xrightarrow{\tau} >_d (\!|I, P_1'|\!)$	par le lemme 5.1.4 (1),
$(\!|I, P_2|\!) \stackrel{\tau}{\Rightarrow} Q_2 \quad \& \quad Q_2 \approx (\!|I, P_1'|\!)$	puisque $(\!|I, P_1|\!) \approx (\!|I, P_2|\!)$
$P_2 \xrightarrow{\tau} P_2'\ \&\ Q_2(>_d)^*(\!|I, P_2'|\!)$	par les lemmes 5.1.7 (1) et 4.2.14.

Ainsi $(\!|I, P_1'|\!) \approx Q_2 \geq_d (\!|I, P_2'|\!)$, ce qui implique $P_1'\ S\ P_2'$.

(*out*) Supposons que $P_1 \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} P_1'$ avec $a \notin I$. Alors :

$(\!|I, P_1|\!) \xrightarrow{(\nu\vec{w},r)\overline{a}\_,\vec{b},r,\_,\_} ((\!|I', P_1'|\!) \mid T_r)$	par le lemme 5.1.4 (2)
$(\!|I, P_2|\!) \stackrel{(\nu\vec{w},r)\overline{a}\_,\vec{b},r,\_,\_}{\Rightarrow} Q_2 \quad \& \quad ((\!|I', P_1'|\!) \mid T_r) \approx Q_2)$	
$P_2 \xrightarrow{(\nu\vec{w})\overline{a}\vec{b}} P_2' \quad \& \quad Q_2(>_d)^*((\!|I', P_2'|\!) \mid T_r)$	par les lemmes 5.1.7, 4.2.14, et par fermeture de diagramme.

Ainsi, $((\!|I', P_1'|\!) \mid T_r) \approx Q_2(>_d)^*((\!|I', P_2'|\!) \mid T_r)$. Par le lemme 5.1.5, $(\!|I', P_1'|\!) \approx (\!|I', P_2'|\!)$ et donc $P_1' \approx P_2'$.

(*in*) Supposons que $P_1 \xrightarrow{a\vec{b}} P_1'$. Alors $(\!|I, P_1|\!) \xrightarrow{a(j,\vec{b},r,c,c')} Q_1 >_d (\!|I, P_1'|\!)$ par le lemme 5.1.4 (1). Il y a deux cas possibles : si

$(\!|I, P_2|\!) \stackrel{a(j,\vec{b},r,c,c')}{\Rightarrow} Q_2 \quad \& \quad Q_1 \approx Q_2$	par $(\!|I, P_1|\!) \approx (\!|I, P_2|\!)$, alors
$P_2 \stackrel{a\vec{b}}{\Rightarrow} P_2' \quad \& \quad Q_2(>_d)^*(\!|I, P_2'|\!)$	par les lemmes 5.1.7 et 4.2.14.

Ainsi, de $(\!|I, P_1'|\!) \approx Q_1 \approx Q_2 \approx (\!|I, P_2'|\!)$, il s'en suit $P_1'\ S\ P_2'$. Dans l'autre cas, si

$$
\begin{array}{rcl}
CM_1(a, \mathit{in}_a) & = & a(m_1).CM_2(a, \mathit{in}_a, m_1) \\
CM_2(a, \mathit{in}_a, m_1) & = & a(m_2).\mathsf{if}\ j_1 = \mathit{in}_a \vee j_2 \neq \mathit{in}_a\ \mathsf{then}\ (CM_1(a, \mathit{in}_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2)) \\
 & & \quad \mathsf{else}\ (\nu c : \mathit{val})(\overline{a}(\_, \_, \_, c, c) \mid \overline{a}(\_, \_, \_, c, \_) \mid \\
 & & \quad\quad CM_3(a, m_1, m_2, c)) \\
CM_3(a, \mathit{in}_a, m_1, m_2, c) & = & a(m_3).\mathsf{if}\ c_3 \neq c\ \mathsf{then}\ (CM_3(a, \mathit{in}_a, m_1, m_2, c) \mid \overline{a}(m_3)) \\
 & & \quad \mathsf{else\ if}\ c_3' = c\ \mathsf{then}\ CM_4(a, \mathit{in}_a, m_1, m_2, c) \\
 & & \quad\quad \mathsf{else}\ CM_5(a, \mathit{in}_a, m_1, m_2, c) \\
CM_4(a, \mathit{in}_a, m_1, m_2, c) & = & a(m_4).\mathsf{if}\ c_4 \neq c\ \mathsf{then}\ (CM_4(a, \mathit{in}_a, m_1, m_2, c) \mid \overline{a}(m_4)) \\
 & & \quad \mathsf{else}\ (CM_1(a, \mathit{in}_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2)) \\
CM_5(a, \mathit{in}_a, m_1, m_2, c) & = & a(m_5).\mathsf{if}\ c_5 \neq c\ \mathsf{then}\ (CM_5(a, \mathit{in}_a, m_1, m_2, c) \mid \overline{a}(m_5)) \\
 & & \quad \mathsf{else}\ (CM_1(a, \mathit{in}_a) \mid \overline{r_2}(\vec{b_1}))
\end{array}
$$

FIG. 5.1: Implantation du gestionnaire de canal

$$
\begin{array}{lll}
(\!|I, P_2|\!) \stackrel{\tau}{\Rightarrow} Q_2 \quad \& \quad Q_1 \approx (Q_2 \mid \overline{a}(j, \vec{b}, r, c, c')) & \text{par } (\!|I, P_1|\!) \approx (\!|I, P_2|\!), \text{ alors} \\
P_2 \stackrel{\tau}{\Rightarrow} P_2' \quad \& \quad Q_2 (>_d)^* (\!|I, P_2'|\!) & \text{par lemmes 5.1.7 et 4.2.14.}
\end{array}
$$

Par le lemme 5.1.6, $(\!|I, P_2' \mid \overline{a}(\vec{b})|\!) \approx ((\!|I, P_2'|\!) \mid \overline{a}(j, \vec{b}, r, c, c'))$. Ainsi, de

$$(\!|I, P_1'|\!) \approx Q_1 \approx (Q_2 \mid \overline{a}(j, \vec{b}, r, c, c')) \approx ((\!|I, P_2'|\!) \mid \overline{a}(j, \vec{b}, r, c, c')) \approx (\!|I, P_2' \mid \overline{a}(\vec{b})|\!)$$

il s'en suit $P_1' \; S \; (P_2' \mid \overline{a}(\vec{b}))$. □

5.2 Implantation des gestionnaires de canaux

Pour terminer la preuve de notre résultat, il reste à montrer que notre gestionnaire de canal $CM(a, \mathit{in}_a)$ peut être implanté de manière adéquate dans π_1^r. La figure 5.1 montre cette implantation. Afin de la décrire de manière concise, il est commode d'utiliser un système d'équations récursives où m_k, pour $k = 1, \ldots, 5$, désigne le vecteur $j_k, \vec{b_k}, r_k, c_k, c_k'$ qui est supposé ne pas contenir in_a. On peut immédiatement convertir ce système d'équations récursives dans notre notation :

$$(\mathsf{rec}\, A_1(a, \mathit{in}_a).a(m_1).\underbrace{a(m_2).\mathsf{if}\ \ldots}_{CM_2(a, m_1)})$$

et de manière similaire pour $CM_i(a, \mathit{in}_a, m_1, m_2, c)$ (qui sont des sous-termes de CM_1, pour $3 \leq i \leq 5$). On peut facilement vérifier que $a \Vdash CM_1(a, \mathit{in}_a)$.

L'implantation $CM_1(a, \mathit{in}_a)$ du gestionnaire du canal a doit nécessairement passer par un état intermédiaire pour transmettre un message sur a : c'est l'état dans lequel il a reçu une requête d'émission et attend une requête de réception. Or, $CM(a, \mathit{in}_a)$ traite simultanément les deux requêtes sans passer par un état intermédiaire (*cf.* règles (CM_τ) et (CM_{in})). D'autre part, quelque soit

l'état de $CM_1(a, in_a)$, ce dernier doit pouvoir simuler son abstraction $CM(a, in_a)$. En particulier, si $CM(a, in_a)$ est dans l'état intermédiaire et si $CM(a, in_a)$ admet une requête de l'environnement (par (CM_{in})), alors $CM(a, in_a)$ doit pouvoir retourner dans l'état initial pour admettre la même requête. C'est pour cette raison que notre implantation utilise un choix interne qui permet, après admission d'une requête d'émission et de réception, de finalement ne pas les traiter en rejetant ces requêtes et en retournant dans son état initial.

Nous commentons plus précisément cette implantation. Le gestionnaire de canal réalise d'abord deux réceptions sur a. La première doit être une requête d'émission et la seconde une requête de réception (sinon il revient à son état initial). Le gestionnaire de canal procède alors à un choix interne. Pour se faire, il génère deux messages $\overline{a}(\_,\_,\_,c,c)$ et $\overline{a}(\_,\_,\_,c,c')$ dont l'un peut être reçu par CM_3, puis l'autre par CM_4 ou CM_5. Si le message reçu par CM_3 est $\overline{a}(\_,\_,\_,c,c)$ le gestionnaire de canal accède à l'état CM_4 et revient à l'état initial. Sinon, il accède à l'état CM_5 et établit la connexion.

On s'aperçoit, dans cette implantation, que si nous n'avons que des requêtes d'émissions, celles-ci seront consommées par le gestionnaire de canal et renvoyées automatiquement dans l'environnement. Cette *attente active*[1] signifie qu'une situation de deadlock est traduite en une situation où on boucle et, par conséquent, où on n'évite pas la perte de messages. Il n'y a donc pas de miracle et ceci n'est pas surprenant : on ne peut pas espérer de notre codage qu'il soit « intelligent » au point de traduire un programme qui perd ses messages dans un programme qui ne les perds pas. Notre codage montre que tous les comportements de π_1 peuvent être programmés dans π_1^r. Nous ne considérons pas que l'attente active soit une bon style de programmation, et en pratique il y a de meilleurs façons de gérer les messages indésirables. Cependant, celles-ci requièrent une bonne compréhension du problème et est encouragée par la discipline de programmation réceptive.

On dénote par $[CM_1/CM]$ l'opération de substitution du gestionnaire de canal abstrait CM par son implantation CM_1 décrite ci-dessus. Ainsi, $[CM_1/CM][\![P]\!]$ et $[CM_1/CM](\!|I, P|\!)$ sont maintenant des termes de π_1^r. Dans la proposition suivante on considère encore des termes du calcul enrichi avec les constantes $CM(a, in_a)$.

Proposition 5.2.1

1. *Soit* $M \cong \Pi_{i\in I}\overline{a_i}(j_i, \vec{b_i}, r_i, c_i, c'_i)$. *Alors* $(CM(a, in_a) \mid M) \approx (CM_1(a, in_a) \mid M)$.
2. *Si* $I \Vdash P$ *alors* $(\!|I, P|\!) \approx [CM_1/CM](\!|I, P|\!)$.

Preuve: (1) Dans cette preuve on utilise, en plus de $m_k = j_k, \vec{b_k}, r_k, c_k, c'_k$ comme dans la définition de CM_i (excepté qu'on demande plus que ce vecteur ne contienne pas in_a), les notations suivantes :

$$\begin{array}{rclcrcl} M & \cong & \Pi_{i\in I}\overline{a}(j_i, \vec{b_i}, r_i, c_i, c'_i) & & R & \cong & \Pi_{j\in J}\overline{r_j}(\vec{b_j}) \\ N & \cong & (M \mid R) & & m_{c_0,c_1} & \cong & \overline{a}(\_,\_,\_,c_0,c_1) \end{array}$$

[1] *busy waiting* en anglais.

Soit $\mathcal{R}$ la relation composée par les paires suivantes (on omet le paramètre in_a pour CM_2 à CM_5) :

$$((CM(a, in_a) \mid N), (CM_1(a, in_a) \mid N)) \tag{5.1}$$

$$((CM(a, in_a) \mid \overline{a}(m_1) \mid N), (CM_2(a, m_1) \mid N)) \tag{5.2}$$

$$((CM(a, in_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2) \mid N), ((\nu c, c')(CM_3(a, m_1, m_2, c) \mid m_{c,c} \mid m_{c,c'}) \mid N)) \ (*) \tag{5.3}$$

$$((CM(a, in_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2) \mid N), ((\nu c, c')(CM_4(a, m_1, m_2, c) \mid m_{c,c'}) \mid N)) \ (*) \tag{5.4}$$

$$((CM(a, in_a) \mid \overline{r_2}(\vec{b_1}) \mid N), ((\nu c, c')(CM_5(a, m_1, m_2, c) \mid m_{c,c'}) \mid N)) \tag{5.5}$$

$(*)$ où $j_1 \neq in_a$ et $j_2 = in_a$. On vérifie que $\mathcal{R}$ est une bisimulation up-to H_1. Soit $(P, Q) \in \mathcal{R}$ et $P \xrightarrow{\alpha} P'$. Pour chacun des cas (5.1-5.5), on prétend qu'on peut trouver une transition correspondante $Q \stackrel{\alpha}{\Rightarrow} Q'$ et retomber dans un des cas (5.1-5.5), c'est-à-dire $(P', Q') \in \mathcal{R}$. Dans la suite on décrit schématiquement les transitions correspondantes pour Q, en omettant en particulier les paramètres des CM_i. Par exemple,

$$CM_4 \xrightarrow{\tau} CM_1 \xrightarrow{a} CM_2 \xrightarrow{\tau} CM_3$$

signifie que Q réalise une transition de réception sur a, avec les arguments appropriés, précédée et suivie de synchronisations internes, et que, suivant ces transitions, le gestionnaire de canal passe de l'état 4 à l'état 3 en traversant les états 1 et 2. Dès qu'une séquence termine dans l'état i, on peut vérifier qu'on tombe dans le i-ième schéma de la définition de $\mathcal{R}$.

- Si α est une transition d'émission, alors celle-ci est produite par un message émanant de R. En effet, le sujet de ce message ne peut être a car ce canal est dans l'interface de P.
- Supposons la transition réalisée par P est CM_τ. On examine les cinq cas possibles.

  (5.1) N doit contenir des messages $\overline{a}(j_1, \vec{b_1}, r_1, c_1, c_1')$ et $\overline{a}(in_a, \vec{b_2}, r_2, c_2, c_2')$ avec $j_1 \neq in_a$. On peut alors poser, sans perte de généralité $N = \overline{a}(m_1) \mid \overline{a}(m_2) \mid N'$. La transition réalisée par P est :

  $$P \equiv (CM(a, in_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2) \mid N') \xrightarrow{\tau} (CM(a, in_a) \mid \overline{r_2}(\vec{b_1}) \mid N')$$

  Les transitions correspondantes pour Q sont :

  $$\begin{aligned}(CM_1(a, in_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2) \mid N') &\xrightarrow{\tau} (CM_2(a, m_1) \mid \overline{a}(m_2) \mid N') \\ &\xrightarrow{\tau} (\nu c)(m_{c,c} \mid (\nu c')m_{c,c'} \mid CM_3(a, m_1, m_2, c)) \mid N' \\ &\xrightarrow{\tau} (\nu c, c')(m_{c,c} \mid CM_5(a, m_1, m_2, c)) \mid N' \\ &\xrightarrow{\tau} (\nu c, c')(CM_1(a, in_a) \mid \overline{r_2}(\vec{b_1})) \mid N'\end{aligned}$$

  Et on a bien $((CM(a, in_a) \mid \overline{r_2}(\vec{b_1}) \mid N'), (\nu c, c')(CM_1(a, in_a) \mid \overline{r_2}(\vec{b_1})) \mid N') \in H_1(\mathcal{R})$.

  (5.2) N contient au moins un messages $\overline{a}(j_2, \vec{b_2}, r_2, c_2, c_2')$, donc $N \equiv \overline{a}(j_2, \vec{b_2}, r_2, c_2, c_2') \mid N'$. P réalise alors la même transition que précédemment, et Q peut réaliser les transitions suivantes :

  $$\begin{aligned}CM_2(a, m_1) \mid \overline{a}(m_2) \mid N' &\xrightarrow{\tau} (\nu c)(m_{c,c} \mid (\nu c')m_{c,c'} \mid CM_3(a, m_1, m_2, c)) \mid N' \\ &\xrightarrow{\tau} (\nu c)((\nu c')m_{c,c'} \mid CM_4(a, m_1, m_2, c)) \mid N' \\ &\xrightarrow{\tau} (\nu c, c')(CM_1(a, in_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2)) \mid N' \\ &\vdots \quad (\xrightarrow{\tau} CM_2 \xrightarrow{\tau} CM_3 \xrightarrow{\tau} CM_5) \\ &\xrightarrow{\tau} (\nu c, c', c'', c''')(CM_1(a, in_a) \mid \overline{r_2}(\vec{b_1})) \mid N'\end{aligned}$$

où les points de suspension sont à remplacer par les mêmes transitions que dans le cas précédent. Encore une fois on a $((CM(a, in_a) \mid \overline{r_2}(\vec{b_1}) \mid N'), (\nu c, c', c'', c''')(CM_1(a, in_a) \mid \overline{r_2}(\vec{b_1})) \mid N') \in H_1(\mathcal{R})$.

(5.3) La suite des transitions pour Q est :

$$\begin{array}{ll} & (\nu c, c')(CM_3(a, m_1, m_2, c) \mid m_{c,c} \mid m_{c,c'}) \mid N \\ \xrightarrow{\tau} & (\nu c, c')(CM_4(a, m_1, m_2, c) \mid m_{c,c'}) \mid N \\ \xrightarrow{\tau} & (\nu c, c')(CM_1(a, in_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2)) \mid N \\ \vdots & (\xrightarrow{\tau} CM_2 \xrightarrow{\tau} CM_3 \xrightarrow{\tau} CM_5) \\ \xrightarrow{\tau} & (\nu c, c', c'', c''')(CM_1(a, in_a) \mid \overline{r_2}(\vec{b_1})) \mid N \end{array}$$

Et on conclue comme dans le cas précédent.

(5.4) Il s'agit essentiellement du même cas que le précédent.

(5.5) On peut, comme dans le premier cas, faire l'hypothèse que $N = \overline{a}(j_3, \vec{b_3}, r_3, c_3, c_3') \mid \overline{a}(in_a, \vec{b_4}, r_4, c_4, c_4') \mid N'$ où $j_3 \neq in_a$. P fait alors la transition τ suivante :

$$CM(a, in_a) \mid \overline{r_2}(\vec{b_1}) \mid \overline{a}(m_3) \mid \overline{a}(m_4) \mid N' \xrightarrow{\tau} CM(a, in_a) \mid \overline{r_2}(\vec{b_1}) \mid \overline{r_4}(\vec{b_3}) \mid N' = P'$$

Q peut réaliser les transitions suivantes :

$$\begin{array}{ll} & (\nu c, c')(CM_5(a, m_1, m_2, c) \mid m_{c,c'}) \mid \overline{a}(m_3) \mid \overline{a}(m_4) \mid N' \\ \xrightarrow{\tau} & (\nu c, c')(CM_1(a, in_a) \mid \overline{a}(m_3) \mid \overline{a}(m_4)) \mid \overline{r_2}(\vec{b_1}) \mid N' \\ \vdots & (\xrightarrow{\tau} CM_2 \xrightarrow{\tau} CM_3 \xrightarrow{\tau} CM_5) \\ \xrightarrow{\tau} & (\nu c, c', c'', c''')(CM_1(a, in_a) \mid \overline{r_4}(\vec{b_3})) \mid \overline{r_2}(\vec{b_1}) \mid N' = Q' \end{array}$$

Et on a $(P', Q') \in H_1(\mathcal{R})$

– Supposons à présent que P réalise une réception. Les gestionnaires de canaux étant les seuls récepteurs, la seule règle applicable est CM_{in}.

(5.1) N doit contenir une requête de réception : $N = \overline{a}(in_a, \vec{b_2}, r_2, c_2, c_2') \mid N'$. P fait la transition suivante (où $j_1 \neq in_a$) :

$$CM(a, in_a) \mid \overline{a}(m_2) \mid N' \xrightarrow{a(m_1)} CM(a, in_a) \mid \overline{r_2}(\vec{b_1}) \mid N' = P'$$

Et Q peut réaliser la séquence de transitions suivantes :

$$\begin{array}{lll} CM_1(a, in_a) \mid \overline{a}(m_2) \mid N' & \xrightarrow{a(m_1)} & CM_2(a, m_1) \mid \overline{a}(m_2) \mid N' \\ & \xrightarrow{\tau} & (\nu c)(m_{c,c} \mid (\nu c')m_{c,c'} \mid CM_3(a, m_1, m_2, c)) \mid N' \\ & \xrightarrow{\tau} & (\nu c, c')(m_{c,c} \mid CM_5(a, m_1, m_2, c)) \mid N' \\ & \xrightarrow{\tau} & (\nu c, c')(CM_1(a, in_a) \mid \overline{r_2}(\vec{b_1})) \mid N' = Q' \end{array}$$

Et on a bien $(P', Q') \in H_1(\mathcal{R})$.

(5.2) Il y a deux cas à prendre en considération : celui où on consomme un message de N (et alors $N = \overline{a}(\mathit{in}_a, \vec{b_2}, r_2, c_2, c_2') \mid N'$) et celui où on consomme le message $\overline{a}(m_1)$ (et donc $j_1 = \mathit{in}_a$). Dans le premier cas P réalise la transition (où $j_3 \neq \mathit{in}_a$)

$$CM(a, \mathit{in}_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2) \mid N' \xrightarrow{a(m_3)} CM(a, \mathit{in}_a) \mid \overline{a}(m_1) \mid \overline{r_2}(\vec{b_3}) \mid N' = P'$$

Q peut réaliser la séquence de transitions correspondante suivante :

$$\begin{array}{rl}
 & CM_2(a, m_1) \mid \overline{a}(m_2) \mid N' \\
\xrightarrow{\tau} & (\nu c)(m_{c,c} \mid (\nu c')m_{c,c'} \mid CM_3(a, m_1, m_2, c)) \mid N' \\
\xrightarrow{\tau} & (\nu c, c')(m_{c,c'} \mid CM_4(a, m_1, m_2, c)) \mid N' \\
\xrightarrow{\tau} & (\nu c, c')(CM_1(a, \mathit{in}_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2)) \mid N' \\
\xrightarrow{a(m_3)} & (\nu c, c')(CM_2(a, m_3) \mid \overline{a}(m_1) \mid \overline{a}(m_2)) \mid N' \\
\xrightarrow{\tau} & (\nu c, c', c'')(m_{c'',c''} \mid (\nu c''')m_{c'',c'''} \mid CM_3(a, m_3, m_2, c'') \mid \overline{a}(m_1)) \mid N' \\
\xrightarrow{\tau} & (\nu c, c', c'', c''')(m_{c'',c''} \mid CM_5(a, m_3, m_2, c'') \mid \overline{a}(m_1)) \mid N' \\
\xrightarrow{\tau} & (\nu c, c', c'', c''')(CM_1(a, \mathit{in}_a) \mid \overline{a}(m_1) \mid \overline{r_2}(\vec{b_3})) \mid N' = Q'
\end{array}$$

Dans le deuxième cas, P réalise la transition (où $j_3 \neq \mathit{in}_a$)

$$CM(a, \mathit{in}_a) \mid \overline{a}(m_1) \mid N \xrightarrow{a(m_3)} CM(a, \mathit{in}_a) \mid \overline{r_1}(\vec{b_3}) \mid N = P'$$

Les transitions correspondantes pour Q sont :

$$\begin{array}{rl}
CM_2(a, m_1) \mid N \xrightarrow{a(m_3)} & CM_1(a, m_1) \mid \overline{a}(m_1) \mid \overline{a}(m_3) \mid N \\
\vdots & (\xrightarrow{\tau} CM_2 \xrightarrow{\tau} CM_3 \xrightarrow{\tau} CM_5) \\
\xrightarrow{\tau} & (\nu c, c')(CM_1(a, \mathit{in}_a) \mid \overline{r_1}(\vec{b_3})) \mid N = Q'
\end{array}$$

Et dans les deux cas on a encore $(P', Q') \in H_1(\mathcal{R})$.

(5.3) On a (où $j_3 \neq \mathit{in}_a$)

$$CM(a, \mathit{in}_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2) \xrightarrow{a(m_3)} CM(a, \mathit{in}_a) \mid \overline{a}(m_1) \mid \overline{r_2}(\vec{b_3})) \mid N = P'$$

Et Q peut réaliser :

$$\begin{array}{rl}
 & (\nu c, c')(CM_3(a, m_1, m_2, c) \mid m_{c,c} \mid m_{c,c'}) \mid N \\
\xrightarrow{\tau} & (\nu c, c')(CM_4(a, m_1, m_2, c) \mid m_{c,c'}) \mid N \\
\xrightarrow{\tau} & (\nu c, c')(CM_1(a, \mathit{in}_a) \mid \overline{a}(m_1) \mid \overline{a}(m_2)) \mid N \\
\xrightarrow{a(m_3)} & (\nu c, c')(CM_2(a, m_3) \mid \overline{a}(m_1) \mid \overline{a}(m_2)) \mid N \\
\vdots & (\xrightarrow{\tau} CM_3 \xrightarrow{\tau} CM_5) \\
\xrightarrow{\tau} & (\nu c, c', c'', c''')(CM_1(a, \mathit{in}_a) \mid \overline{r_3}(\vec{b_3}) \mid \overline{a}(m_1)) \mid N = Q'
\end{array}$$

Et on a bien $(P', Q') \in H_1(\mathcal{R})$.

(5.4) Il s'agit essentiellement du même cas que le précédent.

(5.5) Comme dans le premier cas, on doit faire l'hypothèse que N contient une requête de réception : $N = \overline{a}(in_a, \vec{b_3}, r_3, c_3, c_3') \mid N'$. P réalise alors la transition :

$$CM(a, in_a) \mid \overline{r_2}(\vec{b_1}) \mid \overline{a}(m_3) \mid N' \xrightarrow{a(m_4)} CM(a, in_a) \mid \overline{r_2}(\vec{b_1}) \mid \overline{r_3}(\vec{b_4}) \mid N' = P'$$

Q peut réaliser les transitions correspondantes suivantes :

$$\begin{array}{rl} & (\nu c, c')(CM_5(a, m_1, m_2, c) \mid m_{c,c}) \mid \overline{a}(m_3) \mid N' \\ \xrightarrow{\tau} & (\nu c, c')(CM_1(a, in_a) \mid \overline{r_2}(\vec{b_2}) \mid \overline{a}(m_3)) \mid N' \\ \xrightarrow{a(m_4)} & (\nu c, c')(CM_2(a, m_3) \mid \overline{r_2}(\vec{b_2}) \mid \overline{a}(m_3)) \mid N' \\ \vdots & (\xrightarrow{\tau} CM_3 \xrightarrow{\tau} CM_5) \\ \xrightarrow{\tau} & (\nu c, c', c'', c''')(CM_1(a, in_a) \mid \overline{r_3}(\vec{b_4}) \mid \overline{r_2}(\vec{b_2})) \mid N' = Q' \end{array}$$

Finalement, $(P', Q') \in H_1(\mathcal{R})$.

– Pour l'autre direction, supposons que $(P, Q) \in \mathcal{R}$ et $Q \xrightarrow{\alpha} Q'$.

(5.1) Si Q réalise une action interne, c'est qu'il existe un message tel que $N = (\overline{a}(m_1) \mid N')$ et donc $Q \xrightarrow{\tau} (CM_2(a, m_1) \mid N')$. Dans ce cas, P ne réalise aucune action (c'est-à-dire une transition $P \stackrel{\tau}{\Rightarrow} P$ de longueur nulle) et on se retrouve dans le cas 5.2. De même, si Q réalise une action de réception $Q \xrightarrow{a(m_1)} (CM_2(a, m_1) \mid N)$, P ne fait rien non plus et on se retrouve également dans le cas (5.2). De nouveau, aucune action d'émission n'est observable.

(5.2) Q peut réalise soit une action interne (auquel cas $N = (\overline{a}(m_2) \mid N')$), soit une réception $a(m_2)$. Dans les deux cas, P ne réalise aucune action et si $j_1 = in_a$ et $j_2 \neq in_a$ alors on se retrouve dans le cas (5.1), sinon dans le cas (5.3).

(5.3) Dans Q, si CM_3 consomme un message sur a (par une action τ ou $a(m_3)$) qui n'est ni $m_{c,c}$ ni $m_{c,c'}$, alors Q boucle sur lui même ce qui correspond à ne réaliser aucune transition pour P et on reste dans le cas (5.3). Par contre, si CM_3 consomme le message $m_{c,c}$ alors, en contre partie, P ne réalise pas d'action et on arrive dans le cas (5.4). Enfin, si CM_3 reçoit le message $m_{c,c'}$, en contre partie, P réalise la transition CM_τ avec les messages $\overline{a}(m_1)$ et $\overline{a}(m_2)$, et on aboutit au cas (5.5).

(5.4) Comme dans le cas précédent, CM_4 peut boucler sur lui-même en ne consommant pas $m_{c,c'}$, et dans ce cas, P ne fait aucune action et retombe en (5.4). Si c'est effectivement $m_{c,c'}$ qui est reçu (par une action interne), alors on retombe dans le cas (5.1) en ne réalisant pas de transition pour P.

(5.5) A nouveau, Q peut boucler et on reste dans le même cas en ne faisant rien pour P. Si le message $m_{c,c}$ est consommé par CM_5 (par une action interne) alors, en contre partie P ne fait rien et on tombe dans le cas (5.1).

(2) Dans la traduction, un gestionnaire de canal peut apparaître dans deux positions :

1. Sous un préfixe de réception, dans un processus de la forme :

$$(\nu a)(\nu in_a)(CM(a, in_a) \mid [\![P]\!])$$

Dans ce cas, $CM(a, in_a)$ (et $CM_1(a, in_a)$) ne peuvent pas immédiatement jouer de rôle dans les transitions et ne sont pas affectées par celle-ci à cause des restrictions appliquées sur leurs paramètres.

2. Dans un processus bien formé de la forme, à équivalence structurelle près :

$$(\nu in_a)(CM(a, in_a) \mid P) \quad \text{ou} \quad (\nu a)(\nu in_a)(CM(a, in_a) \mid P)$$

Le premier cas est celui où a est dans l'interface I du terme traduit. De plus, P ne peut exploiter la clef in_a que dans des messages $\overline{a}(in_a, \vec{b}, r, c, c')$. On peut montrer, dans une analyse par cas similaire à celle de la première partie de cette preuve, que le comportement de $CM(a, in_a)$ placé dans un contexte d'évaluation peut être bisimulé par un état adéquat CM_i ($i = 1, \ldots, n$) placé dans le même contexte d'évaluation.

Deuxième partie

Le π-calcul réceptif réparti

CHAPITRE 6

Les modèles de la distribution

LES MODÈLES DE LA DISTRIBUTION PROPOSÉS JUSQU'À PRÉSENT, et les langages implantant explicitement des primitives liées à la notion de distribution, ont été conçus en retenant différents critères. D'un modèle à l'autre, ces critères peuvent être opposés voir complètement différents, et dans un même modèle des critères peuvent être complémentaires. Ce présent chapitre a pour objectif d'analyser, le plus exhaustivement possible, les critères permettant plus ou moins une classification des modèles de la distribution. Pour cela, nous nous appuyons en partie sur le document réalisé dans le cadre du projet MARVEL [BGL99].

La première section est dévolue à l'analyse des critères de la distribution suivant lesquels nous décrirons, dans les sections suivantes, quelques modèles de la distribution. En particulier, nous nous intéresserons au calcul des Ambients, au Join-calcul distribué, à Nomadic-Pict, au $\pi_{1\ell}$-calcul, à KLAIM et enfin à Dπ qui constitue le modèle de référence pour celui que nous développerons dans la suite de cette partie.

6.1 Les critères de la distribution

Il convient tout d'abord d'exhiber le concept permettant de qualifier un modèle de *modèle pour la distribution*. Celui que nous choisissons est le concept de **domaine** qu'on peut définir comme étant une région regroupant (d'un point de vue spatial) un ensemble de processus, et ce de manière implicite ou explicite. Dans la suite, nous employons aussi les termes « site » et « localité » avec le même sens que « domaine ». À la notion de domaine, est souvent associée celle de **migration** permettant la mobilité d'entités entre les différents domaines d'un système. Cette migration est à distinguer de la mobilité usuellement entendue comme la transmission de programmes et telle que définie dans le deuxième chapitre. Nous avons vu que le π-calcul est un exemple de modèle permettant la mobilité dite *mobilité de noms* (où la transmission d'un programme est la connaissance d'un canal de communication permettant d'y avoir accès). Sans la migration, les modèles de la distribution semblent présenter peu

d'intérêt. D'autre part, les langages modernes permettant la programmation d'applications réparties implantent des primitives de migration et ainsi accroissent la variété des applications réalisables.

Notre analyse de la distribution se limite donc à envisager les modèles suivant deux aspects : la **répartition** et la **mobilité**. D'autres dimensions de la distribution pourraient être prises en compte dont par exemple la sécurité et l'implantation de langages. Concernant ces deux derniers aspects, nous référons le lecteur par exemple aux documents [BGL99] et [LBCF99].

6.1.1 Aspect répartition

Cet aspect concerne les critères de comparaison associés à la notion de domaine. Ceux-ci peuvent varier suivant les orientations du modèle : défaillance, communication, migration ou encore sécurité. Les critères décrits ici sont : la **topologie de l'espace des domaines**, les **sémantiques de domaine**, les **capacités d'observation/contrôle des domaines** et enfin les caractéristiques de **communications intra- et inter-domaines**. La défaillance est également abordée comme critère de comparaison.

Topologie de l'espace des domaines

Il s'agit ici de décrire la façon dont les domaines sont structurés. Dans la plupart des cas, la structure est **hiérarchique** : un domaine peut contenir plusieurs sous-domaines. Un système est alors un arbre de domaines ou encore une forêt. L'autre type de structure est en fait un cas particulier du précédent : il s'agit des espaces de domaines **plats**. À notre connaissance il n'existe pas de modèle optant pour une structure de type *ensembliste* permettant l'intersection de domaines sans nécessairement leur inclusion. L'absence de tels modèles peut en partie se justifier par le fait qu'une structure hiérarchique, peut « simuler » des structures ensemblistes suivant les orientations considérées.

Sémantiques de domaine

Les domaines permettent d'unifier le comportement des entités qu'ils regroupent suivant diverses orientations. Nous en distinguons ici trois qui ne sont pas nécessairement exclusives les unes des autres ; souvent d'ailleurs, plusieurs orientations sont considérées par un même modèle.

Dans l'**orientation communication**, un domaine caractérise un lieu où la communication est possible. On parle alors de communication *intra-domaine* ou *locale*. La comunication entre des entités appartenant à des domaines différents est dans ce cas impossible.

Les calculs envisageant l'**orientation mobilité** comportent une primitive atomique de migration du contenu d'une localité. Dans le cas d'une structure hiérarchique, ce contenu peut être un sous-domaine emporté avec tous ses domaines fils (voir l'aspect mobilité).

Enfin, l'**orientation défaillance** se traduit par l'incapacité à émettre et/ou recevoir des messages ou des entités migrantes.

Observabilité / Contrôlabilité des domaines

Il s'agit de l'aptitude du calcul à permettre l'observation et/ou le contrôle des domaines depuis l'application. Si, cette notion est généralement liée à celle de défaillance, le contrôle peut être motivé

par des considérations de sécurité. Ainsi, le contrôle peut par exemple explicitement permettre l'accès à un domaine, ou encore l'activation d'un domaine.

Domaines et communication locale

On regroupe ici les caractéristiques de la **désignation locale**, de la **politique de comunication locale** et de la **sémantique de la réception**.

La désignation d'une ressource (généralement un récepteur) se fait grâce à un nom – éventuellement connu à l'issue d'une communication. Dans un domaine, un même nom peut représenter un ou plusieurs récepteurs. Les calculs étudiés dans ce chapitre peuvent au minimum communiquer des noms de canaux ou de domaines. Il est parfois possible de communiquer des usages spécifiques appelés « capacités ». La réception peut être unique ou multiple (plusieurs récepteurs identifiés par un même nom). Elle est aussi souvent *statique* dans le sens où la réception d'un nom de canal n'alloue que la capacité d'émettre sur celui-ci.

Domaines et communication distante

Dans le même ordre d'idée, on décrit ici les formes de **désignation distante**, de **politiques de communication distante** et de **routage logique**.

La référence à une ressource distante par son nom est obtenue grâce à la notion de portée distribuée, c'est-à-dire que la portée d'un nom peut s'étendre à plusieurs domaines. La communication distante peut être globale, restreinte à un voisinage ou purement locale comme dans le cas précédent. La localité cible de la communication peut être explicite ou implicite (ce cas accompagne souvent celui de la communication global ou restreinte). Le transport du message est soit direct (ou atomique) soit indirect (ou en plusieurs phases où il doit d'abord sortir du domaine source, pénétrer dans le domaine cible et enfin être consommé). Le routage « logique » est l'acheminement dans l'arborescence des domaines. Il peut être explicite ou transparent.

Domaines et défaillance

Certains modèles fondent leur sémantique sur le concept de défaillance. La défaillance d'une localité signifie son incapacité à émettre/recevoir des messages ou des entités migrantes. C'est une sorte de blocage temporaire ou définitif d'un domaine et de son contenu. Généralement, la défaillance est contrôlable et observable depuis l'extérieur ou l'intérieur d'une localité.

6.1.2 Aspect mobilité

L'analyse de la mobilité s'articule autour de trois notions :

- Les **domaines**. Il s'agit de déterminer ici en quelque sorte le sujet de la migration, c'est-à-dire ce que l'on quitte et ce dans quoi on rentre lors des migrations.
- Les **entités migrantes**. C'est ce qui est objet des migrations. On considère qu'elles peuvent toujours comporter un contenu exécutable.
- Le **contexte**. C'est l'environnement (ou la connaissance) d'une entité au moment de sa migration.

Ces notions ne sont pas exclusives ; elles sont inter-dépendantes suivant les choix optés pour chacune d'elles. Une entité migrante est toujours incluse dans un domaine, elle peut même *être* un domaine. Parfois, le contexte est inclu dans l'entité mobile. Cependant, comme on l'a signalé plus haut, le contexte s'étend généralement au-delà de l'entité voir de son domaine initial et peut donc être partagé avec d'autres entités. Il convient alors de déterminer la politique à adopter pour le contexte vis-à-vis de l'entité mobile.

Une première distinction que nous pouvons faire sur les entités mobiles concerne leur caractère **statique** ou **dynamique**. En effet, si elles contiennent du code exécutable, celui-ci peut être soit activé après migration (dans ce cas on dit que l'entité est statique), soit indépendant de la migration, c'est-à-dire que son exécution est interrompue le temps de la migration. Dans ce dernier cas, l'entité est dynamique et on peut encore distinguer deux cas : celui où l'instruction de migration est **bloquante** vis-à-vis de l'entité migrante ou **asynchrone** (elle peut entrer en concurrence avec des instructions de migration).

La mobilité peut s'opérer par **déplacement** ou par **réplication**. Dans le premier cas, l'entité mobile est un code original, dans le second, c'est une copie (ou « clône ») d'une portion de code dont l'original reste en place. En pratique, cette distinction concerne essentiellement le contexte ; pour les entités migrantes les deux types de mobilité sont généralement modélisés (le plus souvent c'est la réplication qui peut être simulée).

On peut également différencier deux points de vues vis-à-vis de l'autorité (ou encore l'instruction) qui commande la migration. L'autorité peut être extérieure à l'entité mobile ; on parle alors de mouvement **objectif**. Dans le cas contraire, c'est-à-dire si la migration est directement commandée par l'entité objet du mouvement, alors on parle de mouvement **subjectif**. Les entités mobiles statiques font toujours l'objet de mouvements objectifs.

Le comportement du contexte « public » vis-à-vis d'une entité en cours de migration est généralement celui de la liaison dynamique : la valeur d'un nom public est celle qu'il a dans le site d'exécution courant. En ce qui concerne la partie « privée » du contexte, celle-ci pouvant être partagée avec des entités fixes, plusieurs comportements sont alors envisageables :

- **Contexte fixe**. Les éléments du contexte restent dans le domaine initial et soit la relation est rompue (avec éventuellement liaison dynamique sur le site d'arrivé), soit la relation est maintenue à distance implicitement ou explicitement (par l'intermédiaire de proxy par exemple).
- **Contexte mobile**. On retrouve ici les types de mobilité par déplacement et par réplication. Le premier cas nous fait revenir dans un problème symétrique au précédent vis-à-vis des entités fixes. Dans le second, il faut déterminer si la cohérence entre les éléments du contexte original et de sa copie doit être maintenue.

6.2 Le calcul des Ambients

Le calcul des Ambients [CG98] est fondé sur l'idée même de migration : il capture de manière primitive les concepts de domaines, de mobilité et d'autorisation d'accès. Si ces primitives sont suffisantes en matière d'expressivité, la communication existe par commodité.

Dans ce calcul, les entités de base sont les « domaines » (*ambients*), les *processus* et les *capacités* (à traverser ou à dissoudre une frontière). Le calcul manipule des noms d'ambients, deux ambients différents pouvant porter le même nom.

6.2.1 Aspect répartition

La topologie de l'espace des domaines est de type hiérarchique, un ambient pouvant contenir aussi bien des processus que d'autres ambients. La sémantique des domaines est orientée communication et mobilité. La migration d'un ambient entraine la mobilité de tout ce qu'il contient et ne peut s'accomplir que sous la condition que l'authorité à l'origine de la migration connaisse le nom de l'ambient en question. Un ambient peut être *ouvert* par un processus qui lui est extérieur (primitive *open*) ayant pour effmigra95(la)-29 dluud1rieura9Dansa9-1(pfronti.Dans)-Deur mani gn209(hilrchiq9se)-244-1()-25 9se ambients

localité.

6.3.1 Aspect répartition

La topologie est hiérarchique et les noms de localités sont uniques à l'échelle globale permettant de nommer d'autres localités de manière non ambiguë. Les localités sont des unités de migration et de défaillance. La migration d'une localité, grâce à une règle structurelle de « repliage » dans une loc-définition, entraîne la migration de tous ses processus et localités filles.

Une localité peut être marquée « vivante » ou « morte », et peut être stoppée de manière subjective. On peut observer de manière objective son état.

Il n'y pas de différence entre la désignation locale et distante car les noms définis dans les « join-pattern » sont globalement uniques. Par conséquent, la désignation d'un canal se fait par un nom primitif. Aussi, il peut y avoir plusieurs récepteurs sur un nom donné mais au sein d'une même définition et doivent être colocalisés. C'est cette dernière propriété de colocalisation qui rend la désignation distante transparente à la localisation. La communication distante s'effectue en deux étapes : d'abord sortie du domaine courant et entrée dans le domaine cible, puis consommation du message.

Les défaillances considérées sont des défaillances permanentes. La localité défaillante ne peut être à la source d'une communication ou d'une migration ; en revanche elle peut en être la cible.

6.3.2 Aspect mobilité

Seule une localité peut migrer et devenir sous-localité d'une localité cible. Ces migrations ont pour seules conséquences de rendre dépendantes les entités migrantes de la défaillance de la nouvelle localité mère ou d'échapper à la défaillance future de la localité d'origine.

L'instruction de migration a la forme $\mathsf{go}(a, \kappa)$ où a est la localité de destination et κ une continuation. Elle est « contenue » dans la localité à migrer et constitue donc une forme subjective de migration. Elle est asynchrone mais la continuation permet de synchroniser la migration avec l'exécution des processus abrités par la localité mobile.

De même que dans le cas des ambients on peut considérer que le contexte d'une localité migrante reste fixe, l'entretien des références distantes ne posant pas de problème puisque ces liens sont complètement transparents.

6.4 Nomadic Pict

Nomadic Pict [SWP99, Uny01] est une extension de Pict [Tur96, PT98, Pie98] (une implantation d'un variant du π-calcul asynchrone) avec des notions de localité, d'agent et de migration.

Les entités de base sont les processus et les agents. Ces derniers sont des processus localisés et nommés. Il n'y a pas de construction dans le langage pour les domaines mais les sites sont nommés et ne peuvent être référencés que comme cible d'une migration. On peut aussi voir les agents comme des « domaines » (contenus dans un site). Le langage permet de créer de nouveaux agents.

6.4.1 Aspect répartition

L'espace des domaines est de type hiérarchique mais restreint à deux niveaux : les sites, qui contiennent des agents qui eux-même abritent des processus. Un site est une unité de communication. À l'intérieur d'un agent, les processus peuvent également communiquer. Un agent peut envoyer un message à un processus hébergé par un autre agent avec lequel il partage le même site d'exécution. Pour cela il doit spécifier le nom de l'agent cible. Il existe également dans Nomadic Pict une construction permettant la communication distante et transparente vis-à-vis des sites (calcul *High-Level*). Cependant, cette construction n'est pas primitive dans le sens où il est montré que celle-ci s'implante avec les constructions primitives de communication locale.

6.4.2 Aspect mobilité

On peut distinguer deux types d'entités mobiles : celles entre sites et incarnées par les agents, et celles entre agents d'un même site qui sont des messages. Les agents ont un caractère dynamique vis-à-vis de la migration : l'instruction de migration des agents est subjective et asynchrone par rapport à l'entité migrante. Elle peut déclencher un processus de continuation permettant la synchronisation de l'agent migrant avec la migration. Ici aussi on peut considérer que le contexte d'un agent migrant est fixe. L'instruction permettant l'envoie d'un message à un agent voisin permet de déclencher un processus (une exception) si cet agent est absent. Ainsi, quand un agent migre, et donc perd son voisinage, l'entretien avec des références distantes peut être géré.

6.5 Le $\pi_{1\ell}$-calcul

Le $\pi_{1\ell}$-calcul [Ama97, Ama00] est un π-calcul asynchrone typé intégrant les concepts de localités, de communication distante, de migration et de défaillance. Une caractéristique de ce calcul est qu'il assure l'unicité globale des récepteurs.

Les entités de base sont les processus, les localités et les configurations. Une localité (ou domaine) contient des processus, une configuration est une composition parallèle de localités et de messages. Ces derniers peuvent être une émission au sens du π-calcul, un processus migrant, ou une instruction de contrôle de l'état d'une localité. À chaque localité est donc associé un processus de contrôle. Le calcul manipule des noms de canaux et de localités.

6.5.1 Aspect répartition

L'espace des domaines est plat. L'espace entre les localités formalise un « éther » où circulent les messages. Un domaine est une unité de défaillance et d'exécution. Une localité défaillante signifie la défaillance de tous ses processus. Elle possède donc un état (*Run/Stop*) qui peut être observé et contrôlé grâce à son processus de contrôle. L'observation et le contrôle peuvent s'effectuer aussi bien depuis l'intérieur que depuis l'extérieur de la localité en question.

La réception locale sur un nom est persistante et statique (on ne peut pas créer un récepteur sur un nom reçu). La désignation distante est directe : elle est transparente à la localisation. C'est l'unicité globale des récepteurs qui assure la non ambiguïté de leur localisation. La communication distante

s'effectue en deux étapes : le message sort du domaine courant (qui ne doit pas être défaillant), puis rentre dans la localité cible pour y être consommé. La communication locale est régie par le même principe.

6.5.2 Aspect mobilité

Les entités migrantes sont les processus. L'instruction $\mathsf{spawn}(a, P)$ envoie le processus P à la localité a pour s'y exécuter ; c'est une instruction objective asynchrone, les entités ayant un caractère statique.

La simple connaissance d'un nom suffit pour communiquer sur celui-ci, la communication est transparente. Par conséquent, un processus migrant conserve tous les liens qu'il peut avoir avec des récepteurs sur les noms de canaux qu'il connaît ; le contexte du migrant est donc fixe.

6.6 KLAIM

KLAIM [DFP98, DFP00, DFP99, DFPV00](*Kernel Language for Agents Interaction and Mobility*) est fondé sur le langage de coordination LINDA [GCCC85]. Ce dernier implante un mécanisme de communication asynchrone par l'intermédiaire d'un environnement global partagé (appelé *Tuple Space*) où sont déposés et sélectionnés (par *pattern matching*) les paramètres effectifs des messages. KLAIM complète ce langage avec des notions de sites (emplacements physiques), de localités (emplacement logiques), d'agents et de migration. Les entités de base sont les *tuples* (paramètres effectifs des messages et processus), les processus (processus CCS) et les sites (globalement uniques et organisés en configurations).

6.6.1 Aspect répartition

L'espace des domaines est plat. Un site est essentiellement une unité de migration. Dans une faible mesure il s'agit aussi d'une unité de communication dans le sens où chaque site possède son propre *tuple space* ; un processus désirant envoyer ou recevoir un tuple sur un site distant doit nommer ce site explicitement.

Seules les localités sont nommées. Chaque site possède sont propre environnement d'allocation des localités dans les sites. Les processus ne manipulent que des localités, et c'est au niveau de la sémantique opérationnelle qu'est gérée la répartition physique. La communication locale est anonyme : une valeur est déposée par un processus dans le tuple space local et un autre processus peut le sélectionner par *pattern matching*. Il n'y a donc pas réellement de notion de récepteur.

Dans la communication distante la localité cible est explicitement nommée. Un processus peut « déposer » et « consommer » une valeur dans le tuple space d'un site distant. La communication distante s'effectue en deux temps : le tuple à transmettre est évalué localement et déposé dans le tuple space cible, puis il est consommé.

6.6.2 Aspect mobilité

Les entités mobiles sont les processus ; elles sont à caractère statique. Il y a deux types de migrations (deux instructions), les deux étant objectives. La première $\mathsf{out}(P)@\ell.Q$ migre un processus P à

la localité ℓ, où il sera exécuté avec le contexte de la localité source, puis déclenche la continuation Q localement. Le contexte est donc ici mobile, par réplication. L'autre migration $\mathsf{eval}(P)@\ell.Q$ migre P en ℓ, où il est évalué dans le contexte de ℓ, puis Q est déclenché localement. Il s'agit ici d'appliquer une politique de liaison dynamique : le contexte reste fixe et la relation avec le domaine initial est perdu.

6.7 Le π-calcul Distribué (Dπ)

Le Dπ-calcul [HR98b, HR98a] est un π-calcul synchrone, enrichi de primitives pour la migration (notion de localité, instruction de migration, noms structurés). Les entités de base sont les processus et les configurations (ou systèmes). Une configuration est un système de processus localisés (dans des domaines) en parallèle. Les domaines se manifestent par une construction du langage associant un nom de localité à un processus. Les noms de localités sont globalement uniques à réarangement près géré par une relation d'équivalence structurelle. Le calcul manipule des noms de canaux, des noms de localités et des noms composés, qui consistent en une paire d'un nom de localité et d'un ensemble de noms de canaux (utilisables à la localité indiquée).

6.7.1 Aspect répartition

La topologie de l'espace des domaines est plat. Une localité est une unité d'exécution et de communication. La désignation locale est directe, elle ne nécessite que le nom du canal de communication.

Il n'y a pas de communication distante : un message pour lequel le récepteur correspondant se trouve dans un domaine distant doit faire l'objet d'une migration explicite vers son destinataire.

6.7.2 Aspect mobilité

Les entités migrantes sont les processus ; elles sont de type statique, la migration est objective et asynchrone. Un système de type permet d'assurer qu'un processus migrant utilise les noms de façon cohérente avec ce qui est prescrit par le type de la localité de destination. Le contexte d'un processus migrant est toujours fixe.

CHAPITRE 7

Le π-calcul réparti réceptif

NOUS PRÉSENTONS DANS CE CHAPITRE LE π-CALCUL RÉPARTI RÉCEPTIF (noté Dπ_1^r), qui est un calcul fondé sur le π-calcul distribué (Dπ) de M. HENNESSY et J. RIELY [HR98a, HR98b] introduit dans le chapitre précédent. Notre calcul réparti est une extension de celui présenté dans le chapitre 3 pour prendre en compte les notions de domaine et de migration. Aussi, nous nous efforcerons à mettre en évidence les caractéristiques le différenciant de Dπ.

Ce chapitre est structuré en deux sections. Dans la première, nous présentons la syntaxe et la sémantique opérationnelle dans le style de la machine chimique abstraite. Dans le second, nous donnons le système de types et montrons que le typage est préservé par la réduction.

7.1 Syntaxe et sémantique opérationnelle

Nous étendons π_1^r avec des constructions primitives permettant la distribution spatiale des processus et leur migration. La topologie de l'espace des domaines est *plat*, ce qui permet de définir les termes grâce à une syntaxe à deux niveaux. Le premier niveau concerne les **processus** (ou threads), le second les **réseaux** (ou systèmes). À la différence de Dπ, la communication est ici asynchrone. Une **localité** est une unité de communication et de migration. La communication étant purement locale, la communication distante n'est possible que par migration de messages. À l'échelle globale il n'y a pas unicité des noms de canaux dans le sens où un nom a une « signification » qui est propre à chaque localité. Ainsi, un même nom de canal peut être utilisé dans une localité ℓ pour transmettre une valeur, et être utilisé dans une localité k pour transmettre des canaux. Aussi, les noms de canaux pouvant s'échanger entre des processus distants (*i.e.* localisés dans des sites différents), il est nécessaire de spécifier la localité à laquelle il faut rattacher ce canal. C'est pourquoi, nous définissons une nouvelle catégorie de noms : les **noms composés** (par opposition aux **noms simples**). Ceux-ci consistent en une paire d'un nom de canal et d'un nom de localité, noté $a@\ell$ et signifiant « le canal a utilisé à la localité ℓ ». On utilise les lettres $u, v, \ldots$ pour dénoter des noms simples ou composés. L'ensemble des noms

$u, v, \ldots$	$::=$		*noms*
		a	*simple*
	$\mid$	$a@\ell$	*composé*
w	$::=$	$u \mid \ell : \psi$	
$P, Q, R \ldots$	$::=$		*processus*
		$\mathbf{0}$	*inaction*
	$\mid$	$\overline{a}(\vec{u})$	*message*
	$\mid$	$a(\vec{u}).P$	*réception sur le canal* a
	$\mid$	$(P \mid Q)$	*composition parallèle*
	$\mid$	$[a = b]P, Q$	*branchement conditionnel*
	$\mid$	$(\nu w)P$	*restriction*
	$\mid$	$\mathsf{go}\,\ell.P$	*migration*
	$\mid$	$T(\vec{u})$	*instance de processus paramétré*
T	$::=$		*processus paramétré*
		A	*identificateur*
	$\mid$	$(\mathsf{rec}\ A(\vec{u}).P)$	*processus paramétré récursif*
S	$::=$		*réseaux*
		$\mathbf{0}$	*réseau vide*
	$\mid$	$\ell[\![P]\!]$	*processus localisé à la localité* ℓ
	$\mid$	$(S \mid S')$	*composition parallèle*
	$\mid$	$(\nu w)S$	*restriction*

FIG. 7.1: Syntaxe de $\mathrm{D}\pi_1^r$

simples $\mathcal{N} = \{a, b, \ldots, x, y, \ldots\}$ est à présent partitionné en trois sous-ensembles : $\mathcal{N}_{chan}$ ensemble des noms de canaux, $\mathcal{N}_{val}$ ensemble des valeurs et $\mathcal{N}_{loc} = \{\ell, k, \ldots\}$ ensemble des noms de localités. Cette partition de l'ensemble des noms n'est pas nécessaire dans cette partie – dans [ABL00] cette partition n'a pas été faite – elle nous sera utile pour l'inférence de types développée dans la troisième partie de cette thèse.

La figure 7.1 donne la syntaxe des termes de $\mathrm{D}\pi_1^r$. Les processus sont ceux de π_1^r excepté qu'ils peuvent maintenant manipuler des noms composés. Ils comportent aussi une nouvelle construction ($\mathsf{go}\,\ell.P$) permettant la migration de processus. La restriction peut porter sur un nom simple de canal ou de valeur, sur un nom composé ou sur une localité pour laquelle on indique explicitement le type. Nous verrons en détail dans la section suivante quels sont ces types. La raison pour laquelle le type d'une localité restreinte doit apparaître explicitement dans le terme sera expliquée dans la troisième section dans l'élaboration du système de bonne formation. On utilisera l'opérateur $\_@\ell$ sur les restrictions, défini par :

$$w@\ell = \begin{cases} a@\ell & \text{si } w = a \in \mathcal{N}_{chan} \\ w & \text{sinon} \end{cases}$$

On définit le *sujet* d'une restriction par :

$$\mathit{subj}(a) = \mathit{subj}(a@\ell) = \mathit{subj}(a : \psi) = a$$

Dans $(\nu w)U$, le sujet de w est le seul nom ayant une occurrence dans w qui soit lié dans U. On désigne par $U, V, \ldots$ un terme de n'importe quelle sorte (*i.e.* processus, processus paramétré ou réseaux).

Exemple 7.1.1 Nous donnons, à titre d'exemple, l'implantation du protocole RPC dans le cadre réparti. Un processus client (*Client*) localisé en ℓ, désire invoquer un service nommé a localisé en k, avec des arguments $\vec{d}$, et attend le résultat qui sera traité avec la continuation P. Le client doit créer un canal de retour privé à sa propre localité, et l'envoyer comme sujet d'un nom composé indiquant la localité de retour, avec les données $\vec{d}$. Un tel processus s'écrit de la façon suivante :

$$\mathit{Client} \stackrel{\mathsf{def}}{=} (\nu r)(\mathsf{go}\, k.\overline{a}(\vec{d}, r@\ell) \mid r(\vec{v}).P)$$

Supposons maintenant que le service (*Service*) localisé en k calcule une fonction $f(\vec{d})$ des arguments et renvoie le résultat au client ; ceci peut simplement s'écrire :

$$\mathit{Service} \stackrel{\mathsf{def}}{=} a^*(\vec{x}, y@z).\mathsf{go}\, z.\overline{y}(f(\vec{x}))$$

Comme ici, il est courant que seul un message soit migré. On introduit donc une notation spécifique pour ce type de migration :

$$\overline{a}@\ell(\vec{u}) \stackrel{\mathsf{def}}{=} \mathsf{go}\, \ell.\overline{a}(\vec{u})$$

Finalement, le réseau obtenu est :

$$\ell[\![\mathit{Client}]\!] \mid k[\![\mathit{Service}]\!]$$

□

Nous décrivons maintenant formellement le comportement des réseaux grâce à une relation de réduction $U \rightarrow U'$. La définition de cette relation induit la notion de *substitution*. Il y a deux types de substitutions : de noms à noms, et de processus paramétrés à identificateurs de processus. Une substitution de noms est une application S d'un sous-ensemble fini $\mathit{dom}(\mathsf{S})$ de $\mathcal{N}$ dans $\mathcal{N}$. On dénote par $\mathit{im}(\mathsf{S})$ l'image de S, c'est-à-dire l'ensemble $\{\mathsf{S}(a) \mid a \in \mathit{dom}(\mathsf{S})\}$. L'application d'une substitution à un terme U est notée $[\mathsf{S}]U$ qui n'est définie que s'il n'y a pas de capture de variable, c'est-à-dire si $\mathit{im}(\mathsf{S}) \cap \mathit{bn}(U) = \emptyset$. Si S est l'application :

$$a_1 \mapsto b_1, \ldots, a_k \mapsto b_k$$

alors $[\mathsf{S}]U$ est aussi dénoté $[b_1/a_1, \ldots, b_k/a_k]U$, ou $[\vec{b}/\vec{a}]U$. On utilise aussi la notation $[v/u]U$ où u et v sont des noms éventuellement composés ; dans ce cas, si $u = x@y$ alors x est supposé être différent de y, et v doit être un nom composé $a@b$, et la vrai substitution est $[a/x, b/y]$; alors que si u est un nom simple, alors v doit également être un nom simple. La substitution de processus paramétrés à des identificateurs de processus ne nous est nécessaire que dans le cas de la substitution d'un seul identificateur, noté $[T/A]P$. Ici aussi, elle n'est définie que s'il n'y a pas de capture de variable, c'est-à-dire si $\mathit{fn}(T) \cap \mathit{bn}(P) = \emptyset$.

On rappelle la définition de l'application d'une substitution pour le cas des constructions liantes (les autres cas étant triviaux), où par convention $\mathsf{S}(a) = a$ si $a \notin \mathit{dom}(\mathsf{S})$:

$$
\begin{array}{rcll}
[\mathsf{S}]a(\vec{u}).P & = & \mathsf{S}(a)(\vec{u}).[\mathsf{S}']P & \mathsf{S}' = \mathsf{S} \upharpoonright (\mathit{dom}(\mathsf{S}) - \mathit{nm}(\vec{u})) \\
[\mathsf{S}](\nu w)P & = & \left\{ \begin{array}{ll} (\nu a)[\mathsf{S}']P & \text{si } w = a \\ (\nu a@\mathsf{S}(\ell))[\mathsf{S}']P & \text{si } w = a@\ell \\ (\nu a : \mathsf{S}(\psi))[\mathsf{S}']P & \text{si } w = a : \psi \end{array} \right\} & \text{où } \mathsf{S}' = \mathsf{S} \upharpoonright (\mathit{dom}(\mathsf{S}) - \{a\}) \\
[\mathsf{S}](\mathsf{rec}\, A(\vec{u}).P) & = & (\mathsf{rec}\, A(\vec{u}).[\mathsf{S}']P) & \mathsf{S}' = \mathsf{S} \upharpoonright (\mathit{dom}(\mathsf{S}) - \mathit{nm}(\vec{u})) \\
[\mathsf{S}](\nu \ell : \psi)S & = & (\nu \ell : \mathsf{S}(\psi))[\mathsf{S}']S & \mathsf{S}' = \mathsf{S} \upharpoonright (\mathit{dom}(\mathsf{S}) - \{\ell\})
\end{array}
$$

Nous reviendrons plus loin sur l'application d'une substitution à un type ψ. La substitution de processus paramétrés à des identificateurs est essentiellement donnée par :

$$
[T/A](\mathsf{rec}\, B(\vec{u}).P) = \left\{ \begin{array}{ll} (\mathsf{rec}\, B(\vec{u}).[T/A]P) & \text{si } A \neq B \\ (\mathsf{rec}\, B(\vec{u}).P) & \text{sinon} \end{array} \right.
$$

La relation $=_\alpha$ d'*α-conversion* est la plus petite congruence satisfaisant les axiomes suivants :

$$
\begin{array}{rcll}
a(\vec{u}).P & =_\alpha & a(\vec{v}).[\vec{v}/\vec{u}]P & \mathit{nm}(\vec{v}) \cap (\mathit{nm}(P) \cup \{a\}) = \emptyset \\
(\nu a)U & =_\alpha & (\nu b)[b/a]U & b \notin \mathit{nm}(U) \\
(\nu a@\ell)U & =_\alpha & (\nu b@\ell)[b/a]U & b \notin \mathit{nm}(U) \cup \{\ell\} \\
(\nu \ell : \psi)U & =_\alpha & (\nu k : \psi)[k/\ell]U & k \notin \mathit{nm}(U) \cup \mathit{nm}(\psi) \\
(\mathsf{rec}\ A(\vec{u}).P) & =_\alpha & (\mathsf{rec}\ B(\vec{v}).[B/A][\vec{v}/\vec{u}]P) & B \notin \mathit{nm}(P),\ \mathit{nm}(\vec{v}) \cap \mathit{nm}(P) = \emptyset
\end{array}
$$

Notons que, d'après la convention implicite selon laquelle tous les noms liés par une réception ou une récursion sont tous distincts, dans le cas des termes $a(\vec{u}).P$ et $(\mathsf{rec}\ A(\vec{u}).P)$ la substitution $[\vec{v}/\vec{u}]$ doit être injective.

Les termes sont considérés identiques s'il sont égaux à renommage près des noms liés, c'est-à-dire s'ils sont α-équivalents. Cependant, ce n'est pas le seul cas d'égalité que nous avons besoin de considérer : pour définir la relation de réduction dans le style de la machine chimique abstraite, on utilise une *relation d'équivalence structurelle*. C'est la plus petite relation d'équivalence contenant les axiomes suivants :

$$
\begin{array}{rcll}
(\mathbf{0} \mid U) & \equiv & U & \textit{neutralité} \\
(U \mid (V \mid W)) & \equiv & ((U \mid V) \mid W) & \textit{associativité} \\
(U \mid V) & \equiv & (V \mid U) & \textit{commutativité} \\
((\nu w)U \mid V) & \equiv & (\nu w)(U \mid V) \quad \mathit{subj}(w) \notin \mathit{fn}(V) & \textit{migration de portée (1)} \\
\ell[\![(\nu w)P]\!] & \equiv & (\nu w@\ell)\ell[\![P]\!] \quad \ell \neq \mathit{subj}(w) & \textit{migration de portée (2)} \\
\ell[\![P \mid Q]\!] & \equiv & \ell[\![P]\!] \mid \ell[\![Q]\!] & \textit{routage}
\end{array}
$$

et satisfaisant les règles suivantes :

$$
\frac{U =_\alpha V}{U \equiv V} \qquad \frac{U \equiv V\ ,\ V \equiv W}{U \equiv W} \qquad \frac{U \equiv V}{\mathbf{E}[U] \equiv \mathbf{E}[V]}
$$

$$
\begin{array}{rcll}
(\overline{a}(\vec{v}) \mid a(\vec{u}).P) & \to & [\vec{v}/\vec{u}]P & \textit{loi de communication} \\
k[\![\mathsf{go}\,\ell.P]\!] & \to & \ell[\![P]\!] & \textit{loi de migration} \\
(\mathsf{rec}\,A(\vec{u}).P)(\vec{v}) & \to & [(\mathsf{rec}\,A(\vec{u}).P)/A][\vec{v}/\vec{u}]P & \textit{dépliage} \\
[a=a]P,Q & \to & P & \textit{branchement-vrai} \\
[a=b]P,Q & \to & Q \qquad a \neq b & \textit{branchement-faux}
\end{array}
$$

FIG. 7.2: Lois de réduction

où $\mathbf{E}$ est un *contexte d'évaluation* donné par la grammaire suivante :

$$\mathbf{E} \ ::= \ [] \mid (\mathbf{E} \mid U) \mid (\nu w)\mathbf{E} \mid \ell[\![\mathbf{E}]\!]$$

La loi de routage est utilisée de gauche à droite pour permettre à un processus migrant de sortir d'une localité ($\ell[\![\mathsf{go}\,k.P \mid Q]\!] \equiv \ell[\![\mathsf{go}\,k.P]\!] \mid \ell[\![Q]\!]$), et de droite à gauche pour permettre au processus de rejoindre sa localité de destination. Modulo cette équivalence, tout processus P peut être mis sous une forme canonique :

$$P \ \equiv \ (\nu\vec{w})(R_1 \mid \ldots \mid R_n)$$

où les R_i sont des « molécules » c'est-à-dire soit des messages ($\overline{a}(\vec{v})$), soit des récepteurs ($a(\vec{u}).Q$), soit des processus conditionnels ($[a=b]Q_1, Q_2$), récursifs ($T(\vec{u})$) ou migrants ($\mathsf{go}\,\ell.Q$). De même, tout réseau peut être mis sous une forme canonique :

$$S \ \equiv \ (\nu\vec{w})(\ell_1[\![P_1]\!] \mid \ldots \mid \ell_n[\![P_n]\!])$$

où chaque P_i est une composition parallèle de molécules, et les ℓ_i sont deux à deux distincts. Par exemple, pour le RPC (exemple 7.1.1), une forme canonique est :

$$\ell[\![\mathit{Client}]\!] \mid k[\![\mathit{Service}]\!] \ \equiv (\nu r@\ell)\big(\ell[\![\mathsf{go}\,k.\overline{a}(\vec{d}, r@\ell) \mid r(\vec{v}).P]\!] \mid k[\![a^*(\vec{x}, y@z).\mathsf{go}\,z.\overline{y}(f(\vec{x}))]\!]\big)$$

Les axiomes de réduction nous disent comment les molécules agissent et interagissent. Ils sont donnés dans la figure 7.2. Il faut y ajouter deux règles indiquant que la réduction est définie à équivalence structurelle près et qu'elle peut s'appliquer sous contexte :

$$\frac{V \equiv U \ , \ U \to U' \ , \ U' \equiv V'}{V \to V'} \qquad\qquad \frac{U \to U'}{\mathbf{E}[U] \to \mathbf{E}[U']}$$

On notera que dans les lois de communication et de dépliage on suppose que la substitution est définie. On dénote par $U \twoheadrightarrow V$ la réduction $\to$ en interdisant la loi de communication. Cette réduction est fortement normalisable, c'est-à-dire, étant donné un terme, toute séquence de réductions de ce terme par $\twoheadrightarrow$ est finie.

Propriété 7.1.2 *Tout terme U est fortement normalisable pour la relation de réduction $\twoheadrightarrow$.*

Preuve: On associe à chaque terme clos U du langage un entier $n(U)$ comme suit :

$$\begin{gathered} n(\mathbf{0}) \;=\; n(\overline{a}(\vec{u})) \;=\; n(a(\vec{u}).P) \;=\; 0 \\ n((\nu w)P) \;=\; n(\ell[\![P]\!]) \;=\; n(P) \\ n(U \mid V) \;=\; n(U) + n(V) \\ n([a=b]P, Q) \;=\; n(P) + n(Q) + 1 \\ n(\mathsf{go}\,\ell.P) \;=\; n((\mathsf{rec}\ A(\vec{u}).P)(\vec{v})) \;=\; n(P) + 1 \\ n((\nu w)S) \;=\; n(S) \end{gathered}$$

On peut facilement montrer que $U \equiv V \;\Rightarrow n(U) = n(V)$. Puisque $n(U)$ ne dépend pas des noms et que toutes les occurrences d'identificateurs sont gardées par des réceptions, on remarque qu'on a en particulier

$$n([(\mathsf{rec}\,A(\vec{u}).P)/A][\vec{v}/\vec{u}]P) \;=\; n(P)$$

Il est alors facile de montrer que $U \rightarrow V \;\Rightarrow n(V) < n(U)$. □

La migration d'un processus est illustrée par la réduction détaillée suivante :

$$\ell[\![\mathsf{go}\,k.P \mid Q]\!] \mid k[\![R]\!] \;\equiv\; \ell[\![\mathsf{go}\,k.P]\!] \mid \ell[\![Q]\!] \mid k[\![R]\!] \;\twoheadrightarrow\; k[\![P]\!] \mid \ell[\![Q]\!] \mid k[\![R]\!] \;\equiv\; \ell[\![Q]\!] \mid k[\![R \mid P]\!]$$

La loi de communication nous indique que celle-ci est effectivement purement locale : il ne peut y avoir réduction du réseau $\ell[\![\overline{a}(\vec{v})]\!] \mid k[\![a(\vec{u}).P]\!]$ si $k \neq \ell$; les messages doivent être explicitement acheminés comme dans $\ell[\![\mathsf{go}\,k.\overline{a}(\vec{v})]\!] \mid k[\![a(\vec{u}).P]\!]$. À la différence de [Ama00, FGL$^+$96] où les messages peuvent traverser les domaines de manière transparente pour rejoindre un récepteur, nous avons une sémantique « migrer et communiquer » à la Dπ qui est aussi celle des Ambients [CG98].

Exemple 7.1.3 Nous reprenons l'exemple du RPC pour montrer son comportement :

$$\begin{aligned} \ell[\![\mathit{Client}]\!] \mid k[\![\mathit{Service}]\!] \;&\equiv\; (\nu r@\ell)\big(\ell[\![\mathsf{go}\,k.\overline{a}(\vec{d}, r@\ell)]\!] \mid \ell[\![r(\vec{v}).P]\!] \mid k[\![\mathit{Service}]\!]\big) \\ &\twoheadrightarrow\; (\nu r@\ell)\big(k[\![\overline{a}(\vec{d}, r@\ell)]\!] \mid \ell[\![r(\vec{v}).P]\!] \mid k[\![\mathit{Service}]\!]\big) \\ &\equiv\; (\nu r@\ell)\big(k[\![\mathit{Service} \mid \overline{a}(\vec{d}, r@\ell)]\!] \mid \ell[\![r(\vec{v}).P]\!]\big) \end{aligned}$$

Puisque $\mathit{Service} \;\twoheadrightarrow\; a(\vec{x}, y@z).(\mathsf{go}\,z.\overline{y}(f(\vec{y})) \mid \mathit{Service})$, après réception du message sur a, on a :

$$\begin{aligned} \ell[\![\mathit{Client}]\!] \mid k[\![\mathit{Service}]\!] \;&\stackrel{*}{\rightarrow}\; (\nu r@\ell)\big(k[\![\mathit{Service} \mid \mathsf{go}\,\ell.\overline{r}(f(\vec{d}))]\!] \mid \ell[\![r(\vec{v}).P]\!]\big) \\ &\equiv\; (\nu r@\ell)\big(k[\![\mathit{Service}]\!] \mid k[\![\mathsf{go}\,\ell.\overline{r}(f(\vec{d}))]\!] \mid \ell[\![r(\vec{v}).P]\!]\big) \\ &\twoheadrightarrow\; (\nu r@\ell)\big(k[\![\mathit{Service}]\!] \mid \ell[\![\overline{r}(f(\vec{d}))]\!] \mid \ell[\![r(\vec{v}).P]\!]\big) \\ &\equiv\; (\nu r@\ell)\big(k[\![\mathit{Service}]\!] \mid \ell[\![\overline{r}(f(\vec{d})) \mid r(\vec{v}).P]\!]\big) \\ &\rightarrow\; (\nu r@\ell)\big(k[\![\mathit{Service}]\!] \mid \ell[\![[f(\vec{d})/\vec{v}]P]\!] \end{aligned}$$

On a ici fait l'hypothèse – qui sera vérifiée par le système de types – que les vecteurs $\vec{d}$ et $\vec{x}$ sont de même longueur, ainsi que $f(\vec{d})$ et $\vec{v}$. □

$$
\begin{array}{lcll}
\tau, \sigma \ldots & ::= & \mathit{val} \mid \gamma \mid \psi \mid \gamma@\psi & \textit{types} \\
\gamma, \delta \ldots & ::= & \mathit{Ch}(\tau_1, \ldots, \tau_n) & \textit{types de canaux} \\
\psi, \phi \ldots & ::= & \{a_1 : \gamma_1, \ldots, a_n : \gamma_n\} & \textit{types de localités}
\end{array}
$$

FIG. 7.3: Types

7.2 Le système de types

Comme nous l'avons vu dans l'exemple du RPC, la communication de noms de localités est cruciale dans notre calcul. Elle permet de connaître la localité où il faut éventuellement migrer ou retourner un résultat. Du point de vue du typage, les canaux devant être utilisés conformément à leurs types, il faut également assigner des types aux noms de localités et aux noms composés. L'idée est que le type d'une localité est l'enregistrement des canaux, et de leurs types, sur lesquels une communication est possible dans cette localité. Dans la terminologie usuelle ces types sont des **types dépendants** en ce sens qu'ils dépendent de noms du calcul. Ils ont la forme $\{a_1 : \gamma_1, \ldots, a_n : \gamma_n\}$ où tous les a_i sont des noms de canaux distincts et l'ordre n'importe pas. On écrira parfois $\{a : \gamma, \psi\}$ pour $\{a : \gamma, a_1 : \gamma_1, \ldots, a_n : \gamma_n\}$ si $\psi = \{a_1 : \gamma_1, \ldots, a_n : \gamma_n\}$, et $\mathit{dom}(\{a_1 : \gamma_1, \ldots, a_n : \gamma_n\})$ pour l'ensemble $\{a_1, \ldots, a_n\}$. On dénote par $\psi \sqcap \phi$ l'union des types de localités ϕ et ψ qui n'est définie que si ψ et ϕ assignent les même types aux canaux qu'elles ont en commun. Le type d'un nom composé $a@\ell$ est simplement un type de la forme $\gamma@\psi$ où γ est le type de a et ψ celui de ℓ. Ces types sont des **types existentiels** : intuitivement, on peut comprendre $\gamma@\psi$ comme $\exists x.\{x : \gamma, \psi\}$. Dans la suite, on écrira $\gamma^{@}$ pour $\gamma@\{\}$. Les types sont reportés dans la figures 7.3. On notera que, par construction, dans les types il ne peut pas y avoir d'occurrence de noms de valeur ou de localité. On fera l'hypothèse qu'aucun nom d'un type ψ – *i.e.* $\mathit{nm}(\psi)$ – qui a une occurrence dans un terme n'est lié par une réception ou un processus récursif dans ce terme, afin que, par exemple, $a(b).(\nu\ell : \{b : \gamma, c : \delta\}).\overline{b}()$ ne puisse pas être écrit. En effet, dans ce dernier terme, si $\gamma \neq \delta$, la réception du nom c provoquerait une erreur de type en cours de réduction. Dans [HR98a], cette hypothèse est directement obtenue en distinguant variables – pouvant seules être liées par une réception – et noms – pouvant seuls être restreints et apparaître dans les types de localité. Cependant, nous ne faisons pas cette distinction car il est facile d'analyser statiquement les termes pour vérifier cette propriété. En effet, on peut par exemple définir l'ensemble des canaux $\mathit{cr}(P)$ qui ont une occurrence dans un type ψ apparaissant dans P de la façon suivante :

$$
\begin{array}{c}
\mathit{cr}((\nu\ell : \psi)U) = \mathit{cr}(U) \cup \mathit{nm}(\psi) \\
\mathit{cr}(\overline{a}(\vec{u})) = \mathit{cr}(\mathbf{0}) = \mathit{cr}(A) = \emptyset \\
\mathit{cr}(a(\vec{u}).P) = \mathit{cr}(\mathsf{rec}\ A(\vec{u}).P) = \mathit{cr}(\mathsf{go}\,\ell.P) = \mathit{cr}(\ell[\![P]\!]) = \mathit{cr}(P) \\
\mathit{cr}((\nu a)U) = \mathit{cr}((\nu a@\ell)U) = \mathit{cr}(U) \\
\mathit{cr}([a = b]P, Q) = \mathit{cr}(P) \cup \mathit{cr}(Q) \\
\mathit{cr}(U \mid V) = \mathit{cr}(U) \cup \mathit{cr}(V) \\
\mathit{cr}(T(\vec{u})) = \mathit{cr}(T)
\end{array}
$$

Ainsi, il suffit de vérifier que tout sous-terme $a(\vec{u}).P$ et $(\mathsf{rec}\ A(\vec{u}).P)$ satisfait $\mathit{nm}(\vec{u}) \cap \mathit{cr}(P) = \emptyset$.

Remarque 7.2.1 Si $\mathit{dom}(\mathsf{S}) \cap \mathit{cr}(U) = \emptyset$ alors $[\mathsf{S}]U$ est défini.

Ceci dit, le fait que nous ne distinguons pas les variables des noms nous permet de typer des termes qui ne le sont pas dans le système de types original de Dπ[HR98a]. Un exemple en est donné dans le chapitre 10.

Comme d'habitude, le système de types doit garantir qu'il n'y aura pas d'erreur d'arité en cours de réduction et qu'un nom de canal (respectivement de localité, de valeur) est utilisé en tant que tel. Nous avons des séquents de la forme $\Gamma \vdash_\ell P$, pour vérifier que P, placé à la localité courante ℓ, est conforme à la spécification de typage Γ, et des séquents $\Gamma \vdash_\ell T : \gamma$ et $\Gamma \vdash S$ pour, respectivement, les processus paramétrés et les réseaux. Comme dans système de types donné dans le chapitre 3 pour π^r, nous utilisons aussi un système de types auxiliaire pour typer les noms qui se révèle maintenant très pratique. Cependant, et à la différence de [ABL00], nous en définissons ici deux dans l'objectif de supprimer les règles explicites d'affaiblissement

$$\frac{\Gamma \vdash_\ell P}{\Gamma, \Delta \vdash_\ell P} \qquad \frac{\Gamma \vdash S}{\Gamma, \Delta \vdash S}$$

et ainsi, ces règles étant source de non-déterminisme, d'anticiper le problème d'inférence de types. L'idée est de reporter les affaiblissements dans les axiomes. Il est toutefois nécessaire de dupliquer les axiomes, chacun ayant une variante « faible » et une variante « forte ». Nous avons donc un système de types *faible* pour les noms dont les séquents sont de la forme $\Gamma \vdash^W_\ell u_1 : \tau_1, \ldots, u_n : \tau_n$, et un système de types fort dont les séquents sont de la forme $\Gamma \vdash_\ell u_1 : \tau_1, \ldots, u_n : \tau_n$. Dans tous les séquents le contexte de typage Γ est une application d'un sous-ensemble fini $\mathit{dom}(\Gamma)$ de $\mathcal{N}_{loc} \cup \mathcal{N}_{val} \cup \mathcal{P}$ dans l'ensemble des types, sujet aux contraintes suivantes :

1. si $a \in \mathit{dom}(\Gamma) \cap \mathcal{N}$ alors $\Gamma(a)$ est soit *val* et $a \in \mathcal{N}_{val}$, soit un type de localité ψ et $a \in \mathcal{N}_{loc}$;
2. si $A \in \mathit{dom}(\Gamma) \cap \mathcal{P}$ alors $\Gamma(A)$ est un type de canal γ ;
3. si $a \in \mathit{dom}(\Gamma)$ alors a n'a pas d'occurrence dans $\mathit{im}(\Gamma)$, c'est-à-dire, a n'apparaît dans aucun type assigné par Γ.

La troisième contrainte n'est en fait qu'une conséquence de la première puisque, comme il a été dit plus haut, seuls des noms de canaux peuvent apparaître dans les types. On écrit les contextes comme d'habitude, c'est-à-dire une liste d'assignations de types :

$$\Gamma \;=\; \ldots, a : \mathit{val}, \ldots, \ell : \{c_1 : \gamma_1, \ldots, c_n : \gamma_n\}, \ldots, A : \mathit{Ch}(\tau), \ldots$$

Dans le système de types on utilise un opérateur partiel d'union sur les contextes, défini par :

$$(\Gamma, \Delta)(x) = \begin{cases} \Delta(x) & \text{si } x \in \mathit{dom}(\Delta) - \mathit{dom}(\Gamma) \text{ ou } \Delta(x) = \Gamma(x) \\ \phi \sqcap \psi & \text{si } \Delta(x) = \phi \;\&\; \Gamma(x) = \psi \\ \Gamma(x) & \text{si } x \in \mathit{dom}(\Gamma) - \mathit{dom}(\Delta) \end{cases}$$

Un contexte Δ, Γ peut être mal défini – ou *illégal* – si, par exemple, ℓ a le type $\{a : \gamma\}$ dans Γ, et $\{a : \delta\}$ dans Δ, et $\gamma \neq \delta$.

$$\frac{}{a : \tau \vdash_\ell a : \tau} \qquad \frac{}{\ell : \{a : \gamma\} \vdash_\ell a : \gamma} \qquad \frac{}{\ell : \{a : \gamma, \psi\} \vdash_k a@\ell : \gamma@\psi}$$

$$\frac{\Gamma \vdash_\ell \vec{u} : \vec{\tau}\ ,\ \Delta \vdash_\ell \vec{v} : \vec{\sigma}}{\Gamma, \Delta \vdash_\ell \vec{u} : \vec{\tau}, \vec{v} : \vec{\sigma}}$$

FIG. 7.4: Système de types fort pour les noms

$$\frac{}{\Gamma, a : \tau \vdash^W_\ell a : \tau} \qquad \frac{}{\Gamma, \ell : \{a : \gamma\} \vdash^W_\ell a : \gamma} \qquad \frac{}{\Gamma, \ell : \{a : \gamma, \psi\} \vdash^W_k a@\ell : \gamma@\psi}$$

$$\frac{\Gamma \vdash^W_\ell \vec{u} : \vec{\tau}\ ,\ \Gamma \vdash^W_\ell \vec{v} : \vec{\sigma}}{\Gamma \vdash^W_\ell \vec{u} : \vec{\tau}, \vec{v} : \vec{\sigma}}$$

FIG. 7.5: Système de types faible pour les noms

$$\frac{}{\Gamma \vdash_\ell \mathbf{0}} \qquad \frac{\Gamma \vdash^W_\ell \vec{u} : \vec{\tau}}{\ell : \{a : \mathit{Ch}(\vec{\tau})\}, \Gamma \vdash_\ell \overline{a}(\vec{u})} \qquad \frac{\Gamma, \Delta \vdash_\ell P \quad , \quad \Delta \vdash_\ell \vec{u} : \vec{\tau}}{\ell : \{a : \mathit{Ch}(\vec{\tau})\}, \Gamma \vdash_\ell a(\vec{u}).P}$$

$$\frac{\Gamma \vdash_\ell P \quad , \quad \Gamma \vdash_\ell Q}{a, b : \mathit{val}, \Gamma \vdash_\ell [a = b]P, Q} \qquad \frac{\Gamma \vdash_\ell P \quad , \quad \Gamma \vdash_\ell Q}{\ell_1, \ell_2 : \{\}, \Gamma \vdash_\ell [\ell_1 = \ell_2]P, Q}$$

$$\frac{a : \mathit{val}, \Gamma \vdash_\ell P \quad , \quad a \in \mathcal{N}_{val}}{\Gamma \vdash_\ell (\nu a)P} \qquad \frac{\ell : \{a : \gamma\}, \Gamma \vdash_\ell P \quad , \quad a \in \mathcal{N}_{chan}}{\Gamma \vdash_\ell (\nu a)P} \qquad \frac{k : \{a : \gamma\}, \Gamma \vdash_\ell P}{\Gamma \vdash_\ell (\nu a@k)P}$$

$$\frac{k : \psi, \Gamma \vdash_\ell P}{\Gamma \vdash_\ell (\nu k : \psi)P} \qquad \frac{\Gamma \vdash_\ell P \quad , \quad \Gamma \vdash_\ell Q}{\Gamma \vdash_\ell P \mid Q} \qquad \frac{\Gamma \vdash_k P}{\Gamma \vdash_\ell \mathsf{go}\, k.P}$$

$$\frac{\Gamma \vdash_\ell P \quad , \quad P =_\alpha Q}{\Gamma \vdash_\ell Q} \qquad \frac{}{A : \mathit{Ch}(\vec{\tau}), \Gamma \vdash_\ell A : \mathit{Ch}(\vec{\tau})}$$

$$\frac{A : \mathit{Ch}(\vec{\tau}), \Gamma, \Delta \vdash_\ell P \quad , \quad \Delta \vdash_\ell \vec{u} : \vec{\tau}}{\Gamma \vdash_\ell (\mathsf{rec}\ A(\vec{u}).P) : \mathit{Ch}(\vec{\tau})} \qquad \frac{\Gamma \vdash_\ell T : \mathit{Ch}(\vec{\tau}) \quad , \quad \Gamma \vdash^W_\ell \vec{u} : \vec{\tau}}{\Gamma \vdash_\ell T(\vec{u})}$$

FIG. 7.6: Système de types pour les processus

$$\frac{}{\Gamma \vdash \mathbf{0}} \qquad \frac{\Gamma \vdash_\ell P}{\Gamma \vdash \ell[\![P]\!]} \qquad \frac{\Gamma \vdash S \quad , \quad \Gamma \vdash S'}{\Gamma \vdash S \mid S'} \qquad \frac{\Gamma \vdash S \quad , \quad S =_\alpha S'}{\Gamma \vdash S'}$$

$$\frac{\ell : \{a : \gamma\}, \Gamma \vdash S}{\Gamma \vdash (\nu a@\ell)S} \qquad \frac{a : \mathit{val}, \Gamma \vdash S \quad , \quad a \in \mathcal{N}_{val}}{\Gamma \vdash (\nu a)S} \qquad \frac{\ell : \psi, \Gamma \vdash S}{\Gamma \vdash (\nu \ell : \psi)S}$$

FIG. 7.7: Système de types pour les réseaux

Dans les figures 7.4 et 7.5, $\Gamma \vdash_\ell u : \tau$ et $\Gamma \vdash_\ell^W u : \tau$ signifient : u a le type τ quand il est localisé en ℓ. Un axiome fort requiert que le contexte Γ comporte exactement ce qui est nécessaire au typage de $u : \tau$ en ℓ et rien d'autre ; alors que le contexte de l'axiome faible peut contenir des informations inutiles au typage de $u : \tau$ en ℓ. En particulier, on a unicité du contexte de typage dans le système de types fort pour les noms :

Remarque 7.2.2 Si $\Gamma \vdash_\ell \vec{u} : \vec{\tau}$ et $\Delta \vdash_\ell \vec{u} : \vec{\tau}$ alors $\Gamma = \Delta$.

On utilise le système de types fort lorsqu'on désire typer les paramètres formels d'une réception ou d'un processus paramétré (figure 7.6). En effet, pour typer le corps d'une réception il faut introduire dans le contexte de typage les types des paramètres formels et uniquement cette information. Réciproquement, on utilise le système faible pour le typage des paramètres effectifs (des messages et des processus paramétrés). Les deux systèmes ont une forte corrélation comme le montre la remarque suivante.

Remarque 7.2.3 Si $\Gamma \vdash_\ell^W \vec{u} : \vec{\tau}$ alors il existe des contextes Δ et Θ tels que $\Gamma = \Delta, \Theta$ et $\Delta \vdash_\ell \vec{u} : \vec{\tau}$. D'autre part, si $\Delta \vdash_\ell \vec{u} : \vec{\tau}$, alors $\Delta \vdash_\ell^W \vec{u} : \vec{\tau}$.

Dans les figures 7.6 et 7.7, par la convention habituelle, on suppose que dans les règles pour les constructions liantes, c'est-à-dire la réception, la restriction et la récursion, les variables liées n'ont pas d'occurrence dans le contexte résultant. Par exemple, dans la construction de réception, les noms dans $\vec{u}$ n'ont pas d'occurrence dans Γ, $\{a, \ell\}$ et $\vec{\tau}$.

En contre partie d'avoir éliminé les règles d'affaiblissement explicites, il faut introduire des règles d'α-conversion. Celles-ci sont nécessaires pour que le typage soit invariant par α-conversion. D'après la convention du paragraphe précédent, en l'absence de ces règles d'α-conversions, nous pourrions inférer $\Gamma, k : \psi \vdash_\ell (\nu k' : \phi)P$ mais pas $\Gamma, k : \psi \vdash_\ell (\nu k : \phi)[k/k']P$, ce qui est généralement attendu et nécessaire à la propriété de préservation du typage par réduction. L'introduction de ces règles ne posera pas de problème pour l'inférence de types car, par chance, elles n'introduisent pas de non-déterminisme. En effet, il est facile de se convaincre que dans l'inférence d'un séquent cette règle peut n'être utilisée qu'une seule fois et ce, en conclusion de l'inférence : si on désire inférer un séquent $\Gamma \vdash S$, on construira l'inférence – sans règle d'α-conversion – qui prouve le séquent $\Gamma \vdash S'$ où $S' =_\alpha S$ et tel que tous les noms liés dans S' n'apparaissent pas dans Γ, sont tous distincts et différents des noms libres de S'.

Remarquons que dans la règle d'émission, l'utilisation du système de type faible pour les noms induit une forme de sous-typage implicite sur les localités. Il est en effet possible de prouver le séquent

$$k : \{b : \gamma_0, c : \gamma_1\}, \ell : \{a : \mathit{Ch}(\{\})\} \vdash_\ell \overline{a}(k)$$

même si le type de k donné par le contexte est plus « généreux » que celui attendu par a. Nous reviendrons en détail sur cette forme de sous-typage dans la troisième partie de cette thèse. Dans les règles concernant le branchement conditionnel, nous voyons qu'il est permis de comparer aussi bien des valeurs que des localités. Pour typer un processus migrant $\mathsf{go}\,\ell.P$, il suffit de typer P à la localité courante ℓ, la localité courante initiale étant sans importance.

Exemple 7.2.4 Le terme $a(x@y).\overline{x}@y()$ permet, après réception d'un nom localisé, de migrer un message sur ce nom à la localité où il est défini. On le type de la manière suivante :

$$\dfrac{\dfrac{\dfrac{}{y : \{x : Ch()\} \vdash_y \overline{x}()}}{y : \{x : Ch()\} \vdash_\ell \mathsf{go}\, y.\overline{x}()} \qquad \dfrac{}{y : \{x : Ch()\} \vdash_\ell x@y : Ch()^{@}}}{\ell : \{a : Ch(Ch()^{@})\} \vdash_\ell a(x@y).\overline{x}@y()}$$

En revanche, le terme $a(x, y).\overline{x}@y()$ n'est pas typable car il faudrait utiliser le séquent valide

$$y : \{x : Ch()\} \vdash_y x : Ch(), y : \{x : Ch()\}$$

mais il nous conduirait à typer $a(x, y).\overline{x}@y()$ à la localité courante y ce qui contredirait notre convention sur les noms liés.
Le terme $a(x@y).(\overline{x}@y() \mid \overline{b}@y())$ montre qu'après réception d'un nom localisé il est possible de communiquer sur d'autres noms de sa localité, noms qui doivent être « publics ». Ce terme est typable si a est un canal communiquant des noms localisés (de types $Ch()$) à des localités où un canal b a le type $Ch()$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash_y \overline{x}()}}{\Gamma \vdash_\ell \mathsf{go}\, y.\overline{x}()} \qquad \dfrac{\dfrac{}{\Gamma \vdash_y \overline{b}()}}{\Gamma \vdash_\ell \mathsf{go}\, y.\overline{b}()}}{y : \{x : Ch(), b : Ch()\} \vdash_\ell \overline{x}@y() \mid \overline{b}@y()} \qquad \dfrac{}{y : \{x : Ch(), b : Ch()\} \vdash_\ell x@y : Ch()@\{b : Ch()\}}}{\ell : \{a : Ch(Ch()@\{b : Ch()\})\} \vdash_\ell a(x@y).(\overline{x}@y() \mid \overline{b}@y())}$$

où $\Gamma = y : \{x : Ch(), b : Ch()\}$.
Enfin, le terme $a(y).\overline{b}@y$ après réception d'une localité, y migre un message sur un canal public b de cette localité. Il est typable de la façon suivante :

$$\dfrac{\dfrac{\dfrac{}{y : \{b : Ch()\} \vdash_y \overline{b}()}}{y : \{b : Ch()\} \vdash_\ell \mathsf{go}\, y.\overline{b}()} \qquad \dfrac{}{y : \{b : Ch()\} \vdash_\ell y : \{b : Ch()\}}}{\ell : \{a : Ch(\{b : Ch()\})\} \vdash_\ell a(y).\overline{x}@y()}$$

□

Nous aurions pu utiliser comme dans [HR98b] des schémas de noms composés plus généraux tels que $\{a_1, \ldots, a_n\}@\ell$ permettant de lier un groupe de noms de canaux à une seule localité. Cependant, cette construction ne nous est pas nécessaire et nous avons préféré simplifier la syntaxe. Aussi, à la place des types existentiels $\gamma@\psi$ nous n'aurions pu prendre que le cas particulier $\gamma^{@}(= \gamma@\{\})$ – ce qui a d'ailleurs été choisi dans [ABL00]) – mais le terme $a(x@y).(\overline{x}@y() \mid \overline{b}@y())$ de l'exemple 7.2.4 n'aurait plus été typable puisqu'alors le type de la localité n'est plus spécifié dans le type d'un nom composé.

Exemple 7.2.5 Nous reprenons ici l'exemple du RPC pour typer le processus $Client$. Faisons l'hypothèse que la continuation P est typable avec un contexte $\vec{v}\colon \vec{val}, \Gamma$ à la localité ℓ du client. Soit $\psi = \{a\colon Ch(\vec{val}, Ch(\vec{val})^{@})$ et $\phi = \{r\colon Ch(\vec{val})\}$, et $\Delta = k\colon \psi, \ell\colon \phi, \vec{d}\colon \vec{val}, \Gamma$, en supposant ce contexte valide. On a alors :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vec{d}\colon \vec{val}, \ell\colon \{r\colon Ch(\vec{val})\}, \ldots \vdash^{W}_{k} \vec{d}\colon \vec{val}, r@\ell\colon Ch(\vec{val})^{@}}}{k\colon \{a\colon Ch(\vec{val}, Ch(\vec{val})^{@})\}, \ldots \vdash_{k} \overline{a}(\vec{d}, r@\ell)}}{\Delta \vdash_{\ell} \mathsf{go}\, k.\overline{a}(\vec{d}, r@\ell)} \qquad \dfrac{\dfrac{\vdots}{\vec{v}\colon \vec{val}, \Delta \vdash_{\ell} P} \quad \dfrac{}{\vec{v}\colon \vec{val} \vdash_{\ell} \vec{v}\colon \vec{val}}}{\Delta \vdash_{\ell} r(\vec{v}).P}}{\Delta \vdash_{\ell} \mathsf{go}\, k.\overline{a}(\vec{d}, r@\ell) \mid r(\vec{v}).P}}{k\colon \psi, \vec{d}\colon \vec{val}, \Gamma \vdash_{\ell} (\nu r)(\mathsf{go}\, k.\overline{a}(\vec{d}, r@\ell) \mid r(\vec{v}).P)}}{k\colon \psi, \vec{d}\colon \vec{val}, \Gamma \vdash \ell[\![(\nu r)(\mathsf{go}\, k.\overline{a}(\vec{d}, r@\ell) \mid r(\vec{v}).P)]\!]}$$

Par le lemme d'affaiblissement énoncé plus loin, nous savons qu'il existe une dérivation du séquent $\vec{v}\colon \vec{val}, \Delta \vdash_{\ell} P$ puisque, par hypothèse, $\vec{v}\colon \vec{val}, \Gamma \vdash_{\ell} P$ est un séquent valide. □

L'exemple du RPC illustre une situation où une requête est migrée vers un service. Que ce passe-t-il si un processus quitte sa localité courante pour migrer vers une localité où une ressource est disponible ? Par exemple, pour typer $(\nu a)(a(\vec{u}).R \mid \mathsf{go}\, \ell.P \mid Q)$ à la localité courante k, on requiert que P soit typable en ℓ avec un contexte contenant $k\colon \{a\colon Ch(\vec{\tau})\}$ qui est l'unique information concernant a. Ceci signifie que P ne peut utiliser a qu'à la localité k, c'est-à-dire, si $k \neq \ell$, dans un nom composé $a@k$ ou sous une instruction de migration $\mathsf{go}\, k$.

Le résultat principal de ce chapitre est la propriété standard de préservation du typage par la réduction. Pour prouver cette propriété nous avons besoin de montrer quelques résultats intermédiaires, mettant principalement en relation la substitution et le typage. On dénote par $U \leq_{\mathcal{T}} V$ le fait que V a plus de contextes de typage que U. Formellement cette relation est définie par :

$$\begin{aligned} P \leq_{\mathcal{T}} Q &\Leftrightarrow \forall \Gamma\, \forall \ell (\Gamma \vdash_{\ell} P \Rightarrow \Gamma \vdash_{\ell} Q) \\ T \leq_{\mathcal{T}} T' &\Leftrightarrow \forall \Gamma\, \forall \ell\, \forall \gamma (\Gamma \vdash_{\ell} T\colon \gamma \Rightarrow \Gamma \vdash_{\ell} T'\colon \gamma) \\ S \leq_{\mathcal{T}} S' &\Leftrightarrow \forall \Gamma (\Gamma \vdash S \Rightarrow \Gamma \vdash S') \end{aligned}$$

Lemme 7.2.6 *La relation $\leq_{\mathcal{T}}$ est une précongruence, c'est-à-dire un préordre compatible avec les constructions syntaxiques.*

Preuve: La preuve de ce lemme est directe puisque le typage d'un terme ne dépend que des constructions syntaxiques. □

On dénote par $=_{\mathcal{T}}$ la relation d'équivalence associée à $\leq_{\mathcal{T}}$, c'est-à-dire $U =_{\mathcal{T}} V$ si et seulement si $U \leq_{\mathcal{T}} V$ et $V \leq_{\mathcal{T}} U$. Cette relation est évidemment une congruence et elle contient $=_{\alpha}$.

Lemme 7.2.7 *Si $P \rightarrow P'$ alors il existe Q et Q' tels que $Q =_{\alpha} P$ et $Q' =_{\alpha} P'$, et $[T/A]Q \rightarrow [T/A]Q'$.*

Preuve: Simple induction sur la preuve $P \rightarrow P'$. L'α-conversion est utilisée pour que les noms libres T ne soient pas capturés par des lieurs dans P et P'. □

Le lemme suivant concerne le système de types fort pour les noms. Il met en évidence les noms du contexte qui apparaissent dans le vecteur de noms à typer. Il nous est utile pour montrer que l'union de deux contextes, dont l'un est le contexte de typage d'un vecteur de noms, est légal.

Lemme 7.2.8 *Soit* $\Gamma \vdash_\ell \vec{u} : \vec{\tau}$ *et* $\forall u_i \in \{\vec{u}\} . u_i = a@k \Rightarrow k \neq \ell$.

1. *si* $\Gamma = a : \sigma, \Gamma'$ *et* $a \neq \ell$, *alors* $a \in nm(\vec{u})$;
2. *si* $\Gamma = \ell : \psi, \Gamma'$, *alors* $dom(\psi) \subseteq nm(\vec{u})$.

Preuve: Simple induction sur la preuve de $\Gamma \vdash_\ell \vec{u} : \vec{\tau}$. □

Lemme 7.2.9 *Soient* $\Gamma, \Delta, \vec{u}$ *et* ℓ *tels que* $\Delta \vdash_\ell \vec{u} : \vec{\tau}$, $\forall u_i \in \{\vec{u}\}. u_i = a@k \Rightarrow k \neq \ell$, *et* $nm(\Gamma) \cap nm(\vec{u}) = \emptyset$. *Alors* Γ, Δ *est un contexte légal.*

Preuve: On remarque tout d'abord, que si les unions, deux à deux, des contextes Γ, Δ et Θ sont légales, alors il en est de même du contexte Γ, Δ, Θ. On procède par induction sur le contexte Γ en posant $\Gamma = x : \tau, \Theta$ avec $\Theta \neq \Gamma$ – et où x peut désigner un identificateur de processus. Montrons que $x : \tau, \Delta$ est légal. Si $x \neq \ell$ alors, par le lemme 7.2.8 (1), $x \notin dom(\Delta)$, donc $x : \tau, \Delta$ est légal. Si $x = \ell$ et $\Delta = \ell : \psi, \Delta'$ avec $\ell \notin dom(\Delta')$ alors, par le lemme 7.2.8 (2), $\psi \sqcap \tau$ est défini et donc $x : \tau, \Delta$ est légal. Enfin, si $x = \ell$ et $\ell \notin dom(\Delta)$, $x : \tau, \Delta$ est trivialement légal. En appliquant l'hypothèse d'induction Θ, Δ est légal, et puisque $x : \tau, \Delta$ est légal, par la remarque précédente on conclue que Γ, Δ est légal. □

Le lemme suivant montre qu'on peut restreindre un contexte de typage aux noms qui ont des occurrences libres dans le terme à typer. On dénote par $a \notin (\Gamma \vdash_\ell P)$ le fait qu'il n'y a pas d'occurrence de a dans Γ (c'est-à-dire ni dans le domaine ni dans les types assignés par ce contexte), ni dans $fn(P)$, et que $a \neq \ell$; x peut dénoter un identificateur de processus.

Lemme 7.2.10 (Renforcement)

1. *Si* $x : \tau, \Gamma \vdash^W_\ell u : \sigma$ *et* $x \notin nm(u) \cup \{\ell\}$ *alors* $\Gamma \vdash^W_\ell u : \sigma$;
2. *si* $x : \tau, \Gamma \vdash_\ell P$ *est prouvable et* $x \notin fn(P) \cup \{\ell\}$ *alors* $\Gamma \vdash_\ell P$ *est prouvable ;*
3. *si* $x : \tau, \Gamma \vdash_\ell T : \gamma$ *est prouvable et* $x \notin fn(T) \cup \{\ell\}$ *alors* $\Gamma \vdash_\ell T : \gamma$;
4. *si* $x : \tau, \Gamma \vdash S$ *est prouvable et* $x \notin fn(S)$ *alors* $\Gamma \vdash S$ *est prouvable ;*
5. *si* $k : \{a : \gamma\}, \Gamma \vdash^W_\ell u : \sigma$ *et* $a \notin nm(u) \cup nm(\sigma)$ *alors* $\Gamma \vdash^W_\ell u : \sigma$;
6. *si* $k : \{a : \gamma\}, \Gamma \vdash_\ell P$ *est prouvable et* $a \notin (\Gamma \vdash_\ell P)$ *alors* $\Gamma \vdash_\ell P$ *est prouvable ;*
7. *si* $k : \{a : \gamma'\}, \Gamma \vdash_\ell T : \gamma$ *est prouvable et* $a \notin (\Gamma \vdash_\ell T : \gamma)$ *alors* $\Gamma \vdash_\ell T : \gamma$ *est prouvable ;*
8. *si* $\ell : \{a : \gamma\}, \Gamma \vdash S$ *est prouvable et* $a \notin (\Gamma \vdash S)$ *alors* $\Gamma \vdash S$ *est prouvable.*

Preuve: Les points (1) et (5) se montrent directement en examinant les cas possibles pour u et σ. Les autres points se montrent par induction sur l'inférence du séquent en utilisant la convention faite pour les règles s'appliquant aux constructions liantes et le lemme 7.2.8. La preuve de ce lemme ne pose aucune difficulté. □

Le lemme suivant est le « dual » du précédent, il montre que le contexte de typage d'un terme typable peut être affaibli. On dénote par $nm(\Gamma)$ l'ensemble des noms ayant une occurrence dans Γ.

Lemme 7.2.11 (Affaiblissement) *Soient Γ et Δ des contextes légaux tels que Γ, Δ est aussi un contexte légal, alors :*

1. *Si $\Gamma \vdash^W_\ell u : \tau$ est prouvable, alors $\Gamma, \Delta \vdash^W_\ell u : \tau$ est prouvable ;*
2. *si $\Gamma \vdash_\ell P$ est prouvable, alors $\Gamma, \Delta \vdash_\ell P$ est prouvable ;*
3. *si $\Gamma \vdash_\ell T : \gamma$ est prouvable, alors $\Delta, \Gamma \vdash_\ell T : \gamma$ est prouvable ;*
4. *si $\Gamma \vdash S$ est prouvable, alors $\Gamma, \Delta \vdash S$ est prouvable.*

Preuve: La preuve du premier point est directe. Les autres points se montrent par induction sur la preuve du séquent. Examinons le cas où $P = a(\vec{u}).Q$. Supposons $nm(\vec{u}) \cap nm(\Delta) \neq \emptyset$, et soient $\vec{v}$ et R tels que $nm(\vec{v}) \cap (nm(\Delta) \cup nm(\Gamma)) = \emptyset$ et $a(\vec{u}).Q =_\alpha a(\vec{v}).R$. Le séquent $\Gamma \vdash_\ell a(\vec{v}).R$ est donc également prouvable avec $\Gamma = \ell : \{a : Ch(\vec{\tau})\}, \Gamma'$ et $\Gamma', \Delta' \vdash_\ell R$ avec $\Delta' \vdash_\ell \vec{v} : \vec{\tau}$. Par le lemme 7.2.9, le contexte Γ, Δ, Δ' étant légal, par hypothèse d'induction, $\Gamma', \Delta', \Delta \vdash_\ell R$ est prouvable et on conclue par application de la règle de typage pour la réception puis celle pour l'α-conversion. □

Dans le lemme suivant on utilise la notion de *hauteur* de preuve d'un séquent. Une preuve pouvant être perçue comme un arbre, un axiome a la hauteur 0, et chaque règle incrémente de 1 le maximum des hauteurs de ses prémisses. On dénote par $\mathsf{S}(\Gamma \vdash_\ell P)$ – et de manière similaire les autres types de séquents – le séquent obtenu par l'application de la substitution S à P, au contexte si le résultat est un contexte légal, et à la localité ℓ ; c'est-à-dire $\mathsf{S}(\Gamma \vdash_\ell P) = \mathsf{S}(\Gamma) \vdash_{\mathsf{S}(\ell)} [\mathsf{S}]P$ où $\mathsf{S}(\Gamma)$ est défini de manière évidente. Pour les séquents $\Gamma \vdash^W_\ell \vec{u} : \vec{\tau}$ et $\Gamma \vdash_\ell \vec{u} : \vec{\tau}$ il faut également appliquer la substitution aux types $\vec{\tau}$ sous réserve que le résultat soit également valide. L'application d'une substitution S à un type ψ qui est non injective sur $dom(\mathsf{S})$ induit une forme de contraction implicite et n'est définie que si le type résultant est un type valide. En d'autres termes, si, par exemple, $\psi = \{a : \gamma, b : \delta\}$ et $\mathsf{S}(a) = \mathsf{S}(b)$ alors on doit avoir $\mathsf{S}(\gamma) = \mathsf{S}(\delta)$, et on obtient $\mathsf{S}(\psi) = \{\mathsf{S}(a) : \mathsf{S}(\gamma)\}$.

Lemme 7.2.12 (Substitution)

1. *Si $\Gamma \vdash_\ell \vec{u} : \vec{\tau}$ (resp. $\Gamma \vdash^W_\ell \vec{u} : \vec{\tau}$) est prouvable avec une preuve de hauteur h, alors $\mathsf{S}(\Gamma \vdash_\ell \vec{u} : \vec{\tau})$ (resp. $\mathsf{S}(\Gamma \vdash^W_\ell \vec{u} : \vec{\tau})$) l'est aussi pour toute substitution S telle que $\mathsf{S}(\Gamma)$ est un contexte légal ;*
2. *Si $\Gamma \vdash_\ell P$ (resp. $\Gamma \vdash_\ell T : Ch(\vec{\tau})$) est prouvable avec une preuve de hauteur h, alors $\mathsf{S}(\Gamma \vdash_\ell P)$ (resp. $\mathsf{S}(\Gamma \vdash_\ell T : Ch(\vec{\tau}))$) l'est aussi pour toute substitution S telle que $[\mathsf{S}]P$ (resp. $[\mathsf{S}]T$) est défini et $\mathsf{S}(\Gamma)$ est un contexte légal (et si $\mathsf{S}(\vec{\tau})$ est un vecteur de types valides) ;*
3. *Si $\Gamma \vdash_\ell S$ est prouvable avec une hauteur de preuve h, alors $\mathsf{S}(\Gamma \vdash_\ell S)$ l'est aussi, pour toute substitution S telle que $[\mathsf{S}]S$ est défini et $\mathsf{S}(\Gamma)$ est un contexte légal ;*

4. *Si* $\Delta \vdash_\ell T : Ch(\vec{\tau})$ *et* $A : Ch(\vec{\tau}), \Gamma \vdash_\ell P$ *(resp.* $A : Ch(\vec{\tau}), \Gamma \vdash_\ell T' : \sigma$*, resp.* $A : Ch(\vec{\tau}), \Gamma \vdash S$*) sont prouvable, et* A *n'a pas d'occurrence dans* Γ*, alors* $\Delta, \Gamma \vdash_\ell [T/A]P$ *(resp.* $\Delta, \Gamma \vdash_\ell [T/A]T' : \sigma$*, resp.* $\Delta, \Gamma \vdash [T/A]S$*) est prouvable, pourvu qu'aucun nom libre de* T *ne soit lié dans* P *(resp.* T', S*) et que* Δ, Γ *soit un contexte légal.*

Preuve:

1. Simple induction sur la hauteur h en remarquant que si $\mathsf{S}(\Gamma, \Delta)$ est légal, alors $\mathsf{S}(\Gamma)$ et $\mathsf{S}(\Delta)$ aussi et, de plus, $\mathsf{S}(\Gamma, \Delta) = \mathsf{S}(\Gamma), \mathsf{S}(\Delta)$.

2. Par induction sur h et par analyse de cas sur la dernière règle utilisée pour inférer le séquent. Si $P = a(\vec{u}).Q$ et $\Gamma = \ell : \{a : Ch(\vec{\tau})\}, \Delta$ avec $\ell : \{a : Ch(\vec{\tau})\}, \Theta, \Delta \vdash_\ell Q$ où $\Theta \vdash_\ell \vec{u} : \vec{\tau}$, alors, puisque par convention $im(\mathsf{S}) \cap nm(\vec{u}) = \emptyset$, le séquent $\mathsf{S}'(\ell : \{a : Ch(\vec{\tau})\}, \Theta, \Delta \vdash_\ell Q)$ est prouvable par hypothèse d'induction, où $\mathsf{S}' = \mathsf{S} \restriction dom(\mathsf{S}) - nm(\vec{u})$. D'autre part, $\mathsf{S}'(\Theta) \vdash_k \vec{u} : \mathsf{S}'(\vec{\tau})$ où $k = \mathsf{S}'(\ell)$, et on peut donc utiliser la règle de typage pour la réception pour conclure. Le cas où $P = (\mathsf{rec}\ A(\vec{u}).Q)$ est très similaire. Les autres cas sont très faciles.

3. La preuve est identique à celle du point précédent.

4. Par induction sur l'inférence du séquent $A : Ch(\vec{\tau}), \Gamma \vdash_\ell P$, de $A : Ch(\vec{\tau}), \Gamma \vdash_\ell T' : \sigma$, ou de $A : Ch(\vec{\tau}), \Gamma \vdash S$. Supposons que $P = a(\vec{u}).Q$ et $nm(\Delta) \cap nm(\vec{u}) \neq \emptyset$. Soient $\vec{v}$ et R tels que $nm(\vec{v}) \cap (nm(\Gamma) \cup nm(\Delta) \cup fn(T)) = \emptyset$ et $a(\vec{u}).Q =_\alpha a(\vec{v}).R$. Alors, $\Gamma = \ell\{a : Ch(\vec{\sigma})\}, \Gamma'$ et $\Gamma, \Theta \vdash_\ell R$ avec $\Theta \vdash_\ell \vec{v} : \vec{\sigma}$. D'autre part, $[T/A]P =_\alpha a(\vec{v})[T/A]R$; et d'après le lemme 7.2.9, Θ, Γ, Δ est légal, d'où, par hypothèse d'induction, $\Theta, \Gamma, \Delta \vdash_\ell [T/A]R$ est prouvable. On conclue par l'application de la règle de typage pour la réception et la règle d'α-conversion.
   Dans le cas où on a l'axiome $A : Ch(\vec{\tau}), \Gamma \vdash_\ell A : Ch(\vec{\tau})$, par le lemme d'affaiblissement 7.2.11, de $\Delta \vdash_\ell T : Ch(\vec{\tau})$ prouvable, on déduit que $\Gamma, \Delta \vdash_\ell T : Ch(\vec{\tau})$ est prouvable.
   Dans les cas où A n'a pas d'occurrence dans P (resp. T' et S), on a $[T/A]P = P$ (resp. $[T/A]T' = T'$ et $[T/A]S = S$). Par le lemme de renforcement 7.2.10, $\Gamma \vdash_\ell P$ (resp. $\Gamma \vdash_\ell T' : \sigma$ et $\Gamma \vdash S$) est prouvable et par le lemme d'affaiblissement $\Gamma, \Delta \vdash_\ell P$ (resp. $\Gamma, \Delta \vdash_\ell T' : \sigma$ et $\Gamma, \Delta \vdash S$) l'est aussi. Les autres cas se traitent facilement ou de manière similaire à la réception.

□

On montre à présent que le typage est invariant par équivalence structurelle.

Proposition 7.2.13 *La relation* $=_\mathcal{T}$ *contient la relation d'équivalence structurelle* $\equiv$.

Preuve: Nous savons déjà que $=_\mathcal{T}$ contient l'α-conversion, et que, d'après le lemme 7.2.6, c'est une congruence. Il est donc suffisant de montrer que pour tout axiome $U \equiv V$ on a $U \leq_\mathcal{T} V$ et $V \leq_\mathcal{T} U$. La preuve est triviale pour les axiomes de neutralité, d'associativité et commutativité de la composition parallèle et de routage.

Soit $\Gamma \vdash_\ell ((\nu w)P \mid Q)$ avec $subj(w) \notin fn(Q)$. On peut faire l'hypothèse que tous les noms liés du terme sont distincts et n'ont pas d'occurrence dans Γ, de sorte que la preuve de ce séquent n'utilise

pas la règle de typage pour l'α-conversion. Cette preuve a donc la forme suivante :

$$\dfrac{\dfrac{\dfrac{\vdots}{x:\tau,\Gamma\vdash_\ell P}}{\Gamma\vdash_\ell (\nu w)P} \qquad \dfrac{\vdots}{\Gamma\vdash_\ell Q}}{\Gamma\vdash_\ell ((\nu w)P \mid Q)}$$

où $x:\tau$ est, respectivement, $\ell:\{a:\gamma\}$ si $w = a \in \mathcal{N}_{chan}$, $k:\{a:\gamma\}$ si $w = a@k$, $k:\psi$ si $w = k:\psi$, ou $a:$ ***val*** si $w = a \in \mathcal{N}_{val}$. Puisque ***subj***$(w) \notin$ ***nm***(Γ), le contexte $x:\tau,\Gamma$ est bien formé, et donc, par le lemme d'affaiblissement 7.2.11, le séquent $x:\tau,\Gamma\vdash_\ell Q$ est prouvable, et finalement, par application de règle de typage pour la composition parallèle, on prouve $\Gamma\vdash_\ell (\nu w)(P \mid Q)$. Inversement, une preuve de $\Gamma\vdash(\nu w)(P\mid Q)$ a la forme :

$$\dfrac{\dfrac{\dfrac{\vdots}{x:\tau,\Gamma\vdash_\ell P} \qquad \dfrac{\vdots}{x:\tau,\Gamma\vdash_\ell Q}}{x:\tau,\Gamma\vdash_\ell P\mid Q}}{\Gamma\vdash(\nu w)(P\mid Q)}$$

Si $x =$ ***subj***(w), par le lemme de renforcement 7.2.10 (2), $\Gamma\vdash_\ell Q$ est prouvable. Si $w = a@k$ ou $w = a \in \mathcal{N}_{chan}$, par le point (6) du même lemme, $\Gamma\vdash_\ell Q$ est encore prouvable. D'autre part, $\Gamma\vdash_\ell (\nu w)P$, et donc $\Gamma\vdash_\ell ((\nu w)P \mid Q)$. La preuve du cas de $((\nu w)S \mid S') \equiv (\nu w)(S \mid S')$ est identique.

Pour $\ell[\![(\nu w)P]\!] \equiv (\nu w@\ell)\ell[\![P]\!]$, on fait la même hypothèse que précédemment et supposons que $\Gamma\vdash\ell[\![(\nu w)P]\!]$ est prouvable. La preuve de ce séquent a donc la forme suivante :

$$\dfrac{\dfrac{\dfrac{\vdots}{x:\tau,\Gamma\vdash_\ell P}}{\Gamma\vdash_\ell (\nu w)P}}{\Gamma\vdash\ell[\![(\nu w)P]\!]}$$

où $x:\tau$ est, respectivement, $\ell:\{a:\gamma\}$ si $w = a \in \mathcal{N}_{chan}$, ou $k:\{a:\gamma\}$ si $w = a@k$, ou $a:$ ***val*** si $w = a \in \mathcal{N}_{val}$, ou encore $k:\psi$ si $w = k:\psi$. Dans tous les cas l'inférence suivante est valide :

$$\dfrac{\dfrac{\dfrac{\vdots}{x:\tau,\Gamma\vdash_\ell P}}{x:\tau,\Gamma\vdash\ell[\![P]\!]}}{\Gamma\vdash(\nu w@\ell)\ell[\![P]\!]}$$

La preuve du cas symétrique est similaire. □

On en arrive enfin à l'énoncé du résultat principal de ce chapitre : la préservation du typage par la réduction.

Théorème 7.2.14 (Préservation du typage) $U \to V \Rightarrow U \leq_{\mathcal{T}} V$

Preuve: Par induction sur la preuve de $U \to V$. Puisque $\leq_{\mathcal{T}}$ est une précongruence contenant l'équivalence structurelle, il suffit de considérer les axiomes de réduction. Les cas concernant les lois de migrations et les branchements sont triviaux.

Pour le cas de la communication $(\overline{a}(\vec{v}) \mid a(\vec{u}).P) \to [\vec{v}/\vec{u}]P$, soit un contexte Γ et une localité ℓ tels que $\Gamma \vdash_\ell (\overline{a}(\vec{v}) \mid a(\vec{u}).P)$ est prouvable. La preuve de ce séquent a la forme suivante :

$$\dfrac{\dfrac{\dfrac{\vdots}{\Theta \vdash_\ell^W \vec{v} : \vec{\tau}}}{\Gamma \vdash_\ell \overline{a}(\vec{v})} \qquad \dfrac{\dfrac{\vdots}{\Theta, \Delta \vdash_\ell P} \qquad \dfrac{\vdots}{\Delta \vdash_\ell \vec{u} : \vec{\tau}}}{\Gamma \vdash_\ell a(\vec{u}).P}}{\Gamma \vdash_\ell \overline{a}(\vec{v}) \mid a(\vec{u}).P}$$

où on fait à nouveau l'hypothèse que les noms liés ont été renommés de sorte qu'on n'ait pas à utiliser de règles d'α-conversion, et $\Gamma = \ell : \{a : \mathit{Ch}(\vec{\tau})\}, \Theta$. Les noms de $\vec{u}$ n'ayant pas d'occurrence dans $\vec{\tau}$, le contexte $[\vec{v}/\vec{u}]\Delta$ est légal et, par le lemme de substitution 7.2.12 (1) on a $[\vec{v}/\vec{u}]\Delta \vdash_\ell \vec{v} : \vec{\tau}$. Par la remarque 7.2.3, $\Theta = (\Theta_1, \Theta_2)$ avec $\Theta_1 \vdash_\ell \vec{v} : \vec{\tau}$, et par la remarque 7.2.2, $\Theta_1 = [\vec{v}/\vec{u}]\Delta$. Puisque $\mathit{nm}(\vec{u}) \cap \mathit{nm}(\Gamma) = \emptyset$, on a $[\vec{v}/\vec{u}](\Theta, \Delta) = (\Theta, [\vec{v}/\vec{u}]\Delta) = \Theta$ qui est un contexte légal. Par conséquent, on peut appliquer le lemme de substitution 7.2.12 (2), et $\Theta \vdash_\ell [\vec{v}/\vec{u}]P$ est prouvable.

Considérons maintenant le cas de la règle de dépliage de la récursion

$$(\mathsf{rec}\ A(\vec{u}).P)(\vec{v}) \to [(\mathsf{rec}\ A(\vec{u}).P)/A][\vec{v}/\vec{u}]P$$

et supposons que le séquent $\Gamma \vdash_\ell (\mathsf{rec}\ A(\vec{u}).P)(\vec{v})$ est prouvable. La preuve de ce séquent a la forme suivante :

$$\dfrac{\dfrac{\dfrac{\vdots}{A : \mathit{Ch}(\vec{\tau}), \Gamma, \Delta \vdash_\ell P} \qquad \dfrac{\vdots}{\Delta \vdash_\ell \vec{u} : \vec{\tau}}}{\Gamma \vdash_\ell (\mathsf{rec}\ A(\vec{u}).P) : \mathit{Ch}(\vec{\tau})} \qquad \dfrac{\vdots}{\Gamma \vdash_\ell^W \vec{v} : \vec{\tau}}}{\Gamma \vdash_\ell (\mathsf{rec}\ A(\vec{u}).P)(\vec{v})}$$

Comme dans le cas précédent, le séquent $A : \mathit{Ch}(\vec{\tau}), \Gamma \vdash_\ell [\vec{v}/\vec{u}]P$ est prouvable. Par α-conversion, on peut faire l'hypothèse qu'aucun nom libre dans $(\mathsf{rec}\ A(\vec{u}).P)$ n'est lié dans $[\vec{v}/\vec{u}]P$, d'où, par le lemme de substitution 7.2.12 (4), on conclue que $\Gamma \vdash_\ell [(\mathsf{rec}\ A(\vec{u}).P)/A][\vec{v}/\vec{u}]P$ est prouvable. □

CHAPITRE 8

Livrabilité des messages

LE SYSTÈME DE TYPE ÉLABORÉ DANS LE CHAPITRE PRÉCÉDENT ne garantit évidemment pas la propriété de sûreté que nous recherchons, à savoir la réceptivité. En effet, nous pouvons typer des processus envoyant des messages qui ne seront jamais reçus tel que $(\nu a)\overline{a}()$. Comme dans la partie précédente, nous construisons dans ce chapitre un système de « bonne formation » des termes qui, conjointement au système de type, garanti la propriété de réceptivité. Dans la première section nous présentons ce système de bonne formation, et dans la suivante nous prouvons les propriétés de réceptivité et de livrabilité des messages.

8.1 Le système de bonne formation

Nous reprenons ici le système de la section 2.3 et l'adaptons afin de prendre en compte les constructions liées à la distribution. Rappelons que l'idée est toujours d'interdire les imbrications de réceptions sur des noms différents et de vérifier que les récepteurs sont persistants et immédiatement accessibles. Tout récepteur doit donc être « encapsulé » dans un processus récursif, lui permettant ainsi de se « réincarner », éventuellement dans un état différent, dès qu'il a reçu un message. De plus, on impose, comme dans π_1 [Ama97], l'unicité des récepteurs pour chaque nom, c'est-à-dire que deux composants en parallèles d'un processus ne sont pas habilités à recevoir sur un même nom. Enfin, il faut garantir que pour tout nom de canal restreint, un récepteur est défini pour ce nom.

Les processus bien formés sont ceux pour lesquels un jugement $I \Vdash U$ est prouvable ; on dit alors que « U *est bien formé avec l'interface* I ». L'**interface** d'un terme est maintenant un ensemble de noms qui peuvent être composés et pour lesquels le terme offre un récepteur.

Nous décrivons à présent les règles de bonne formation, donnée dans la figure 8.1, sans revenir sur celles déjà présentées dans le chapitre 3. Comme d'habitude, l'ensemble $I = \{u_1, \ldots, u_n\}$ est présenté comme la liste $u_1, \ldots, u_n$ de ses éléments, est l'union d'interfaces I et J est écrite I, J. Dans les règles concernant la restriction $(\nu w)U$ on fait l'hypothèse implicite que le sujet de la restriction n'ap-

$$\frac{}{\Vdash \mathbf{0}} \qquad \frac{}{\Vdash \overline{a}(\vec{u})} \qquad \frac{a \Vdash P}{a \Vdash a(\vec{u}).P} \qquad \frac{I \Vdash U \quad , \quad J \Vdash V}{I, J \Vdash (U \mid V)}\, I \cap J = \emptyset$$

$$\frac{u, I \Vdash U}{I \Vdash (\nu u)U}\, u \notin \mathcal{N}_{val} \qquad \frac{I \Vdash U}{I \Vdash (\nu a)U}\, a \in \mathcal{N}_{val} \qquad \frac{\mathit{dom}(\psi)@\ell, I \Vdash U}{I \Vdash (\nu \ell : \psi)U}$$

$$\frac{I \Vdash P \quad , \quad I \Vdash Q}{I \Vdash [a = b]P, Q} \qquad \frac{}{\Vdash A} \qquad \frac{a \Vdash P}{\Vdash (\mathsf{rec}\ A(a, \vec{u}).P)} \qquad \frac{\Vdash T}{a \Vdash T(a, \vec{u})}$$

$$\frac{I \Vdash P}{I@\ell \Vdash \mathsf{go}\,\ell.P}\, \{a \mid a \in I \text{ et } a@\ell \in I\} = \emptyset \qquad \frac{I \Vdash P}{I@\ell \Vdash \ell[\![P]\!]}\, \{a \mid a \in I \text{ et } a@\ell \in I\} = \emptyset$$

FIG. 8.1: Système de bonne formation pour $\mathrm{D}\pi_1^r$

paraît pas dans le contexte résultant, c'est-à-dire $\mathit{subj}(w) \notin \mathit{nm}(I)$. Dans le cas où la restriction porte sur une localité $w = \ell : \psi$ (et $\ell \notin \mathit{nm}(I)$), si $\mathit{dom}(\psi) = \{a_1, \ldots, a_n\}$, alors on requiert qu'un récepteur existe en ℓ pour chaque nom $a_1, \ldots, a_n$ (rappelons qu'alors $\mathit{dom}(\psi)@\ell = \{a_1@\ell, \ldots, a_n@\ell\}$). Le type ψ d'une localité restreinte ℓ représente donc l'ensemble des ressources disponibles pour les agents à la localité ℓ. On rappelle que dans les termes, tout identificateur de processus $A(\vec{u})$ est gardé par une réception.

La contrainte donnée pour la composition parallèle pourrait être relâchée et nous pourrions toujours montrer la livrabilité des messages mais pas nécessairement à un récepteur unique. On requiert néanmoins l'unicité des récepteurs car elle semble être une hypothèse importante tant du point de vue de la spécification que de l'implantation. Cette propriété est également garantie par les contraintes imposées pour $I@\ell \Vdash \mathsf{go}\,\ell.P$ et $I@\ell \Vdash \ell[\![P]\!]$. Pour la comprendre il faut remarquer que les noms simples présents dans l'interface d'un processus P sont ceux que peut directement utiliser P à sa localité courante. Cette dernière n'étant pas explicitement spécifiée dans le jugement $I \Vdash P$, elle peut en particulier être ℓ, et donc l'application de l'opérateur $\_@\ell$ peut entraîner la duplication d'un nom de canal et donc d'un récepteur.

Exemple 8.1.1 Pour que le processus *Client* du protocole RPC soit bien formé, $r(\vec{v}).P$ doit être bien formé avec l'interface $\{r\}$. Ce canal n'étant utilisé qu'une seule fois pour recevoir un résultat, on utilise comme dans le chapitre 3, la construction dérivée de réception linéaire :

$$\mathit{Client} \stackrel{\mathsf{def}}{=} (\nu r)(\overline{a}@k(\vec{d}, r@\ell) \mid r(\vec{v}) : P)$$

où le jugement $r \vdash r(\vec{v}) : P$ est valide, si $\Vdash P$. La preuve de bonne formation de *Client* est donc :

$$\dfrac{\dfrac{}{\Vdash \overline{a}@k(\vec{d}, r@\ell)} \qquad \dfrac{\dfrac{\vdots}{\Vdash P}}{r \Vdash r(\vec{v}) : P}}{\dfrac{r \Vdash \overline{a}@\ell(\vec{d}, r@\ell) \mid r(\vec{v}) : P}{\Vdash (\nu r)(\overline{a}@k(\vec{d}, r@\ell) \mid r(\vec{v}) : P)}}$$

De même, il est facile de montrer que le processus *Service* est bien formé à la localité k avec l'interface $\{a@k\}$. □

Dans le chapitre précédent nous avons vu que le typage impose des contraintes sur l'utilisation des canaux au regard de la localité où ils peuvent être utilisés. Le système de bonne formation impose également ce type de contraintes en forçant les interfaces à être, en quelque sorte, persistantes. Par exemple, un terme tel que $a(\vec{u}).(P \mid \mathsf{go}\,\ell.a(\vec{v}).Q)$ n'est pas acceptable, car le sous-terme $(P \mid \mathsf{go}\,\ell.a(\vec{v}).Q)$ devrait avoir l'interface $\{a\}$, alors qu'il n'est acceptable qu'avec une interface contenant $a@\ell$. Ceci reste vrai quand le terme initial est placé à la localité courante ℓ – et dans ce cas la migration n'a aucun effet – car la localité courante dans le système de bonne formation n'est pas « enregistrée » dans le jugement. Cette remarque signifie que les serveurs ne sont pas très facilement déplaçables, ce qui semble naturel étant donné la propriété que nous cherchons à obtenir. En effet, on ne désire pas qu'un récepteur disparaisse en allant dans une autre localité après avoir reçu un message. Nous verrons dans le chapitre suivant comment gérer des ressources migrantes.

8.2 Livrabilité des messages

On prouve maintenant notre résultat principal, c'est-à-dire la *livrabilité de message*. À cette fin, nous avons préalablement besoin de montrer une propriété de préservation de la bonne formation par la réduction, et une propriété de réceptivité. La remarque suivante est évidente :

Remarque 8.2.1 Si $I \Vdash U$ est prouvable, alors $nm(I) \subseteq fn(U)$.

Il faut à nouveau établir une propriété reliant bonne formation et substitution. Dans le lemme suivant on dénote par $\mathsf{S}(I \Vdash U)$, le jugement $\mathsf{S}(I) \Vdash \mathsf{S}(U)$.

Lemme 8.2.2 *Soit $I \Vdash U$ et S une substitution telle que $[\mathsf{S}]U$ est défini et S est injective sur $nm(I) \cup cr(U)$. Alors, $\mathsf{S}(I \Vdash U)$, et si $I \Vdash [\mathsf{S}]U$ alors $\mathsf{S}^{-1}(I) \Vdash U$. Si $I \Vdash P$ et $\Vdash T$, et $fn(T) \cap bn(P) = \emptyset$ alors $I \Vdash [T/A]P$.*

Preuve: Par induction sur l'inférence de $I \Vdash U$. Nous ne considérons que les cas les plus significatifs, c'est-à-dire ceux où U est une réception, ou une restriction de localité ou un processus migrant. Le cas où U est l'identificateur de processus A est trivial.
• Si $U = a(\vec{u}).Q$ alors $I = \{a\}$ et, si $\mathsf{S}' = \mathsf{S} \restriction dom(\mathsf{S}) - nm(\vec{u})$, $[\mathsf{S}]U = \mathsf{S}(a)(\vec{u}).[\mathsf{S}']Q$. Comme $a \notin \{\vec{u}\}$ et $\{a\} \Vdash Q$, par hypothèse d'induction on a $\{\mathsf{S}'(a)\} \Vdash [\mathsf{S}']Q$ et, par la règle pour la réception,

$\{\mathsf{S}(a)\} \Vdash [\mathsf{S}]U$. Ensuite, si $I \Vdash \mathsf{S}(a)(\vec{u}).[\mathsf{S}']Q$, alors nécessairement $I = \{\mathsf{S}(a)\}$ et $\{\mathsf{S}(a)\} \Vdash [\mathsf{S}']Q$; comme $a \notin \mathit{nm}(\vec{u})$, on a $\mathsf{S}(a) = \mathsf{S}'(a)$ et, par hypothèse d'induction, $\{a\} \Vdash Q$ et donc $a \Vdash a(\vec{u}).Q$. Enfin, si $\Vdash T$, par définition $[T/A]U = a(\vec{u}).[T/A]Q$, et puisque $\{a\} \Vdash Q$, par hypothèse d'induction $\{a\} \Vdash [T/A]Q$; et par application de la règle pour la réception $\{a\} \Vdash [T/A]U$.
• Si $U = (\nu\ell : \psi)V$, alors $[\mathsf{S}]U = (\nu\ell : \mathsf{S}(\psi))[\mathsf{S}']V$ où $\mathsf{S}' = \mathsf{S} \restriction \mathit{dom}(\mathsf{S}) - \{\ell\}$ et $\mathit{dom}(\psi)@\ell, I \Vdash V$. S' étant injective sur $\mathit{nm}(I) \cup \mathit{dom}(\psi) \cup \{\ell\}$, par hypothèse d'induction, on a $\mathsf{S}'(I, \mathit{dom}(\psi)@\ell) \Vdash [\mathsf{S}']V$, et comme $\ell \notin \mathit{dom}(\mathsf{S}') \cup \mathit{nm}(\psi) \cup \mathit{nm}(I)$, on a $\mathsf{S}(I), \mathit{dom}(\mathsf{S}(\psi))@\ell \Vdash [\mathsf{S}']V$. Finalement, on peut appliquer de la règle pour la restriction de localités et $\mathsf{S}(I) \Vdash [\mathsf{S}]U$. Ensuite, si $I \Vdash (\nu\ell : \mathsf{S}(\psi))[\mathsf{S}']V$, alors $I, \mathit{dom}(\mathsf{S}(\psi))@\ell \Vdash [\mathsf{S}']V$, et par hypothèse d'induction, $\mathsf{S}'^{-1}(I), \mathsf{S}'^{-1}(\mathit{dom}(\mathsf{S}(\psi))@\ell) \Vdash V$. Étant données les hypothèses, $\mathsf{S}(\psi) = \mathsf{S}'(\psi)$ et $\ell \notin \mathit{nm}(I)$, donc $\mathsf{S}^{-1}(I), \mathit{dom}(\psi)@\ell \Vdash V$ et donc $\mathsf{S}^{-1}(I) \Vdash U$. Enfin, la substitution d'un processus paramétré se montre comme dans le cas précédent.
• Si $U = \mathsf{go}\,\ell.P$ et $I@\ell \Vdash \mathsf{go}\,\ell.P$ alors $I \Vdash P$ avec $\{a \mid a \in I \text{ et } a@\ell \in I\} = \emptyset$. Par hypothèse d'induction, $\mathsf{S}(I) \Vdash [\mathsf{S}]P$ et S étant injective sur $\mathit{nm}(I@\ell)$, on a $\{a \mid a \in \mathsf{S}(I) \text{ et } a@\mathsf{S}(\ell) \in \mathsf{S}(I)\} = \emptyset$, d'où $\mathsf{S}(I)@\mathsf{S}(\ell) \Vdash [\mathsf{S}]U$. Les preuves de $I \Vdash [\mathsf{S}]U \Rightarrow \mathsf{S}^{-1}(I) \Vdash U$ et de $\Vdash [T/A]P$ sont aussi simples. Le cas où $U = \ell[\![P]\!]$ se traite de manière identique. □

Nous définissons à présent la relation $\leq_{\mathcal{R}}$ sur les termes bien formés qui, informellement, signifie que si $U \leq_{\mathcal{R}} V$ alors toute interface de U est aussi une interface de V. Plus formellement :

$$U \leq_{\mathcal{R}} V \Leftrightarrow \forall I (I \Vdash U \Rightarrow I \Vdash V)$$

Clairement, cette relation est une précongruence. Soit $=_{\mathcal{R}}$ la congruence associée.

Corollaire 8.2.3 *La relation $=_{\mathcal{R}}$ contient la relation $=_\alpha$ d'α-conversion.*

Preuve: La preuve, par induction sur la définition de $=_\alpha$, est très facile. □

Lemme 8.2.4 *La relation $=_{\mathcal{R}}$ contient la relation d'équivalence structurelle $\equiv$.*

Preuve: Puisque $\leq_{\mathcal{R}}$ est une précongruence qui contient l'α-conversion, il suffit de montrer que pour chaque axiome $U \equiv V$ on a $U \leq_{\mathcal{R}} V$ et $V \leq_{\mathcal{R}} U$. Les cas concernant les axiomes de neutralité, d'associativité et de commutativité de la composition parallèle et de routage sont triviaux.
• Soit $I \Vdash ((\nu w)U \mid V)$ avec $\mathit{subj}(w) \notin \mathit{fn}(V)$. Par le corollaire 8.2.3, on peut faire l'hypothèse que $\mathit{subj}(w)$ n'a pas d'occurrence dans I. Examinons le cas où $w = u \notin \mathcal{N}_{val}$. On a

$$\dfrac{\dfrac{\dfrac{\vdots}{u, I_1 \Vdash U}}{I_1 \Vdash (\nu u)U} \qquad \dfrac{\vdots}{I_2 \Vdash V}}{I_1, I_2 \Vdash ((\nu u)U \mid V)}\, I_1 \cap I_2 = \emptyset$$

où $I_1, I_2 = I$. Par la remarque 8.2.1 on a $u \notin I_2$, donc $I \Vdash (\nu w)(U \mid V)$. Les autres cas sont similaires, ainsi que le cas symétrique où $I \Vdash (\nu w)(U \mid V)$.

• Si $I \Vdash \ell[\![(\nu w)P]\!]$ et $w = u \notin \mathcal{N}_{val}$, alors on a

$$\dfrac{\dfrac{\dfrac{\vdots}{u, J \Vdash P}}{J \Vdash (\nu u)P}}{I \Vdash \ell[\![(\nu u)P]\!]}\, \{a \mid a \in J \text{ et } a@\ell \in J\} = \emptyset$$

où *subj*$(u) \notin$ *nm*(J) et $I = J@\ell$. Alors, on a aussi $\{a \mid a \in J$ et $a@\ell \in u, J\} = \emptyset$ et par conséquent, l'inférence suivante est valide :

$$\dfrac{\dfrac{\dfrac{\vdots}{u, J \Vdash P}}{u@\ell, J@\ell \Vdash \ell[\![P]\!]}}{I \Vdash (\nu u@\ell)\ell[\![P]\!]}$$

Dans le cas où $w = k\!:\!\psi$, on a *dom*$(\psi)@k, J \Vdash P$ où $\{a \mid a \in J$ et $a@\ell \in J\} = \emptyset$ et $k \notin$ *nm*(J). Donc, $\{a \mid a \in J$ et $a@\ell \in J,$ *dom*$(\psi)@k\} = \emptyset$ et on peut construire la même inférence que précédemment en rappelant que $w@\ell = w$ et *dom*$(\psi)@k@\ell =$ *dom*$(\psi)@k$. Le cas où $w \in \mathcal{N}_{val}$ est facile, ainsi que le cas symétrique où l'hypothèse est $I \Vdash (\nu u@\ell)\ell[\![P]\!]$. □

Proposition 8.2.5 (Préservation de la bonne formation) $U \to V \Rightarrow U \leq_{\mathcal{R}} V$

Preuve: Par induction sur la définition de $U \to V$. Encore une fois, il suffit de considérer les axiomes de la réduction. Les cas concernant la loi de migration et le branchement conditionnel sont triviaux.
• Pour la loi de communication, posons $U = (\overline{a}(\vec{v}) \mid a(\vec{u}).P)$ et $V = [\vec{v}/\vec{u}]P$, on a

$$\dfrac{\dfrac{}{\Vdash \overline{a}(\vec{v})} \qquad \dfrac{\dfrac{\vdots}{a \Vdash P}}{a \Vdash a(\vec{u}).P}}{a \Vdash (\overline{a}(\vec{v}) \mid a(\vec{u}).P)}$$

avec $a \notin$ *nm*$(\vec{u})$. Par convention, *nm*$(\vec{u}) \cap$ *cr*$(P) = \emptyset$, donc par la remarque 7.2.1, $[\vec{v}/\vec{u}]P$ est défini et par le lemme 8.2.2, on a aussi $a \Vdash [\vec{v}/\vec{u}]P$.
• Dans le cas du dépliage, posons $U = (\mathsf{rec}\ A(\vec{u}).P)(\vec{v})$ et $V = [\mathsf{rec}\ A(\vec{u}).P/_A][\vec{v}/\vec{u}]P$. On a donc l'inférence :

$$\dfrac{\dfrac{\dfrac{\vdots}{b \Vdash P}}{\Vdash (\mathsf{rec}\ A(b, \vec{u'}).P)}}{a \Vdash (\mathsf{rec}\ A(b, \vec{u'}).P)(a, \vec{v'})}$$

où $\vec{u} = b, \vec{u'}$ et $\vec{v} = a, \vec{v'}$, et on conclue comme dans le cas précédent. □

$$\frac{}{a(\vec{u}).P \downarrow a} \qquad \frac{U \downarrow u}{(U \mid V) \downarrow u} \qquad \frac{V \downarrow u}{(U \mid V) \downarrow u} \qquad \frac{U \downarrow u}{(\nu w)U \downarrow u}\ \mathit{nm}(u) \cap \mathit{subj}(w) = \emptyset$$

$$\frac{P \not\downarrow u}{\mathsf{go}\,\ell.P \downarrow u@\ell} \qquad \frac{P \downarrow u}{\ell[\![P]\!] \downarrow u@\ell} \qquad \frac{U \twoheadrightarrow^* V\ ,\ V \downarrow u}{U \not\downarrow u}$$

FIG. 8.2: Définition des prédicats $U \downarrow u$ et $U \not\downarrow u$

La propriété de réceptivité établit que si un nom est dans l'interface d'un processus, alors ce processus offre une ressource (un récepteur) sur ce nom. Cependant cette ressource peut ne pas être directement disponible, mais on peut montrer qu'une réduction normalisante peut toujours la faire apparaître. Pour exprimer formellement cette propriété, on définit les prédicat $U \downarrow u$ et $U \not\downarrow u$ par induction sur la structure de U (figure 8.2 où nous rappelons que $U \twoheadrightarrow V$ est une réduction sans communication). Dans la clause concernant $\mathsf{go}\,\ell.P$ de cette définition, bien que ce terme ne se réduise pas, il faut anticiper le fait qu'après migration, P offrira un récepteur. Il facile de voir que si $U \downarrow u$ et $V \equiv U$ alors $V \downarrow u$, et que

$$\begin{array}{rcll} P \downarrow a & \Leftrightarrow & P \equiv (\nu\vec{w})(a(\vec{u}).R \mid Q) & a \notin \mathit{subj}(\vec{w}) \\ S \downarrow a@\ell & \Leftrightarrow & S \equiv (\nu\vec{w})(k[\![P]\!] \mid S') & P \downarrow v, v@k = a@\ell \text{ et } a, \ell \notin \mathit{subj}(\vec{w}) \end{array}$$

Le lemme suivant montre qu'après une première phase de réductions normalisantes où des migrations peuvent être réalisées, tous les récepteurs qu'un réseau peut exhiber sont immédiatement accessibles.

Lemme 8.2.6 *Si $P \downarrow a@\ell$ alors il existe un processus Q tel que $Q \downarrow a$ et $k[\![P]\!] \twoheadrightarrow^* (\nu\vec{w})(\ell[\![Q]\!] \mid S)$ pour un certain réseau S, avec $a \notin \mathit{subj}(\vec{w})$.*

Preuve: Par induction sur la définition de $P \downarrow a@\ell$.
• Si $P = \mathsf{go}\,\ell'.R$ avec $R \not\downarrow u$ et $u@\ell' = a@\ell$, on a $k[\![P]\!] \twoheadrightarrow \ell'[\![R]\!]$, et par définition de $R \not\downarrow u$ il existe Q tel que $R \twoheadrightarrow^* Q$ et $Q \downarrow u$. On a donc $k[\![P]\!] \twoheadrightarrow \ell'[\![Q]\!]$, et il y a deux cas : soit $u = a$ et alors $\ell' = \ell$ et c'est fini ; soit $u = a@\ell$ et il suffit d'appliquer hypothèse d'induction à $\ell'[\![Q]\!]$.
• Si $P = (R \mid R')$ avec, par exemple, $R \downarrow a@\ell$, alors $k[\![P]\!] \equiv k[\![R]\!] \mid k[\![R']\!]$, et par hypothèse d'induction, il existe Q tel que $Q \downarrow a$ et $k[\![R]\!] \twoheadrightarrow^* (\nu\vec{w})(\ell[\![Q]\!] \mid S)$, d'où $k[\![P]\!] \twoheadrightarrow^* (\nu\vec{w})(\ell[\![Q]\!] \mid S \mid k[\![R']\!])$.
• Si $P = (\nu w)R$ avec $R \downarrow a@\ell$ et $a \neq \mathit{subj}(w)$ (ou avec $\ell \neq \ell'$ si $w = \ell' : \psi$), on a $k[\![P]\!] \equiv (\nu w@k)k[\![R]\!]$, et on termine en utilisant l'hypothèse d'induction. □

Proposition 8.2.7 (Réceptivité) *Soit U un terme typé et bien formé. Alors :*

1. *si $I \Vdash U$, alors $u \in I \Leftrightarrow U \not\downarrow u$;*
2. *si $U \not\downarrow u$ et $U \to V$, alors $V \not\downarrow u$.*

Preuve:
1. ($\Rightarrow$) On prouve une propriété plus générale, c'est-à-dire : $u, I \Vdash U \Rightarrow$ [S] ψI

message $\overline{a}(\vec{u})$ ou $\mathbf{0}$.
• Si $U = a(\vec{u}).P$, alors $a \Vdash U$ et $[\mathsf{S}]U = \mathsf{S}(a)(\vec{u}).[\mathsf{S}']P$ avec $\mathsf{S}' = \mathsf{S} \restriction \mathit{dom}(\mathsf{S}) - \mathit{nm}(\vec{u})$, et par définition $\mathsf{S}(a)(\vec{u}).[\mathsf{S}']P \downarrow \mathsf{S}(a)$ et donc $\mathsf{S}(a)(\vec{u}).[\mathsf{S}']P \not\downarrow \mathsf{S}(a)$.
• Si $U = V \mid W$, et $I = I_1, I_2$ avec, par exemple, $u, I_1 \Vdash V$ et $I_2 \Vdash W$, par hypothèse d'induction $[\mathsf{S}]V \not\downarrow \mathsf{S}(u)$ et par définition, il existe V' tel que $[\mathsf{S}]V \twoheadrightarrow^* V'$ et $V' \downarrow \mathsf{S}(u)$. Donc, $[\mathsf{S}](V \mid W) \twoheadrightarrow^* (V' \mid [\mathsf{S}]W)$ et $V' \mid [\mathsf{S}]W \downarrow \mathsf{S}(u)$, et finalement, $[\mathsf{S}]U \not\downarrow \mathsf{S}(u)$. La preuve du cas où $U = (\nu w)V$ avec est similaire.
• Si $U = \mathsf{go}\,\ell.P$ alors $u, I = (v, J)@\ell$ avec $v@\ell = u$ et $J@\ell = I$ et $v, J \Vdash P$. Par hypothèse d'induction, $[\mathsf{S}]P \not\downarrow \mathsf{S}(v)$, donc $\mathsf{go}\,\mathsf{S}(\ell).[\mathsf{S}]P \downarrow \mathsf{S}(v)@\mathsf{S}(\ell)$ et finalement, $[\mathsf{S}]U \not\downarrow \mathsf{S}(u)$. Le cas où $U = \ell[\![P]\!]$ est semblable.
• Si $U = [a = b]P, Q$, alors $u, I \Vdash P$ et $u, I \Vdash Q$. Par hypothèse d'induction $[\mathsf{S}]P \not\downarrow \mathsf{S}(u)$ et $[\mathsf{S}]Q \not\downarrow \mathsf{S}(u)$. Donc $\exists V_1, V_2$ tels que $[\mathsf{S}]P \twoheadrightarrow^* V_1$ et $[\mathsf{S}]Q \twoheadrightarrow^* V_2$ avec $V_1 \downarrow \mathsf{S}(u)$ et $V_2 \downarrow \mathsf{S}(u)$. Puisque soit $[\mathsf{S}]U \twoheadrightarrow [\mathsf{S}]P$, soit $[\mathsf{S}]U \twoheadrightarrow [\mathsf{S}]Q$, $\exists V$ tel que $[\mathsf{S}]U \twoheadrightarrow^* V$ et $V \downarrow \mathsf{S}(u)$ et donc $[\mathsf{S}]U \not\downarrow \mathsf{S}(u)$.
• Si $U = T(a, \vec{u})$ avec $I = \emptyset$, on a $T = (\mathsf{rec}\ A(b, \vec{v}).P)$ puisque U est gardé, et $\vec{v}$ et $\vec{u}$ ont même longueur car U est typé. L'inférence de $a \Vdash U$ a donc la forme suivante :

$$\dfrac{\dfrac{\dfrac{\vdots}{b \Vdash P}}{\Vdash (\mathsf{rec}\ A(b, \vec{v}).P)}}{a \Vdash (\mathsf{rec}\ A(b, \vec{v}).P)(a, \vec{u})}$$

On a $[\mathsf{S}](\mathsf{rec}\ A(b, \vec{v}).P)(a, \vec{u}) = T'(c, \vec{u'})$ avec $T' = (\mathsf{rec}\ A(b, \vec{v}).[\mathsf{S}']P)$ où $\mathsf{S}' = \mathsf{S} \restriction \mathit{dom}(\mathsf{S}) - \mathit{nm}(b, \vec{v})$, et $(c, \vec{u'}) = [\mathsf{S}](a, \vec{u})$ et donc

$$[\mathsf{S}]((\mathsf{rec}\ A(b, \vec{v}).P)(a, \vec{u})) \twoheadrightarrow [T'/A][c, \vec{u'}/b, \vec{v}][\mathsf{S}']P$$

P étant également typable et gardé, par hypothèse d'induction, $[c, \vec{u'}/b, \vec{v}][\mathsf{S}']P \not\downarrow c$ et donc, d'après le lemme 7.2.7, $[\mathsf{S}]((\mathsf{rec}\ A(b, \vec{v}).P)(a, \vec{u})) \not\downarrow \mathsf{S}(a)$.
($\Leftarrow$) On montre que si $I \Vdash U$ et $U \downarrow u$ ou $U \not\downarrow u$, alors $u \in I$, par induction sur la définition des prédicats $U \downarrow u$ et $U \not\downarrow u$. On traite le cas où $U = \ell[\![P]\!]$. On a donc, $P \not\downarrow v$ avec $v@\ell = u$, et $J \Vdash P$ avec $J@\ell = I$. Par hypothèse d'induction $v \in J$ et donc $u = v@\ell \in J@\ell = I$. Si $U \not\downarrow u$, avec $U \twoheadrightarrow^* V$ et $V \downarrow u$, par la proposition 8.2.5 (et V est typable par la proposition 7.2.14), $I \Vdash V$, et par hypothèse d'induction $u \in I$. Les autres cas sont aussi simples.
2. Supposons que $I \Vdash U$ alors, par le point précédent, $u \in I$. Par la proposition 8.2.5 et le théorème 7.2.14, V est typable et $I \Vdash V$, ce qui implique, par $1.(\Rightarrow)$ que $V \not\downarrow u$. □

Une conséquence immédiate de cette proposition est qu'on connaît statiquement la localité de tous les récepteurs dans un terme : pour un nom public, elle est indiquée dans l'interface – la localité d'un récepteur sur a est la localité courante si $a \in I$ et ℓ si $a@\ell \in I$ (il est à noter qu'un même nom de canal peut désigner plusieurs récepteurs mais à des localités différentes). Pour un sous-terme $(\nu u)P$ d'un processus bien formé, on sait que u – quand il ne s'agit pas d'une valeur – est dans l'interface de P. Notons cependant, que dans $(\nu a@\ell)P$, la localité ℓ est libre et peut donc être reçue comme paramètre d'un message. Enfin, dans des termes $(\nu \ell : \psi)U$, toute localité nouvellement créée précise l'ensemble de ses récepteurs disponibles. Une conséquence immédiate de la réceptivité est :

Corollaire 8.2.8 *Si $a(\vec{u}).P$ est bien formé et typable, alors $P \not\downarrow a$.*

Cette propriété nous dit qu'un récepteur est *persistants*, dans le sens où il sera toujours disponible, sous le même nom et quelquesoient les messages qui lui sont envoyés. Par conséquent, on peut appeler un récepteur un « *serveur* », qui accepte toujours des requêtes (*i.e.* des messages), ou encore une « ressource ». Nous passons maintenant à la preuve de livrabilité des messages, montrant que si un message est envoyé (à une certaine localité) sur un canal dont la portée est connue dans un processus bien formé et typé, alors le processus offrira un récepteur pour ce nom.

Théorème 8.2.9 (Livrabilité des messages) *Soit S un terme typable et bien formé avec $I \Vdash S$. Si $S \to^* (\nu\vec{w})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)$ avec $a@\ell \in I$ ou $a \in \mathit{subj}(\vec{w})$, alors il existe R et S_1 tels que $S_0 \twoheadrightarrow^* (\nu\vec{w'})(\ell[\![a(\vec{v}).R]\!] \mid S_1)$ avec $a \notin \mathit{subj}(\vec{w'})$.*

Preuve: Par la proposition 8.2.5, on a $I \Vdash (\nu\vec{w})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)$ avec $a@\ell \in I$ ou $a \in \mathit{subj}(\vec{w})$. On montre d'abord par induction sur la longueur de $\vec{w}$ que ceci implique $S_0 \not\downarrow a@\ell$. Si la longueur est 0, on a $I \Vdash S_0$ et $a@\ell \in I$, et on conclue par la proposition 8.2.7. Sinon, soit $\vec{w} = w, \vec{w_0}$ et supposons que $a = \mathit{subj}(w)$ et $a \notin \mathit{subj}(\vec{w})$ (dans le cas contraire soit $a@\ell \in I$, soit $a \in \mathit{subj}(\vec{w_0})$ et on applique simplement l'hypothèse d'induction). Puisque S est typé, disons avec $\Gamma \vdash S$, on a aussi $\Gamma \vdash (\nu\vec{w})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)$ par le théorème 7.2.14. Si on avait $w = a : \psi$ ou $w = a \in \mathcal{N}_{val}$, on devrait avoir

$$\dfrac{\dfrac{\vdots}{a : \tau, \Gamma \vdash (\nu\vec{w_0})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)}}{\Gamma \vdash (\nu w)(\nu\vec{w_0})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)}$$

où $\tau = \psi$ si $w = a : \psi$, ou $\tau = \mathit{val}$ si $w \in \mathcal{N}_{val}$. Mais puisque a est libre dans $(\nu\vec{w_0})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)$, Γ doit contenir $\ell : \{a : \gamma\}$ (voir plus bas) et la règle de typage pour restreindre w ne peut être appliquée. Par conséquent, nécessairement $w = a@k$, et la preuve du séquent précédent a la forme suivante (où $a \notin \Gamma$) :

$$\dfrac{\dfrac{\vdots}{k : \{a : \gamma\}, \Gamma \Vdash (\nu\vec{w_0})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)}}{\Gamma \vdash (\nu a@k)(\nu\vec{w_0})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)}$$

Puisque a est libre dans $(\nu\vec{w_0})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)$ et que le typage de $\ell[\![\overline{a}(\vec{v})]\!]$ a la forme

$$\dfrac{\dfrac{\dfrac{\vdots}{\Theta \vdash_\ell \vec{v} : \vec{\tau}}}{\ell : \{a : \mathit{Ch}(\vec{\tau})\}, \Theta \vdash_\ell \overline{a}(\vec{v})}}{\Delta \vdash \ell[\![\overline{a}(\vec{v})]\!]}$$

avec $\Delta = \ell : \{a : \mathit{Ch}(\vec{\tau})\}, \Theta \subseteq k : \{a : \gamma\}, \Gamma$, le contexte $k : \{a : \gamma\}, \Gamma$ doit contenir l'hypothèse $\ell : \{a : \mathit{Ch}(\vec{\tau})\}$. Or $a \notin \Gamma$, donc $k = \ell$ et $\gamma = \mathit{Ch}(\vec{\tau})$. Par conséquent, on a

$$\dfrac{\dfrac{\vdots}{a@\ell, I \Vdash (\nu \vec{w_0})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)}}{I \Vdash (\nu a@\ell)(\nu \vec{w_0})(\ell[\![\overline{a}(\vec{v})]\!] \mid S_0)}$$

d'où, par hypothèse d'induction, $S_0 \not\downarrow a@\ell$.
A présent, par définition, il existe S_1 tel que $S_0 \twoheadrightarrow^* S_1$ et $S_1 \downarrow a@\ell$, c'est-à-dire (par la remarque faite plus haut) $S_1 \equiv (\nu \vec{w_1})(k[\![P]\!] \mid S_2)$ avec $P \downarrow v$, $v@k = a@\ell$ et $a, \ell \notin \mathit{subj}(\vec{w_1})$. Alors, on a soit $P \downarrow a$ et $k = \ell$, soit $P \downarrow a@\ell$. Dans le premier cas on a

$$S_1 \equiv (\nu \vec{w_1})(\ell[\![(\nu \vec{w_2})(a(\vec{u}).R \mid Q)]\!] \mid S_2) \equiv (\nu \vec{w_3})(\ell[\![a(\vec{u}).R]\!] \mid \ell[\![Q]\!] \mid S_2)$$

avec $a \notin \mathit{subj}(\vec{w_1}, \vec{w_2}, \vec{w_3})$ et ce qui conclue la preuve pour ce cas. Si $P \downarrow a@\ell$ alors, par le lemme 8.2.6, il existe Q, S_3 tel que $Q \downarrow a$ et $k[\![P]\!] \twoheadrightarrow^* (\nu \vec{w_2})(\ell[\![Q]\!] \mid S_3)$, avec $a \notin \mathit{subj}(\vec{w_3})$, et on conclue comme dans le cas précédent. □

On notera que ce résultat est faux pour les termes non typés. Par exemple, on a $\Vdash (\nu a)\overline{a}$ si $a \in \mathcal{N}_{val}$ ou, $(\nu a : \{\})\overline{a}$, alors que ces termes contiennent un message qui ne sera jamais reçu.

CHAPITRE 9

Le style de programmation réceptif

DANS CE CHAPITRE NOUS DONNONS UNE SÉRIE D'EXEMPLES afin d'illustrer le « style » de programmation que nous devons adopter pour se conformer à la discipline de réceptivité. L'objectif de ce chapitre est également de montrer que des « programmes » ou «protocoles » répartis et à priori non réceptifs, peuvent être implantés dans ce style et ainsi, en quelque sorte, nous donner une idée de son expressivité. Nous donnons trois exemples répartis en trois sections : un serveur d'objets, un objet distribué et une cellule migrante.

Rappelons tout d'abord les constructions dérivées ainsi que les conditions de leur typage et de leur bonne formation.

$$
\begin{array}{rcll}
T_a & \stackrel{\mathsf{def}}{=} & (\mathsf{rec}\ A(a).a(\vec{u}).A(a))(a) & \\
a(\vec{u}){:}P & \stackrel{\mathsf{def}}{=} & a(\vec{u}).(P \mid T_a) & \textit{réception linéaire} \\
a^*(\vec{u}).P & \stackrel{\mathsf{def}}{=} & (\mathsf{rec}\ A(a).a(\vec{u}).(P \mid A(a)))(a) & \textit{réception répliquée}
\end{array}
$$

La définition de T_a, et par conséquent son typage, dépendent bien entendu du type de a : on a $\ell\mathbf{:}\{a\mathbf{:}\mathit{Ch}(\vec{\tau})\} \vdash_\ell T_a$ sous réserve qu'il existe un contexte Δ tel que le vecteur $\vec{u}$ dans la définition de T_a valide le séquent $\Delta \vdash \vec{u}\mathbf{:}\vec{\tau}$. D'autre part, on a toujours $a \Vdash T_a$.
Le séquent $\Gamma \vdash_\ell a(\vec{u}).P$ est prouvable si, et seulement, $\Gamma \vdash_\ell a(\vec{u}){:}P$ est prouvable ; et $\Vdash P$ si et seulement si $a \Vdash a(\vec{u}){:}P$.
De même, en faisant l'hypothèse que, pour la réception répliquée, il n'y a pas d'occurrence de A dans P, le séquent $\Gamma \vdash_\ell a^*(\vec{u}).P$ est prouvable si, et seulement, le séquent $\Gamma \vdash_\ell a(\vec{u}).P$ est prouvable. Aussi, $\Vdash P$ est prouvable si, et seulement si, $a \Vdash a^*(\vec{u}).P$ est prouvable.
On se permettra donc d'utiliser les règles de typage et de bonne formation dérivées suivantes :

$$
\frac{\Gamma, \Delta \vdash_\ell P\ ,\ \Delta \vdash_\ell \vec{u}\mathbf{:}\vec{\tau}}{\ell\mathbf{:}\{a\mathbf{:}\mathit{Ch}(\vec{\tau})\}, \Gamma \vdash_\ell a(\vec{u}){:}P}
\qquad
\frac{\Gamma, \Delta \vdash_\ell P\ ,\ \Delta \vdash_\ell \vec{u}\mathbf{:}\vec{\tau}}{\ell\mathbf{:}\{a\mathbf{:}\mathit{Ch}(\vec{\tau})\}, \Gamma \vdash_\ell a(\vec{u})^*.P}
\qquad
\frac{\Vdash P}{a \Vdash a(\vec{u}){:}P}
\qquad
\frac{\Vdash P}{a \Vdash a^*(\vec{u}).P}
$$

9.1 Un serveur d'objets

Supposons l'existence d'un « objet » générique $\mathit{obj}(b, \vec{d_0})$, de nom b – c'est-à-dire $b \Vdash \mathit{obj}(b, \vec{d_0})$ – et de paramètres d'état $\vec{d_0}$ – avec par exemple $\{\vec{d_0}\} \subset \mathcal{N}_{val}$ –, et que nous désirons créer à une localité ℓ un serveur $\mathit{ServObj}(s)$ pour ce type d'objets, qui va en délivrer des instances à chaque requête établie sur un canal s – de manière semblable au *code à la demande*. On peut écrire un tel serveur de la façon suivante :

$$\mathit{ServObj}(s) \stackrel{\mathsf{def}}{=} s^*(c@k_0, \vec{d_0}).\mathsf{go}\, k_0.(\nu b)(\overline{c}(b) \mid \mathit{obj}(b, \vec{d_0}))$$

Un client $\mathit{ClientObj}(k)$ à une localité k qui désire acquérir sa propre instance de l'objet, et l'utilise dans un processus P peut être écrit :

$$\mathit{ClientObj}(k) \stackrel{\mathsf{def}}{=} (\nu c)(\overline{s}@\ell(c@k, \vec{d}) \mid c(b_0){:}P)$$

Supposons que pour toute localité ℓ', $\mathit{obj}(b, \vec{d_0})$ peut être typé avec $\ell' : \{b : \gamma\}, \vec{d_0} : \vec{val} \vdash_{\ell'} \mathit{obj}(b, \vec{d_0})$ dont la preuve est $p_{\ell'}$; montrons alors que $\ell : \{s : \mathit{Ch}(\mathit{Ch}(\gamma)^{@}, \vec{val})\} \vdash_\ell \mathit{ServObj}(s)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\ldots, k_0 : \{c : \mathit{Ch}(\gamma), b : \gamma\}, \vec{d_0} : \vec{val} \vdash_{k_0} \overline{c}(b)} \quad \overset{p_{k_0}}{\vdots}}{\ldots, k_0 : \{c : \mathit{Ch}(\gamma), b : \gamma\}, \vec{d_0} : \vec{val} \vdash_{k_0} \overline{c}(b) \mid \mathit{obj}(b, \vec{d_0})}}{\ell : \{s : \mathit{Ch}(\mathit{Ch}(\gamma)^{@}, \vec{val})\}, \Delta \vdash_{k_0} (\nu b)(\overline{c}(b) \mid \mathit{obj}(b, \vec{d_0}))} \quad \overset{(*)}{\vdots}}{\ell : \{s : \mathit{Ch}(\mathit{Ch}(\gamma)^{@}, \vec{val})\}, \Delta \vdash_\ell \mathsf{go}\, k_0.(\nu b)(\overline{c}(b) \mid \mathit{obj}(b, \vec{d_0}))}}{\ell : \{s : \mathit{Ch}(\mathit{Ch}(\gamma)^{@}, \vec{val})\} \vdash_\ell s^*(c@k_0, \vec{d_0}).\mathsf{go}\, k_0.(\nu b)(\overline{c}(b) \mid \mathit{obj}(b, \vec{d_0}))}$$

où $(*)$ est la preuve de $\Delta \vdash_\ell c@k_0 : \mathit{Ch}(\gamma)^{@}, \vec{d_0} : \vec{val}$ avec $\Delta = k_0 : \{c : \mathit{Ch}(\gamma)\}, \vec{d} : \vec{val}$. De la même manière, on peut montrer que $\ell : \{s : \mathit{Ch}(\mathit{Ch}(\gamma)^{@}, \vec{val})\}, \vec{d} : \vec{val}, \Gamma \vdash_k \mathit{ClientObj}(k)$ si $c(b_0){:}P$ est typable en k avec le contexte $\Gamma, k : \{c : \mathit{Ch}(\gamma)\}$. D'autre part, par affaiblissement, le séquent $\ell : \{s : \mathit{Ch}(\mathit{Ch}(\gamma)^{@}, \vec{val})\}, \vec{d} : \vec{val}, \Gamma \vdash_\ell \mathit{ServObj}(s)$ est valide, et par conséquent on a :

$$\ell : \{s : \mathit{Ch}(\mathit{Ch}(\gamma)^{@}, \vec{val})\}, \vec{d} : \vec{val}, \Gamma \vdash \ell[\![\mathit{ServObj}(s)]\!] \mid k[\![\mathit{ClientObj}(k)]\!]$$

Montrons que $s \Vdash \mathit{ServObj}(s)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Vdash \overline{c}(b)} \quad \dfrac{\vdots}{b \Vdash \mathit{obj}(b, \vec{d_0})}}{b \Vdash \overline{c}(b) \mid \mathit{obj}(b, \vec{d_0})}}{\Vdash (\nu b)(\overline{c}(b) \mid \mathit{obj}(b, \vec{d_0}))}}{\Vdash \mathsf{go}\, k_0.(\nu b)(\overline{c}(b) \mid \mathit{obj}(b, \vec{d_0}))}}{s \Vdash \mathit{ServObj}(s)}$$

De la même manière, on peut montrer que $\Vdash$ *ClientObj*(k). On en déduit que $s@\ell \Vdash \ell[\![$*ServObj*$(s)]\!]$ et $\Vdash k[\![$*ClientObj*$(k)]\!]$, et finalement :

$$s@\ell \Vdash \ell[\![\textit{ServObj}(s)]\!] \mid k[\![\textit{ClientObj}(k)]\!]$$

Ce qui montre que toute requête envoyée au serveur sur le canal s à la localité ℓ sera effectivement traitée.

Modifions à présent l'objet pour lui permettre d'utiliser une ressource *res*(r) de nom public r – avec $r \Vdash$ *res*(r) – dans sa localité d'exécution, et supprimons, pour simplifier, ses paramètres d'état $\vec{d_0}$. Nous notons un tel objet *obj*(b, r), et nous faisons l'hypothèse qu'il est bien typé dans toute localité ℓ avec $\ell : \{b : \gamma, r : \delta\} \vdash_\ell$ *obj*(b, r). Le serveur s'écrit à présent :

$$\textit{ServObj}(s) \stackrel{\text{def}}{=} s^*(c@k_0).\mathsf{go}\, k_0.(\nu b)(\overline{c}(b) \mid \textit{obj}(b, r))$$

Le client doit maintenant comporter la ressource nécessaire à l'exécution de l'objet qu'il invoque :

$$\textit{ClientObj}(k, r) \stackrel{\text{def}}{=} (\nu c)(\overline{s}@\ell(c@k) \mid c(b_0){:}P \mid \textit{res}(r))$$

Maintenant s reçoit un canal communiquant des noms de type γ – qui est le type du nom de l'objet – localisé dans un site où est disponible une ressource r de type δ. On peut montrer que *ServObj*(s) est bien typé avec $\ell : \{s : \mathit{Ch}(\mathit{Ch}(\gamma)@\{r : \delta\})\} \vdash_\ell$ *ServObj*(s), et que *ClientObj*(k, r) est également bien typé avec $\ell : \{s : \mathit{Ch}(\mathit{Ch}(\gamma)@\{r : \delta\})\}, k : \{r : \delta\}, \Gamma \vdash_k$ *ClientObj*(k, r), si $c(b_0){:}P$ est bien typé avec $k : \{c : \mathit{Ch}(\gamma)\}, \Gamma \vdash_k c(b_0){:}P$. Concernant la bonne formation il est facile de vérifier qu'on a toujours $s \Vdash$ *ServObj(s)*, et que $r \Vdash$ *ClientObj*(k, r), d'où

$$s@\ell, r@k \Vdash \ell[\![\textit{ServObj(s)}]\!] \mid k[\![\textit{ClientObj}(k, r)]\!]$$

9.2 Un objet distribué

Dans cette exemple on traite le cas d'un « objet » localisé à la fois en ℓ_0 et en ℓ_1 (avec le même nom), et possédant un état partagé et éventuellement distribué. Plus particulièrement, on programme un « bouton » distribué de nom a. « Appuyer sur le bouton » en ℓ_i signifie envoyer un message sur le canal a en ℓ_i. Appuyer deux fois sur le bouton en ℓ_0 ou en ℓ_1 a pour effet d'envoyer un message sur le canal c en ℓ. En particulier, si on appuie une fois sur le bouton en ℓ_0 et une fois sur le bouton en ℓ_1, un message est envoyé sur c en ℓ. Ceci induit évidemment un politique de coopération entre les deux localités. Nous cherchons à apporter une solution à ce problème qui fonctionne même si l'une des deux localités arrête de s'exécuter. Cette condition évacue la solution consistant à centraliser le fonctionnement du bouton où une seule et même localité décide des émissions. D'autre part, un certain degré d'asymétrie semble désirable pour éviter la situation d'inter-blocage où chaque localité essaie « d'emprunter » le bouton pour produire une émission. Dans le solution présentée, le bouton à la localité ℓ_0 peut être emprunté par celui de la localité ℓ_1 mais l'inverse n'est pas permis. Le bouton distribué utilise une valeur privée b pour emprunter le bouton :

$$S \stackrel{\text{def}}{=} (\nu b)(\ell_0[\![P_0(a, a@\ell_1, c@\ell, b)]\!] \mid \ell_1[\![P_1(a, a@\ell_2, c@\ell, b)]\!])$$

où

$$\begin{array}{rcl} P_0 & \stackrel{\mathsf{def}}{=} & \mathsf{rec}\ A(x, y@z, c'@\ell', b').x(i).(\overline{y}@z(b') \mid x(i).(A(x, y@z, c'@\ell', b') \mid \overline{c'}@\ell'())) \\ P_1 & \stackrel{\mathsf{def}}{=} & \mathsf{rec}\ B(x, y@z, c'@\ell', b').x(i).\mathsf{if}\ i = b'\ \mathsf{then}\ (B(x, y@z, c'@\ell', b') \mid \overline{x}(b')) \\ & & \qquad \mathsf{else}\ x(i).\mathsf{if}\ i = b'\ \mathsf{then}\ (B(x, y@z, c'@\ell', b') \mid \overline{y}@z(b')) \\ & & \qquad\qquad \mathsf{else}\ (B(x, y@z, c'@\ell', b') \mid \overline{c'}@\ell'()) \end{array}$$

On a donc, par dépliage de la récursion :

$$\begin{array}{rcl} P_0(a, a@\ell_1, c@\ell, b) & \to & a(i).(\overline{a}@\ell_1(b) \mid a(i).(P_0(a, a@\ell_1, c@\ell, b) \mid \overline{c}@\ell())) \\ P_1(a, a@\ell_0, c@\ell, b) & \to & a(i).\mathsf{if}\ i = b\ \mathsf{then}\ (P_1(a, a@\ell_0, c@\ell, b) \mid \overline{a}(b)) \\ & & \qquad \mathsf{else}\ a(i).\mathsf{if}\ i = b\ \mathsf{then}\ (P_1(a, a@\ell_0, c@\ell, b) \mid \overline{a}@\ell_0(b)) \\ & & \qquad\qquad \mathsf{else}\ (P_1(a, a@\ell_0, c@\ell, b) \mid \overline{c}@\ell()) \end{array}$$

Afin de mettre en perspective le fonctionnement de ce bouton, nous écrivons

$$\begin{array}{rcl} Q_0 & \stackrel{\mathsf{def}}{=} & P_0(a, a@\ell_1, c@\ell, b) \\ Q_1 & \stackrel{\mathsf{def}}{=} & P_1(a, a@\ell_0, c@\ell, b) \\ Q'_0 & \stackrel{\mathsf{def}}{=} & a(i).(Q_0 \mid \overline{c}@\ell()) \\ Q'_1 & \stackrel{\mathsf{def}}{=} & a(i).\mathsf{if}\ i = b\ \mathsf{then}\ (Q_1 \mid \overline{a}@\ell_0(b))\ \mathsf{else}\ (Q_1 \mid \overline{c}@\ell()) \end{array}$$

où, intuitivement, Q'_0 et Q'_1 sont les états, respectivement, de Q_0 et Q_1 après réception d'un message sur a, et en particulier pour Q'_1 d'une valeur différente de b. Soient $\vec{m}$ un vecteur de messages quelconques sur a, et $\vec{n_1}$ (resp. $\vec{n_2}$) un vecteur de messages sur a communiquant la valeur b (resp. une valeur quelconque d différente de b). Il est facile de vérifier que les transitions suivantes sont valides :

$$\ell_0[\![Q_0 \mid (\overline{a}(e) \mid \vec{m})]\!] \mid \ell_1[\![Q \mid \vec{n_1} \mid \vec{n_2}]\!] \to^* \ell_0[\![Q'_0 \mid \vec{m'}]\!] \mid \ell_1[\![Q \mid (\vec{n_1} \mid \overline{a}(b)) \mid \vec{n_2}]\!] \quad (1)$$

$$\ell_0[\![Q'_0 \mid (\overline{a}(e) \mid \vec{m})]\!] \mid \ell_1[\![Q \mid \vec{n_1} \mid \vec{n_2}]\!] \to^* \ell_0[\![Q_0 \mid \vec{m'}]\!] \mid \ell_1[\![Q \mid \vec{n_1} \mid \vec{n_2}]\!] \mid \ell[\![\overline{c}()]\!] \quad (2)$$

$$\ell_0[\![Q \mid \vec{m}]\!] \mid \ell_1[\![Q_1 \mid \vec{n_1} \mid (\vec{n'_2} \mid \overline{a}(d))]\!] \to^* \ell_0[\![Q \mid \vec{m}]\!] \mid \ell_1[\![Q'_1 \mid \vec{n_1} \mid \vec{n'_2}]\!] \quad (3)$$

$$\ell_0[\![Q \mid \vec{m}]\!] \mid \ell_1[\![Q'_1 \mid \vec{n_1} \mid (\vec{n'_2} \mid \overline{a}(d))]\!] \to^* \ell_0[\![Q \mid \vec{m}]\!] \mid \ell_1[\![Q_1 \mid \vec{n_1} \mid \vec{n'_2}]\!] \mid \ell[\![\overline{c}()]\!] \quad (4)$$

$$\ell_0[\![Q \mid \vec{m}]\!] \mid \ell_1[\![Q'_1 \mid (\vec{n'_1} \mid \overline{a}(b)) \mid \vec{n_2}]\!] \to^* \ell_0[\![Q \mid \vec{m} \mid \overline{a}(b)]\!] \mid \ell_1[\![Q_1 \mid \vec{n'_1} \mid \vec{n_2}]\!] \quad (5)$$

où Q peut désigner Q_1 ou Q'_1 dans les transitions (1) et (2), et Q_0 ou Q'_0 dans les transitions (3) à (5), et e est une valeur quelconque. Les états Q'_0 ou Q'_1 sont atteints quand un message sur a a été reçu une fois ; les états Q_0 ou Q_1 sont retrouvés à la réception d'un deuxième message ce qui entraîne automatiquement l'émission d'un message sur c en ℓ – à condition, dans le cas de Q_1, que la valeur reçue soit différente de b. Ainsi, l'envoi du message $\overline{a}@\ell_0(b)$ par Q_1 est interprété par Q_0 comme la délégation à Q_0 de la pression du bouton (ou encore l'emprunt du bouton de Q_0 par Q_1). Le message $\overline{a}@\ell_1(b)$ envoyé par Q_0 autorise à Q_1 cette délégation.

Soit $\gamma = \mathit{Ch}(\mathit{val})$, montrons que $\Gamma \vdash_{\ell_0} Q_0$ avec $\Gamma = \ell_0, \ell_1 : \{a : \gamma\}, \ell : \{c : \mathit{Ch}()\}, b : \mathit{val}$ – on omet les preuves de $\Gamma \vdash^W_{\ell_0} (a, a@\ell_1, c@\ell, b) : \vec{\tau}$ et de $\Delta \vdash_{\ell_0} \vec{u} : \vec{\tau}$ où

$$\begin{array}{rcl} \Delta & = & \ell_0 : \{x : \gamma\}, z : \{y : \gamma\}, \ell' : \{c' : \mathit{Ch}()\}, b' : \mathit{val} \\ \vec{u} & = & (x, y@z, c'@\ell', b') \\ \vec{\tau} & = & (\gamma, \gamma^{@}, \mathit{Ch}()^{@}, \mathit{val}) \end{array}$$

$$\dfrac{\dfrac{\dfrac{\vdots}{\ldots, \Delta \vdash_z \overline{y}(b')}}{\ldots, \Delta \vdash_{\ell_0} \overline{y}@z(b')} \qquad \dfrac{\dfrac{\dfrac{\vdots}{A : Ch(\vec{\tau}), \Delta, \ldots \vdash_{\ell_0} A(\vec{u})} \qquad \dfrac{\dfrac{\vdots}{\Delta, \ldots \vdash_{\ell'} \overline{c'}()}}{\Delta, \ldots \vdash_{\ell_0} \overline{c'}@\ell'()}}{A : Ch(\vec{\tau}), \Delta, j : val, \ldots \vdash_{\ell_0} A(\vec{u}) \mid \overline{c'}@\ell'()}}{A : Ch(\vec{\tau}), \Delta, \ldots \vdash_{\ell_0} x(j).(A(\vec{u}) \mid \overline{c'}@\ell'())}}{A : Ch(\vec{\tau}), \Gamma, \Delta, i : val \vdash_{\ell_0} \overline{y}@z(b') \mid x(i).(A(\vec{u}) \mid \overline{c'}@\ell'())}$$
$$\dfrac{A : Ch(\vec{\tau}), \Gamma, \Delta \vdash_{\ell_0} x(i).(\overline{y}@z(b') \mid x(j).(A(\vec{u}) \mid \overline{c'}@\ell'()))}{\Gamma \vdash_{\ell_0} P_0 : Ch(\vec{\tau})}$$
$$\Gamma \vdash_{\ell_0} Q_0$$

On peut aussi vérifier que $\Gamma \vdash_{\ell_1} Q_1$ est valide. Par conséquent, on a

$$\ell_0, \ell_1 : \{a : Ch(val)\}, \ell : \{c : Ch()\} \vdash (\nu b)(\ell_0[\![Q_0]\!] \mid \ell_1[\![Q_1]\!])$$

Q_0 et Q_1 sont bien formés avec l'interface $\{a\}$. Vérifions le pour Q_0.

$$\dfrac{\dfrac{\dfrac{\vdots}{\Vdash \overline{y}@z(b')} \qquad \dfrac{\dfrac{\Vdash A}{x \Vdash A(\vec{u})} \qquad \dfrac{\vdots}{\Vdash \overline{c'}@\ell'()}}{x \Vdash A(\vec{u}) \mid \overline{c'}@\ell'()}}{x \Vdash x(j).(A(\vec{u}) \mid \overline{c'}@\ell'())}}{x \Vdash x(i).(\overline{y}@z(b') \mid x(j).(A(\vec{u}) \mid \overline{c'}@\ell'()))}$$
$$\dfrac{\Vdash P_0}{a \Vdash Q_0}$$

Ainsi, on a

$$a@\ell_0, a@\ell_1 \Vdash (\nu b)(\ell_0[\![Q_0]\!] \mid \ell_1[\![Q_1]\!])$$

9.3 Une cellule migrante

Dans cette section on adopte une notation empruntée aux « objets » de TyCO [Vas94], c'est-à-dire qu'on écrit

$$a\{m_1 = (\vec{u_1})P_1, \ldots, m_n = (\vec{u_n})P_n, Q\}$$

pour le processus

$$a(m, \vec{u_1}, \ldots, \vec{u_n}).[m = m_1]\, P_1,$$
$$\vdots$$
$$[m = m_n]\, P_n, Q$$

où les m_i sont supposés être de type *val*. On peut facilement vérifier que si $a \Vdash P_i$ pour tout i, et $a \Vdash Q$, alors le processus précédent est également bien formé avec l'interface a. Un tel agent se comporte comme un objet : il offre des « méthodes » $m_1, \ldots, m_n$, et modifie éventuellement son état – mais pas son identité – après réception d'un message. Les messages pour un tel agent seront notés $\overline{a} \triangleleft m_i(\vec{u_i})$ comme abréviation pour $\overline{a}(m_i, \_, \ldots, \vec{u_i}, \ldots, \_)$. Dans la plupart des cas, ces objets ont un comportement récursif simple tel que le suivant :

$$\begin{aligned} \mathsf{rec}\, A(a,\vec{v}).a(m,\vec{u_1},\ldots,\vec{u_n}).[m=m_1]\,(P_1 \mid A(a,\vec{v})), \\ \vdots \\ [m=m_n]\,(P_n \mid A(a,\vec{v})), A(a,\vec{v}) \end{aligned}$$

où $\Vdash P_i$ pour tout i. Ces agents réagissent uniformément aux appels de méthodes : ils déclenchent le processus P_i sollicité et se réincarnent en un objet identique.

Dans le même ordre d'idée que le buffer (exemple 3.4.2), on peut programmer une *cellule* (voir aussi [Tur96]) qu'on peut lire sans détruire son contenu. Pour donner plus d'intérêt à cet exemple dans le cadre distribué, on suppose que cette cellule ne fait pas que répondre à des requêtes de lecture et d'écriture, mais qu'elle répond également à des requêtes de *migration* vers un localité donnée. Le nom c de la cellule a le type $\gamma = \mathit{Ch}(\mathit{val}, \mathit{Ch}(\mathit{val}), \mathit{val}, \mathit{Ch}(), \{\})$, où le premier argument est la clef read (resp. write, resp. migr) pour la lecture (resp. l'écriture, resp. la migration), les second et quatrième arguments sont des canaux de retour, pour les lecteurs et écrivains respectivement, le troisième est le contenu de la cellule, et le dernier la localité de destination en cas de migration. Pour maintenir « l'accessibilité » d'une cellule migrante, celle-ci prend un nouveau nom à la localité de destination et un « proxy » est laissé à la localité d'origine pour maintenir la liaison entre la cellule et un processus client. Un tel mécanisme est nécessaire pour maintenir une politique de réceptivité : si la cellule garde le même nom, elle n'est pas bien formée, de même s'il n'y a pas un récepteur sur c à la localité d'origine pour prendre la relève. À la place du forwarder, laisser un processus tel que T_c aurait pu suffire pour garantir la réceptivité mais dans ce cas la cellule ne serait plus accessible et les messages ne seraient plus traités. On pourrait également imaginer un récepteur qui retourne un message indiquant que la cellule a migré. Nous voyons donc que la contrainte de réceptivité oblige le programmeur à prendre en compte les messages éventuellement adressés à la cellule à une localité qu'elle a quitté.

La cellule migrante $\mathit{MigCell}(c, x, \ell)$ a pour paramètres x et ℓ, qui sont sa valeur et sa localité courantes :

$$\begin{aligned} \mathit{MigCell}(c,x,\ell) \stackrel{\mathsf{def}}{=} \mathsf{rec}\, A(c,x,\ell).c\{\mathsf{read} = (y)(\overline{y}(x) \mid A(c,x,\ell))\,, \\ \mathsf{write} = (x',z)(\overline{z}() \mid A(c,x',\ell))\,, \\ \mathsf{migr} = M\ ,\ A(c,x,\ell)\} \end{aligned}$$

où

$$\begin{aligned} M &\stackrel{\mathsf{def}}{=} (k)(\nu c'@k)\big(\mathsf{go}\,k.A(c',x,k) \mid \mathit{Proxy}\big) \\ \mathit{Proxy} &\stackrel{\mathsf{def}}{=} c^*(m,y,x,z,t).\mathsf{go}\,k.(\nu y')\big(\mathit{Fwd}(y',y@\ell) \mid \\ &\qquad\qquad (\nu z')\big(\mathit{Fwd}(z',z@\ell) \mid \overline{c'}(m,y',x,z',t)\big)\big) \\ \mathit{Fwd}(a,b@\ell) &\stackrel{\mathsf{def}}{=} a^*(\vec{x}).\overline{b}@\ell(\vec{x}) \end{aligned}$$

Dans le cas d'une requête de lecture $\overline{c}\triangleleft\mathsf{read}(y)$, la cellule retourne sa valeur courante x sur le canal y du client. Dans le cas d'une opération d'écriture, $\overline{c}\triangleleft\mathsf{write}(x', z)$, la cellule est mise à jour avec la nouvelle valeur x', et un accusé de réception $\overline{z}()$ est envoyé au processus qui a invoqué cette opération. Notons que les processus clients sont supposés être locaux, c'est-à-dire, qu'ils sont dans la même localité que la cellule. Finalement, une instruction de migration $\overline{c} \triangleleft \mathsf{migr}(k)$ a pour effet de transférer la cellule à la localité k, avec un nouveau nom c', son contenu courant x et sa nouvelle localité courante k, et un « proxy » gérant les appels à c est laissé à la localité quittée par la cellule.

On peut vérifier que la cellule est bien typée avec $\ell : \{c : \gamma\}, x : \mathit{val} \vdash_\ell \mathit{MigCell}(c, x, \ell)$. Il est facile de voir que les forwarders du proxy sont bien typés avec $k : \{y' : \mathit{Ch}(\mathit{val})\}, \ell : \{y : \mathit{Ch}(\mathit{val})\} \vdash_k \mathit{Fwd}(y', y@\ell)$ et $k : \{z' : \mathit{Ch}()\}, \ell : \{z : \mathit{Ch}()\} \vdash_k \mathit{Fwd}(z, z'@\ell)$. On peut donc typer le proxy dont la preuve est schématisée par l'inférence suivante – on omet à nouveau la preuve du séquent $\Delta \vdash_\ell (m, y, x, z, t) : (\mathit{val}, \mathit{Ch}(\mathit{val}), \mathit{val}, \mathit{Ch}(), \{\})$:

$$\dfrac{\dfrac{\dfrac{\vdots}{\ldots, \Delta, k : \{y' : \mathit{Ch}(\mathit{val})\} \vdash_k \mathit{Fwd}(y', y@\ell)} \qquad \dfrac{\dfrac{\vdots}{\ldots, k : \{z' : \mathit{Ch}()\} \vdash_k \mathit{Fwd}(z', z@\ell) \mid \overline{c'}(m, y', x, z', t)}}{\Gamma, \Delta, k : \{y' : \mathit{Ch}(\mathit{val})\} \vdash_k P}}{\Gamma, \Delta, k : \{y' : \mathit{Ch}(\mathit{val})\} \vdash_k \mathit{Fwd}(y', y@\ell) \mid P}}{\Gamma, \Delta \vdash_\ell \mathsf{go}\, k.(\nu y')(\mathit{Fwd}(y', y@\ell) \mid P)}}{\Gamma \vdash_\ell \mathit{Proxy}}$$

où

$$\begin{aligned}
\Gamma &= \ell : \{c : \gamma\}, k : \{c' : \gamma\} \\
\Delta &= \ell : \{y : \mathit{Ch}(\mathit{val}), z : \mathit{Ch}()\}, t : \{\}, m, x : \mathit{val} \\
P &= (\nu z')(\mathit{Fwd}(z', z@\ell) \mid \overline{c'}(m, y', x, z', t))
\end{aligned}$$

Les séquents suivants sont valides, nous omettons leurs preuves qui sont faciles :

$$\begin{aligned}
A : \mathit{Ch}(\gamma, \mathit{val}, \{\}), \ell : \{c : \gamma, y : \mathit{Ch}(\mathit{val})\}, x : \mathit{val} &\vdash_\ell \overline{y}(x) \mid A(c, x, \ell) \\
A : \mathit{Ch}(\gamma, \mathit{val}, \{\}), \ell : \{c : \gamma, z : \mathit{Ch}()\}, x' : \mathit{val} &\vdash_\ell \overline{z}() \mid A(c, x', \ell) \\
A : \mathit{Ch}(\gamma, \mathit{val}, \{\}), \ell : \{c : \gamma\}, x : \mathit{val} &\vdash_\ell (\nu c'@k)\big(\mathsf{go}\, k.A(c', x, k) \mid \mathit{Proxy}\big)
\end{aligned}$$

Et on peut facilement en déduire le typage de la cellule :

$$x, \mathsf{read}, \mathsf{write}, \mathsf{migr} : \mathit{val}, \ell : \{c : \gamma\} \vdash_\ell \mathit{MigCell}(c, x, \ell)$$

Dans le même ordre d'idée on peut rapidement vérifier que la cellule est bien formée on observant que les jugements suivants sont valides :

$$\begin{aligned}
y', z' &\Vdash \mathit{Fwd}(y', y@\ell) \mid \mathit{Fwd}(z', z@\ell) \\
c &\Vdash \mathit{Proxy} \\
c &\Vdash \overline{y}(x) \mid A(c, x, \ell) \\
c &\Vdash \overline{z}() \mid A(c, x', \ell) \\
c &\Vdash (\nu c'@k)\big(\mathsf{go}\, k.A(c', x, k) \mid \mathit{Proxy}\big)
\end{aligned}$$

Et finalement $c \Vdash \mathit{MigCell}(c, x, \ell)$.

Dans cette version, une cellule distante n'est accessible que par des messages émis depuis sa localité d'origine. Nous pourrions également programmer une cellule disponible à sa nouvelle localité avec un nom c' transmis au client sur un canal a reçu en argument, c'est-à-dire en modifiant M de la façon suivante :

$$M' \stackrel{\text{def}}{=} (a@k)(\nu c'@k)(\mathsf{go}\, k.(\overline{a}(c') \mid A(c', x, k)) \mid Proxy)$$

Alors, un processus désirant migrer cette cellule à une localité k avec une partie de son état s'écrira :

$$Q = (\nu a@k)(\overline{c} \triangleleft \mathsf{migr}(a@k) \mid \mathsf{go}\, k.a(c'){:}P)$$

On peut facilement vérifier qu'on obtient l'exécution suivante :

$$\ell[\![\mathit{MigCell}(c, x, \ell) \mid Q \mid R]\!] \rightarrow^* (\nu c'@k)(\ell[\![\mathit{Proxy} \mid R]\!] \mid k[\![\mathit{MigCell}(c', x, k) \mid P \mid (\nu a)T_a]\!])$$

Cependant, le typage de cette nouvelle version de la cellule migrante nécessite des types récursifs. Ceux-ci sont simplement obtenus en ajoutant des variables de type et la récursion, comme suit :

$$\tau \ ::= \ \ldots \mid t \mid \mu t.\tau$$

Il sont utilisés dans le système d'inférence de types comme tout autre type, cependant l'égalité de types récursifs est interprétée comme l'égalité des arbres (infinis, réguliers) qu'ils représentent. Ainsi, en posant

$$\delta = \mu t.\mathit{Ch}(\mathit{val}, \mathit{Ch}(\mathit{val}), \mathit{val}, \mathit{Ch}(), \mathit{Ch}(t)^{@})$$

on peut typer la nouvelle cellule avec c de type δ.

Cette exemple montre déjà que l'ajout de la récursion au calcul Dπ – qui n'avait « que » l'opérateur de réplication – permet d'obtenir un certain degré de « dynamicité » pour les entités migrantes. Cependant, notre cellule migrante n'est qu'une forme parmi d'autres de « mobilité d'état ». Dans le cas de la deuxième version avec types récursifs, et pour faire référence au chapitre 6, nous sommes en présence d'une mobilité par déplacement. Si on considère le nom c de la cellule en ℓ comme faisant partie du contexte d'un processus local, ce contexte est fixe lors de la migration de la cellule mais la liaison est explicitement maintenue à distance, par l'intermédiaire des proxys ou des références distantes. T. SEKIGUCHI et A. YONEZAWA, dans [SY97], identifient ce type de mobilité sous l'appellation « *carry type* », c'est-à-dire que la donnée – ici notre cellule – est « physiquement » perdue pour le site d'origine mais reste accessible depuis ce dernier grâce à une référence distante.

Les auteurs dans [SY97] identifient aussi d'autres types de mobilité dont le « *resident type* », le dual de « *carry type* ». Dans ce cas, une instruction de migration de la cellule a pour effet de laisser la cellule à la localité d'origine et de la rendre accessible depuis la localité de destination via un proxy qui est l'entité réellement migrée. On écrit alors :

$$M_{resident} \stackrel{\text{def}}{=} (a@k)(\mathsf{go}\, k.(\nu c')(\overline{a}(c') \mid \mathit{Proxy}') \mid A(c, x, \ell))$$

où *Proxy*$'$ est le dual de *Proxy* : il accepte des requêtes sur c' en k et les renvoie sur c en ℓ.

On peut également facilement donner une version de la cellule où la migration est réalisée par duplication sans cohérence maintenue entre la copie et son original (appelé « *copy type* » dans [SY97]) comme suit :

$$M_{copy} \stackrel{\mathsf{def}}{=} (a@k)(\mathsf{go}\, k.(\nu c')(\overline{a}(c') \mid A(c', x, k)) \mid A(c, x, \ell))$$

Enfin, deux types de mobilité sont également définis dans [SY97] : « *proper* » et « *takeaway* ». Le premier caractérise la migration d'une entité (un utilisateur de la cellule) dont le contexte reste fixe (ici la cellule) et la relation avec celui-ci est perdue. Le second est son symétrique : le contexte est mobile et la relation avec celui-ci est également perdue pour les processus sollicitant un élément du contexte dans le domaine initial. Il n'est pas possible de représenter exactement ces types de mobilité en respectant le principe de réceptivité, car « perdre la relation » signifie pour nous « solliciter une ressource qui a disparu », ou, en d'autres termes, envoyer un message sur un canal qui n'a plus de récepteur. On peut cependant s'en tirer en utilisant les « trous noirs » T_c – qui absorbent tous les messages émis sur c – pour remplacer la cellule correspondante :

$$\begin{aligned} M_{proper} &\stackrel{\mathsf{def}}{=} (a@k)(\mathsf{go}\, k.(\nu c')(\overline{a}(c') \mid T_{c'}) \mid A(c, x, \ell)) \\ M_{takeaway} &\stackrel{\mathsf{def}}{=} (a@k)(\mathsf{go}\, k.(\nu c')(\overline{a}(c') \mid A(c', x, k)) \mid T_c) \end{aligned}$$

À la place de T_c nous pourrions, par exemple, programmer un processus qui, à la réception d'un message sur c, envoie un signal au processus qui a émis ce message lui indiquant que le service n'est plus disponible. Pour conclure, nous avons vu à travers cette exemple, que la discipline de programmation réceptive impose au programmeur de prendre garde à ne pas transférer des ressources sans fournir à la localité qui est quittée une ressource de rechange avec éventuellement un comportement différent. Par conséquent, on ne peut pas supprimer (ou déplacer) une ressource, mais on peut subvenir à cette lacune en changeant dynamiquement le comportement d'une ressource et en créant de nouvelles ressources distantes.

Troisième partie

Inférence de type

CHAPITRE 10

Préliminaires

CE CHAPITRE A POUR OBJECTIF D'INTRODUIRE L'ALGORITHME D'INFÉRENCE DE TYPES que nous décrivons formellement dans les deux chapitres suivants. Nous commençons par restreindre le langage et discutons de ces restrictions. Ensuite, nous donnons l'idée générale de l'algorithme de manière très intuitive en exposant des solutions envisageables et enfin la solution adoptée.

10.1 Révisions (mineures) des termes et du système de types

L'algorithme d'inférence de types repose sur beaucoup de détails techniques qui peuvent rendre sa compréhension délicate. Pour conséquent, pour simplifier l'algorithme, nous avons choisi de ne traiter que la version monadique de Dπ, bien que l'extension de l'algorithme au cas polyadique ne pose pas de problème théorique. Aussi, puisque les types vont comporter à présent des variables de types auxquelles nous appliquons des substitutions, nous supprimons les informations de types pouvant se trouver dans les termes, c'est-à-dire dans les abstractions de localités. En réalité, cette restriction n'en est pas une car l'information de types pour les localités restreintes seront inférées. Supprimer les types des termes nous permet ainsi de ne pas avoir à appliquer de substitutions sur les termes ce qui simplifie d'autant les preuves des propriétés de l'algorithme et la formalisations de différents concepts introduits. Ces restrictions ont pour unique objectif d'accroire la lisibilité et non de permettre de résoudre la problème de l'inférence de type. Les termes du langage sont donc décrits

par la grammaire suivante :

noms	$u, v, \ldots$	$::=$	$a \mid a@b$
processus	$P, Q, \ldots$	$::=$	$\mathbf{0} \mid \overline{a}(u) \mid a(u).P \mid (P \mid Q) \mid [a = b]P, Q$
		$\mid$	$(\nu u)P \mid \mathsf{go}\,\ell.P \mid T(u)$
processus paramétrés	T	$::=$	$A \mid (\mathsf{rec}\, A(u).P)$
réseaux	S	$::=$	$\mathbf{0} \mid \ell[\![P]\!] \mid (S \mid S') \mid (\nu u)S$

Comme dans la plupart des compilateurs, la première étape suivant l'analyse lexicale et syntaxique d'un programme consiste à renommer les variables liées portant sur le même nom et ayant des occurrences dans des portées lexicales qui se recouvrent. C'est par exemple le cas de a dans $(\nu a)(P \mid (\nu a)Q)$ qui sera par exemple réécrit en $(\nu a)(P \mid (\nu b)[b/a]Q)$. Dans D$\pi$, la liaison des noms étant statique (*i.e.* pas de capture de variables), nous pouvons opérer à ce même type de manipulations préliminaires. Plus précisément, puisque nous typons également des termes qui peuvent être ouverts, nous procédons à un renommage des noms liés de sorte que tous les noms liés soient distincts entre eux et distincts des noms libres. Les algorithmes réalisant cette réécriture seront triviaux et classiques, nous n'en donnons donc pas la description. Du point de vue du typage, les règles permettant de typer les constructions liantes (abstractions et restrictions) requièrent que le nom lié n'ait pas d'occurrence dans le contexte de typage. Si tel est le cas il faut d'abord utiliser la règle d'α-conversion pour pouvoir ensuite appliquer la règle pour la construction liante. On voit alors tout l'intérêt de ne considérer que les termes dont tous les noms liés sont distincts entre eux et distincts des noms libres : si un tel terme est typable alors il y a une preuve du typage qui n'utilise pas de règle d'α-conversion. Par conséquent, le système de types que nous considérons dans cette partie est celui des figures 7.4 à 7.7 du chapitre 7 pour des termes monadiques et sans règle d'α-conversion. D'autre part, le type d'une localité restreinte est maintenant « deviné » car il n'est plus fourni par le terme. Nous reportons les règles d'inférence dans les figures 10.1 à 10.4. Enfin remarquons que si $\Gamma \vdash S$ est prouvable, alors Γ ne comporte aucun nom qui soit lié dans S.

10.2 Méthodologie

Classiquement, l'inférence de types consiste en deux étapes distinctes :

1. la génération de contraintes ou d'équations entre des schémas de types ;
2. le calcul d'une substitution solution des contraintes précédemment produites.

La première étape nécessite un contexte de typage initial et est fortement dirigée par les constructeurs du terme dont on cherche un typage. Le contexte initial peut se calculer très facilement, il suffit généralement d'assigner à chaque nom libre du terme une variable de type. Par exemple, pour le λ-terme xy, on aura le contexte $x : \alpha, y : \beta$ où α et β sont des variables de types. Le terme étant l'application de x à y, x a donc un type fonctionnel dont l'argument est de même type que celui de y. On produira alors les contraintes $\alpha \doteq \alpha_1 \to \alpha_2$ et $\alpha_1 = \beta$ où α_1 et α_2 sont des variables de types fraîches. Une fois toutes les contraintes produites, la deuxième étape produit une substitution μ qui les résout et dont l'application au contexte initial donne le contexte de typage effectif du terme initial. Si nous sommes

$$\frac{}{a:\tau \vdash_\ell a:\tau} \qquad \frac{}{\ell:\{a:\gamma\} \vdash_\ell a:\gamma} \qquad \frac{}{\ell:\{a:\gamma,\psi\} \vdash_k a@\ell:\gamma@\psi}$$

FIG. 10.1: Système de types fort pour les noms

$$\frac{}{\Gamma, a:\tau \vdash^W_\ell a:\tau} \qquad \frac{}{\Gamma, \ell:\{a:\gamma\} \vdash^W_\ell a:\gamma} \qquad \frac{}{\Gamma, \ell:\{a:\gamma,\psi\} \vdash^W_k a@\ell:\gamma@\psi}$$

FIG. 10.2: Système de types faible pour les noms

$$\frac{}{\Gamma \vdash_\ell \mathbf{0}} \qquad \frac{\Gamma \vdash^W_\ell u:\tau}{\ell:\{a:\mathit{Ch}(\tau)\}, \Gamma \vdash_\ell \overline{a}(u)} \qquad \frac{\Gamma, \Delta \vdash_\ell P \quad , \quad \Delta \vdash_\ell u:\tau}{\ell:\{a:\mathit{Ch}(\tau)\}, \Gamma \vdash_\ell a(u).P}$$

$$\frac{\Gamma \vdash_\ell P \quad , \quad \Gamma \vdash_\ell Q}{a, b:\mathit{val}, \Gamma \vdash_\ell [a=b]P, Q} \qquad \frac{\Gamma \vdash_\ell P \quad , \quad \Gamma \vdash_\ell Q}{\ell_1, \ell_2:\{\}, \Gamma \vdash_\ell [\ell_1=\ell_2]P, Q}$$

$$\frac{a:\mathit{val}, \Gamma \vdash_\ell P \quad , \quad a \in \mathcal{N}_{val}}{\Gamma \vdash_\ell (\nu a)P} \qquad \frac{\ell:\{a:\gamma\}, \Gamma \vdash_\ell P \quad , \quad a \in \mathcal{N}_{chan}}{\Gamma \vdash_\ell (\nu a)P} \qquad \frac{k:\{a:\gamma\}, \Gamma \vdash_\ell P}{\Gamma \vdash_\ell (\nu a@k)P}$$

$$\frac{k:\psi, \Gamma \vdash_\ell P}{\Gamma \vdash_\ell (\nu k)P} \qquad \frac{\Gamma \vdash_\ell P \quad , \quad \Gamma \vdash_\ell Q}{\Gamma \vdash_\ell P \mid Q} \qquad \frac{\Gamma \vdash_k P}{\Gamma \vdash_\ell \mathsf{go}\, k.P} \qquad \frac{}{A:\mathit{Ch}(\vec{\tau}), \Gamma \vdash_\ell A:\mathit{Ch}(\vec{\tau})}$$

$$\frac{A:\mathit{Ch}(\vec{\tau}), \Gamma, \Delta \vdash_\ell P \quad , \quad \Delta \vdash_\ell \vec{u}:\vec{\tau}}{\Gamma \vdash_\ell (\mathsf{rec}\ A(\vec{u}).P):\mathit{Ch}(\vec{\tau})} \qquad \frac{\Gamma \vdash_\ell T:\mathit{Ch}(\vec{\tau}) \quad , \quad \Gamma \vdash^W_\ell \vec{u}:\vec{\tau}}{\Gamma \vdash_\ell T(\vec{u})}$$

FIG. 10.3: Système de types pour les processus

$$\frac{}{\Gamma \vdash \mathbf{0}} \qquad \frac{\Gamma \vdash_\ell P}{\Gamma \vdash \ell[\![P]\!]} \qquad \frac{\Gamma \vdash S \quad , \quad \Gamma \vdash S'}{\Gamma \vdash S \mid S'}$$

$$\frac{\ell:\{a:\gamma\}, \Gamma \vdash S}{\Gamma \vdash (\nu a@\ell)S} \qquad \frac{a:\mathit{val}, \Gamma \vdash S \quad , \quad a \in \mathcal{N}_{val}}{\Gamma \vdash (\nu a)S} \qquad \frac{\ell:\psi, \Gamma \vdash S}{\Gamma \vdash (\nu \ell)S}$$

FIG. 10.4: Système de types pour les réseaux

intéressés par un **typage principal**, on désirera alors trouver un contexte générique, c'est-à-dire tel que tout contexte de typage est une instance de ce contexte générique. Ainsi, ce dernier est un « représentant » de tous les typages possibles pour le terme désiré. L'existence d'un typage principal dépend de l'existence d'une substitution la plus générale – ou **substitution principale** – de l'ensemble de contraintes produites par la première étape de l'inférence. L'algorithme d'inférence de type que nous donnons génère un typage principal.

10.2.1 Les variables de rangées

Comme nous l'avons mentionné plus haut, nous devons considérer des **schémas de types**, c'est-à-dire des types pouvant comporter des variables de types que nous allons pouvoir substituer pour résoudre les contraintes. Nous noterons $\tau, \gamma, \psi, \ldots$ les schémas de types, nous donnerons leur syntaxe exacte dans le chapitre suivant. La difficulté de l'inférence de types pour notre système réside essentiellement dans le fait que nous sommes en présence de **types dépendants** (les types de localités), et plus précisément dépendant de noms de canaux pouvant être liés. Mais nous décrivons un peu plus loin ces difficultés. Concernant les types de localités il va être nécessaire de pouvoir les étendre, c'est-à-dire que l'application d'une substitution à un type de localité doit permettre d'obtenir un type de localité qui type davantage de canaux et ainsi être en mesure, par exemple, de résoudre des contraintes telles que $\{a : \gamma\} \doteq \{b : \delta\}$. Pour cela, nous pouvons nous inspirer de travaux réalisés dans le cadre de l'inférence de types pour les langages à objets [Wan87, Rém93b, Rém93a, JM93]. En effet, le type d'un objet n'est rien d'autre qu'une collection de labels typés tout comme nos types de localités sont des collections de canaux typés et dont l'ordre est sans importance. L'extension des types d'objets est permise grâce à l'introduction de variables de rangée ρ_L comportant en indice un ensemble de noms de labels (des canaux dans notre cas). Les types de localité ont donc la forme

$$\psi = \{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$$

L'indice L sert à conserver la propriété de l'unicité du typage des noms dans un type de localité : tous les a_i sont distincts deux à deux. Intuitivement, cet indice enregistre les noms a_i et le type ψ précédent est un type bien formé si tous les a_i sont distincts entre eux et si $\{a_1, \ldots, a_n\} \subseteq L$ (1). Par exemple le type $\{a : \gamma, \rho_{\{a\}}\}$ est bien formé. La substitution d'un type de localité ϕ à ρ_L n'est alors valide que si ϕ n'assigne pas des types à des noms présents dans L, c'est-à-dire si $\mathit{dom}(\phi) \cap L = \emptyset$ (2). Ainsi, par exemple, $[\{a : \delta, \rho'_{\{a\}}\}/\rho_{\{a\}}]$ n'est pas une substitution correcte car, appliquée au type précédent, elle mène à un type de localité assignant deux types à un même nom de canal : $\{a : \gamma, a : \delta, \rho'_{\{a\}}\}$. Cependant, pour maintenir le domaine du type de localité inclus dans l'indice L, il faut qu'une substitution $[\{a_1 : \gamma, \ldots, a_n : \gamma_n, \rho'_{L'}\}/\rho_L]$ satisfasse également $L \subseteq L'$ (3). Si tel n'est pas le cas, l'application de $[\{b : \delta, \rho'_{\{b\}}\}/\rho_{\{a\}}]$ (qui satisfait (2) mais pas (3)) à $\{a : \gamma, \rho_{\{a\}}\}$ donne $\{a : \gamma, b : \delta, \rho'_{\{b\}}\}$ qui type bien des noms distincts mais ne satisfait déjà plus (1). Et, si on lui applique ensuite $[\{a : \gamma', \rho''_{\{a\}}\}/\rho'_{\{b\}}]$ (qui satisfait toujours (1)) on obtient $\{a : \gamma, b : \delta, a : \gamma', \rho''_{\{a\}}\}$ qui n'est plus du tout bien formé.

10.2.2 Le typage principal

La définition – informelle – de typage principal que nous avons donnée plus haut n'est pas suffisante pour notre système. En effet, un contexte générique peut être trop permissif pour pouvoir exprimer les typages possibles d'un terme et uniquement ceux-ci : il peut y avoir des instances de ce contexte qui ne sont pas des contextes de typage pour le terme considéré. La raison est que dans notre calcul un même nom peut être utilisé dans deux localités différentes avec deux types différents et qu'une forme de sous-typage implicite existe entre les types de localités. Considérons par exemple le terme :

$$P = a(\ell).\mathsf{go}\,\ell.\overline{b}(c) \mid \overline{a}(k_1) \mid \overline{a}(k_2)$$

où c est une valeur. En l'absence d'informations supplémentaires sur les localités k_1 et k_2, ce terme a une infinité de typages possibles. Cependant les types de k_1 et k_2 doivent au minimum fournir le type $\mathit{Ch}(\mathit{val})$ à b. Ainsi, en considérant que P soit localisé en k, un contexte valide est :

$$\Gamma = k : \{a : \mathit{Ch}(\{b : \mathit{Ch}(\mathit{val})\})\}, k_1 : \{b : \mathit{Ch}(\mathit{val})\}, k_2 : \{b : \mathit{Ch}(\mathit{val})\}, c : \mathit{val}$$

Aussi, les types de k_1 et k_2 peuvent typer d'autres canaux que b et pas nécessairement les mêmes. Et a peut se voir attribué un type plus « généreux » (en typant un canal b' par exemple) à condition que les types de k_1 et k_2 correspondent. Par conséquent, le typage suivant est encore un typage correcte pour P :

$$\Delta = \begin{cases} k : \{a : \mathit{Ch}(\{b : \mathit{Ch}(\mathit{val}), b' : \gamma\})\}, \\ k_1 : \{b : \mathit{Ch}(\mathit{val}), b' : \gamma, d_1 : \delta_1\}, \\ k_2 : \{b : \mathit{Ch}(\mathit{val}), b' : \gamma, d_2 : \delta_2\}, \\ c : \mathit{val} \end{cases}$$

A priori le typage le plus général est le plus « petit » c'est-à-dire obtenu à partir de Γ augmenté de variables de types pour pouvoir obtenir les autres typages possibles. Ce qui donnerait :

$$\mathbf{\Gamma} = \begin{cases} k : \{a : \mathit{Ch}(\{b : \mathit{Ch}(\mathit{val}), \rho_{\{b\}}\}), \rho'_{\{a\}}\}, \\ k_1 : \{b : \mathit{Ch}(\mathit{val}), \rho''_{\{b\}}\}, \\ k_2 : \{b : \mathit{Ch}(\mathit{val}), \rho'''_{\{b\}}\}, \\ c : \mathit{val} \end{cases}$$

Malheureusement, il est facile de voir que toutes les instances de $\mathbf{\Gamma}$ ne donnent pas un contexte valide pour P. Par exemple, par la substitution $\mu = [\{b' : \gamma\}/\rho_{\{b\}}, \{\}/\rho'_{\{a\}}, \{\}/\rho''_{\{b\}}, \{\}/\rho'''_{\{b\}}]$, on obtient l'instance de $\mathbf{\Gamma}$ suivante :

$$\mu\mathbf{\Gamma} = \begin{cases} k : \{a : \mathit{Ch}(\{b : \mathit{Ch}(\mathit{val}), b' : \gamma\})\}, \\ k_1 : \{b : \mathit{Ch}(\mathit{val})\}, \\ k_2 : \{b : \mathit{Ch}(\mathit{val})\}, \\ c : \mathit{val} \end{cases}$$

c'est-à-dire un contexte où a attend une localité où un canal b' est disponible avec le type γ, alors que k_1 et k_2 transmis sur a ne le fournissent pas.

Dans le contexte générique d'un terme il manque donc une information permettant de contraindre des types de localités entre eux. Dans l'exemple précédent il s'agit de mettre en relation le type de localité transmis par a avec ceux de k_1 et de k_2. Pour cela, il suffit de revenir a une **relation de sous-typage** explicite entre types de localités. Comme dans [HR98b], nous notons $\psi <: \phi$ si ψ « contient » ϕ ; par exemple, on a $\{a : \gamma, b : \delta, \rho_{\{a,b\}}\} <: \{a : \gamma, \rho'_{\{a\}}\}$. Ainsi, on peut maintenant définir le typage principal d'un terme par la conjonction d'un contexte générique et d'un ensemble d'**assertions de sous-typage**. Plus particulièrement, il est suffisant de n'avoir qu'un ensemble d'assertions de sous-typage **atomiques** c'est-à-dire dont le membre droit de chaque inéquation est réduit à une variable de rangée. Une instance du contexte générique sera un contexte valide pour le terme considéré que si l'instanciation préserve les assertions de sous-typages. Dans l'exemple précédent, si on retient $\mathbf{\Gamma}$ comme contexte générique, on peut définir l'ensemble $\mathcal{A}$ des assertions de sous-typage, par

$$\mathcal{A} = \left\{ \rho''_{\{b\}} <: \rho_{\{b\}} \ , \ \rho'''_{\{b\}} <: \rho_{\{b\}} \right\}$$

Ainsi, la substitution

$$\lambda = [\{b' : \gamma\}/\rho_{\{b\}}, \{\}/\rho'_{\{a\}}, \{b' : \gamma\}/\rho''_{\{b\}}, \{b' : \gamma\}/\rho'''_{\{b\}}]$$

appliquée à $\mathcal{A}$ mène à des assertions de sous-typage encore valides, et en l'appliquant à $\mathbf{\Gamma}$ on obtient Δ qui, comme on l'a vu, est effectivement un contexte valide. En revanche, l'application de μ à $\mathcal{A}$ produit l'assertion $\{\} <: \{b' : \gamma\}$ qui est évidemment incorrecte. Ceci nous permet donc de rejeter μ comme instanciation possible de $\mathbf{\Gamma}$.

Notre algorithme d'inférence de types va donc produire, dans la première étape, non seulement un ensemble d'équations entre schémas de types mais également un ensemble d'inéquations entre schémas de types de localités éventuellement incorrectes. La deuxième étape, l'unification, va donc consister à trouver une solution (si elle existe) aux équations qui rend également les inéquations de sous-typage valides.

10.2.3 Gérer les types dépendants

Nous avons vu que les variables de rangée permettent de contrôler la multiplicité des noms de canaux dans les types de localité et qu'elles sont suffisantes pour les types d'objets. Cependant, les types de localités sont des types dépendants au sens où, contrairement aux types des objets, les labels ne sont pas des constantes pures mais des noms du langage de la même catégorie syntaxique que les noms liés. Nous tachons ici de mettre en évidence les difficultés qu'il en résulte pour l'inférence de types.

Nous avons déjà signalé que les noms liés d'un terme ne peuvent apparaître dans son contexte de typage. Or, lors de la destructuration d'un terme – et donc dans la première étape de l'inférence de types – des noms initialement liés apparaissent plus tard libres et il devient difficile de les distinguer. Considérons par exemple le terme, localisé en ℓ,

$$R = a(b).\mathsf{go}\, k.\overline{b}(c)$$

Celui-ci n'est évidemment pas typable car R reçoit un canal b qu'il est sensé utiliser à sa localité courante ℓ alors qu'il l'utilise à la localité k. Anticipons un peu sur la suite et essayons de produire les contraintes pour ce terme. Il y a deux localités et une valeur, un contexte initial peut donc être : $\mathbf{\Gamma}_0 = k : \rho_\emptyset, \ell : \rho'_\emptyset, c : \mathit{val}$. R étant la réception d'un nom local b sur un canal a, il faut contraindre le type de ℓ à donner un type à a, c'est-à-dire produire la contrainte $\rho'_\emptyset \doteq \{a : \mathit{Ch}(\alpha), \rho''_{\{b\}}\}$. D'autre part, pour le reste du terme (c'est-à-dire $\mathsf{go}\, k.\overline{b}(c)$), le nom b est libre et donc le contexte devient $\mathbf{\Gamma}_1 = k : \rho_\emptyset, \ell : \{b : \alpha, \rho^3_{\{b\}}\}, c : \mathit{val}$. Il faut encore relier le type de la localité de ℓ dans $\mathbf{\Gamma}_1$ avec celui qu'elle avait dans $\mathbf{\Gamma}_0$: $\rho_\emptyset \doteq \rho^3_{\{b\}}$. Ensuite, l'opérateur de migration nous dit simplement que la localité courante devient k. Enfin nous avons l'émission d'une valeur sur le canal b. À cette étape, b nous apparaît comme un nom libre et comme pour a il suffit de produire une contrainte typant b en k : $\rho_\emptyset \doteq \{b : \mathit{Ch}(\mathit{val}), \rho^4_{\{b\}}\}$. On obtient donc l'ensemble des contraintes suivant :

$$\mathcal{C} = \left\{\rho'_\emptyset \doteq \{a : \mathit{Ch}(\alpha), \rho''_{\{b\}}\}, \rho_\emptyset \doteq \rho^3_{\{b\}}, \rho_\emptyset \doteq \{b : \mathit{Ch}(\mathit{val}), \rho^4_{\{b\}}\}\right\}$$

Une substitution solution de $\mathcal{C}$ est par exemple :

$$\mu = [\{b\colon \mathit{Ch}(\mathit{val})\}/\rho_{\emptyset}, \{a\colon \mathit{Ch}(\gamma)\}/\rho'_{\emptyset}, \{\}/\rho''_{\{b\}}, \{b\colon \mathit{Ch}(\mathit{val})\}/\rho^3_{\{b\}}, \{\}/\rho^4_{\{b\}}]$$

où γ est n'importe quel type de canal et l'application de μ au contexte initial nous donne

$$\mu\mathbf{\Gamma} = k\colon \{b\colon \mathit{Ch}(\mathit{val})\}, \ell\colon \{a\colon \mathit{Ch}(\gamma)\}, c\colon \mathit{val}$$

Nous pourrions simplement dire que μ n'est pas valide car elle a dans son image un type contenant un nom lié (b). Malheureusement, on ne peut pas généraliser cette condition car on risque de rejeter des termes qui sont typables. C'est ce que nous essayons de montrer dans les exemples suivants.

Reprenons l'idée du processus P donné plus haut en liant de différentes façons les noms a et b. Soit $\gamma = Ch(\mathit{val})$ et c une valeur, alors le terme

$$S_1 = \ell[\![(\nu b)(\nu a)Q]\!] \quad \text{où} \quad Q = (\overline{a}(\ell) \mid a(k).\mathsf{go}\, k.\overline{b}(c))$$

est typable avec le contexte $\Gamma = \ell\colon \{\}, c\colon \mathit{val}$ car tous les noms de canaux sont abstraits. Q est typable avec le contexte $\Delta = \ell\colon \{a\colon \mathit{Ch}(\{b\colon \gamma\})\}, c\colon \mathit{val}$ et donc a a un type dans lequel il y a une occurrence d'un nom lié. Ceci n'est pas contradictoire avec la constatation que nous avions faite et selon laquelle les noms liés n'apparaissent pas dans le contexte de typage. En effet, du point de vue de Q le nom b est libre et l'application de la règle de typage pour la restriction fait disparaître a (et son type) du contexte de typage, et dans Γ il n'y a plus d'occurrence de b. Remarquons que si on intervertit les restrictions on obtient le terme $\ell[\![(\nu a)(\nu b)Q]\!]$ qui n'est plus typable car le contexte de typage de $(\nu b)Q$ ne doit pas avoir d'occurrence de b alors que le type de a le requiert. Cette exemple nous montre que des noms libres pour Q peuvent comporter dans leur types des noms restreints pour le terme initial (*i.e.* ici pour S_1). Si nous procédions à la simulation de l'inférence de type comme dans le paragraphe précédent, le typage de S_1 conduit, après destructuration de la construction de localité et des abstractions, au typage de Q. On produit alors des contraintes permettant de trouver le type de a et, nécessairement, une substitution solution de ces contraintes aura dans son image un type contenant b. Si on interdit cette substitution comme on l'avait suggéré, on conclue que S_1 n'est pas typable alors qu'il l'est !

Essayons d'autres types de liaisons pour mieux comprendre ces mécanismes. Lions à présent a et b par des abstractions :

$$S_2 = \ell[\![d(b).e(a).Q]\!]$$

Ce terme n'est pas typable car e est un nom libre pour S_2 dont le type est « canal transmettant des noms de même type que a », c'est-à-dire $\mathit{Ch}(\mathit{Ch}(\{b\colon \gamma\}))$. Or, pour S_2, b est un nom lié. Si on intervertit les réceptions de a et de b on obtient le terme $\ell[\![d(a).e(b).Q]\!]$ qui n'est évidemment toujours pas typable.

Enfin, lions b par une abstraction et a par une restriction :

$$S_3 = \ell[\![d(b).(\nu a)Q]\!]$$

Ce terme est typable avec le contexte $\ell\colon \{d\colon \mathit{Ch}(\gamma)\}, c\colon \mathit{val}$ où aucun nom lié n'apparaît. Q étant toujours typable avec le contexte Δ donné précédemment, on conclue que des noms libres pour Q

(ici il s'agit en particulier de a) peuvent comporter dans leur types des noms abstraits pour le terme initial (S_3). Remarquons que ce terme n'est pas typable dans le Dπ original de M. HENNESSY et J. RIELY. En effet, leur calcul est explicitement typé et les noms sont distingués des variables (seules pouvant être abstraites). D'autre part, les types apparaissant dans les termes sont clos c'est-à-dire sans occurrence de variable. Or dans leur calcul le terme S_3 devrait s'écrire

$$\ell[\![d(x:\gamma).(\nu a:\mathit{Ch}(\{x:\gamma\}))(\overline{a}(\ell) \mid a(k).\mathsf{go}\, k.\overline{x}(c))]\!]$$

où a a un type non clos.

La solution que nous proposons pour gérer les noms liés et interdire correctement leur apparition dans les types, consiste à généraliser les indices des variables de rangées à tous les types dans une forme plus forte. Ainsi, on attribue à chaque variable de type α un indice L de sorte que α^L ne peut être substituée que par des types dans lesquels aucun nom de L n'apparait. Par exemple, la substitution $[\mathit{Ch}(\{a:\gamma\})/_{\alpha^{\{a\}}}]$ n'est pas valide, alors que $[\mathit{Ch}(\{b:\gamma\})/_{\alpha^{\{a\}}}]$ l'est si $a \notin \mathit{nm}(\gamma)$. Cette généralisation est plus forte que les indices de variables de rangées dans le sens où ces derniers ne s'intéressent qu'au domaine de la localité substituée : la substitution $[\{b:\mathit{Ch}(\{a:\gamma\})\}/_{\rho_{\{a\}}}]$ est correcte bien que a apparaisse dans le type substitué à $\rho_{\{a\}}$. Par conséquent, ces indices ont des fonctions différentes : ceux des variable de types sont utilisés pour contrôler la multiplicité des noms de canaux alors que leurs généralisations sont utilisés pour gérer les noms liés. Cependant, les indices généralisés peuvent parfois recouvrir la fonction de ceux des variables de rangées. Par exemple, $\{a:\gamma, \rho_{\emptyset}^{\{a\}}\}$ sera considéré comme bien formé car la variable $\rho_{\emptyset}^{\{a\}}$ ne sera jamais substituée à une localité contenant a et donc en particulier à une localité ayant a dans son domaine. Une autre différence est que chaque variable de rangée possède un indice constant alors que les indices généralisés seront dynamiquement mis à jour au cours de l'inférence de types. C'est d'ailleurs pour cette raison que nous n'écrirons pas α^L mais nous utiliserons une relation finie $\mathcal{R}$ de $\mathcal{N}_{chan} \times \mathcal{V}$ (où $\mathcal{V}$ est l'ensemble des variables de types), de sorte que $[\tau/\alpha]$ est correcte si pour tout $a \in \mathit{nm}(\tau), (a,\alpha) \notin \mathcal{R}$.

Si nous reprenons l'exemple du processus R avec une relation $\mathcal{R}$ initialement vide, à la réception de b, $\mathcal{R}$ est mis à jour en associant b à toutes les variables de types connues à cet instant. En particulier, on a $(b, \rho_\emptyset) \in \mathcal{R}$ et donc $\mathcal{R}$ invalide la substitution μ.

Essayons de décrire informellement ce mécanisme pour l'exemple plus significatif du terme S_3. Lorsque l'algorithme est à l'étape de typer $(\nu a)Q$, la relation $\mathcal{R}$ associe b à toutes les variables de types connues à cet instant. La restriction de a nous fait alors mettre à jour le contexte de typage pour que celui-ci type a avec une nouvelle variable de type α inconnue de $\mathcal{R}$. En particulier, $(b,\alpha) \notin \mathcal{R}$ et donc on pourra librement substituer le type effectif $\mathit{Ch}(\{b:\gamma\})$ de a à α.

On peut également voir la relation $\mathcal{R}$ comme une trace d'exécution de l'algorithme : si une variable de type α est associée à davantage de noms qu'une variable α' dans $\mathcal{R}$, ceci signifie que α' a été introduite « après » α.

CHAPITRE 11

Unification

CONTRAIREMENT À L'ORDRE DONNÉ DANS LE CHAPITRE PRÉCÉDENT, nous commençons par décrire l'algorithme d'unification. En effet, celui-ci nous permet d'introduire tous les concepts nécessaires à la génération des contraintes et de s'y familiariser. Nous définissons d'abord formellement la notion de relation de liaison ainsi que les types et leur bonne formation. Nous décrivons ensuite le type de problème d'unification par lequel nous sommes concerné. Dans la section suivante nous donnons une série de propriétés relatives aux substitutions et aux relations de liaisons. Enfin, nous exposons l'algorithme d'unification et établissons sa correction et sa complétude.

11.1 Quelques définitions

Soit $\mathcal{V} = \{\alpha, \alpha', \ldots\}$ un ensemble dénombrable de variables de types partitionné en trois sous-ensembles : l'ensemble des variables de type générales $\mathcal{V}_{types} = \{t, t', \ldots\}$, l'ensemble des variables de type de canaux $\mathcal{V}_{chan} = \{h, h', \ldots\}$ et l'ensemble des variables de type de localités (aussi appelées **variables de rangées**) $\mathcal{V}_{loc} = \{\rho_L, \rho'_L, \ldots\}$ où $L \in \mathcal{P}_{fin}(\mathcal{N}_{chan})$. Les **schémas de types** ont la forme suivante :

types	$\tau, \sigma, \ldots$	$::=$	$\psi \mid \gamma \mid \gamma@\psi \mid \mathit{val} \mid t$
types de canaux	$\gamma, \delta, \ldots$	$::=$	$\mathit{Ch}(\tau) \mid h$
types de localités	$\psi, \phi, \ldots$	$::=$	$\{a : \gamma, \psi\} \mid \{\} \mid \rho_L$

On dénote par $\mathit{var}(\tau)$ l'ensemble des variables de type ayant une occurrence dans τ et, comme dans le chapitre 7, on note $\tau, \gamma, \psi, \ldots$ les **types de base**, c'est-à-dire les types τ tels que $\mathit{var}(\tau) = \emptyset$.

On note $nm(\tau)$ l'ensemble des noms ayant une occurrence dans τ et défini formellement par :

$$nm(\alpha) = nm(val) = nm(\{\}) = \emptyset$$
$$nm(Ch(\tau)) = nm(\tau)$$
$$nm(\gamma@\psi) = nm(\gamma) \cup nm(\psi)$$
$$nm(\{a : \gamma, \psi\}) = \{a\} \cup nm(\gamma) \cup nm(\psi)$$

On écrira souvent simplement $\{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$ et $\{a_1 : \gamma_1, \ldots, a_n : \gamma_n\}$ pour les type de localité $\{a_1 : \gamma_1, \{\ldots, \{a_n : \gamma_n, \rho_L\}\}\ldots\}$ et $\{a_1 : \gamma_1, \{\ldots, \{a_n : \gamma_n, \{\}\}\}\ldots\}$. Le premier, comportant une variable de rangée, sera parfois qualifié de **type de localité extensible**. La variable de rangée d'un type de localité est obtenu grâce à la fonction partielle ρvar définie formellement par $\rho var(\rho_L) = \rho_L$ et $\rho var(\{a : \gamma, \psi\}) = \rho var(\psi)$. On dénote également l'ensemble des noms de canaux typés par un type de localité par $dom(\rho_L) = dom(\{\}) = \emptyset$ et $dom(a : \gamma, \psi) = \{a\} \cup dom(\psi)$.

Une **relation de liaison** est une partie finie de $\mathcal{N}_{chan} \times \mathcal{V}$. L'idée de cette relation est d'interdire la substitution d'un type ayant une occurrence d'un nom a à une variable α si $(a, \alpha) \in \mathcal{R}$. On note $\mathcal{R}(\alpha)$ l'ensemble $\{a \mid (a, \alpha) \in \mathcal{R}\}$ et $im(\mathcal{R})$ l'ensemble $\{a \mid \exists \alpha.(a, \alpha) \in \mathcal{R}\}$.

Tous les types décrits par la grammaire précédente ne sont pas des types valides. Comme pour les types de record nous devons être vigilant quant aux typages multiples d'un même canal dans un type de localité ; en d'autres termes, dans $\{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$ les noms $a_1, \ldots, a_n$ doivent être distincts deux à deux. Dans le chapitre 7, il suffisait de faire cette vérification une fois pour toute. Ici les types de localité sont extensibles et peuvent être amenés à typer de nouveaux noms de canaux lorsqu'un type de localité est substitué à leur variable de rangée. L'indice L de la variable de rangée a pour mission d'éviter que celle-ci soit substituée par un type de localité typant un nom déjà présent : c'est la « mémoire des noms interdits ». Cependant, une relation de liaison peut parfois remplir cette fonction comme nous l'avons vu dans le chapitre précédent, c'est pourquoi la bonne formation des types dépend aussi de cette relation. On dira que τ est bien formé pour $\mathcal{R}$, s'il existe une preuve du jugement $\tau :: \mathcal{R}$ dans le système d'inférence suivant :

$$\frac{}{val :: \mathcal{R}} \qquad \frac{}{t :: \mathcal{R}} \qquad \frac{}{h :: \mathcal{R}} \qquad \frac{\tau :: \mathcal{R}}{Ch(\tau) :: \mathcal{R}} \qquad \frac{\gamma :: \mathcal{R} \quad , \quad \psi :: (L, \mathcal{R})}{\gamma@\psi :: \mathcal{R}}$$

$$\frac{\psi :: (L, \mathcal{R})}{\psi :: \mathcal{R}} \qquad \frac{}{\{\} :: (L, \mathcal{R})} \qquad \frac{\gamma :: \mathcal{R} \quad , \quad \psi :: (L \uplus \{a\}, \mathcal{R})}{\{a : \gamma, \psi\} :: (L, \mathcal{R})} \qquad \frac{L' = L \cup \mathcal{R}(\rho_L)}{\rho_L :: (L', \mathcal{R})}$$

où on note $\uplus$ l'union disjointe et on rappelle que $\mathcal{R}(\rho_L) = \{a \mid (a, \rho_L) \in \mathcal{R}\}$. On dira simplement qu'un type τ est bien formé s'il existe une relation de liaison $\mathcal{R}$ telle que $\tau :: \mathcal{R}$.

Exemple 11.1.1 Soit $\gamma = Ch(val)$ et $\mathcal{R}$ la relation $\{(a, \rho_{\{b,c\}})\}$, alors le type

$$\psi = \{a : \gamma, b : \gamma, c : \gamma, \rho_{\{b,c\}}\}$$

est bien formé pour $\mathcal{R}$. En effet, il est clair que $\gamma :: \mathcal{R}$ et on peut montrer que $\psi :: (\emptyset, \mathcal{R})$:

$$\dfrac{\dfrac{}{\gamma :: \mathcal{R}} \quad \dfrac{\{a,b,c\} = \{b,c\} \cup \{a\}}{\rho_{\{b,c\}} :: (\{a,b,c\}, \mathcal{R})}}{\dfrac{\dfrac{}{\gamma :: \mathcal{R}} \quad \{c : \gamma, \rho_{\{b,c\}}\}\} :: (\{a,b\}, \mathcal{R})}{\dfrac{\dfrac{}{\gamma :: \mathcal{R}} \quad \{b : \gamma, \{c : \gamma, \rho_{\{b,c\}}\}\} :: (\{a\}, \mathcal{R})}{\{a : \gamma, \{b : \gamma, \{c : \gamma, \rho_{\{b,c\}}\}\}\} :: (\emptyset, \mathcal{R})}}}$$

□

Remarquons que tout type de base est bien formé pour toute relation de liaison dès lors que les types de localité apparaissant dans ce type n'assignent pas plusieurs types à un même nom de canal. Ainsi, les types définis dans le chapitre 7 sont bien formés pour le système décrit ici.

Il convient enfin de définir la notion de substitution. Une substitution, notée $\mu, \lambda, \ldots$ est une application finie des variables de types dans les schémas de types. Si

$$\mu = [\tau_1/\alpha_1, \ldots, \tau_n/\alpha_n]$$

on dénote par $\mathit{dom}(\mu)$ l'ensemble $\{\alpha_1, \ldots, \alpha_n\}$ et $\mathit{vrang}(\mu)$ l'ensemble $\bigcup_{i \in \{1,\ldots,n\}} \mathit{var}(\tau_i)$. On appelle **substitution de base** les substitutions μ telles que $\mathit{vrang}(\mu) = \emptyset$. Toutes les substitutions considérées sont supposées être idempotentes, et, plus particulièrement $\mathit{dom}(\mu) \cap \mathit{vrang}(\mu) = \emptyset$. On fait également l'hypothèse que les substitutions respectent la grammaire des types : par exemple la substitution d'une variable de type de canal est un type de canal. En dehors de son domaine, une substitution est supposée être l'identité. Les substitutions sont trivialement étendues en des morphismes sur les types. L'extension des substitutions sur les autres objets que nous considérons dans la suite (tels que les séquents, les assertions de sous-typage, etc.) sont laissées informelles car elles sont très classiques. On note $\lambda\mu$ la composition des substitutions λ et μ, et $\emptyset$ la substitution vide. On remarquera qu'ainsi définie, la substitution ne préserve pas nécessairement la bonne formation des types. Par conséquent, une substitution préserve la bonne formation d'un type si elle satisfait certaines propriétés vis-à-vis de la relation pour laquelle les types considérés sont bien formés.

Définition 11.1.2 *Soit $\mathcal{R}$ une relation de liaison, on dit qu'une substitution μ* respecte *$\mathcal{R}$ si*

1. $\forall \alpha \in \mathit{dom}(\mu)$, $\mathit{nm}(\mu\alpha) \cap \mathcal{R}(\alpha) = \emptyset$ *et* $\mu\alpha :: \mathcal{R}$ *;*
2. $\forall \rho_L \in \mathit{dom}(\mu)$, $\mathit{dom}(\mu\rho_L) \cap L = \emptyset$, *et si* $\rho\mathit{var}(\mu\rho_L) = \rho'_{L'}$, *alors* $\mathcal{R}(\rho_L) \subseteq \mathcal{R}(\rho'_{L'})$ *et* $L \subseteq L' \cup \mathcal{R}(\rho'_{L'})$.

Le premier point de cette définition nous dit qu'une substitution μ qui respecte $\mathcal{R}$ ne doit pas remplacer une variable α par un type contenant un nom a si justement $\mathcal{R}$ interdit a pour α (*i.e.* si $a \in \mathcal{R}(\alpha)$). Aussi, les types introduits doivent évidemment être bien formés pour $\mathcal{R}$.

Le deuxième point permet de conserver la propriété de bonne formation des types de localités après substitution. L'exemple suivant donne un idée sur la nécessité de ces contraintes.

Exemple 11.1.3 Soit $\psi = \{a : \gamma, b : \gamma', \rho_{\{b\}}\}$ et $\mathcal{R}$ la relation de liaison $\{(a, \rho_{\{b\}})\}$, alors il est clair que $\psi :: \mathcal{R}$. Considérons la substitution $\mu = [\{c : \delta, \rho'_{\{b,c\}}\}/\rho_{\{b\}}]$, alors

$$\mu\psi = \{a : \gamma, b : \gamma', c : \delta, \rho'_{\{b,c\}}\}$$

mais $\mu\psi$ n'est plus bien formé car $\mathcal{R}(\rho'_{\{b,c\}}) = \emptyset$ et $a \in \mathit{dom}(\mu\psi)$. En effet, si μ satisfait effectivement le premier point de la définition 11.1.2, par contre elle ne satisfait pas la contrainte $\mathcal{R}(\rho_{\{b\}}) \subseteq \mathcal{R}(\rho'_{\{b,c\}})$ du deuxième point. Considérons à présent la relation $\mathcal{R}' = \{(a, \rho_{\{b\}}), (a, \rho'_{\{c\}})\}$, alors on a bien sur toujours $\psi :: \mathcal{R}'$. La substitution $\lambda = [\{c : \delta, \rho'_{\{c\}}\}/\rho_{\{b\}}]$ appliquée à ψ donne

$$\lambda\psi = \{a : \gamma, b : \gamma', c : \delta, \rho'_{\{c\}}\}$$

qui n'est plus bien formé car $b \in \mathit{dom}(\mu\psi)$ et $\mathcal{R}(\rho'_{\{c\}}) = \{a\}$. En effet, λ satisfait bien $\mathcal{R}(\rho'_{\{c\}}) \subseteq \mathcal{R}(\rho_{\{b\}})$ mais $\{c\} \not\subseteq \{b\} \cup \mathcal{R}(\rho_{\{b\}})$. □

Nous montrons dans la section 11.3 qu'une substitution qui respecte $\mathcal{R}$ préserve la bonne formation des types pour $\mathcal{R}$.

11.2 Sur le problème d'unification

Le problème classique de l'unification est : étant donné un ensemble d'équations entre des schémas de types, existe-t-il une substitution des variables de types qui satisfait les équations ? Comme dans le cas de l'unification de types de records, nous sommes concernés par un type d'unification dit « **unification équationnelle** », c'est-à-dire que l'égalité syntaxique est remplacée par l'égalité modulo une théorie équationnelle définie par un ensemble d'identités E. Cette théorie équationnelle est la plus petite relation de congruence sur les schémas de types qui est close par substitution et qui contient E. Pour ce qui nous intéresse, cet ensemble est réduit à une seule équation spécifiant que l'ordre des champs d'un type de localité n'a pas d'importance. Par exemple, $\{a : \gamma, b : \delta, \rho_{\{a,b\}}\}$ et $\{b : \delta, a : \gamma, \rho_{\{a,b\}}\}$, bien que syntaxiquement différents, désignent le même type de localité. Par conséquent, l'équation E qui nous intéresse est :

$$\{a : h, \{b : h', \rho_L\}\} \quad =_E \quad \{b : h', \{a : h, \rho_L\}\}$$

À partir de maintenant toutes les égalités de types sont supposées être modulo E.

Définition 11.2.1 (Contraintes de Types) *Une* contrainte de types *est un ensemble $\mathcal{C}$ d'équations entre des schémas de types*

$$\{\tau_1 \doteq \sigma_1, \ldots, \tau_n \doteq \sigma_n\}$$

On dit qu'une substitution μ est une solution *de $\mathcal{C}$ (ou un* unificateur *pour $\mathcal{C}$) si elle unifie toute paire de types dans $\mathcal{C}$, c'est-à-dire telle que $\mu\tau_i = \mu\sigma_i$ pour tout $i \in \{1, \ldots, n\}$. Enfin on note $\mathcal{C} :: \mathcal{R}$ si $\forall i \in \{1, \ldots, n\}, \tau_i :: \mathcal{R}$ et $\sigma_i :: \mathcal{R}$.*

Dans les théories syntaxiques, suivant la terminologie de [Rob65], dès que deux types τ et σ sont unifiables, il existe un **unificateur principal** (ou **unificateur le plus général**), c'est-à-dire une substitution μ telle que pour toute solution λ de $\tau \doteq \sigma$, λ est une instance de μ (*i.e.* il existe un

substitution μ' telle que $\lambda = \mu'\mu$). De manière générale, dans les théories équationnelles un unificateur principal n'existe pas toujours comme le montre l'exemple suivant (voir aussi par exemple [BS99] pour en savoir davantage sur la théorie de l'unification).

Exemple 11.2.2 Soit f une fonction binaire et E_c l'équation

$$f(x,y) \quad =_{E_c} \quad f(y,x)$$

établissant la commutativité de f. Dans cette théorie équationnelle, la contrainte $f(x,y) \doteq f(a,b)$ où a et b sont des symboles de constantes, a deux solutions $\mu_1 = [a/x, b/y]$ et $\mu_2 = [a/y, b/x]$. Cependant, il n'existe pas de substitution μ' telle que $\mu_1 = \mu'\mu_2$ ou $\mu_2 = \mu'\mu_1$. □

Néanmoins, il est possible de définir un préordre sur les substitutions permettant d'approcher une définition d'unificateur principal. On note $\mu =_{\mathcal{X}} \lambda$ où $\mathcal{X}$ est un ensemble de variables de types, si $\mu\alpha = \lambda\alpha$ pour tout $\alpha \in \mathcal{X}$, et on dit, comme dans [JM93], que μ **est égal à** λ **sur** $\mathcal{X}$.

Définition 11.2.3 *Soient μ et λ des substitutions et $\mathcal{X}$ un ensemble de variables de types, μ est dit* plus général que *λ sur $\mathcal{X}$ si et seulement s'il existe une substitution μ' telle que $\lambda =_{\mathcal{X}} \mu'\mu$. Dans ce cas on écrit $\mu \leq_{\mathcal{X}} \lambda$.*

Étant donné une contrainte de types $\mathcal{C}$ et un ensemble de variables de types $\mathcal{X}$ tels que $var(\mathcal{C}) \subseteq \mathcal{X}$, on ne considère plus une solution la plus générale mais un ensemble minimal complet de solutions $\mathcal{S}_\mathcal{C}$ satisfaisant :

1. pour toute solution λ de $\mathcal{C}$, il existe $\mu \in \mathcal{S}_C$ telle que $\mu \leq_{\mathcal{X}} \lambda$;
2. les éléments de $\mathcal{S}_\mathcal{C}$ sont incomparables vis-à-vis de $\leq_{\mathcal{X}}$.

On dit qu'une contrainte de types $\mathcal{C}$ est **unitaire** si et seulement si elle a un ensemble minimal complet de solutions de cardinalité 1. Une théorie équationnelle est dite **unitaire** si pour toute contrainte $\mathcal{C}$ dans cette théorie, tous les ensembles minimaux complets de solutions de $\mathcal{C}$ sont de cardinalité au plus 1. Dans [Rém93b, Rém93a], D. RÉMY établit la propriété que dans l'algèbre des types de records, l'unification est décidable et unitaire. Nos types de localité sont syntaxiquement identiques aux types des records et nous avons la même équation E, par conséquent ce résultat s'applique également dans notre cas.

On introduit la relation $\psi <: \phi$ sur les types de localités (similaire à celle de [HR98b]) qui signifie : « ϕ type des canaux également typés dans ψ avec les même types ». Celle-ci peut être vue comme l'inverse de la relation d'inclusion. Plus formellement,

Définition 11.2.4 *On définie la relation $<:$ sur les types de localités comme suit :*

- $\psi <: \{\}$;
- $\psi <: \rho_L$;
- $\{a : \gamma, \psi\} <: \{a : \gamma, \phi\}$ *si* $\psi <: \phi$.

Cette relation définie des assertions de sous-typage. *Les assertions de sous-typage de la forme $\psi <: \rho_L$ sont dites* atomiques. *De plus, on note $\psi \equiv \phi$ si $\psi <: \phi$ et $\phi <: \psi$.*

Dans [PS93] et [HR98b], le caractère asymétrique de la communication est mis en évidence par des arguments de sous-typage. Dans la communication de localités, ceci signifie que la réception contraint le type du canal sujet de la communication. Par exemple, si le corps P d'un processus $a(\ell).P$ utilise les canaux $a_1, \ldots, a_n$ à la localité ℓ, alors a doit avoir le type $Ch(\psi)$ où ψ fournie au moins un type pour chaque a_i. Ainsi, dans P, ℓ a un type ϕ tel que $\psi <: \phi$. Inversement, dans l'émission d'une localité k sur a, le type ϕ' de k doit assigner à chaque a_i un type identique à celui assigné par ψ ; c'est-à-dire $\phi' <: \psi$.

Définition 11.2.5 *Soit $\mathcal{A}$ un ensemble d'assertions de sous-typage atomiques, on dit que μ est une $\mathcal{A}$-*substitution *si elle préserve les assertions de $\mathcal{A}$.*

Puisque nous sommes à la recherche d'un typage principal pour les termes de notre calcul réparti, nous avons également affaire à des contraintes de sous-types dont la définition est la suivante.

Définition 11.2.6 (Contraintes de sous-types) *Une* contrainte de sous-types *est un ensemble $\mathcal{I}$ d'inéquations de sous-typage entre types de localités*

$$\{\psi_1 \lessdot \phi_1, \ldots, \psi_n \lessdot \phi_n\}$$

On dit qu'une substitution μ est solution *de $\mathcal{I}$ si elle valide les inéquations de $\mathcal{I}$, c'est-à-dire telle que $\mu\psi_i <: \mu\phi_i$ pour tout $i \in \{1, \ldots, n\}$. On utilise la notation $\psi \doteq \phi$ comme abréviation pour $\psi \lessdot \phi \wedge \phi \lessdot \psi$. Enfin on note $\mathcal{I} :: \mathcal{R}$ si pour tout $i \in \{1, \ldots, n\}, \psi_i :: \mathcal{R}$ et $\phi_i :: \mathcal{R}$.*

Exemple 11.2.7 La contrainte de sous-types $\mathcal{I} = \{a : \gamma, \rho_{\{a\}}\} \lessdot \{b : \delta, \rho'_{\{b\}}\}$ a par exemple pour solution

$$\mu = [\{\}/\rho'_{\{b\}}, \{b : \delta\}/\rho_{\{a\}}]$$

car $\mu\mathcal{I} = \{\{a : \gamma, b : \delta\} <: \{b : \delta\}\}$. □

Les contraintes de types et de sous-types que nous sommes intéressés de résoudre sont maîtrisées par une relation de liaison. En effet, comme nous l'avons informellement décrit dans le chapitre 10, notre algorithme d'inférence de types produit, dans une première étape, un ensemble d'équations et d'inéquations de types mais aussi une relation de liaison pour gérer les noms liés rencontrés au cours de cette étape.

Définition 11.2.8 (Contraintes) *On appelle* contrainte *un triplet $(\mathcal{C}, \mathcal{I})_{\mathcal{R}}$ où $\mathcal{R}$ une relation de liaison, $\mathcal{C}$ est une contrainte de types telle que $\mathcal{C} :: \mathcal{R}$ et $\mathcal{I}$ une contrainte de sous-types telle que $\mathcal{I} :: \mathcal{R}$.*

Enfin, il convient de définir la notion de **solution principale** d'une contrainte.

Définition 11.2.9 (Solution principale) *Une substitution μ est dite* solution principale de $(\mathcal{C}, \mathcal{I})_{\mathcal{R}}$ sur $\mathcal{X}$*, si pour toute substitution de base λ qui respecte $\mathcal{R}$ qui est solution de $\mathcal{C}$ et de $\mathcal{I}$, on a $\mu \leq_{\mathcal{X}} \lambda$.*

11.3 Substitutions et relations de liaison

Dans cette section nous donnons une série de propriétés sur les substitutions et leur relations avec la bonne formation des types. Les deux lemmes suivant établissent que les substitutions qui respectent une relation $\mathcal{R}$ préservent la bonne formation des types pour $\mathcal{R}$.

Lemme 11.3.1 *Si μ respecte $\mathcal{R}$, $\rho_L :: (L_1, \mathcal{R})$ et $\mu\rho_L$ est extensible, alors $\mu\rho_L :: (L_2, \mathcal{R})$ avec $L_1 \subseteq L_2$.*

Preuve: De $\rho_L :: (L_1, \mathcal{R})$ on déduit que $L_1 = \mathcal{R}(\rho_L) \cap L$ (1), et de μ respecte $\mathcal{R}$, on a $L \subseteq L' \cup \rho var(\rho'_{L'})$ et $\mathcal{R}(\rho_L) \subseteq \mathcal{R}(\rho'_{L'})$. Et donc, par (1), $L_1 \subseteq L' \cup \mathcal{R}(\rho'_{L'})$ (2). On peut facilement vérifier que de $\mu\rho_L :: (L_2, \mathcal{R})$ on a $\rho'_{L'} :: (L_2 \uplus dom(\mu\rho_L), \mathcal{R})$ et donc $L_2 \uplus dom(\mu\rho_L) = L' \cup \mathcal{R}(\rho'_{L'})$. Avec (2), on déduit que $L_1 \subseteq L_2 \uplus dom(\mu\rho_L)$. Or, pour tout $a \in dom(\mu\rho_L)$, on a $a \notin L$ et $a \notin \mathcal{R}(\rho_L)$ car μ respecte $\mathcal{R}$, et donc finalement $L_1 \subseteq L_2$. □

Lemme 11.3.2 *Si $\tau :: \mathcal{R}$ et μ respecte $\mathcal{R}$ alors $\mu\tau :: \mathcal{R}$.*

Preuve: On montre plus précisément les deux points suivants :

- Si $\tau :: \mathcal{R}$ et μ respecte $\mathcal{R}$ alors $\mu\tau :: \mathcal{R}$
- si $\psi :: (L_1, \mathcal{R})$ et μ respecte $\mathcal{R}$ alors il existe L_2 tel que $\mu\psi :: (L_2, \mathcal{R})$ avec $L_1 \subseteq L_2$.

On procède par induction sur la hauteur des preuves de $\tau :: \mathcal{R}$ et de $\psi :: (L_1, \mathcal{R})$. Les preuves de hauteur 1 correspondent aux cas où τ est *val*, h ou t et au cas où ψ est une variable ρ_L ou $\{\}$ et sont directement montrés par le fait que μ respecte $\mathcal{R}$ et par le lemme 11.3.1. Supposons que ce lemme soit vrai pour tout prédicat pouvant se montrer par une preuve de hauteur au plus n. Dans le cas où τ est $Ch(\sigma)$ ou $\gamma@\phi$ ou ψ et $\tau :: \mathcal{R}$ se montre par une preuve de hauteur $n+1$, on applique simplement l'hypothèse d'induction. Supposons que $\psi = \{a : \gamma, \phi\}$ et $\{a : \gamma, \phi\} :: (L_1, \mathcal{R})$ se montre par une preuve de hauteur $n+1$, alors $\gamma :: \mathcal{R}$ et $\phi :: (L_1 \uplus \{a\}, \mathcal{R})$ où $a \notin L_1$ se montrent par des preuves de hauteur au plus n. Donc, par hypothèse d'induction, $\mu\gamma :: \mathcal{R}$ et $\mu\phi :: (L_3, \mathcal{R})$ avec $L_1 \uplus \{a\} \subseteq L_3$. Soit $L_2 = L_3 - a$, on a donc $\mu\phi :: (L_2 \uplus \{a\}, \mathcal{R})$ et on peut conclure que $\mu\psi :: (L_2, \mathcal{R})$ avec $L_1 \subseteq L_2$. □

Si un type est bien formé pour une relation $\mathcal{R}$ alors il est bien formé pour toute relation de liaison qui contient $\mathcal{R}$.

Lemme 11.3.3 *Si $\mathcal{R} \subseteq \mathcal{R}'$ et $\tau :: \mathcal{R}$, alors $\tau :: \mathcal{R}'$.*

Preuve: On montre plus précisément que :

- si $\tau :: \mathcal{R}$ alors $\tau :: \mathcal{R}'$;
- si $\psi :: (L_1, \mathcal{R})$, alors il existe $\psi :: (L_2, \mathcal{R}')$ avec $L_1 \subseteq L_2$.

et on procède par induction comme dans la preuve du point précédent. Regardons le cas des types de localités. Si $\psi = \rho_L$, alors $L_1 = L \cup \mathcal{R}(\rho_L)$ et il suffit de définir L_2 par $L_2 = L \cup \mathcal{R}'(\rho_L)$ qui satisfait bien $L_1 \subseteq L_2$. Si $\psi = \{a : \gamma, \phi\} :: (L_1, \mathcal{R})$, alors $\gamma :: \mathcal{R}$ et $\phi :: (L_1 \uplus \{a\}, \mathcal{R})$ et par hypothèse d'induction $\gamma :: \mathcal{R}$ et il existe L_3 tel que $\phi :: (L_3, \mathcal{R}')$ avec $L_1 \uplus \{a\} \subseteq L_3$. Donc $L_3 = L_2 \uplus \{a\}$ et par conséquent $\{a : \gamma, \phi\} :: (L_2, \mathcal{R}')$ avec $L_1 \subseteq L_2$. □

Si λ et μ sont des substitutions qui respectent $\mathcal{R}$ malheureusement la composition $\lambda\mu$ de ces substitutions ne respecte pas nécessairement $\mathcal{R}$ comme le montre l'exemple suivant.

Exemple 11.3.4 Soient $\mu = [\rho_\emptyset/t]$ et $\lambda = [\{a : \gamma, \rho'_{\{a\}}\}/\rho_\emptyset]$, alors ces deux substitutions respectent la relation $\mathcal{R} = \{(a,t)\}$. Cependant, leur composition $\lambda\mu = [\{a : \gamma, \rho'_{\{a\}}\}/\rho_\emptyset, \{a : \gamma, \rho'_{\{a\}}\}/t]$ ne respecte pas $\mathcal{R}$ car $a \in \mathit{nm}(\lambda\mu t)$ alors que $a \in \mathcal{R}(t)$. □

Pour que la composition de substitutions continue de respecter une relation de liaison il faut que celle-ci satisfasse certains critères vis-à-vis de la substitution appliquée en premier. Dans l'exemple précédent, si la composition ne respecte pas $\mathcal{R}$ c'est parce que μ assigne une variable $\rho_\emptyset$ à t alors que $\mathcal{R}$ « interdit » des noms à t qui ne le sont pas pour $\rho_\emptyset$ (ici a). En revanche, si $\mu\alpha = \tau$ et que pour toute variable α' de τ on a $\mathcal{R}(\alpha) \subseteq \mathcal{R}(\alpha')$, alors la composition d'une substitution λ qui respecte $\mathcal{R}$ avec μ respectera $\mathcal{R}$.

Définition 11.3.5 *On dit qu'une relation $\mathcal{R}$ est μ-*close *si*

$$\forall \alpha \in \mathit{dom}(\mu)\, \forall \alpha' \in \mathit{var}(\mu\alpha)\, .\mathcal{R}(\alpha) \subseteq \mathcal{R}(\alpha')$$

Lemme 11.3.6 *Soit μ une substitution qui respecte $\mathcal{R}$ et $\mathcal{R}$ est μ-close alors pour toute substitution λ qui respecte $\mathcal{R}$, $\lambda\mu$ respecte $\mathcal{R}$.*

Preuve: Soient $\mu = [\tau_1/\alpha_1 \dots \tau_n/\alpha_n]$ et $\lambda = [\sigma_1/\alpha'_1 \dots \sigma_m/\alpha'_m]$, et supposons, sans perte de généralité, que $\{\alpha_1, \dots, \alpha_n\} \cap \{\alpha'_1, \dots, \alpha'_m\} = \emptyset$. On a donc

$$\lambda\mu = [\sigma_1/\alpha'_1 \dots \sigma_m/\alpha'_m, \lambda\tau_1/\alpha_1 \dots \lambda\tau_n/\alpha_n]$$

Pour montrer que $\lambda\mu$ respecte $\mathcal{R}$ il faut montrer que pour tout $i \in \{1, \dots, n\}$ on a $\mathit{nm}(\lambda\tau_i) \cap \mathcal{R}(\alpha_i) = \emptyset$ (1). On a $\mathit{nm}(\lambda\tau_i) = \mathit{nm}(\tau_i) \cup \bigcup_{\alpha \in \mathit{var}(\tau_i) \cap \mathit{dom}(\lambda)} \mathit{nm}(\lambda\alpha)$. Puisque μ respecte $\mathcal{R}$, on a déjà $\mathit{nm}(\tau_i) \cap \mathcal{R}(\alpha_i) = \emptyset$. Soit $\alpha \in \mathit{var}(\tau_i) \cap \mathit{dom}(\lambda)$, λ respectant $\mathcal{R}$, on en déduit que $\mathit{nm}(\lambda\alpha) \cap \mathcal{R}(\alpha) = \emptyset$ et de $\mathcal{R}$ μ-close, on a $\mathcal{R}(\alpha_i) \subseteq \mathcal{R}(\alpha)$ et donc $\mathit{nm}(\lambda\alpha) \cap \mathcal{R}(\alpha_i) = \emptyset$ ce qui montre (1). D'autre part, il clair, par le lemme 11.3.2 et le fait que λ et μ respectent $\mathcal{R}$, que pour tout $\alpha \in \mathit{dom}(\lambda\mu)$ on a $\lambda\mu\alpha :: \mathcal{R}$. Enfin, pour montrer le point 2 de la définition 11.1.2, soit $\rho_L \in \mathit{dom}(\lambda\mu)$ avec $\rho_L \in \mathit{dom}(\mu)$ et $\rho\mathit{var}(\mu\rho_L) = \rho'_{L'}$, et de μ respecte $\mathcal{R}$, on déduit que $\mathcal{R}(\rho_L) \subseteq \mathcal{R}(\rho'_{L'})$ (2). Alors, soit $\rho'_{L'} \notin \mathit{dom}(\lambda)$ et donc $\rho\mathit{var}(\lambda\mu\rho_L) = \rho'_{L'}$ et c'est fini ; soit $\rho_{L'} \in \mathit{dom}(\lambda)$ et de λ respecte $\mathcal{R}$ on a $\mathcal{R}(\rho_{L'}) \subseteq \mathcal{R}(\rho\mathit{var}(\lambda\rho'_{L'}))$. Or $\rho\mathit{var}(\lambda\rho'_{L'}) = \rho\mathit{var}(\lambda\mu\rho_L)$ et donc, avec (2) on conclue que $\mathcal{R}(\rho_L) \subseteq \mathcal{R}(\rho\mathit{var}(\lambda\mu\rho_L))$. Dans le cas où $\rho_L \in \mathit{dom}(\lambda)$ on conclue directement par le fait que λ respecte $\mathcal{R}$. Montrons que si $\rho_L \in \mathit{dom}(\mu)$ alors $\mathit{dom}(\lambda\mu\rho_L) \cap L = \emptyset$ (3). Soit $\rho'_{L'} = \rho\mathit{var}(\mu\rho_L)$, on a donc $\mathit{dom}(\lambda\mu\rho_L) = \mathit{dom}(\mu\rho_L) \cup \mathit{dom}(\lambda\rho'_{L'})$. De μ respecte $\mathcal{R}$, on a $\mathit{dom}(\mu\rho_L) \cap L = \emptyset$ et $L \subseteq L' \cup \mathcal{R}(\rho'_{L'})$ (4), et de λ respecte $\mathcal{R}$, on a $\mathit{dom}(\lambda\rho'_{L'}) \cap L' = \emptyset$. De plus, $\mathit{dom}(\lambda\rho'_{L'}) \subseteq \mathit{nm}(\lambda\rho'_{L'})$, et $\mathit{nm}(\lambda\rho'_{L'}) \cap \mathcal{R}(\rho'_{L'}) = \emptyset$, et donc $\mathit{dom}(\lambda\rho'_{L'}) \cap (L' \cup \mathcal{R}(\rho'_{L'})) = \emptyset$. Et, avec (4), finalement on a montré (3). Montrons enfin que si $\rho''_{L''} = \rho\mathit{var}(\lambda\mu\rho_L)$, on a $L \subseteq L'' \cup \mathcal{R}(\rho''_{L''})$ (5). Soit $\rho'_{L'} = \rho\mathit{var}(\mu\rho_L)$, de μ respecte $\mathcal{R}$, on a $L \subseteq L' \cup \mathcal{R}(\rho'_{L'})$, et de λ respecte $\mathcal{R}$, on a $L' \subseteq L'' \cup \mathcal{R}(\rho''_{L''})$ et $\mathcal{R}(\rho'_{L'}) \subseteq \mathcal{R}(\rho''_{L''})$ et donc on a (5). □

Étant donné une relation $\mathcal{R}$ et une substitution μ, il est facile d'étendre $\mathcal{R}$ afin que celle-ci soit μ-close.

Définition 11.3.7 *Soit $\mathcal{R}$ une relation de liaison et μ une substitution qui respecte $\mathcal{R}$, on appelle μ-clôture de $\mathcal{R}$ (ou* clôture de $\mathcal{R}$ par μ*) la relation $\mathcal{R} \cup \bigcup_{\alpha \in dom(\mu)} \bigcup_{\alpha' \in var(\mu\alpha)} \mathcal{R}(\alpha) \times \{\alpha'\}$.*

Remarque 11.3.8

- La μ-clôture d'une relation $\mathcal{R}$ est μ-close et c'est la plus petite relation μ-close contenant $\mathcal{R}$.
- Si μ respecte $\mathcal{R}$ et $\mathcal{R}'$ est la μ-clôture de $\mathcal{R}$, alors μ respecte $\mathcal{R}'$. Le premier point de la définition 11.1.2 se montre simplement en constatant que, par définition des substitutions, $dom(\mu) \cap vrang(\mu) = \emptyset$ et donc pour tout $\alpha \in (dom(\mu))$ on a $\mathcal{R}'(\alpha) = \mathcal{R}(\alpha)$. Et par le lemme 11.3.3 on a $\mu\alpha :: \mathcal{R}'$. Le deuxième point est vérifié par le fait que $\mathcal{R}'$ est μ-close.

Nous avons encore besoin de trois lemmes qui nous seront utiles pour montrer, dans la section suivante, la préservation des solutions.

Lemme 11.3.9 *Si $\mathcal{R}$ est μ-close, μ respecte $\mathcal{R}$, $vrang(\lambda) \cap dom(\mu) = \emptyset$ et $\mathcal{R}'$ est la λ-clôture de $\mathcal{R}$, alors $\mathcal{R}'$ est μ-close et μ respecte $\mathcal{R}'$.*

Preuve: Soit $\mathcal{R}'' = \bigcup_{\alpha \in dom(\lambda)} \bigcup_{\alpha' \in var(\lambda\alpha)} \mathcal{R}(\alpha) \times \{\alpha'\}$, on a donc $\mathcal{R}' = \mathcal{R} \cup \mathcal{R}''$. Soient $\alpha \in dom(\mu)$ et $\alpha' \in var(\mu(\alpha))$, on a $\mathcal{R}(\alpha) \subseteq \mathcal{R}(\alpha')$ (car μ respecte $\mathcal{R}$). Puisque $\alpha \notin vrang(\lambda)$, par définition de $\mathcal{R}''$, $\mathcal{R}''(\alpha) = \emptyset$ et donc $\mathcal{R}'(\alpha) = \mathcal{R}(\alpha)$. D'autre part, $\mathcal{R}(\alpha') \subseteq \mathcal{R}'(\alpha')$, d'où $\mathcal{R}'(\alpha) \subseteq \mathcal{R}'(\alpha')$ ce qui montre que $\mathcal{R}'$ est μ-close. Montrons que μ respecte $\mathcal{R}'$, c'est-à-dire que pour $\alpha \in dom(\mu)$ on a $nm(\mu(\alpha)) \cap \mathcal{R}'(\alpha) = \emptyset$. On a vu que dans ce cas $\mathcal{R}'(\alpha) = \mathcal{R}(\alpha)$ et comme μ respecte $\mathcal{R}$, c'est fini. Pour montrer que $\forall \alpha \in dom(\mu),\ \mu\alpha :: \mathcal{R}'$ on se réfère au lemme 11.3.3. Si $\rho_L \in dom(\mu)$, on a $\mathcal{R}'(\rho_L) \subseteq \mathcal{R}'(\rho var(\mu\rho_L))$, car dans ce cas $\mathcal{R}'(\rho_L) = \mathcal{R}(\rho_L)$. □

Lemme 11.3.10 *Si $\mathcal{R}$ est μ-close, μ et λ respectent $\mathcal{R}$ avec $(vrang(\lambda) \cup dom(\lambda)) \cap dom(\mu) = \emptyset$, et $\mathcal{R}'$ est la λ-clôture de $\mathcal{R}$, alors $\lambda\mu$ respecte $\mathcal{R}'$ et $\mathcal{R}'$ est $\lambda\mu$-close.*

Preuve: $\mathcal{R}$ étant μ-close et $\mathcal{R}'$ sa λ-clôture et puisque $vrang(\lambda) \cap dom(\mu) = \emptyset$, par le lemme 11.3.9, $\mathcal{R}'$ est également μ-close et μ respecte $\mathcal{R}'$. De plus, comme λ respecte $\mathcal{R}'$ (par la remarque 11.3.8), par le lemme 11.3.6, $\lambda\mu$ respecte $\mathcal{R}'$. Montrons que $\mathcal{R}'$ est $\lambda\mu$-close, c'est-à-dire que pour $\alpha \in dom(\lambda\mu)$ et $\alpha' \in var(\lambda\mu\alpha)$ on a $\mathcal{R}'(\alpha) \subseteq \mathcal{R}'(\alpha')$. D'après les hypothèses on a $dom(\lambda\mu) = dom(\lambda) \uplus dom(\mu)$. Supposons que $\alpha \in dom(\lambda)$, on a $\alpha' \in var(\lambda\mu\alpha) = var(\lambda\alpha)$, et $\mathcal{R}'$ étant λ-close on conclue directement. Supposons que $\alpha \in dom(\mu)$, on a

$$\alpha' \in var(\lambda\mu\alpha) = \big(\bigcup_{\alpha'' \in var(\mu\alpha) \cap dom(\lambda)} var(\lambda\alpha'') \big) \cup (var(\mu\alpha) - dom(\lambda))$$

Si $\alpha' \in (var(\mu\alpha) - dom(\lambda))$, $\mathcal{R}'$ étant μ-close, on a directement $\mathcal{R}'(\alpha) \subseteq \mathcal{R}'(\alpha')$. Si

$$\alpha' \in \bigcup_{\alpha'' \in var(\mu\alpha) \cap dom(\lambda)} var(\lambda\alpha'')$$

Soit $\alpha'' \in var(\mu\alpha) \cap dom(\lambda)$ tel que $\alpha' \in var(\lambda\alpha'')$. $\mathcal{R}'$ étant λ-close, on a $\mathcal{R}'(\alpha'') \subseteq \mathcal{R}'(\alpha')$. D'autre part, de $\alpha'' \in var(\mu\alpha)$ et $\mathcal{R}'$ est μ-close, on $\mathcal{R}'(\alpha) \subseteq \mathcal{R}'(\alpha'')$, et finalement $\mathcal{R}'(\alpha) \subseteq \mathcal{R}'(\alpha')$. □

Lemme 11.3.11 *Si $\lambda\mu$ est une substitution de base qui respecte $\mathcal{R}$ et $dom(\lambda) \cap dom(\mu) = \emptyset$, et si $\mathcal{R}'$ est la μ-clôture de $\mathcal{R}$, alors λ respecte $\mathcal{R}'$.*

Preuve: Commençons par montrer que λ et μ respectent $\mathcal{R}$. Puisque $dom(\lambda) \cap dom(\mu) = \emptyset$, pour tout $\alpha \in dom(\lambda)$ on a $\lambda\alpha = \lambda\mu\alpha$ et donc $nm(\lambda\alpha) = nm(\lambda\mu\alpha)$. D'où, de $\lambda\mu$ respecte $\mathcal{R}$ on déduit que $nm(\lambda\alpha) \cap \mathcal{R}(\alpha) = \emptyset$. Il est évident pour tout $\alpha \in dom(\mu)$, on a $nm(\mu\alpha) \cap \mathcal{R}(\alpha) = \emptyset$ car $nm(\mu\alpha) \subseteq nm(\lambda\mu\alpha)$. Les autres points de la définition 11.1.2 se montrent facilement car les domaines de λ et μ sont disjoints et inclus dans celui de $\lambda\mu$, donc λ et μ respectent $\mathcal{R}$. On a $\mathcal{R}' = \mathcal{R} \cup \mathcal{R}''$ où $\mathcal{R}'' = \bigcup_{\alpha \in dom(\mu)} \bigcup_{\alpha' \in var(\mu\alpha)} \mathcal{R}(\alpha) \times \{\alpha'\}$. On a vu que λ respecte $\mathcal{R}$, il faut donc montrer que pour tout $\alpha \in dom(\lambda)$ on a $nm(\lambda\alpha) \cap \mathcal{R}''(\alpha) = \emptyset$ $(*)$. On a deux cas :

- si $\alpha \notin vrang(\mu)$, alors $\mathcal{R}''(\alpha) = \emptyset$ et c'est fini ;
- si $\alpha \in vrang(\mu)$, pour montrer $(*)$ il faut montrer que pour tout $\alpha' \in dom(\mu)$ tel que $\alpha \in var(\mu\alpha')$ on a $nm(\lambda\alpha) \cap \mathcal{R}(\alpha') = \emptyset$. Or $nm(\lambda\alpha) \subseteq nm(\lambda\mu\alpha')$ et comme $\lambda\mu$ respecte $\mathcal{R}$ c'est fini.

On a $\forall \alpha \in dom(\lambda)$, $\lambda\alpha :: \mathcal{R}'$ par le lemme 11.3.3. □

11.4 L'algorithme d'unification

Nous décrivons l'algorithme d'unification à l'aide d'une relation de réduction $\rightsquigarrow$. Celle-ci consiste en la réécriture de tuples (ou configurations) $(\mathcal{C}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{R}}$ où $\mathcal{C}$ et $\mathcal{I}$ sont des contraintes de types et de sous-types, μ une substitution, $\mathcal{R}$ une relation de liaison et $\mathcal{X}$ un ensemble fini de variables de type tel que $var(\mathcal{C}, \mathcal{I}) \subseteq \mathcal{X}$. Intuitivement, l'idée de cette relation est : étant donné une contrainte $(\mathcal{C}, \mathcal{I})_{\mathcal{R}}$, si

$$(\mathcal{C}, \mathcal{I}, \emptyset)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow^* (\emptyset, \mathcal{A}, \mu)^{\mathcal{Y}}_{\mathcal{R}'} \not\rightsquigarrow$$

alors toute substitution de base solution de $\mathcal{C}$ et $\mathcal{I}$ qui respecte $\mathcal{R}$ est une instance de μ. S'il n'y a pas de solution de $\mathcal{C}$ et $\mathcal{I}$ qui respecte $\mathcal{R}$, alors $(\mathcal{C}, \mathcal{I}, \emptyset)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow^* \bot$ où $\bot$ est la configuration d'échec. Les règles de transition sont reportés dans la figure 11.1 où on utilise l'abréviation $\psi \triangleleft \rho_L$ pour $[\rho_L/\rho var(\psi)]\psi$.

La réduction $\rightsquigarrow$ consiste essentiellement en la décomposition de paires de schémas de types jusqu'à ce que l'un d'eux soit une variable de type, c'est-à-dire jusqu'à obtenir une paire $\alpha \doteq \tau$. Alors, la substitution courante μ est composée avec la substitution du type τ à la variable α, c'est-à-dire $[\tau/\alpha]$. Celle-ci est également appliquée à la contrainte restante (voir la règle (*elim*)). La condition $\alpha \notin var(\tau)$ garantie qu'il ne restera plus d'occurrences de la variable α dans la contrainte résultante, et donc que cette variable a bien été éliminée (on évite aussi d'éventuelles réductions infinies). D'autre part, la relation de liaison $\mathcal{R}$ est « renforcée » en sa clôture par $[\tau/\alpha]$ afin que sa composition avec la substitution courante μ respecte cette nouvelle relation. Si la condition $\alpha \notin var(\tau)$ n'est pas vérifiée, la règle (*oc*) (pour *occurs check*) s'applique et nous laisse dans la configuration d'échec. Les règles (*trivial*) et ($trivial_{val}$) éliminent les équations triviales (*i.e.* celles pour lesquels toute substitution est solution). Les règles (*chan*) et (*at*) décomposent les types de façon évidente (par exemple, toutes les solution de $Ch(\tau) \doteq Ch(\sigma)$ sont celles de $\tau \doteq \sigma$). La règle (*loc*) nous dit que pour unifier $\{a : \gamma, \psi\}$ avec $\{a : \delta, \phi\}$, nous devons unifier γ avec δ, et ψ avec ϕ. Ceci est justifié par le lemme suivant.

Lemme 11.4.1 *Soit μ est une substitution de base qui respecte $\mathcal{R}$ et $\mathcal{C} :: \mathcal{R}$.*

1. *Si $\{a : \gamma, \psi\} :: \mathcal{R}$ et $\{a : \delta, \phi\} :: \mathcal{R}$, alors μ est une solution de $\big\{\{a : \gamma, \psi\} \doteq \{a : \delta, \phi\}\big\} \cup \mathcal{C}$ si et seulement si c'est une solution de $\{\gamma \doteq \delta, \psi \doteq \phi\} \cup \mathcal{C}$.*

(*trivial*) $(\{\alpha \doteq \alpha\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}}$

(*trivial*$_{val}$) $(\{val \doteq val\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}}$

(*elim*) $(\{\alpha \doteq \tau\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow ([\tau/\alpha]\mathcal{C}\,,\,[\tau/\alpha]\mathcal{I}\,,\,[\tau/\alpha]\mu)^{\mathcal{X}}_{\mathcal{R}'}$
si $\alpha \notin var(\tau) \cup \mathcal{V}_{loc},, \alpha \notin dom(\mu), nm(\tau) \cap \mathcal{R}(\alpha) = \emptyset$ et où $\mathcal{R}'$ est la cloture par $[\tau/\alpha]$ de $\mathcal{R}$.

(*chan*) $(\{Ch(\tau) \doteq Ch(\sigma)\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{\tau \doteq \sigma\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}}$

(*at*) $(\{\gamma@\psi \doteq \delta@\phi\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{\gamma \doteq \delta, \psi \doteq \phi\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}}$

(*loc*) $(\{\{a : \gamma, \psi\} \doteq \{a : \delta, \phi\}\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{\gamma \doteq \delta\} \cup \{\psi \doteq \phi\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}}$

(*loc_end*) $(\{\psi \doteq \phi\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\lambda\mathcal{C},\,\lambda\mathcal{I}\,,\,\lambda\mu)^{\mathcal{Y}}_{\mathcal{R}'}$
où $\mathcal{Y} = \mathcal{X} \uplus \{\rho''_{L''}\}, \psi' = \psi \triangleleft \rho''_{L''},\ \phi' = \phi \triangleleft \rho''_{L''}$ et $\lambda = [\psi'/\rho'_{L'}, \phi'/\rho_L]$ avec
$\rho_L = \rho var(\psi), \rho'_{L'} = \rho var(\phi)$, et $L'' = (L - \mathcal{R}(\rho'_{L'})) \cup (L' - \mathcal{R}(\rho_L))$ et si
- $dom(\psi) \cap dom(\phi) = \emptyset$
- $L' \cap dom(\psi) = \emptyset$ et $L \cap dom(\phi) = \emptyset$
- $nm(\psi) \cap \mathcal{R}(\rho'_{L'}) = \emptyset$ et $nm(\phi) \cap \mathcal{R}(\rho_L) = \emptyset$
- $\rho_L \notin var(\phi) - \rho'_{L'}$ et $\rho'_{L'} \notin var(\psi) - \rho_L$
- $\mathcal{R}'$ est la λ-cloture de $\mathcal{R}$.

(*clash*) $(\{\sigma \doteq \tau\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow \bot$
si σ et τ ne sont pas des variables et n'ont pas le même symbol de tête, ou sont des types de localités avec des domaines disjoints mais ne vérifiant pas les conditions de *(loc_end)*.

(*oc*) $(\{\alpha \doteq \tau\} \cup \mathcal{C}\,,\,\mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow \bot$
si $\alpha \in var(\tau)$.

(*st_ok*) $(\mathcal{C}\,,\,\{\{a : \gamma, \psi\} \lessdot \{a : \delta, \phi\}\} \cup \mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{\gamma \doteq \delta\} \cup \mathcal{C}\,,\,\{\psi \lessdot \phi\} \cup \mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}}$

(*st_wrg*) $(\mathcal{C}\,,\,\{\psi \lessdot \phi\} \cup \mathcal{I}\,,\,\mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\lambda\mathcal{C}\,,\,\{\psi \triangleleft \rho''_{L''} \lessdot \rho'_{L'}\} \cup \lambda\mathcal{I}\,,\,\lambda\mu)^{\mathcal{Y}}_{\mathcal{R}'}$
où $\mathcal{Y} = \mathcal{X} \uplus \{\rho''_{L''}\}, \lambda = [\phi \triangleleft \rho''_{L''}/\rho_L]$ avec $\rho_L = \rho var(\psi), \rho'_{L'} = \rho var(\phi)$, et si $\phi \notin \mathcal{V}_{loc}$,
$dom(\psi) \cap dom(\phi) = \emptyset, L \cap dom(\phi) = \emptyset, \rho_L \notin var(\phi) - \rho'_{L'}, nm(\phi) \cap \mathcal{R}(\rho_L) = \emptyset$,
$L'' = (L - \mathcal{R}(\rho'_{L'})) \cup (L' - \mathcal{R}(\rho_L)), \mathcal{R}' = \mathcal{R}'' \cup \mathcal{R}(\rho'_{L'}) \times \{\rho''_{L''}\}$
où $\mathcal{R}''$ est la λ-cloture de $\mathcal{R}$.

FIG. 11.1: Relation de réduction pour l'unification

2. *Soient* ψ *et* ϕ *des types de localités tels que* $dom(\psi) \cap dom(\phi) = \emptyset$, $\psi :: \mathcal{R}$, $\phi :: \mathcal{R}$, $\rho_L = \rho var(\psi)$ *et* $\rho'_{L'} = \rho var(\phi)$, *alors* μ *est solution de* $\{\psi \doteq \phi\} \cup \mathcal{C}$ *si et seulement si*
   (a) $\rho_L \notin var(\phi) - \rho'_{L'}$ *et* $\rho'_{L'} \notin var(\psi) - \rho_L$;
   (b) $dom(\psi) \cap L' = \emptyset$ *et* $dom(\phi) \cap L = \emptyset$;
   (c) $nm(\psi) \cap \mathcal{R}(\rho'_{L'}) = \emptyset$ *et* $nm(\phi) \cap \mathcal{R}(\rho_L) = \emptyset$;
   (d) *pour tout ensemble fini de variables de types* $\mathcal{Y}$, *pour tout* $\rho''_{L''} \notin \mathcal{X} \cup \mathcal{Y} \cup dom(\mathcal{R})$ *(où* $\mathcal{X} = var(\{\psi \doteq \phi\} \cup \mathcal{C})$ *et* $L'' = (L - \mathcal{R}(\rho'_{L'})) \cup (L' - \mathcal{R}(\rho_L))$*), il existe une substitution* λ *telle que* $\mu =_{\mathcal{X} \cup \mathcal{Y}} \lambda\mu'$ *où*

$$\mu' = [\psi \lhd \rho''_{L''}/\rho'_{L'}, \phi \lhd \rho''_{L''}/\rho_L],$$

   et $\mu'\mathcal{C} :: \mathcal{R}'$ *et* λ *est une solution de* $\mu'\mathcal{C}$ *qui respecte* $\mathcal{R}'$ *où* $\mathcal{R}'$ *est la* μ'*-clôture de* $\mathcal{R}$. *Enfin,* μ' *respecte* $\mathcal{R}'$.

Preuve: La preuve du premier point est directe, nous donnons celle du point 2.
($\Rightarrow$) Pour montrer (*a*) il suffit de raisonner par l'absurde en supposons que $\rho_L \in var(\phi) - \rho'_{L'}$ et on met en évidence que dans ce cas on obtient des types infinis. Pour montrer (*b*), supposons que $\psi = \{a : \gamma, \psi'\}$ avec $a \in L'$. Si μ est une solution de $\psi \doteq \phi$, on a $\mu\phi = \{a : \mu\gamma, \mu\psi'\}$ et donc il existe un ϕ' tel que $\mu\rho'_{L'} = \{a : \mu\gamma, \phi'\}$ car $a \notin dom(\phi)$. Or, de μ respecte $\mathcal{R}$, on a $dom(\mu\rho'_{L'}) \cap L' = \emptyset$ et donc $a \notin L'$. Pour montrer (*c*), il suffit de remarquer que $nm(\psi) \subseteq nm(\mu\rho'_{L'})$ et, comme μ respecte $\mathcal{R}$, on a $nm(\mu\rho'_{L'}) \cap \mathcal{R}(\rho'_{L'}) = \emptyset$ et donc $nm(\psi) \cap \mathcal{R}(\rho'_{L'}) = \emptyset$. Pour montrer (*d*), supposons que $\psi = \{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\}$ et $\phi = \{b_1 : \delta_1, \ldots, b_m : \delta_m, \rho'_{L'}\}$. Puisque μ unifie ψ et ϕ, on a $\{a_1 : \mu\gamma_1, \ldots, a_n : \mu\gamma_n, \mu\rho_L\} = \{b_1 : \mu\delta_1, \ldots, b_m : \mu\delta_m, \mu\rho'_{L'}\}$, et puisque $\{a_1, \ldots, a_n\} \cap \{b_1, \ldots, b_m\} = \emptyset$, il existe ψ tel que

$$\mu\rho_L = \{b_1 : \mu\delta_1, \ldots, b_m : \mu\delta_m, \psi\} \text{ et } \mu\rho'_{L'} = \{a_1 : \mu\gamma_1, \ldots, a_n : \mu\gamma_n, \psi\}$$

Soit λ défini par

$$\lambda\alpha = \begin{cases} \psi & \text{si } \alpha = \rho''_{L''} \\ \mu\alpha & \text{si } \alpha \neq \rho_L \text{ et } \alpha \neq \rho'_{L'} \end{cases}$$

Alors, puisque $\rho_L, \rho'_{L'} \notin (var(\psi) \cup var(\phi)) - \{\rho_L, \rho'_{L'}\}$ il est facile de vérifier que $\mu\rho_L = \lambda\mu'\rho_L$ et $\mu\rho'_{L'} = \lambda\mu'\rho'_{L'}$ et donc $\mu =_{\mathcal{X} \cup \mathcal{Y}} \lambda\mu'$. Pour montrer que $\psi \lhd \rho''_{L''} :: \mathcal{R}'$, il suffit de vérifier que $\{a_1, \ldots, a_n\} \subseteq L'' \cup \mathcal{R}'(\rho''_{L''})$ (1). On peut voir que $L'' \cup \mathcal{R}'(\rho''_{L''}) = L \cup L' \cup \mathcal{R}(\rho_L) \cup \mathcal{R}(\rho'_{L'})$, et de $\psi :: \mathcal{R}$ on déduit que $\{a_1, \ldots, a_n\} \subseteq L \cup \mathcal{R}(\rho_L)$ et donc (1). De la même manière on montre que $\phi \lhd \rho''_{L''} :: \mathcal{R}'$. Par le point (*c*) et le fait que $\mathcal{R}'(\rho_L) = \mathcal{R}(\rho_L)$ et $\mathcal{R}'(\rho'_{L'}) = \mathcal{R}(\rho'_{L'})$, on a $nm(\psi \lhd \rho''_{L''}) \cap \mathcal{R}'(\rho'_{L'}) = \emptyset$ et $nm(\phi \lhd \rho''_{L''}) \cap \mathcal{R}'(\rho_L) = \emptyset$. Enfin, par le point (*b*) et par définition de la clôture de $\mathcal{R}$ par μ', on vérifie facilement les conditions de la définition 11.1.2, ce qui montre que μ' respecte $\mathcal{R}'$. Donc, de $\mathcal{C} :: \mathcal{R}$ on a $\mu'\mathcal{C} :: \mathcal{R}$. Puisque μ respecte $\mathcal{R}$, la restriction de $\lambda\mu'$ au domaine $dom(\lambda\mu') - \rho''_{L''}$ respecte $\mathcal{R}$ et par le lemme 11.3.11 la restriction de λ au domaine $dom(\lambda) - \rho''_{L''}$ respecte $\mathcal{R}'$. Il ne reste plus qu'à montrer que $[\psi/\rho''_{L''}]$ respecte $\mathcal{R}'$. De $\psi :: \mathcal{R}$ on déduit que $\psi :: \mathcal{R}'$. Et, de μ respecte $\mathcal{R}$, on a $nm(\psi) \cap \mathcal{R}(\rho_L) = nm(\psi) \cap \mathcal{R}(\rho'_{L'}) = \emptyset$. Puisque $\mathcal{R}'(\rho''_{L''}) = \mathcal{R}(\rho_L) \cup \mathcal{R}(\rho'_{L'})$, on a bien $nm(\psi) \cap \mathcal{R}'(\rho''_{L''}) = \emptyset$. Donc, finalement $[\psi/\rho''_{L''}]$ respecte $\mathcal{R}'$ et donc λ aussi.

($\Leftarrow$) Il suffit de montrer que la restriction λ' de $\lambda\mu'$ au domaine $\mathit{dom}(\lambda\mu') - \{\rho''_{L''}\}$ est une solution de $\psi \doteq \phi$. Soient

$$\psi = \{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\} \quad \text{et} \quad \phi = \{b_1 : \delta_1, \ldots, b_m : \delta_m, \rho'_{L'}\}$$

Puisque, par hypothèse, $\rho_L \notin \mathit{var}(\phi) - \rho'$

Preuve: Soient n_v et n_t des entiers naturels définis par :

- n_v est le nombre de variables distinctes ayant une occurrence dans $(\mathcal{C}, \mathcal{I})$;
- n_t est la somme des tailles des types dans $(\mathcal{C}, \mathcal{I})$ (en supposant que les variables et les constantes de types ont la taille 1).

En associant la paire $(0, 0)$ à $\bot$, la table suivante montre la décroissance stricte (pour l'ordre lexicographique) de (n_v, n_t) pour chaque règle excepté la règle (*st_wrg*).

	$(\mathit{trivial}_{(\mathit{val})})/(\mathit{chan})/(\mathit{at})/(\mathit{loc})/(\mathit{st\_ok})$	$(\mathit{loc\_end})/(\mathit{clash})/(\mathit{oc})/(\mathit{elim})$
n_v	$=$	$<$
n_t	$<$	

La règle (*st_wrg*) peut sembler problématique car, alors que le nombre de variables de types reste inchangé, la taille totale peut augmenter. En réalité, on peut montrer que cette règle ne peut être appliquée qu'un nombre fini de fois. Le nombre d'inéquations dans $\mathcal{I}$ étant constant, on peut les ordonner ainsi : $\psi_1 \lessdot \phi_1, \ldots, \psi_n \lessdot \phi_n$. On remarque que si

$$(\mathcal{C}, \{\psi_1 \lessdot \phi_1, \ldots, \psi_n \lessdot \phi_n\}, \mu) \rightsquigarrow (\mathcal{C}', \{\psi'_1 \lessdot \phi'_1, \ldots, \psi'_n \lessdot \phi'_n\}, \mu')$$

alors, pour tout $i \in \{1, \ldots, n\}$, si $\rho\mathit{var}(\psi_i) = \rho_L$ et $\rho\mathit{var}(\phi_i) = \rho'_{L'}$, alors $L \subseteq L'$. En particulier, si la règle appliquée est (*st_wrg*) à une inéquation $\psi_i \lessdot \phi_i$, alors l'inclusion est stricte (*i.e.* $L \subset L'$). Puisque dans L on ajoute que des noms de canaux ayant une occurrence dans $(\mathcal{C}, \mathcal{I}, \mu)$, la croissance de L est bornée ce qui montre qu'on ne peut appliquer (*st_wrg*) à une inéquation qu'un nombre fini de fois. Donc, après un nombre fini de pas dans la réduction, on atteint une configuration dans laquelle toute règle peut être appliquée sauf (*st_wrg*) et donc à partir de laquelle (n_v, n_t) décroit et donc termine soit sur $\bot$, soit sur une configuration $(\emptyset, \mathcal{I}', \mu')$ où $\mathcal{I}'$ est nécessairement atomique. □

Il y a un certain nombre d'invariants naturels que nous désirons préserver au cours de la réduction. Par exemple, le fait que les types présents dans le tuple sont bien formés et d'autres qui nous sont utiles pour montrer la correction de l'algorithme. On regroupe ces propriétés dans la définition suivante.

Définition 11.4.3 *On dit que $(\mathcal{C}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{R}}$ est une* bonne configuration *si :*

1. $\mathit{var}(\mathcal{C}, \mathcal{I}, \mu) \subseteq \mathcal{X}$;
2. $\mathit{dom}(\mu) \cap \mathit{var}(\mathcal{C}, \mathcal{I}) = \emptyset$;
3. μ *respecte* $\mathcal{R}$;
4. $\mathcal{R}$ *est* μ*-close ;*
5. $\mathcal{C} :: \mathcal{R}$ *et* $\mathcal{I} :: \mathcal{R}$.

Lemme 11.4.4 (Invariance de bonne configuration) *Si $(\mathcal{C}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{R}}$ est une bonne configuration et $(\mathcal{C}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{C}', \mathcal{I}', \mu')^{\mathcal{Y}}_{\mathcal{R}'}$, alors $(\mathcal{C}', \mathcal{I}', \mu')^{\mathcal{Y}}_{\mathcal{R}'}$ est une bonne configuration.*

Preuve: Regardons le cas des règles (*elim*), (*loc_end*), et (*st_wrg*), les autres cas étant triviaux.
(*elim*) De $\mathit{nm}(\tau) \cap \mathcal{R}(\alpha)$ et des hypothèses $\tau :: \mathcal{R}$ et $t \notin \mathcal{V}_{\mathit{loc}}$, on déduit que $[\tau/\alpha]$ respecte $\mathcal{R}$. Donc, d'après le lemme 11.3.10, $[\tau/\alpha]\mu$ respecte $\mathcal{R}'$ qui est clos par $[\tau/t]\mu$. Puisque $[\tau/\alpha]$ respecte $\mathcal{R}'$ et que $\mathcal{C} :: \mathcal{R}$ et $\mathcal{I} :: \mathcal{R}$, par le lemme 11.3.3 on a $\mathcal{C} :: \mathcal{R}'$ et $\mathcal{I} :: \mathcal{R}'$ et par le lemme 11.3.2 on a donc

$[\tau/t]\mathcal{C} :: \mathcal{R}'$ et $[\tau/\alpha]\mathcal{I} :: \mathcal{R}'$.
(*loc_end*) D'après le lemme 11.4.1 sous les conditions de cette règle, $\lambda = [\psi'/\rho'_{L'}, \phi'/\rho_L]$ respecte $\mathcal{R}'$. Par le lemme 11.3.9 μ respecte $\mathcal{R}'$ qui est μ-close et donc, par le lemme 11.3.10, $\lambda\mu$ respecte $\mathcal{R}'$ et $\mathcal{R}'$ est $\lambda\mu$-close.
(*st_wrg*) Montrons que $\lambda = [\phi \lhd \rho''_{L''}/\rho_L]$ respecte $\mathcal{R}'$. On a $nm(\phi \lhd \rho''_{L''}) = nm(\phi)$ et on peut facilement constater que $\mathcal{R}'(\rho_L) = \mathcal{R}(\rho_L)$ car $\rho_L \notin var(\phi) - \rho'_{L'}$ donc $nm(\phi \lhd \rho''_{L''}) \cap \mathcal{R}'(\rho_L) = nm(\phi) \cap \mathcal{R}(\rho_L) = \emptyset$ par les conditions de la règle. On peut voir que $\mathcal{R}'(\rho''_{L''}) = \mathcal{R}(\rho_L) \cup \mathcal{R}(\rho'_{L'})$ (1) et donc on a bien $\mathcal{R}'(\rho''_{L''}) \subseteq \mathcal{R}'(\rho_L)$ car $\mathcal{R}'(\rho_L) = \mathcal{R}(\rho_L)$. Enfin, montrons par induction sur la preuve de $\phi :: (L_0, \mathcal{R})$ que $\phi \lhd \rho''_{L''} :: (L_1, \mathcal{R}')$ avec $L_0 \subseteq L_1$. Si $\phi = \rho'_{L'}$, on a $\rho'_{L'} :: (L_0, \mathcal{R})$ avec $L_0 = L' \cup \mathcal{R}(\rho'_{L'})$. Et $\rho''_{L''} :: (L_1, \mathcal{R}')$ avec $L_1 = L'' \cup \mathcal{R}'(\rho''_{L''}) = L \cup L' \cup \mathcal{R}(\rho_L) \cup \mathcal{R}(\rho'_{L'})$ et donc $L_0 \subseteq L_1$. Si $\phi = \{a : \gamma, \phi'\} :: (L_0, \mathcal{R})$, alors $\gamma :: \mathcal{R}$ et $\phi' :: (L_0 \uplus \{a\}, \mathcal{R})$. Alors, par hypothèse d'induction, $\phi' \lhd \rho''_{L''} :: (L_2, \mathcal{R}')$ avec $L_0 \cup \{a\} \subseteq L_2$ et donc il existe un L_1 tel que $L_2 = L_1 \uplus \{a\}$. De plus $\gamma :: \mathcal{R}'$ et par conséquent $\phi \lhd \rho''_{L''} = \{a : \gamma, \phi' \lhd \rho''_{L''}\} :: (L_1, \mathcal{R}')$ avec $L_0 \subseteq L_1$, ce qui montre que λ respecte $\mathcal{R}'$. Exactement de la même façon on peut vérifier que $\psi \lhd \rho''_{L''} :: \mathcal{R}'$. On peut facilement voir que μ respecte $\mathcal{R}'$ car μ respecte $\mathcal{R}''$ (par le lemme 11.3.10) et $\rho''_{L''} \notin var(\mu)$. Enfin, comme dans le cas précédent, on peut facilement vérifier que $\lambda\mu$ respecte $\mathcal{R}'$, et que $\mathcal{R}'$ est $\lambda\mu$-close, $\lambda\mathcal{C} :: \mathcal{R}'$ et $\lambda\mathcal{I} :: \mathcal{R}'$. □

Le lemme suivant établit la propriété que les solutions des contraintes dans les tuples sont préservées par la relation de réduction.

Lemme 11.4.5 (Préservation) *Si $(\mathcal{C}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow^* (\mathcal{C}', \mathcal{I}', \mu')^{\mathcal{Y}}_{\mathcal{R}'}$ et $(\mathcal{C}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{R}}$ est une bonne configuration, alors $\mu' = \mu''\mu$ et,*

1. *pour tout λ solution de base de $(\mathcal{C}, \mathcal{I})$ qui respecte $\mathcal{R}$, il existe λ' telle que $\lambda =_{\mathcal{X}} \lambda'\mu''$ et λ' est solution de $(\mathcal{C}', \mathcal{I}')$ qui respecte $\mathcal{R}'$.*
2. *pour tout λ solution de $(\mathcal{C}', \mathcal{I}')$ qui respecte $\mathcal{R}'$, $\lambda\mu''$ est une solution de $(\mathcal{C}, \mathcal{I})$ qui respecte $\mathcal{R}'$.*

Preuve: On procède par induction sur la longueur de la séquence de réduction. On n'explore que le cas de base (un pas de réduction), le reste de la preuve étant trivial. D'après le lemme 11.4.4 $(\mathcal{C}', \mathcal{I}', \mu')^{\mathcal{Y}}_{\mathcal{R}'}$ est une bonne configuration. Dans le cas des règles (*trivial*), (*trivial*$_{(val)}$), (*chan*), (*at*), (*loc*) et (*st_ok*), μ'' est simplement la substitution vide, $\mathcal{Y} = \mathcal{X}$ et $\mathcal{R}' = \mathcal{R}$ et la preuve est directe. Dans le cas de la règle (*loc_end*), on utilise simplement le lemme 11.4.1.

(*elim*) On a $\mu'' = [\tau/\alpha]$ qui respecte $\mathcal{R}$ et $\mathcal{Y} = \mathcal{X}$. Puisque $\mathcal{R}$ est μ-close, que $\mathcal{R}'$ est la μ''-clôture de $\mathcal{R}$ et que $dom(\mu) \cap (var(\tau) \cup \{\alpha\}) = \emptyset$, par lemme 11.3.10, $\mathcal{R}'$ est $\mu''\mu$-close et $\mu''\mu$ respecte $\mathcal{R}'$.
(1) De λ solution de $\{\alpha \doteq \tau\} \cup \mathcal{C}$ qui respecte $\mathcal{R}$, on déduit que λ a la forme

$$[\tau_1/\alpha_1, \ldots, \tau_n/\alpha_n, \lambda\tau/\alpha]$$

Soit $\lambda' = [\tau_1/\alpha_1, \ldots, \tau_n/\alpha_n]$, alors $\lambda =_{\mathcal{X}} \lambda'[\tau/\alpha]$ car $\alpha \notin var(\tau)$. Comme $\lambda'[\tau/\alpha]$ respecte $\mathcal{R}$ et que $\mathcal{R}'$ est la clôture par $[\tau/\alpha]$ de $\mathcal{R}$, par lemme 11.3.11, λ' respecte $\mathcal{R}'$. Enfin, puisque λ est solution de $(\{\alpha \doteq \tau\} \cup \mathcal{C}, \mathcal{I})$, λ' est évidemment solution de $([\tau/\alpha]\mathcal{C}, [\tau/\alpha]\mathcal{I})$.
(2) Si λ est solution de $([\tau/\alpha]\mathcal{C}, [\tau/\alpha]\mathcal{I})$ qui respecte $\mathcal{R}'$, alors $\lambda[\tau/\alpha]$ est évidemment solution

de $(\mathcal{C},\mathcal{I})$. Puisque λ et $[\tau/\alpha]$ respectent $\mathcal{R}'$ qui close par $[\tau/\alpha]$, d'après le lemme 11.3.6 $\lambda[\tau/\alpha]$ respecte $\mathcal{R}'$.

(*st_wrg*) On a $\mathcal{Y} = \mathcal{X} \uplus \{\rho''_{L''}\}$ et $\mu'' = [\phi \lhd \rho''_{L''}/\rho_L]$ qui respecte $\mathcal{R}'$ (voir preuve du lemme 11.4.5).

(1) Soit λ une solution de $(\mathcal{C}, \{\psi \lessdot \phi\} \cup \mathcal{I})$ qui respecte $\mathcal{R}$ et

$$\psi = \{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\},\ \phi = \{b_1 : \delta_1, \ldots, b_m : \delta_m, \rho'_{L'}\}$$

Puisque λ est solution de $\psi \lessdot \phi$ et que $dom(\psi) \cap dom(\phi) = \emptyset$, nécessairement

$$\lambda\rho_L = \{b_1 : \lambda\delta_1, \ldots, b_m : \lambda\delta_m, \psi\}$$

pour un certain ψ, et on en déduit que λ a la forme suivante

$$[\tau_1/\alpha_1, \ldots, \tau_n/\alpha_n, \{b_1 : \lambda\delta_1, \ldots, b_m : \lambda\delta_m, \psi\}/\rho_L]$$

On définissant $\lambda' = [\tau_1/\alpha_1, \ldots, \tau_n/\alpha_n, \psi/\rho''_{L''}]$, alors on a bien $\lambda =_{\mathcal{X}} \lambda'\mu''$ et λ est une solution de $(\mu''\mathcal{C}, \{\psi \lhd \rho''_{L''} \lessdot \rho'_{L'}\} \cup \mu''\mathcal{I})$ qui, par le lemme 11.3.11, respecte $\mathcal{R}''$. Et, puisque $\rho''_{L''} \notin var(\lambda)$, λ respecte $\mathcal{R}'$.

(2) Soient

$$\psi = \{a_1 : \gamma_1, \ldots, a_n : \gamma_n, \rho_L\},\ \phi = \{b_1 : \delta_1, \ldots, b_m : \delta_m, \rho'_{L'}\}$$

et λ préserve $\psi \lhd \rho''_{L''} \lessdot \rho'_{L'}$ et respecte $\mathcal{R}'$. On a

$$\mu''\psi = \{a_1 : \gamma_1, \ldots, a_n : \gamma_n, b_1 : \delta_1, \ldots, b_m : \delta_m, \rho''_{L''}\} \text{ et } \mu''\phi = \phi$$

Il faut montrer que $\lambda\mu''\psi <: \lambda\phi$, or

$$\lambda\mu''\psi = \{a_1 : \lambda\gamma_1, \ldots, a_n : \lambda\gamma_n, b_1 : \lambda\delta_1, \ldots, b_m : \lambda\delta_m, \lambda\rho''_{L''}\}$$

et

$$\lambda\phi = \{b_1 : \lambda\delta_1, \ldots, b_m : \lambda\delta_m, \lambda\rho'_{L'}\}$$

donc il suffit de montrer que $\{a_1 : \lambda\gamma_1, \ldots, a_n : \lambda\gamma_n, \lambda\rho''_{L''}\} <: \lambda\rho'_{L'}$ ce qui est vrai puisque, par hypothèse, λ préserve $\psi \lhd \rho''_{L''} \lessdot \rho'_{L'}$. Enfin, puisque $\mathcal{R}'$ est μ''-close, et que λ respecte $\mathcal{R}'$, $\lambda\mu''$ respecte $\mathcal{R}'$ (par lemme 11.3.6). □

On peut enfin établir la correction de notre algorithme.

Proposition 11.4.6 (Correction) *Si $(\mathcal{C},\mathcal{I},\emptyset)^{\mathcal{X}}_{\mathcal{R}}$ est une bonne configuration et*

$$(\mathcal{C},\mathcal{I},\emptyset)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow^* (\emptyset,\mathcal{I}',\mu)^{\mathcal{Y}}_{\mathcal{R}'} \not\rightsquigarrow$$

alors μ est une solution principale de $(\mathcal{C},\mathcal{I})_{\mathcal{R}}$ sur $\mathcal{X}$ et μ respecte $\mathcal{R}'$.

Preuve: On a directement le fait que μ respecte $\mathcal{R}'$ par le lemme d'invariance de bonne configuration 11.4.4. Montrons que μ est solution de $(\mathcal{C},\mathcal{I})$. D'après le lemme de préservation 11.4.5 (2), pour toute solution λ de $(\emptyset,\mathcal{I}')$ qui respecte $\mathcal{R}'$, $\lambda\mu$ est solution de $(\mathcal{C},\mathcal{I})$ qui respecte $\mathcal{R}'$. En particulier $\lambda = \emptyset$ est solution de $(\emptyset,\mathcal{I}')$ donc μ est bien solution de $(\mathcal{C},\mathcal{I})$. Montrons que c'est une solution principale de $(\mathcal{C},\mathcal{I})$ sur $\mathcal{X}$. Soit λ une solution de $(\mathcal{C},\mathcal{I})$ qui respecte $\mathcal{R}$, alors d'après le lemme 11.4.5 (1), il existe une substitution λ' telle que $\lambda =_{\mathcal{X}} \lambda'\mu$. □

Lemme 11.4.7 $(\mathcal{C}, \mathcal{I}, \emptyset)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow^* \perp$ *si et seulement s'il n'existe pas de solution de* $\mathcal{C}$ *et* $\mathcal{I}$ *qui respecte* $\mathcal{R}$.

Preuve: ($\Rightarrow$) La preuve est directe, il suffit d'analyser les cas menant à une configuration d'erreur $\perp$.
($\Leftarrow$) Par contradiction à partir du lemme de terminaison et de la proposition 11.4.6. □

Proposition 11.4.8 (Complétude) *S'il existe une solution de* $\mathcal{C}$ *et* $\mathcal{I}$ *qui respecte* $\mathcal{R}$, *si* $\mathcal{C} :: \mathcal{R}$ *et* $\mathcal{I} :: \mathcal{R}$, *alors*

$$(\mathcal{C}, \mathcal{I}, \emptyset)^{var(\mathcal{C},\mathcal{I})}_{\mathcal{R}} \rightsquigarrow^* (\emptyset, \mathcal{I}', \mu)^{\mathcal{Y}}_{\mathcal{R}'} \not\rightsquigarrow$$

Preuve: C'est un corollaire du lemme de terminaison et du lemme 11.4.7. □

CHAPITRE 12

Génération de Contraintes

NOUS DÉCRIVONS ICI L'ALGORITHME DE GÉNÉRATION DES CONTRAINTES qui peut se résumer ainsi : partant d'un réseau (*i.e.* le terme que nous cherchons à typer) et d'un contexte initial, l'algorithme génère une contrainte dont la solution principale appliquée au contexte initial nous procure un typage principal. On entend par contexte initial, pour un terme S, l'ensemble des noms de localités et des valeurs ayant une occurrence libre dans S, associés à une variable de rangée ou au type *val*. Un tel contexte peu très facilement être généré.

12.1 L'algorithme de génération des contraintes

L'idée de l'algorithme est de reconstruire de manière incrémentale la preuve du typage d'un terme, c'est-à-dire l'arbre d'inférence dans le système décrit dans les figures 10.1 à 10.4. Ceci est réalisé grâce à un système de réécriture de tuples (ou configuration)

$$(\mathcal{J}, \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}}$$

où $\mathcal{J}$ est un ensemble de jugements de types ($\mathbf{\Gamma} \vdash S$, $\mathbf{\Gamma} \vdash_\ell P$, ou $\mathbf{\Gamma} \vdash^W_\ell u : \tau$), $(\mathcal{C}, \mathcal{I})_{\mathcal{R}}$ est la contrainte générée et $\mathcal{X}$ un ensemble de variables de types contenant celles de $\mathcal{J}$,$\mathcal{C}$ et $\mathcal{I}$. La réduction est très proche du système de types. En effet, étant donné un séquent dans le tuple, la réduction consiste essentiellement à remplacer celui-ci par les jugements en prémisse de la règle correspondante dans le système de types, en ajoutant éventuellement les contraintes adaptées. Par exemple, supposons que $\mathcal{J} = \{\mathbf{\Gamma} \vdash S \mid S'\} \cup \mathcal{J}'$, la règle pour typer la composition parallèle de réseaux étant

$$\frac{\mathbf{\Gamma} \vdash S \quad , \quad \mathbf{\Gamma} \vdash S'}{\mathbf{\Gamma} \vdash S \mid S'}$$

La transition correspondante pour notre relation de réduction est alors simplement :

$$(\{\mathbf{\Gamma} \vdash S \mid S'\} \cup \mathcal{J}', \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}} \quad \rightsquigarrow \quad (\{\mathbf{\Gamma} \vdash S, \mathbf{\Gamma} \vdash S'\} \cup \mathcal{J}', \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}}$$

$$(n_1) \quad (\{\Gamma \vdash^W_\ell k : t\} \uplus \mathcal{J}, \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{J}, \mathcal{C} \cup \{t \doteq \rho_\emptyset\}, \mathcal{I} \cup \{\Gamma(k) \lessdot \rho_\emptyset\})^{\mathcal{Y}}_{\mathcal{R}}$$
$\mathcal{Y} = \mathcal{X} \uplus \{\rho_\emptyset\}$

$$(n_2) \quad (\{\Gamma \vdash^W_\ell a : t\} \uplus \mathcal{J}, \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{J}, \mathcal{C} \cup \{t \doteq h\}, \{\Gamma(\ell) \lessdot \{a : h, \rho_{\{a\}}\}\} \cup \mathcal{I})^{\mathcal{Y}}_{\mathcal{R}}$$
$\mathcal{Y} = \mathcal{X} \uplus \{\rho_{\{a\}}, h\}$

$$(n_3) \quad (\{\Gamma \vdash^W_\ell a@k : t\} \uplus \mathcal{J}, \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{J}, \mathcal{C} \cup \{t \doteq h@\rho'_\emptyset\}, \mathcal{I} \cup \{\rho_{\{a\}} \doteqdot \rho'_\emptyset, \Gamma(k) \lessdot \{a : h, \rho_{\{a\}}\}\})^{\mathcal{Y}}_{\mathcal{R}}$$
$\mathcal{Y} = \mathcal{X} \uplus \{h, \rho_{\{a\}}, \rho'_\emptyset\}$

$$(n_4) \quad (\{\Gamma \vdash^W_\ell a : t\} \uplus \mathcal{J}, \mathcal{C}, \_)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{J}_n, \mathcal{C} \cup \{\Gamma(a) \doteq \mathit{val}, t \doteq \mathit{val}\}, \_)^{\mathcal{X}}_{\mathcal{R}}$$
si $a \in \mathcal{N}_{val}$

$$(nc) \quad (\{\Gamma \vdash^W_\ell u : t\} \uplus \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow \bot$$
si Γ n'est pas légal ou si les conditions des règles précédentes ne peuvent pas s'appliquer.

FIG. 12.1: Génération de contraintes pour les noms

Dans les règles de transition des figures 12.1 à 12.4 le symbole « _ » représente dans un tuple tout élément invariant pour la règle considérée, et $\Gamma, \Delta, \ldots$ sont des **schémas de contextes**, c'est-à-dire des contextes de typage où les types sont des schémas de types.

La figure 12.1 concerne les règles de transition s'appliquant aux jugements de typage pour les noms. Les règles (n_2) et (n_4) ne font que générer des contraintes entre le type du nom donné par le contexte et celui attendu. On remarque que les règles (n_1) à (n_3) introduisent de nouvelles inéquations de sous-typage dans $\mathcal{I}$. Par exemple, dans la règle (n_3) on attend un type t pour le canal a à la localité k. Alors, conformément à la règle de typage correspondante dans la figure 10.2, on contraint le type de k dans Γ à typer a avec une variable de type h, et t à être unifiable avec un type existentiel, c'est-à-dire $h@\rho'_\emptyset$. De plus, le type donné à k par Γ doit assigner des types au moins aux canaux attendus par t pour k. C'est la raison d'être des inéquations générées dans (n_3). Dans ces règles et dans les suivantes, toute occurrence de variable de type non définie est supposée fraîche, c'est-à-dire n'ayant pas d'occurrence dans le tuple initial.

Les règles des figures 12.2 et 12.3 traitent les jugements de typage pour les processus et éventuellement ajoutent de nouveaux jugements à $\mathcal{J}$. Par exemple, la règle (p_1) génère, à partir d'un séquent pour le typage d'un message $\overline{a}(u)$, un séquent dans $\mathcal{J}$ pour typer u avec le type t. La contrainte produite spécifie que le contexte Γ doit fournir au moins un type $\mathit{Ch}(t)$ à a dans le type de la localité courante. Les règles (p_{10}) et (p_{11}) pour le typage des processus paramétrés est très semblable.

En ce qui concerne la règle pour la réception, rappelons que le système de types utilise le système auxiliaire de types fort pour les noms. Ce dernier étant très simple et complètement déterministe, pour un nom u, un type τ et une localité ℓ, on peut très facilement déterminer le contexte Γ tel que $\Gamma \vdash_\ell u : \tau$. En réalité, pour ce qui nous concerne, nous n'avons besoin que du cas où τ est une variable de types et u n'est pas un nom de canal. Ainsi le rôle de la fonction $\mathit{gen}(u : t, \ell)$ est essentiellement de fournir un typage principal pour typer u de type t.

$$(p_0) \qquad (\{\mathbf{\Gamma} \vdash_\ell \mathbf{0}\} \cup \mathcal{J}, , \_, \_)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}}$$

$$(p_1) \qquad (\{\mathbf{\Gamma} \vdash_\ell \overline{a}(u)\} \cup \mathcal{J}, \mathcal{C}, \_)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{J} \cup \{\mathbf{\Gamma} \vdash^W_\ell u : t\}, \mathcal{C} \cup \{\mathbf{\Gamma}(\ell) \doteq \{a : Ch(t), \rho_{\{a\}}\}\}, \_)^{\mathcal{Y}}_{\mathcal{R}}$$

où $\mathcal{Y} = \mathcal{X} \uplus \{t, \rho_L\}$

$$(p_{2a}) \qquad (\{\mathbf{\Gamma} \vdash_\ell a(b).P\} \cup \mathcal{J}, \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{\mathbf{\Delta} \vdash_\ell P\} \cup \mathcal{J}, \mathcal{C} \cup \{\rho \doteq \{a : Ch(\ell)\}\}$$

$$(p_{10}) \quad (\{\mathbf{\Gamma} \vdash_\ell A(u)\} \uplus \mathcal{J}, \mathcal{C}, \_)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{\mathbf{\Gamma} \vdash^W_\ell u : t\} \uplus \mathcal{J}, \{\mathbf{\Gamma}(A) \doteq \mathit{Ch}(t)\} \uplus \mathcal{C}, \_)^{\mathcal{Y}}_{\mathcal{R}}$$

$\mathcal{Y} = \mathcal{X} \uplus \{t\}$ et $A \in \mathit{dom}(\mathbf{\Gamma})$

$$(p_{11}) \quad (\{\mathbf{\Gamma} \vdash_\ell T(u)\} \uplus \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{\mathbf{\Gamma} \vdash_\ell T : \mathit{Ch}(t), \mathbf{\Gamma} \vdash^W_\ell u : t\} \uplus \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}}$$

si $T \notin \mathcal{P}$ et avec $\mathcal{Y} = \mathcal{X} \uplus \{t\}$

$$(p_{12a}) \quad (\{\mathbf{\Gamma} \vdash_\ell \mathsf{rec}\, A(b).P : \mathit{Ch}(t)\} \uplus \mathcal{J}, \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{A : \mathit{Ch}(h), \mathbf{\Delta} \vdash_\ell P\} \uplus \mathcal{J}, \mathcal{C} \cup \{t \doteq h\}, \mathcal{I} \cup \{\rho_\emptyset \doteqdot \rho'_\emptyset\})^{\mathcal{Y}}_{\mathcal{R}'}$$

où $\mathbf{\Delta} =_{\mathit{dom}(\mathbf{\Gamma})-\ell} \mathbf{\Gamma}, \mathbf{\Delta}(\ell) = [\{b : h, \rho'_\emptyset\}/\rho_\emptyset]\mathbf{\Gamma}(\ell)$ avec $\rho_L = \rho\mathit{var}(\mathbf{\Gamma}(\ell)), \mathcal{Y} = \mathcal{X} \uplus \{h, \rho'_\emptyset\}$ et $\mathcal{R}' = \mathcal{R} \cup (\mathcal{R}(\rho_\emptyset) \times \{\rho'_\emptyset\}) \cup (\{b\} \times \mathcal{Y})$ et $A \notin \mathit{dom}(\mathbf{\Gamma})$

$$(p_{12b}) \quad (\{\mathbf{\Gamma} \vdash_\ell \mathsf{rec}\, A(u).P : \mathit{Ch}(t)\} \uplus \mathcal{J}, \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{A : \mathit{Ch}(t), \mathbf{\Gamma}, \mathbf{\Delta} \vdash_\ell P\} \uplus \mathcal{J}, \mathcal{C} \cup \mathcal{C}', \mathcal{I} \cup \mathcal{I}')^{\mathcal{Y}}_{\mathcal{R}'}$$

où $(\mathbf{\Delta}, \mathcal{C}', \mathcal{I}', \mathcal{X}', \mathcal{R}'') = \mathit{gen}(u : t, \ell), \mathcal{Y} = \mathcal{X} \uplus \mathcal{X}'$ et $\mathcal{R}' = \mathcal{R} \cup \mathcal{R}'' \cup (\mathit{chan}(u) \times \mathcal{Y})$ et $A \notin \mathit{dom}(\mathbf{\Gamma})$

$$(pc) \quad (\{\mathbf{\Gamma} \vdash_\ell P\} \uplus \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow \bot$$

si $\mathbf{\Gamma}$ n'est pas légal ou si les conditions des règles précédentes ne peuvent pas s'appliquer.

FIG. 12.3: Génération de contraintes pour les processus

Définition 12.1.1 *On définit la fonction* $\mathit{gen}(u : t, \ell) = (\mathbf{\Gamma}, \mathcal{C}, \mathcal{I}, \mathcal{X}, \mathcal{R})$ *comme suit :*

$$\begin{array}{rcl} \mathit{gen}(a : t, \ell) & = & (a : \mathit{val}, \{t \doteq \mathit{val}\}, \emptyset, \emptyset, \emptyset) \quad \text{si } a \in \mathcal{N}_{val} \\ \mathit{gen}(k : \rho_\emptyset, \ell) & = & (k : \rho_\emptyset, \{t \doteq \rho'_\emptyset\}, \{\rho_\emptyset \doteqdot \rho'_\emptyset\}, \{\rho_\emptyset, \rho'_\emptyset\}, \emptyset) \\ \mathit{gen}(a@k : t, \ell) & = & (k : \{a : h, \rho_\emptyset\}, \{t \doteq h@\rho'_\emptyset\}, \{\rho'_\emptyset \doteqdot \rho_\emptyset\}, \{h, \rho_\emptyset, \rho'_\emptyset\}, \{a\} \times \{h, \rho_\emptyset\}) \end{array}$$

Où $\ell \neq k$ *et* $\rho_\emptyset, \rho'_\emptyset$ *et* h *sont des variables fraîches.*

Les contraintes associées au contexte fournis par cette fonction permettent de forcer la variable de types donnée en argument à avoir une forme valide. Par exemple, pour typer $a@k$, la variable de type t doit s'unifier avec un type existentiel. Dans le lemme suivant on établit que la fonction *gen* fournit bien le résultat attendu. On note $\mathit{chan}(u)$ pour désigner le canal a quand $u = a@k$.

Lemme 12.1.2 *Si* $\mathit{gen}(u : t, \ell) = (\mathbf{\Gamma}, \mathcal{C}, \mathcal{I}, \mathcal{X}, \mathcal{R})$ *alors* $\mathcal{C} :: \mathcal{R}$, $\mathcal{I} :: \mathcal{R}$, $\mathbf{\Gamma} :: \mathcal{R}$, $\mathit{var}(\mathbf{\Gamma}, \mathcal{C}, \mathcal{I}) = \mathcal{X} \cup \{t\}$ *et*

1. *si* μ *est solution de base de* $(\mathcal{C}, \mathcal{I})$ *qui respecte* $\mathcal{R}$ *alors* $\mu\mathbf{\Gamma} \vdash_\ell u : \mu t$ *;*
2. *si* $\Delta \vdash_\ell u : \tau$, $u \notin \mathcal{N}_{chan}$ *et* $\mathit{chan}(u) \notin \mathit{nm}(\tau)$, *alors il existe une solution* μ *de* $(\mathcal{C}, \mathcal{I})$ *qui respecte* $\mathcal{R}$ *telle que* $\Delta = \mu\mathbf{\Gamma}$ *et* $\tau = \mu t$.

Preuve: La preuve de la bonne formation de $\mathcal{C}$, $\mathcal{I}$ et $\mathbf{\Gamma}$ pour $\mathcal{R}$ est triviale ainsi que la preuve de $\mathit{var}(\mathbf{\Gamma}, \mathcal{C}, \mathcal{I}) = \mathcal{X} \cup \{t\}$.

$$
\begin{array}{lrcl}
(\textit{net}_0) & (\{\mathbf{\Gamma} \vdash \mathbf{0}\} \uplus \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}} & \rightsquigarrow & (\mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}} \\
(\textit{net}_1) & (\{\mathbf{\Gamma} \vdash \ell[\![P]\!]\} \uplus \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}} & \rightsquigarrow & (\{\mathbf{\Gamma} \vdash_\ell P\} \uplus \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}} \\
(\textit{net}_2) & (\{\mathbf{\Gamma} \vdash S \mid S'\} \uplus \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}} & \rightsquigarrow & (\{\mathbf{\Gamma} \vdash S, \mathbf{\Gamma} \vdash S'\} \uplus \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}} \\
(\textit{net}_3) & (\{\mathbf{\Gamma} \vdash (\nu a@\ell)S\} \uplus \mathcal{J}, \_, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}} & \rightsquigarrow & (\{\mathbf{\Delta} \vdash S\} \uplus \mathcal{J}, \_, \mathcal{I} \uplus \{\rho_\emptyset \doteqdot \rho'_\emptyset\})^{\mathcal{Y}}_{\mathcal{R}'}
\end{array}
$$

où $\mathbf{\Delta} =_{\mathit{dom}(\mathbf{\Gamma})-\ell} \mathbf{\Gamma}$ et $\mathbf{\Delta}(\ell) = [\{a : h, \rho'_\emptyset\}/\rho_\emptyset]\mathbf{\Gamma}(\ell)$ avec $\rho_\emptyset = \rho\mathit{var}(\mathbf{\Gamma}(\ell))$
$\mathcal{Y} = \mathcal{X} \uplus \{h, \rho'_\emptyset\}$ et $\mathcal{R}' = \mathcal{R} \cup (\mathcal{R}(\rho_\emptyset) \times \{\rho'_\emptyset\}) \cup (\{a\} \times \mathcal{Y})$

$$
(\textit{net}_4) \quad (\{\mathbf{\Gamma} \vdash (\nu\ell)S\} \uplus \mathcal{J}, \mathcal{C}, \_)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{\ell : \rho_\emptyset, \mathbf{\Gamma} \vdash S\} \uplus \mathcal{J}, \mathcal{C}, \_)^{\mathcal{Y}}_{\mathcal{R}}
$$

$\mathcal{Y} = \mathcal{X} \uplus \{\rho_\emptyset\}$

$$
(\textit{net}_5) \quad (\{\mathbf{\Gamma} \vdash (\nu a)S\} \uplus \mathcal{J}, \mathcal{C}, \_)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\{a : \mathit{val}, \mathbf{\Gamma} \vdash S\} \uplus \mathcal{J}, \mathcal{C}, \_)^{\mathcal{X}}_{\mathcal{R}}
$$

si $a \in \mathcal{N}_{val}$

$$
(\textit{netc}) \quad (\{\mathbf{\Gamma} \vdash S\} \uplus \mathcal{J}, \_, \_)^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow \bot
$$

si $\mathbf{\Gamma}$ n'est pas légal ou si les conditions des règles précédentes ne peuvent pas s'appliquer.

FIG. 12.4: Génération de contraintes pour les réseaux

(1) Si $u = k$ alors $\mu t = \mu\rho'_\emptyset = \psi$ et de $\mu\rho_\emptyset \equiv \mu\rho_\emptyset$ on a $\mu\rho_\emptyset = \psi$ et donc $k : \mu\rho_\emptyset \vdash_\ell k : \mu t$.
Si $u = a@k$ alors $\mu t = \mu h@\mu\rho'_\emptyset = \gamma@\psi$ et $\mu\rho_\emptyset = \psi$ et donc $k : \{a : \gamma, \psi\} \vdash_\ell a@k : \gamma@\psi$.
(2) Si $u = k$, alors $\Delta = k : \psi$ et il suffit de définir μ par $\mu t = \mu\rho_\emptyset = \mu\rho'_\emptyset = \psi$.
Si $u = a@k$ on a $\Delta = k : \{a : \gamma, \psi\} \vdash_\ell a@k : \gamma@\psi$, définissons μ par

$$\mu\alpha = \begin{cases} \psi & \text{si } \alpha = \rho_\emptyset \text{ ou } \alpha = \rho'_\emptyset \\ \gamma & \text{si } \alpha = h \end{cases}$$

De la condition $a \notin \mathit{nm}(\gamma@\psi)$, on a bien μ qui respecte $\{a\} \times \{h, \rho_\emptyset\}$. □

Dans la règle (p_2), on utilise cette fonction quand le nom reçu n'est pas un nom simple. On produit alors un séquent pour typer le corps de la réception dans le contexte initial étendu avec le contexte fourni par *gen*. La composition des deux contextes ne pose pas de problème puisque les noms reçus n'ont pas d'occurrence dans le contexte initial et toutes les variables de types introduites sont fraîches. Par exemple, si le paramètre formel est $u = a@k$, alors le contexte initial $\mathbf{\Gamma}$ est étendu avec $k : \{a : h, \rho_{\{a\}}\}$ et, puisque k était un nom lié, $\mathbf{\Gamma}$ n'assigne pas de type à k. Finalement, les contraintes de types $\mathcal{C}$ et de sous-types $\mathcal{I}$ sont complétées avec celles produites par *gen* ainsi qu'avec une contrainte sur le type de la localité courante dans $\mathbf{\Gamma}$ qui garantit l'adéquation entre le type du canal récepteur et le type de l'objet $a@k$ de la communication. Il est encore à noter que la relation de liaison $\mathcal{R}$ est mise à jour de sorte que toutes les variables de types couramment connues ne puissent être substituées par un type contenant a (c'est-à-dire le nom de canal lié).

Quand le nom reçu est un nom simple b (règle p_{2a}), le type ψ assigné par $\mathbf{\Gamma}$ à la localité courante ℓ doit être étendu avec un type pour b. Cette extension est réalisée par la substitution d'un type de localité qui type b à la variable de rangée $\rho_\emptyset$ de ψ. Le contexte $\mathbf{\Delta}$ obtenu ainsi est décrit par la

condition de la règle (p_{2a}). Afin de garder la cohérence entre la nouvelle variable de rangée du type de ℓ dans $\mathbf{\Delta}$ avec celle dans $\mathbf{\Gamma}$ (pouvant encore avoir des occurrences dans d'autres types), on rajoute une contrainte égalisant ces variables de rangée ($\mathcal{C}''$).

Les règles de transition (p_5) à (p_6) pour la restriction ainsi (p_{12a}) et (p_{12b}) pour le corps des processus paramétrés appliquent le même principe que pour les restrictions.

Exemple 12.1.3 À titre de petit exemple, supposons qu'on souhaite générer une contrainte pour le réseau

$$S = \ell[\![\overline{a}(b)]\!]$$

Partant du contexte minimal $\ell : \rho_\emptyset$ et avec $\mathcal{X} = \{\rho_\emptyset\}$, la séquence de réductions est la suivante :

$$\begin{array}{lll}
 & (\{\ell : \rho_\emptyset \vdash \ell[\![\overline{a}(b)]\!]\}, \emptyset, \emptyset)^{\mathcal{X}}_{\emptyset} & \\
\rightsquigarrow & (\{\ell : \rho_\emptyset \vdash_\ell \overline{a}(b)\}, \emptyset, \emptyset)^{\mathcal{Y}}_{\emptyset} & \mathcal{Y} = \mathcal{X} \cup \{t, \rho'_{\{a\}}\} \quad (net_1) \\
\rightsquigarrow & (\{\ell : \rho_\emptyset \vdash^W_\ell b : t\}, \{\rho_\emptyset \doteq \{a : Ch(t), \rho'_{\{a\}}\}\}, \emptyset)^{\mathcal{Y}}_{\emptyset} & (p_1) \\
\rightsquigarrow & (\emptyset, \{\rho_\emptyset \doteq \{a : Ch(t), \rho'_{\{a\}}\}, \rho_\emptyset \doteq \{b : h, \rho''_{\{b\}}\}, t \doteq h\}, \emptyset)^{\mathcal{Z}}_{\emptyset} & \mathcal{Z} = \mathcal{Y} \cup \{\rho''_{\{b\}}, h\} \quad (n_2)
\end{array}$$

La contrainte générée $(\{\rho_\emptyset \doteq \{a : Ch(t), \rho'_{\{a\}}\}, \rho_\emptyset \doteq \{b : t, \rho''_{\{b\}}\}, t \doteq h\}, \emptyset)_\emptyset$ a pour solution principale sur $\mathcal{Z} = \{t, \rho_\emptyset, \rho'_{\{a\}}, \rho''_{\{b\}}\}$

$$\mu = [\{a : Ch(h), b : t, \rho'''_{\{a,b\}}\}/\rho_\emptyset, \{b : h, \rho'''_{\{b\}}\}/\rho'_{\{a\}}, \{a : Ch(h), \rho'''_{\{a,b\}}\}/\rho''_{\{b\}}, h/t]$$

Soit $\lambda = [\{\}/\rho'''_{\{a,b\}}, \gamma/h]$, alors en appliquant $\lambda\mu$ au contexte initial on obtient le séquent valide

$$\ell : \{a : Ch(\gamma), b : \gamma, \rho'''_{\{a,b\}}\} \vdash \ell[\![\overline{a}(b)]\!]$$

□

On donne à présent les résultats principaux concernant la réduction que nous venons de décrire. Le premier établit sa terminaison.

Lemme 12.1.4 (Terminaison) *Toute séquence de réductions*

$$(\mathcal{J}, \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{J}_1, \mathcal{C}_1, \mathcal{I}_1)^{\mathcal{X}_1}_{\mathcal{R}_1} \rightsquigarrow \dots$$

termine soit sur $\bot$ *soit sur* $(\emptyset, \mathcal{C}', \mathcal{I}')^{\mathcal{Y}}_{\mathcal{R}'}$.

Preuve:

- n_t : la somme des tailles des réseaux et des processus dans $\mathcal{J}$;
- n_n : le nombres de séquents $\mathbf{\Gamma} \vdash^W_\ell$ dans $\mathcal{J}$.

En associant la paire (n_t, n_n) à chaque tuple $(\mathcal{J}, \mathcal{C}, \mathcal{I})^{\mathcal{X}}_{\mathcal{R}}$, et $(0, 0)$ à $\bot$, il est facile de voir que cette paire décroît strictement pour l'ordre lexicographique. En particulier, les règles (n_1) à (nc) de la figure 12.1 font décroître n_n en laissant inchangé n_t. Toutes les autres règles des figures 12.2 à 12.4 font décroître n_t. Ceci montre la terminaison de $\rightsquigarrow$. □

Comme pour l'algorithme d'unification on définit la notion de **bonne configuration** pour les tuples $(\mathcal{C}, \mathcal{I}, \mu)^{\mathcal{X}}_{\mathcal{R}}$.

Définition 12.1.5 *On dit que $(\mathcal{J},\mathcal{C},\mathcal{I})^{\mathcal{X}}_{\mathcal{R}}$ est une* bonne configuration *si les noms liés de $\mathcal{J}$ sont distincts entre eux et distincts des noms libres et si :*

1. *var$(\mathcal{J},\mathcal{C},\mathcal{I}) \subseteq \mathcal{X}$;*
2. *$\mathcal{J} :: \mathcal{R}$, $\mathcal{C} :: \mathcal{R}$, et $\mathcal{I} :: \mathcal{R}$;*
3. *bn$(\mathcal{J}) \cap$ im$(\mathcal{R}) = \emptyset$;*
4. *Pour tout $\mathbf{\Gamma} \in \mathcal{J}$ on a dom$(\mathbf{\Gamma}) \cap$ bn$(\mathcal{J}) = \emptyset$ et tout $\ell \in$ dom$(\mathbf{\Gamma})$ on a $\mathbf{\Gamma}(\ell)$ est de la forme $\{a_1 : h_1, \ldots, a_n : h_n, \rho_\emptyset\}$ avec $\mathcal{R}(\rho_\emptyset) =$ im$(\mathcal{R})$,*

La propriété de bonne configuration est préservée par réduction.

Lemme 12.1.6 (Invariance de bonne configuration) *Si $(\mathcal{J},\mathcal{C},\mathcal{I})^{\mathcal{X}}_{\mathcal{R}}$ est une bonne configuration et $(\mathcal{J},\mathcal{C},\mathcal{I})^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow (\mathcal{J}',\mathcal{C}',\mathcal{I}')^{\mathcal{Y}}_{\mathcal{R}'}$ alors $(\mathcal{J}',\mathcal{C}',\mathcal{I}')^{\mathcal{Y}}_{\mathcal{R}'}$ est une bonne configuration.*

Preuve: Les preuves des points 1, 3 et 4 de la définition 12.1.5 sont triviales. On ne fait la preuve de $\mathcal{J}' :: \mathcal{R}', \mathcal{C}' :: \mathcal{R}', \mathcal{I}' :: \mathcal{R}'$ que pour les règles les plus significatives, les autres règles se traitant de manière similaire ou sont triviales.
(n_2) Puisque $\mathbf{\Gamma} :: \mathcal{R}$, par le lemme 11.3.3, $\mathbf{\Gamma}(\ell) :: \mathcal{R}'$. Et on a bien $\{a : h, \rho_{\{a\}}\} :: (\emptyset, \mathcal{R}')$.
(p_{2a}) On montre seulement que $[\{b : h, \rho'_\emptyset\}/\rho_\emptyset]\mathbf{\Gamma}(\ell) :: \mathcal{R}'$. Soit $\mathcal{R}'' = \mathcal{R}' - (b, \rho_\emptyset)$, on montre que $[\{b : h, \rho'_\emptyset\}/\rho_\emptyset]$ respecte $\mathcal{R}''$. Il est clair que $\{b : h, \rho'_\emptyset\} :: \mathcal{R}''$ car $b \in \mathcal{R}''(\rho'_\emptyset)$. D'autre part, puisque $b \in$ bn$(\mathcal{J} \cup \{\mathbf{\Gamma} \vdash_\ell a(b).P\})$, par le point 3 de l'hypothèse de bonne configuration $b \notin \mathcal{R}''(\rho_\emptyset)$, donc nm$(\{b : h, \rho'_\emptyset\}) \cap \mathcal{R}''(\rho_\emptyset) = \emptyset$. Enfin, par définition de $\mathcal{R}'$ (et donc de $\mathcal{R}''$) on a bien $\mathcal{R}''(\rho_\emptyset) \subseteq \mathcal{R}''(\rho'_\emptyset)$, donc $[\{b : h, \rho'_\emptyset\}/\rho_\emptyset]$ respecte $\mathcal{R}''$. De $\mathbf{\Gamma} :: \mathcal{R}$ et par le lemme 11.3.2 on a alors $[\{b : h, \rho'_\emptyset\}/\rho_\emptyset]\mathbf{\Gamma}(\ell) :: \mathcal{R}''$ et par le lemme 11.3.2, $[\{b : h, \rho'_\emptyset\}/\rho_\emptyset]\mathbf{\Gamma} :: \mathcal{R}'$. Enfin, $[\{b : h, \rho'_L\}/\rho_L]\mathbf{\Gamma}(\ell)$ satisfait évidemment le point 4 de la définition de bonne formation.
(p_{2b}) Par les lemmes 12.1.2 et 11.3.3, on a $\mathcal{I}' :: \mathcal{R}'$, $\mathcal{C}' :: \mathcal{R}'$ et $\Delta :: \mathcal{R}'$. Encore une fois, la condition 4 de bonne formation est très facile à vérifier pour Δ. □

Proposition 12.1.7 (Préservation) *Soit $(\mathcal{J},\mathcal{C},\mathcal{I})^{\mathcal{X}}_{\mathcal{R}}$ est une bonne configuration et*

$$(\mathcal{J},\mathcal{C},\mathcal{I})^{\mathcal{X}}_{\mathcal{R}} \rightsquigarrow^* (\mathcal{J}',\mathcal{C}',\mathcal{I}')^{\mathcal{Y}}_{\mathcal{R}'}$$

1. *Si λ est solution de base de $(\mathcal{J},\mathcal{C},\mathcal{I})$ qui respecte $\mathcal{R}$ avec nm(im$(\lambda)) \cap$ bn$(\mathcal{J}) = \emptyset$, alors il existe $\mu =_{\mathcal{X}} \lambda$ avec nm(im$(\mu)) \cap$ bn$(\mathcal{J}') = \emptyset$, et μ est solution de $(\mathcal{J}',\mathcal{C}',\mathcal{I}')$ qui respecte $\mathcal{R}'$.*
2. *Si λ est solution de base de $(\mathcal{J}',\mathcal{C}',\mathcal{I}')$ qui respecte $\mathcal{R}'$, alors λ est aussi solution de $(\mathcal{J},\mathcal{C},\mathcal{I})$.*

Preuve: (1) Par induction sur la longueur de la séquence de réductions. On ne fait la preuve que pour les règles les plus significatives.
(n_1) De $\lambda\mathbf{\Gamma} \vdash^W_\ell k : \lambda t$ on déduit que $\lambda\mathbf{\Gamma} = \Delta, k : \lambda t \sqcap \psi$ avec $k \notin \mathbf{\Delta}'$. Soient $\mathbf{\Gamma}'$ et ψ tels que $\mathbf{\Gamma} = \mathbf{\Gamma}', k : \psi$, alors $\Delta = \lambda\mathbf{\Gamma}'$ et $\lambda\psi = \lambda t \sqcap \psi$ $(*)$. Définissons μ par

$$\mu\alpha = \begin{cases} \lambda t & \text{si } \alpha = \rho_\emptyset \\ \lambda\alpha & \text{sinon.} \end{cases}$$

Il est clair que μ respecte $\mathcal{R}$. De $(*)$, on a $\lambda t \sqcap \phi <: \lambda t$ et donc μ satisfait $\mathbf{\Gamma}(k) \lessdot \rho_\emptyset$.
(n_2) De $\lambda\mathbf{\Gamma} \vdash^W_\ell a\mathbf{:}\,\lambda t$, on a $\lambda\mathbf{\Gamma} = \Delta, \ell\mathbf{:}\,\{a\mathbf{:}\,\lambda t\} \sqcap \phi$ et donc $\lambda\mathbf{\Gamma}(\ell) = \{a\mathbf{:}\,\lambda t\} \sqcap \phi$ $(*)$. Définissons μ par

$$\mu\alpha = \begin{cases} \lambda t & \text{si } \alpha = h \\ \{\} & \text{si } \alpha = \rho_{\{a\}} \\ \lambda\alpha & \text{sinon.} \end{cases}$$

μ respecte bien $\mathcal{R}$ et par $(*)$, c'est bien une solution de $\mathbf{\Gamma}(\ell) \lessdot \{a\mathbf{:}\,h, \rho_{\{a\}}\}$.
(n_3) De $\lambda\mathbf{\Gamma} \vdash^W_\ell a@k\mathbf{:}\,\lambda t$ on déduit que $\lambda t = \gamma@\psi$ et $\lambda\mathbf{\Gamma} = \Delta, k\mathbf{:}\,\{a\mathbf{:}\,\gamma, \psi\} \sqcap \phi$ (avec $k \notin \mathit{dom}(\Delta)$). Il suffit donc de définir μ par

$$\mu\alpha = \begin{cases} \gamma & \text{si } \alpha = h \\ \psi & \text{si } \alpha = \rho_{\{a\}} \text{ ou } \alpha = \rho'_\emptyset \\ \lambda\alpha & \text{sinon.} \end{cases}$$

(p_1) De $\lambda\mathbf{\Gamma} \vdash_\ell \overline{a}(u)$ on a $\lambda\mathbf{\Gamma}(\ell) = \{a\mathbf{:}\,\mathit{Ch}(\tau), \psi\}$ (1) et $\lambda\mathbf{\Gamma} \vdash^W_\ell u\mathbf{:}\,\tau$ (2). Définissons μ par

$$\mu\alpha = \begin{cases} \tau & \text{si } \alpha = t \\ \psi & \text{si } \alpha = \rho_{\{a\}} \\ \lambda\alpha & \text{sinon.} \end{cases}$$

Et, μ est bien solution de $\mathbf{\Gamma} \vdash^W_\ell u\mathbf{:}\,t$ (par (2)) et de $\mathbf{\Gamma}(\ell) \doteq \{a\mathbf{:}\,\mathit{Ch}(t), \rho_{\{a\}}\}$ (par (1)).
(p_{2a}) De $\lambda\mathbf{\Gamma} \vdash_\ell a(b).P$ on a $\lambda\mathbf{\Gamma}, \ell\mathbf{:}\,\{b\mathbf{:}\,\gamma\} \vdash_\ell P$ et $\lambda\mathbf{\Gamma}(\ell) = \{a\mathbf{:}\,\mathit{Ch}(\gamma), \psi\}$ avec $b \notin \mathit{nm}(\lambda\mathbf{\Gamma})$ $(*)$. Définissons μ par

$$\mu\alpha = \begin{cases} \gamma & \text{si } \alpha = h \\ \psi & \text{si } \alpha = \rho''_{\{a\}} \\ \lambda\rho_\emptyset & \text{si } \alpha = \rho'_\emptyset \\ \lambda\alpha & \text{sinon.} \end{cases}$$

μ est bien solution de $\mathbf{\Gamma}(\ell) \doteq \{a\mathbf{:}\,\mathit{Ch}(h), \rho''_{\{a\}}\}$ et de $\rho_\emptyset \doteqdot \rho'_\emptyset$, et on peut facilement vérifier que $\mu\mathbf{\Delta} = \lambda\mathbf{\Gamma}, \ell\mathbf{:}\,\{b\mathbf{:}\,\gamma\}$. Montrons que μ respecte $\mathcal{R}'$. Si $\alpha = h$, de $\mathcal{R}'(h) = \{b\}$ et de $(*)$ on a $b \notin \mathit{nm}(\gamma)$ et donc $\mathit{nm}(\mu h) \cap \mathcal{R}'(h) = \emptyset$. D'autre part, γ étant un type de base bien formé, on a directement $\gamma :: \mathcal{R}'$. Si $\alpha = \rho''_{\{a\}}$, de la même façon on a $\mathit{nm}(\mu\rho''_{\{a\}}) \cap \mathcal{R}'(\rho''_{\{a\}}) = \emptyset$ et $\psi :: \mathcal{R}'$. De $\{a\mathbf{:}\,\mathit{Ch}(\gamma), \psi\}$ bien formé, on a $a \notin \mathit{dom}(\psi)$. Si $\alpha = \rho'_\emptyset$, de $\mathcal{R}'(\rho'_\emptyset) = \mathcal{R}(\rho_\emptyset) \cup \{b\}$ et de λ respecte $\mathcal{R}$, on a $\mathit{nm}(\mu\rho'_\emptyset) \cap \mathcal{R}'(\rho'_\emptyset)$ et $\mu\rho'_\emptyset :: \mathcal{R}$ et par le lemme 11.3.2 $\mu\rho'_\emptyset :: \mathcal{R}'$. Enfin. si $\alpha \in \mathcal{X}$, $\mathcal{R}'(\alpha) = \mathcal{R}(\alpha) \cup \{b\}$, or de $(*)$, $b \notin \mathit{nm}(\mu\alpha)$. De plus, de λ respecte $\mathcal{R}$, on a $\mathit{nm}(\mu\alpha) \cap \mathcal{R}(\alpha) = \emptyset$ et donc $\mathit{nm}(\mu\alpha) \cap \mathcal{R}'(\alpha) = \emptyset$. De $\mu\alpha :: \mathcal{R}$, par le lemme 11.3.2, on a $\mu\alpha :: \mathcal{R}'$. Finalement μ respecte $\mathcal{R}'$.
(p_{2b}) De $\lambda\mathbf{\Gamma} \vdash_\ell a(u).P$, on déduit que $\lambda\mathbf{\Gamma} = \Gamma', \ell\mathbf{:}\,\{a\mathbf{:}\,\mathit{Ch}(\tau), \psi\}$ avec $\ell \notin \mathit{dom}(\Gamma')$, et $\lambda\mathbf{\Gamma}, \Delta \vdash_\ell P$ avec $\Delta \vdash_\ell u\mathbf{:}\,\tau$. Puisque u est lié, il n'a pas d'occurrence dans $\lambda\mathbf{\Gamma}$ et donc en particulier dans τ. On peut alors appliquer le lemme 12.1.2 et il existe une substitution μ' telle que $\mu'\mathbf{\Delta} = \Delta$ et $\mu' t = \tau$. Définissons μ par

$$\mu\alpha = \begin{cases} \mu'\alpha & \text{si } \alpha \in \mathcal{X}' \cup \{t\} \\ \psi & \text{si } \alpha = \rho_{\{a\}} \\ \lambda\alpha & \text{sinon.} \end{cases}$$

Alors, il est clair que $\mu\mathbf{\Gamma}, \mu\mathbf{\Delta} = \lambda\mathbf{\Gamma}, \Delta$ et donc que μ est solution de $\mathbf{\Gamma}, \mathbf{\Delta} \vdash_\ell P$ et de $\mathbf{\Gamma}(\ell) = \{a : Ch(h), \rho_{\{a\}}\}$. Montrons que μ respecte $\mathcal{R}'$. Si $\alpha \in \mathcal{X}'$ on a $\mathcal{R}'(\alpha) = \mathcal{R}''(\alpha)$, et du lemme 12.1.2, μ' respecte $\mathcal{R}''$, on en déduit que α satisfait les conditions de la définition 11.1.2 pour $\mathcal{R}'$. Si $\alpha = t$ ou , on a $\mathcal{R}'(t) = \mathit{chan}(u)$, or on sait que $\mathit{chan}(u) \notin \mathit{nm}(\tau)$. Si $\alpha = \rho_{\{a\}}$, on a $\mathcal{R}'(\rho_{\{a\}}) = \mathit{chan}(u)$, et par hypothèse $\mathit{chan}(u) \notin \mathit{nm}(\rho_{\{a\}})$. D'autre part, $\{a : Ch(\tau), \psi\}$ étant bien formé, on a $a \notin \mathit{dom}(\psi)$. Si $\alpha \in \mathcal{X}$, alors $\mathcal{R}'(\alpha) = \mathcal{R}(\alpha) \cup \mathit{chan}(u)$ et il suffit de rappeler que λ respecte $\mathcal{R}$ et que, par hypothèse, $\mathit{nm}(\mathit{im}(\lambda)) \cap \mathit{chan}(u) = \emptyset$. Et finalement, μ respecte $\mathcal{R}$.
(2) Par induction sur la longueur de la séquence de réductions. On ne fait la preuve que pour les règles les plus significatives. On suppose que $\mathcal{Y} \subseteq \mathit{dom}(\lambda)$.
(n_2) Si λ est solution de base de $t \doteq h$ et $\mathbf{\Gamma} \lessdot \{a : h, \rho_{\{a\}}\}$ $(*)$, alors il existe ψ et γ tels que $\lambda h = \lambda t = \gamma$ et $\lambda\rho_{\{a\}} = \psi$. Alors, de $(*)$, $\lambda\mathbf{\Gamma}(\ell) = \{a : \gamma, \psi\} \sqcap \phi$ et donc $\lambda\mathbf{\Gamma}(\ell) \vdash^W_\ell a : \gamma$ d'où λ est solution de $\mathbf{\Gamma} \vdash^W_\ell a : t$.
(n_3) Si λ est solution de base de $t \doteq h@\rho_\emptyset$, $\rho_{\{a\}} \dot{\equiv} \rho'_\emptyset$ et $\mathbf{\Gamma}(k) \lessdot \{a : h, \rho_{\{a\}}\}$ $(*)$, alors il existe γ et ψ tels que $\lambda t = \gamma@\psi$, $\lambda h = \gamma$, $\lambda\rho'_\emptyset = \lambda\rho_\emptyset = \psi$. Donc, de $(*)$ il existe ϕ tel que $\lambda\mathbf{\Gamma}(k) = \{a : \gamma, \psi\} \sqcap \phi$ et donc $\lambda\mathbf{\Gamma}(k) \vdash^W_\ell a@k : \gamma@\psi$. Et finalement λ est solution de $\mathbf{\Gamma} \vdash^W_\ell a@k : t$.
(p_1) Si λ solution de base de $\mathbf{\Gamma} \vdash^W_\ell u : t$ et de $\mathbf{\Gamma}(\ell) \doteq \{a : Ch(t), \rho_{\{a\}}\}$, il existe τ et ψ tel que $\lambda t = \tau$ et $\rho_{\{a\}} = \psi$. Donc, de $\lambda\mathbf{\Gamma}(\ell) = \{a : Ch(\tau), \psi\}$ et de $\lambda\mathbf{\Gamma} \vdash^W_\ell u : \tau$, par la règle de typage pour l'émission, on a $\lambda\mathbf{\Gamma} \vdash_\ell \overline{a}(u)$.
(p_{2a}) Si λ solution de base de $\mathbf{\Delta} \vdash_\ell P$ (1), $\mathbf{\Gamma}(\ell) \doteq \{a : Ch(h), \rho''_{\{a\}}\}$ (2) et de $\rho_\emptyset \dot{\equiv} \rho'_\emptyset$ (3), alors il existe ψ et γ tels que $\lambda h = \gamma$ et $\lambda\rho''_{\{a\}} = \psi$. De (2), on a $\lambda\mathbf{\Gamma}(\ell) = \{a : Ch(\gamma), \psi\}$ et par définition de $\mathbf{\Delta}$ et par (3), $\lambda\mathbf{\Delta}(\ell) = \{b : \gamma, \lambda\mathbf{\Gamma}(\ell)\}$. D'autre part, puisque pour tout α dans $\mathcal{Y}$, $b \in \mathcal{R}'(\alpha)$ et que λ respecte $\mathcal{R}'$, on a $b \notin \lambda\mathbf{\Gamma}$. D'où, finalement on peut appliquer la règle de typage pour la réception d'un nom simple et $\lambda\mathbf{\Gamma} \vdash_\ell a(b).P$.
(p_{2b}) Si λ solution de base de $\{\mathbf{\Gamma}, \mathbf{\Delta} \vdash_\ell P\}$, $\{\mathbf{\Gamma}(\ell) \doteq \{a : Ch(t), \rho_{\{a\}}\}\} \cup \mathcal{C}'$ et $\mathcal{I}'$ qui respecte $\mathcal{R}'$. Alors par le lemme 12.1.2, on a $\lambda\mathbf{\Delta} \vdash_\ell u : \lambda t$. D'autre part, par hypothèse de bonne configuration et par le fait que $\mathit{chan}(u) \in \mathcal{R}(\alpha)$ pour tout $\alpha \in \mathcal{Y}$ et λ respecte $\mathcal{R}'$, on a $\mathit{nm}(u) \cap \mathit{nm}(\lambda\mathbf{\Gamma}) = \emptyset$. On peut donc appliquer la règle de typage pour la réception, et on a $\lambda\mathbf{\Gamma} \vdash_\ell a(u).P$. □

Avant d'établir les théorèmes de correction et de complétude de notre algorithme d'inférence il convient de définir formellement la notion de typage principal que nous avions esquissé dans le chapitre préliminaire. Mais faisons auparavant une remarque.

Remarque 12.1.8 On rappelle la remarque faite dans le chapitre 10 de manière plus formelle : si $\mathbf{\Gamma}$ est un contexte initial pour S et que $\Delta \vdash S$, alors il existe une substitution λ telle que $\mathit{nm}(\mathit{im}(\lambda)) \cap \mathit{bn}(S) = \emptyset$ et $\lambda\mathbf{\Gamma} =_{\mathit{dom}(\mathbf{\Gamma})} \Delta$ et $\lambda\mathbf{\Gamma} \vdash S$.

Définition 12.1.9 *On dit que $\mathbf{\Gamma}; \mathcal{A}$ est un typage principal pour S, si*

1. *pour toute $\mathcal{A}$-substitution de base λ telle que $\mathit{nm}(\mathit{im}(\lambda)) \cap \mathit{bn}(S) = \emptyset$ on a $\lambda\mathbf{\Gamma} \vdash S$;*
2. *pour tout Δ tel que $\Delta \vdash S$, il existe une substitution de base λ qui préserve $\mathcal{A}$ telle que $\Delta =_{\mathit{dom}(\mathbf{\Gamma})} \lambda\mathbf{\Gamma}$ et $\lambda\mathbf{\Gamma} \vdash S$.*

Théorème 12.1.10 (Correction) *Soit S un réseau dont tous les noms liés sont distincts entre eux*

et distincts des noms libres, $\mathbf{\Gamma}$ *un contexte initial pour* S*, et* $\mathcal{X} = var(\mathbf{\Gamma})$*. Si*

$$(\mathbf{\Gamma} \vdash S, \emptyset, \emptyset)^{\mathcal{X}}_{\emptyset} \rightsquigarrow^* (\emptyset, \mathcal{C}, \mathcal{I})^{\mathcal{Y}}_{\mathcal{R}} \quad et \quad (\mathcal{C}, \mathcal{I}, \emptyset)^{\mathcal{Y}}_{\mathcal{R}} \rightsquigarrow^* (\emptyset, \mathcal{A}, \mu)^{\mathcal{Z}}_{\mathcal{R}'} \not\rightsquigarrow$$

alors $\mu\mathbf{\Gamma}; \mathcal{A}$ *est un typage principal pour* S*.*

Preuve: On vérifie facilement que $(\mathbf{\Gamma} \vdash S, \emptyset, \emptyset)^{\mathcal{X}}_{\emptyset}$ et $(\mathcal{C}, \mathcal{I}, \emptyset)^{\mathcal{Y}}_{\mathcal{R}}$ sont de bonnes configurations. Montrons le point 1 de la définition 12.1.9. Soit λ une substitution qui préserve $\mathcal{A}$ telle que $nm(im(\lambda)) \cap bn(S) = \emptyset$. Faisons l'hypothèse – raisonnable – que λ ne substitue que des types de bases bien formés. On peut facilement vérifier que $im(\mathcal{R}') = bn(S)$ et donc que λ respecte $\mathcal{R}'$. Par la deuxième réduction de l'énoncé et le lemme 11.4.5 (2), $\lambda\mu$ est solution de $(\mathcal{C}, \mathcal{I})$ qui respecte $\mathcal{R}'$. $\lambda\mu$ étant une substitution de base et $\mathcal{R} \subseteq \mathcal{R}'$, on vérifie facilement que $\lambda\mu$ respecte également $\mathcal{R}$. Donc, par la deuxième réduction et par le lemme 12.1.7 (2) $\lambda\mu$ est solution de $\mathbf{\Gamma} \vdash S$ et donc $\lambda\mu\mathbf{\Gamma} \vdash S$.
Montrons le point 2 de la définition 12.1.9. D'après la remarque 12.1.8, si $\Delta \vdash S$, alors il existe une substitution λ telle que $\lambda\mathbf{\Gamma} \vdash S$ et $\Delta =_{dom(\mathbf{\Gamma})} \lambda\mathbf{\Gamma}$. De la première réduction de l'énoncé et par la proposition 12.1.7 (1), il existe λ' telle que $\lambda' =_{\mathcal{X}} \lambda$ et solution de $(\mathcal{C}, \mathcal{I})$ qui respecte $\mathcal{R}$. De la deuxième réduction et par la proposition 11.4.5 (1), il existe λ'' tel que $\lambda' = \lambda''\mu$ et λ'' préserve $\mathcal{A}$. □

Lemme 12.1.11 *Soient* $\mathbf{\Gamma}$ *un contexte initial pour* S *et* $\mathcal{X} = var(\mathbf{\Gamma})$*, si* $(\mathbf{\Gamma} \vdash S, \emptyset, \emptyset)^{\mathcal{X}}_{\emptyset} \rightsquigarrow^* \perp$ *alors* S *n'est pas typable.*

Preuve: On sait qu'il n'y a pas de séquence infinie de réductions, on procède simplement par induction sur la longueur de la séquence de réductions en examinant les cas menant à une configuration d'erreur. □

Théorème 12.1.12 (Complétude) *Soit* S *un réseau typable,* $\mathbf{\Gamma}$ *un contexte initial pour* S*, et* $\mathcal{X} = var(\mathbf{\Gamma})$*, alors*

$$(\mathbf{\Gamma} \vdash S, \emptyset, \emptyset)^{\mathcal{X}}_{\emptyset} \rightsquigarrow^* (\emptyset, \mathcal{C}, \mathcal{I})^{\mathcal{Y}}_{\mathcal{R}} \quad et \quad (\mathcal{C}, \mathcal{I}, \emptyset)^{\mathcal{Y}}_{\mathcal{R}} \rightsquigarrow^* (\emptyset, \mathcal{A}, \mu)^{\mathcal{Z}}_{\mathcal{R}'} \not\rightsquigarrow$$

Preuve: Par la remarque 12.1.8, il existe λ telle que $\lambda\mathbf{\Gamma} \vdash S$. D'autre part, par le lemme de terminaison et le lemme 12.1.11, on a la première réduction. Par lemme 11.4.5, $(\mathcal{C}, \mathcal{I})$ a une solution qui respecte $\mathcal{R}$, et donc par le lemme de terminaison et le lemme 11.4.7 on a la deuxième réduction. □

12.2 Un exemple

Dans cette section nous donnons un exemple complet de l'inférence du typage principal du terme S_3 donné dans le chapitre 10.

$$S_3 = \ell[\![d(b).(\nu a)Q]\!] \quad \text{où} \quad Q = (\overline{a}(\ell) \mid a(k).\mathsf{go}\, k.\overline{b}(c)$$

Soit le contexte initial $\mathbf{\Gamma} = \ell : \rho_\emptyset, c : \mathit{val}$ pour S_3, et $\mathcal{X}_0 = \{\rho_\emptyset\}$, alors la séquence de génération des contraintes est la suivante.

$$
\begin{array}{ll}
& (\{\ell : \rho_\emptyset, c : \mathit{val} \vdash S_3\},\ \emptyset,\ \emptyset)^{\mathcal{X}_0}_{\emptyset} \\
\rightsquigarrow & (\{\ell : \rho_\emptyset, c : \mathit{val} \vdash_\ell d(b).(\nu a)Q\},\ \emptyset,\ \emptyset)^{\mathcal{X}_0}_{\emptyset} \qquad (\mathit{net}_1) \\
\rightsquigarrow & (\{\ell : \{b : h, \rho'_\emptyset\}, c : \mathit{val} \vdash_\ell (\nu a)Q\},\ \mathcal{C}_0,\ \mathcal{I}_0)^{\mathcal{X}_1}_{\mathcal{R}_0} \qquad (p_{2a}) \\
& \text{où } \mathcal{C}_0 = \{\rho_\emptyset \doteq \{d : \mathit{Ch}(h), \rho^2_{\{d\}}\}\}, \mathcal{I}_0 = \{\rho_\emptyset \hat{=} \rho'_\emptyset\}, \mathcal{X}_1 = \mathcal{X}_0 \cup \{h, \rho'_\emptyset, \rho^2_{\{b\}}\}, \mathcal{R}_0 = \{b\} \times \mathcal{X}_1 \\
\rightsquigarrow & (\{\ell : \{b : h, a : h', \rho^3_\emptyset\}, c : \mathit{val} \vdash_\ell Q\},\ \mathcal{C}_0,\ \mathcal{I}_1)^{\mathcal{X}_2}_{\mathcal{R}_1} \qquad (p_5) \\
& \text{où } \mathcal{I}_1 = \mathcal{I}_0 \cup \{\rho'_\emptyset \hat{=} \rho^3_\emptyset\}, \mathcal{X}_2 = \mathcal{X}_1 \cup \{\rho^3_\emptyset, h'\}, \mathcal{R}_1 = \mathcal{R}_0 \cup \{b\} \times \{\rho^3_\emptyset\} \cup \{a\} \times \mathcal{X}_2 \\
\rightsquigarrow & (\{\ldots \vdash_\ell a(k).\mathsf{go}\, k.\overline{b}(c), \ell : \{b : h, a : h', \rho^3_\emptyset\}, c : \mathit{val} \vdash_\ell \ell : t\},\ \mathcal{C}_1,\ \mathcal{I}_1)^{\mathcal{X}_3}_{\mathcal{R}_1} \qquad (p_1) \\
& \text{où } \mathcal{C}_1 = \mathcal{C}_0 \cup \{\{b : h, a : h', \rho^3_\emptyset\} \doteq \{a : \mathit{Ch}(t), \rho^4_{\{a\}}\}\}, \mathcal{X}_3 = \mathcal{X}_2 \cup \{\rho^4_{\{a\}}, t\} \\
\rightsquigarrow & (\{\ell : \{b : h, a : h', \rho^3_\emptyset\}, c : \mathit{val} \vdash_\ell a(k).\mathsf{go}\, k.\overline{b}(c)\},\ \mathcal{C}_2,\ \mathcal{I}_2)^{\mathcal{X}_4}_{\mathcal{R}_1} \qquad (n_1) \\
& \text{où } \mathcal{C}_2 = \mathcal{C}_1 \cup \{t \doteq \rho^5_\emptyset\}, \mathcal{I}_2 = \mathcal{I}_1 \cup \{\{b : h, a : h', \rho^3_\emptyset\} \lessdot \rho^5_\emptyset\}, \mathcal{X}_4 = \mathcal{X}_3 \cup \{\rho^5_\emptyset, h\} \\
\rightsquigarrow & (\{k : \rho^6_\emptyset, \ell : \{b : h, a : h', \rho^3_\emptyset\}, c : \mathit{val} \vdash_\ell \mathsf{go}\, k.\overline{b}(c)\},\ \mathcal{C}_3,\ \mathcal{I}_3)^{\mathcal{X}_5}_{\mathcal{R}_1} \qquad (p_{2b}) \\
& \text{où } \mathcal{C}_3 = \mathcal{C}_2 \cup \{\{b : h, a : h', \rho^3_\emptyset\} \doteq \{a : \mathit{Ch}(t'), \rho^8_{\{a\}}\}, t' \doteq \rho^6_\emptyset\}, \mathcal{I}_3 = \mathcal{I}_2 \cup \{\rho^6_\emptyset \hat{=} \rho^7_\emptyset\} \\
& \quad \mathcal{X}_5 = \{t', \rho^6_\emptyset, \rho^7_\emptyset, \rho^8_{\{a\}}\} \\
\rightsquigarrow & (\{k : \rho^6_\emptyset, \ell : \{b : h, a : h', \rho^3_\emptyset\}, c : \mathit{val} \vdash_k \overline{b}(c)\},\ \mathcal{C}_3,\ \mathcal{I}_3)^{\mathcal{X}_5}_{\mathcal{R}_1} \qquad (p_9) \\
\rightsquigarrow & (\{k : \rho^6_\emptyset, \ell : \{b : h, a : h', \rho^3_\emptyset\}, c : \mathit{val} \vdash_k c : t''\},\ \mathcal{C}_4,\ \mathcal{I}_3)^{\mathcal{X}_5}_{\mathcal{R}_1} \qquad (p_1) \\
& \text{où } \mathcal{C}_4 = \mathcal{C}_3 \cup \{\rho^6_\emptyset \doteq \{b : \mathit{Ch}(t''), \rho^9_{\{b\}}\}\} \\
\rightsquigarrow & (\emptyset,\ \mathcal{C}_5,\ \mathcal{I}_3)^{\mathcal{X}_6}_{\mathcal{R}_1} \qquad (n_4) \\
& \text{où } \mathcal{C}_5 = \mathcal{C}_4 \cup \{\mathit{val} \doteq \mathit{val}, t'' \doteq \mathit{val}\}, \mathcal{X}_6 = \mathcal{X}_5 \cup \{t''\}
\end{array}
$$

On obtient donc les contraintes de types et de sous-types suivantes (où on a retiré l'équation triviale $\mathit{val} \doteq \mathit{val}$) :

$$\begin{aligned}
\mathcal{C}_5 &= \{\rho_\emptyset \doteq \{d\colon \mathit{Ch}(h), \rho^2_{\{d\}}\},\ \{b\colon h, a\colon h', \rho^3_\emptyset\} \doteq \{a\colon \mathit{Ch}(t), \rho^4_{\{a\}}\},\ t \doteq \rho^5_\emptyset,\\
&\quad \{b\colon h, a\colon h',\ \rho^3_\emptyset\} \doteq \{a\colon \mathit{Ch}(t'), \rho^8_{\{a\}}\},\ t' \doteq \rho^6_\emptyset,\ \rho^6_\emptyset \doteq \{b\colon \mathit{Ch}(t''), \rho^9_{\{b\}}\},\ t'' \doteq \mathit{val}\}\\
\mathcal{I}_3 &= \{\rho_\emptyset \doteqdot \rho'_\emptyset,\ \rho'_\emptyset \doteqdot \rho^3_\emptyset,\ \{b\colon h, a\colon h', \rho^3_\emptyset\} \lessdot \rho^5_\emptyset,\ \rho^6_\emptyset \doteqdot \rho^7_\emptyset\}
\end{aligned}$$

Et la relation de liaison :

$$\mathcal{R}_1 = \{a, b\} \times \{\rho_\emptyset, \rho'_\emptyset, \rho^2_{\{d\}}, \rho^3_\emptyset, h\} \cup \{a\} \times \{h'\}$$

Nous donnons à présent la séquence de réductions pour l'unification de la contrainte $(\mathcal{C}_5, \mathcal{I}_3)_{\mathcal{R}_1}$.

$$\begin{aligned}
&(\{t'' \doteq \mathit{val}\} \cup \mathcal{C}_4,\ \mathcal{I}_3,\ \emptyset)^{\mathcal{X}_6}_{\mathcal{R}_1}\\
\rightsquigarrow\ &(\{\rho^6_\emptyset \doteq \{b\colon \mathit{Ch}(\mathit{val}), \rho^9_{\{b\}}\}\} \cup \mu_0\mathcal{C}_3,\ \mu_0\mathcal{I}_3,\ \mu_0)^{\mathcal{X}_6}_{\mathcal{R}_1} && (\mathit{elim})\\
&\text{où } \mu_0 = [\mathit{val}/t'']\\
\rightsquigarrow\ &(\{\{b\colon h, a\colon h', \rho^3_\emptyset\} \doteq \{a\colon \mathit{Ch}(t'), \rho^8_{\{a\}}\}\} \cup \mathcal{C}'_2,\ \mu_1\mathcal{I}_3,\ \mu_1)^{\mathcal{X}_7}_{\mathcal{R}_1} && (\mathit{loc_end})\\
&\text{où } \mu_1 = [\{b\colon \mathit{Ch}(\mathit{val}), \rho^{10}_{\{b\}}\}/\rho^6_\emptyset,\ \rho^{10}_{\{b\}}/\rho^9_{\{b\}}]\mu_0,\ \mathcal{C}'_2 = \{t' \doteq \{b\colon \mathit{Ch}(\mathit{val}), \rho^{10}_{\{b\}}\}\} \cup \mu_1\mathcal{C}_2,\\
&\quad \mathcal{X}_7 = \mathcal{X}_6 \cup \{\rho^{10}_{\{b\}}\}\\
\rightsquigarrow\ &(\{h' \doteq \mathit{Ch}(t'), \{b\colon h, \rho^3_\emptyset\} \doteq \rho^8_{\{a\}}\} \cup \mathcal{C}'_2,\ \mu_1\mathcal{I}_3,\ \mu_1)^{\mathcal{X}_7}_{\mathcal{R}_1} && (\mathit{loc})\\
\rightsquigarrow\ &(\{\{b\colon h, \rho^3_\emptyset\} \doteq \rho^8_{\{a\}}\} \cup \mathcal{C}''_2,\ \mu_2\mathcal{I}_3,\ \mu_2)^{\mathcal{X}_8}_{\mathcal{R}_2} && (\mathit{elim})\\
&\text{où } \mu_2 = [\mathit{Ch}(t')/h']\mu_1, \mathcal{C}''_2 = [\mathit{Ch}(t')/h']\mathcal{C}'_2, \mathcal{R}_2 = \mathcal{R}_1 \cup \{a\} \times \{t'\}\\
\rightsquigarrow\ &(\{t' \doteq \{b\colon \mathit{Ch}(\mathit{val}), \rho^{10}_{\{b\}}\}\} \cup \mu_3\mathcal{C}_2,\ \mu_3\mathcal{I}_3,\ \mu_3)^{\mathcal{X}_9}_{\mathcal{R}_3} && (\mathit{loc_end})\\
&\text{où } \mu_3 = [\rho^{11}_\emptyset/\rho^3_\emptyset,\ \{b\colon h, \rho^{11}_\emptyset\}/\rho^8_{\{a\}}]\mu_2,\ \mathcal{R}_3 = \mathcal{R}_2 \cup \{a, b\} \times \{\rho^{11}_\emptyset\},\ \mathcal{X}_9 = \mathcal{X}_8 \cup \{\rho^{11}_\emptyset\}\\
\rightsquigarrow\ &(\{t \doteq \rho^5_\emptyset\} \cup \mu_4\mathcal{C}_1,\ \mu_4\mathcal{I}_3,\ \mu_4)^{\mathcal{X}_9}_{\mathcal{R}_4} && (\mathit{elim})\\
&\text{où } \mu_4 = [\{b\colon \mathit{Ch}(\mathit{val}), \rho^{10}_{\{b\}}\}/t']\mu_3,\ \mathcal{R}_4 = \mathcal{R}_3 \cup \{a\} \times \{\rho^{10}_{\{b\}}\}\\
\rightsquigarrow\ &(\{\{b\colon h, a\colon \mathit{Ch}(\{b\colon \mathit{Ch}(\mathit{val}), \rho^{10}_{\{b\}}\}), \rho^{11}_\emptyset\} \doteq \{a\colon \mathit{Ch}(\rho^5_\emptyset), \rho^4_{\{a\}}\}\} \cup \mu_5\mathcal{C}_0,\\
&\quad \mu_5\mathcal{I}_3,\ \mu_5)^{\mathcal{X}_9}_{\mathcal{R}_4} && (\mathit{elim})\\
&\text{où } \mu_5 = [\rho^5_\emptyset/t]\mu_4\\
\rightsquigarrow^*\ &(\{\{b\colon \mathit{Ch}(\mathit{val}), \rho^{10}_{\{b\}}\} \doteq \rho^5_\emptyset,\ \{b\colon h, \rho^{11}_\emptyset\} \doteq \rho^4_{\{a\}}\} \cup \mu_5\mathcal{C}_0,\ \mu_5\mathcal{I}_5,\ \mu_5)^{\mathcal{X}_9}_{\mathcal{R}_4} && (\mathit{loc} + \mathit{chan})\\
\rightsquigarrow\ &(\{\{b\colon h, \rho^{11}_\emptyset\} \doteq \rho^4_{\{a\}}\} \cup \mu_6\mathcal{C}_0,\ \mu_6\mathcal{I}_3,\ \mu_6)^{\mathcal{X}_{10}}_{\mathcal{R}_5} && (\mathit{loc_end})\\
&\text{où } \mu_6 = [\rho^{12}_{\{b\}}/\rho^{10}_\emptyset,\ \{b\colon \mathit{Ch}(\mathit{val}), \rho^{12}_{\{b\}}\}/\rho^5_\emptyset]\mu_5,\ \mathcal{R}_5 = \mathcal{R}_4 \cup \{a\} \times \{\rho^{12}_{\{b\}}\},\ \mathcal{X}_{10} = \mathcal{X}_9 \cup \{\rho^{12}_{\{b\}}\}\\
\rightsquigarrow\ &(\{\rho_\emptyset \doteq \{d\colon \mathit{Ch}(h), \rho^2_{\{d\}}\}\}, \mu_7\mathcal{I}_3,\ \mu_7)^{\mathcal{X}_{11}}_{\mathcal{R}_6} && (\mathit{loc_end})\\
&\text{où } \mu_7 = [\rho^{13}_\emptyset/\rho^{11}_\emptyset, \{b\colon h, \rho^{13}_\emptyset\}/\rho^4_{\{a\}}]\mu_6,\ \mathcal{R}_6 = \mathcal{R}_5 \cup \{a, b\} \times \{\rho^{13}_\emptyset\},\ \ X_{11} = \mathcal{X}_{10} \cup \{\rho^{13}_{\{d\}}\}
\end{aligned}$$

$$\rightsquigarrow \quad (\emptyset\,,\{\{b\colon Ch(val),\rho^{12}_{\{b\}}\}\doteqdot\rho^{7}_{\emptyset}\}\cup\mu_8\mathcal{I}_2,\ \mu_8)^{\mathcal{X}_{12}}_{\mathcal{R}_7} \qquad (\textit{loc\_end})$$

où $\mu_8 = [\rho^{14}_{\{d\}}/\rho^{2}_{\{d\}},\{d\colon Ch(h),\rho^{14}_{\{d\}}\}/\rho_{\emptyset}],\mathcal{R}_7 = \mathcal{R}_7\cup\{a,b\}\times\{\rho^{14}_{\{d\}}\},\ \mathcal{X}_{14} = \mathcal{X}_{10}\cup\{\rho^{14}_{\{d\}}\}$

$$\rightsquigarrow^* \quad (\emptyset,\ \{\{b\colon h,a\colon Ch(\{b\colon Ch(val),\rho^{12}_{\{b\}}\}),\rho^{13}_{\emptyset}\}\lessdot\{b\colon Ch(val),\rho^{12}_{\{b\}}\}\}\cup$$
$$\{\rho^{15}_{\{b\}}\doteqdot\rho^{12}_{\{b\}}\}\cup\mu_9\mathcal{I}_1,\ \mu_9)^{\mathcal{X}_{13}}_{\mathcal{R}_8} \qquad (\textit{st\_wrg})$$

où $\mu_9 = [\{b\colon Ch(val),\rho^{15}_{\{b\}}\}/\rho^{7}_{\emptyset}]\mu_8,\ \mathcal{R}_8 = \mathcal{R}_7\cup\{a\}\times\{\rho^{15}_{\{b\}}\},\ \mathcal{X}_{13} = \mathcal{X}_{12}\cup\{\rho^{15}_{\{b\}}\}$

$$\rightsquigarrow^* \quad (\emptyset,\ \{\{a\colon Ch(\{b\colon Ch(val),\rho^{12}_{\{b\}}\}),\rho^{13}_{\emptyset}\}\lessdot\rho^{12}_{\{b\}},\ \rho^{15}_{\{b\}}\doteqdot\rho^{12}_{\{b\}},\ \rho'_{\emptyset}\doteqdot\rho^{13}_{\emptyset}\}\cup$$
$$\{\{\rho'_{\emptyset}\doteqdot\{d\colon Ch(Ch(val)),\rho^{14}_{\{d\}}\}\},\ \mu_{10})^{\mathcal{X}_{13}}_{\mathcal{R}_8} \qquad (\textit{st\_ok + elim})$$

où $\mu_{10} = [Ch(val)/h]\mu_9$

$$\rightsquigarrow^* \quad (\emptyset,\ \{\{a\colon Ch(\{b\colon Ch(val),\rho^{12}_{\{b\}}\}),\rho^{13}_{\emptyset}\}\lessdot\rho^{12}_{\{b\}},\ \rho^{16}_{\{d\}}\doteqdot\rho^{14}_{\{d\}}\}\cup$$
$$\{\{d\colon Ch(Ch(val)),\rho^{16}_{\{d\}}\}\doteqdot\rho^{13}_{\emptyset}\},\ \mu_{11})^{\mathcal{X}_{14}}_{\mathcal{R}_9} \qquad (\textit{st\_wrg})$$

où $\mu_{11} = [\{d\colon Ch(Ch(val)),\rho^{16}_{\{d\}}\}/\rho'_{\emptyset}]\mu_{10},\ \mathcal{R}_9 = \mathcal{R}_8\cup\{a,b\}\times\{\rho^{16}_{\{d\}}\},\ \mathcal{X}_{14} = \mathcal{X}\cup\{\rho^{16}_{\{d\}}\}$

$$\rightsquigarrow^* \quad (\emptyset,\ \{\{a\colon Ch(\{b\colon Ch(val),\rho^{12}_{\{b\}}\}),d\colon Ch(Ch(val)),\rho^{17}_{\{d\}}\}\lessdot\rho^{12}_{\{b\}}\}\cup$$
$$\{\rho^{16}_{\{d\}}\doteqdot\rho^{14}_{\{d\}},\ \rho^{16}_{\{d\}}\doteqdot\rho^{17}_{\{d\}}\},\ \mu_{12})^{\mathcal{X}_{16}}_{\mathcal{R}_{10}} \qquad (\textit{st\_wrg})$$

où $\mu_{12} = [\{d\colon Ch(Ch(val)),\rho^{17}_{\{d\}}\}/\rho^{13}_{\emptyset}]\mu_{11},\ \mathcal{R}_{10} = \mathcal{R}_9\cup\{a,b\}\times\{\rho^{17}_{\{d\}}\},$
$\mathcal{X}_{16} = \mathcal{X}_{15}\cup\{\rho^{17}_{\{d\}}\}$

On obtient finalement l'ensemble d'assertions de sous-typages atomiques :

$$\mathcal{A} = \{\{a\colon Ch(\{b\colon Ch(val),\rho^{12}_{\{b\}}\}),d\colon Ch(Ch(val)),\rho^{17}_{\{d\}}\}<\colon\rho^{12}_{\{b\}},\ \rho^{16}_{\{d\}}\equiv\rho^{14}_{\{d\}},\ \rho^{16}_{\{d\}}\equiv\rho^{17}_{\{d\}}\}$$

et l'application de μ_{12} au contexte initial $\mathbf{\Gamma}$ donne :

$$\mu_{12}\mathbf{\Gamma} = \ell\colon\{d\colon Ch(Ch(val)),\rho^{14}_{\{d\}}\},c\colon val$$

À l'issue de cet exemple nous pouvons faire plusieurs remarques. Tout d'abord, on constate qu'un grand nombre de variables de types intermédiaires ont été générées et certaines demeurent inutilement dans l'ensemble $\mathcal{A}$. En effet, on pourrait évidemment simplifier $\mathcal{A}$ en

$$\mathcal{A}' = \{\{a\colon Ch(\{b\colon Ch(val),\rho^{12}_{\{b\}}\}),d\colon Ch(Ch(val)),\rho^{14}_{\{d\}}\}<\colon\rho^{12}_{\{b\}}\}$$

Nous ne nous sommes pas intéressés dans cette thèse à la simplification de l'ensemble des assertions de sous-typages ni à l'optimisation du nombre de variables intermédiaires générées. Cependant, il semble que l'application de méthodes similaires à celles développées dans [Pot01, Pot96] soient facilement adaptables à nos assertions. D'autre part, l'ensemble $\mathcal{A}$ n'est pas le « plus petit » ensemble d'assertions que nous pourrions associer à $\mu_{12}\mathbf{\Gamma}$ pour avoir un typage principal. En effet, il comporte encore de l'information pour des noms liés dans le terme initial, alors que $\mu_{12}\mathbf{\Gamma}$ n'en comporte pas et que nous ne sommes intéressés que par des instances de $\mu_{12}\mathbf{\Gamma}$ où n'apparaissent pas ces noms liés. Cependant, l'élimination de l'information inutile concernant les noms liés ne peut se faire qu'à posteriori car pour l'inférence de types nous avons besoin trouver les types de tous les noms y compris

des noms liés et donc des contraintes de sous-typages dont ils peuvent faire l'objet. Il resterait donc à déterminer précisément la notion de « *minimalité* » des assertions de sous-typage pour le typage principal et si celle-ci existe.

CHAPITRE 13

Conclusion et perspectives

ARRIVÉS AU TERME DE CETTE THÈSE, nous résumons les travaux réalisés et les résultats obtenus, et, l'ère de l'information suivant son cours, nous ouvrons sur de nouvelles perspectives de recherche.

Nous nous sommes tout d'abord intéressés à la propriété de *livrabilité des messages* dans le cadre des systèmes concurrents. Cette propriété peut simplement être résumée par : au cours de l'évolution d'un système, pour tout message émis il existera un récepteur susceptible de le consommer. Nous avons analysé cette propriété en prenant pour modèle de référence le π-calcul asynchrone et réussi à donner une méthode d'analyse statique des termes simple et décidable qui la satisfont. Cette analyse statique consiste en la conjonction d'un système de types et d'un système de bonne formation, et dans le langage restreint obtenu, appelé le *π-calcul réceptif* et noté π^r, nous avons répondu par l'affirmative au problème ci-dessus : dans l'exécution des processus de π^r il n'y aura jamais de messages errants. Par ailleurs, nous obtenons également facilement la propriété de l'unicité des récepteurs : pour tout nom de canal, il n'existe toujours au plus qu'un seul récepteur pour les messages émis sur celui-ci ; cette restriction porte le nom de π_1^r. Les contraintes imposées aux termes sont très fortes et nous sommes loins de capturer l'ensemble des termes du π-calcul qui satisfont la propriété de livrabilité des messages. Par exemple, on force l'imbrication des réceptions à ne porter que sur un même nom de canal : un processus tel que $a().b()$ est directement rejeté et donc $\overline{a}() \mid \overline{b}() \mid a().b()$ le sera aussi, bien qu'il satisfasse la propriété recherchée. Une conséquence de cette restriction est qu'un nom reçu ne peut être utilisé pour créer un récepteur, autrement dit seule la *capacité à émettre* des noms peut être transmise. C'est une des principales propriétés qui caractérisent le *π-calcul Local* [MS98]. Une autre restriction est que nous forçons la persistance (non-uniforme contrairement à [San99]) des récepteurs même si ceux-ci ne recevront jamais de messages. Malgré ces contraintes fortes, nous avons montré que le π-calcul réceptif reste suffisamment expressif. Nous sommes parvenu à cette conclusion par un codage du π_1 dans le π_1^r qui est *complètement adéquat* (ou *fully-abstract*), c'est-à-dire que deux

termes de π_1 sont bisimilaires si et seulement si leur traduction le sont. La bisimulation considérée est la bisimulation asynchrone faible pour laquelle nous avons été amenés à étendre les techniques de preuve « *up-to* » de la bisimulation synchrone [SM92, San95, Mil89]. Combiné avec les codages du π-calcul dans le Join-calcul [FG96] et de ce dernier dans π_1 [Ama00], on obtient une méthode générale pour compiler un processus du π-calcul dans le π-calcul réceptif à récepteurs uniques.

Ce travail comporte plusieurs pistes de recherche qu'il reste à explorer. La première serait un raffinement du calcul lui permettant de relâcher la contrainte de persistance des récepteurs, par exemple par l'introduction de types linéaires [KPT96] ce qui semble d'ailleurs ne pas poser de problèmes. Par ailleurs, il serait intéressant de voir si le codage de π_1 reste complètement adéquat pour d'autres relations de bisimulation comme par exemple la bisimulation à barbe [MS92]. Enfin il est à noter que notre codage n'est pas *uniforme* au sens de C. PALAMIDESSI [Pal97], et en particulier qu'il ne préserve pas la distribution car on introduit un coordinateur centralisé (les gestionnaires de canaux). Déterminer s'il existe un codage uniforme reste une question ouverte.

Dans la deuxième partie de cette thèse nous nous sommes penchés sur la propriété de livrabilité des messages dans le cadre des systèmes distribués où la notion de localité (ou de domaine) est formalisée. Nous avons choisi de traiter ce problème dans le cadre du π-calcul distribué [HR98a] (dans sa version asynchrone), principalement pour les raisons suivantes : il s'agit du modèle de la distribution le plus proche du π-calcul, il comporte une instruction de migration simple et la communication purement locale donne tout son intérêt à l'étude du problème qui nous intéresse pour la distribution. En effet, dans Dπ, les noms de canaux n'ont de « sens » que vis-à-vis d'un domaine. Une conséquence de cela est que les processus doivent être colocalisés pour correspondre et qu'un même nom de canal peut être exploité dans deux localités avec des usages (*i.e.* des types) différents. Le typage dans Dπ résout les ambiguïtés qu'il pourrait éventuellement en résulter. Par exemple, dans le terme $\ell[\![a(b).(\overline{b}() \mid \mathrm{go}\, k.\overline{b}())]\!]$ l'ambiguïté réside dans le fait qu'après réception d'un nom de canal b à la localité ℓ, ce canal est utilisé à la fois localement et dans une localité différente. Le système de types rejette ces termes. La communication purement locale liée à la capacité des processus à migrer donne au problème de livrabilité des messages une nouvelle dimension. En effet, les restrictions du π-calcul réceptif ne sont plus suffisantes. En particulier, les récepteurs, même s'ils sont persistants, peuvent tout de même disparaître suite à une migration. Par le système de bonne formation, nous avons donc imposé de nouvelles contraintes pour la migration et qui consistent grossièrement à « interdire » la migration des récepteurs. Et, nous avons montré que nous obtenons bien la propriété de livrabilité des messages. Si interdire la migration des récepteurs est encore une contrainte très forte, il reste néanmoins possible de recourir à des mécanismes connu pour permettre la migration de serveurs par exemple. C'est ce que nous avons montré par une série d'exemples donnant également le « style » de programmation réceptif que l'on peut adopter.

Une suite naturelle et certainement importante à la définition du π-calcul réparti réceptif concerne l'expressivité du calcul. En effet, si nous avons donné une idée de son expressivité par une suite d'exemples même assez significatifs, la preuve de celle-ci n'a pas été faite. Une piste serait probablement de réaliser un codage du Dπ original dans Dπ_1^r puisque Dπ est notre modèle de référence. Cependant, il nous manquait une notion de bisimulation pour Dπ, mais celle-ci vient très récemment d'être proposée [HMR02].

Enfin, dans la dernière partie de cette thèse nous avons traité le problème de l'inférence de types. En effet, contrairement au Dπ de [HR98a], les termes de Dπ_1^r ne sont que partiellement explicitement typés : seules les restrictions de localités comportent de l'information de types. Cependant, celles-ci ne sont utiles que pour connaître les canaux d'une localité nouvellement créée lors de la vérification de la propriété de réceptivité. Nous traitons donc l'inférence de types pour des termes n'ayant aucune information de types. L'algorithme proposé, décrit par une relation de réduction, procède en deux temps. Tout d'abord nous reconstruisons la preuve de typage d'un terme en produisant des contraintes qui consistent en des équations et des inéquations entre schémas de types. Ensuite, nous résolvons ces contraintes dont la solution est une substitution. Nous avons mis en évidence les difficultés liées aux traitement des types dépendants que sont les types de localité. Pour les résoudre nous avons introduit la notion de *relation de liaison* qui donne une « trace » grossière mais suffisante de l'ordre dans lesquels les variables de types fraîches ont été introduites vis-à-vis des noms de canaux liés rencontrés dans le terme. Celle-ci est construite lors de la première étape de l'inférence de types et renforcée dans la deuxième. Son rôle est de contraindre les substitutions à respecter la liaison des noms dans le terme que nous cherchons à typer. Finalement, grâce à l'utilisation d'une relation de sous-typage explicite, nous avons été en mesure de définir une notion de *typage principal*, ce que produit notre algorithme.

Certainement la première suite possible à ce travail est l'extension de l'algorithme pour le traitement des *types récursifs*. Cependant, nous avons imposer des contraintes relativement fortes sur les schémas de types et plus particulièrement sur les variables de rangée pour permettre l'unification des types de localités. Il faudrait voir si ces contraintes ne sont pas un obstacle important pour le traitement des types récursifs. Nous avons déjà signalé et constaté que le nombre de variables de types produites est très important et qu'il mène éventuellement à la production d'un ensemble d'assertions de sous-typage qui n'est pas « minimal ». L'optimisation du nombre de variables de types, la simplification des assertions de sous-typages et la définition d'une notion de minimalité pour celles-ci restent à explorer. Enfin, les applications de l'inférence de type pour les modèles de la distributions sont nombreuses. Nous pensons notamment au *typage des systèmes ouverts* très proches de la réalité et dans lesquels seule une partie du réseau est connue et donc typable à un moment donné (*cf.* par exemple [**?**]). L'inférence de type peut alors être utilisée pour typer dynamiquement le code migrant à l'intérieure d'un domaine dont on a le contrôle et donc dont on connait le typage. Et surtout, le typage principal nous permet de vérifier que le code entrant utilise les ressources conformément au domaine qu'il investit. Il serait intéressant d'étudier, au delà de la sûreté de types, si les contraintes de sous-typages peuvent être exploitées pour accorder ou interdire l'accès à des ressources locales pour les codes migrants. Une autre approche consiste à introduire dans le langage des *niveaux de sécurité* comme M. HENNESSY et J. RIELY l'ont fait pour Dπ [**?**]. Cependant, dans cet article, l'information de sécurité apparaît directement dans le typage alors que par nature l'inférence de types permet d'abstenir le programmeur d'écrire les types. On pourrait néanmoins profiter du travail réalisé dans cette thèse en regardant si les types peuvent être « séparés » de l'information de sécurité qui elle resterait explicite dans les termes. Ce travail, que j'ai commencé à explorer m'amène à penser que ceci est réalisable, au prix semble-t-il, toutefois, d'une perte d'expressivité, mais d'une simplification non négligeable.

Bibliographie

[ABL00] R. Amadio, G. Boudol, and C. Lhoussaine. The receptive distributive π-calculus. Technical Report 4080, INRIA, Sophia Antipolis, 2000.

[ACS98] R. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π-calculus. *Theoretical Computer Sciences*, 195 :291–324, 1998.

[Ama97] R. Amadio. An asynchronous model of locality, failure, and process mobility. In D. Garlan and D. Le Métayer, editors, *Proceedings of the 2nd International Conference on Coordination Languages and Models (COORDINATION'97)*, volume 1282, pages 374–391, Berlin, Germany, 1997. Springer-Verlag.

[Ama00] R. Amadio. On modeling mobility. *Journal of Theoretical Computer Science*, 240(1) :147–176, 2000.

[BB90] G. Berry and G. Boudol. The Chemical Abstract Machine. In *Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 81–94, San Francisco, California, January 17–19, 1990. ACM Press, New York.

[BGL99] G. Boudol, F. Germain, and M. Lacoste. Projet MARVEL : Document d'analyse des langages et modèles de la mobilité. Technical report, INRIA - France Télécom R& D, 1999.

[Bou92] G. Boudol. Asynchrony and the π-calculus. Technical Report 1702, INRIA, Sophia Antipolis, 1992.

[Bou93] G. Boudol. Some Chemical Abstract Machines. In J. deBakker, W. deRoever, and G. Rozenberg, editors, *A Decade of Concurrency*, volume 803, pages 92–123. Springer-Verlag, Berlin, 1993.

[Bou97a] G. Boudol. The π-calculus in direct style. In *Symposium on Principles of Programming Languages*, pages 228–241, 1997.

[Bou97b] G. Boudol. Typing the use of resources in a concurrent calculus. In *Proc. ASIAN 97*, number 1345 in Springer Lectures Notes in Computer Science, pages 239–253, 1997.

[BS99] F. Baader and W. Snyder. *Handbook of Automated Reasoning*, chapter Unification theory. Elsevier Science Publishers, 1999. To appear.

[Car85] L. Cardelli. An implementation model of rendezvous communication. In *Seminar on Concurrency*, volume 197 of *Lecture Notes in Computer Science*, pages 449–457. Springer Verlag, 1985.

[CG98] L. Cardelli and A.D. Gordon. Mobile ambients. In M. Nivat, editor, *Proceedings of Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 1378, pages 140–155. Springer-Verlag, Berlin, Germany, 1998.

[Dal99] S. Dal-Zilio. *Le Calcul bleu : types et objets.* PhD thesis, Université de Nice - Sophia Antipolis, 1999.

[DFP98] R. De Nicola, G. Ferrari, and R. Pugliese. KLAIM : A kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24(5) :315–330, May 1998. Special Issue : Mobility and Network Aware Computing.

[DFP99] R. De Nicola, G. Ferrari, and R. Pugliese. Types as specifications of access policies. *Lecture Notes in Computer Science*, 1603 :117–146, 1999.

[DFP00] R. De Nicola, G. Ferrari, and R. Pugliese. Programming access control : The KLAIM experience. In *International Conference on Concurrency Theory*, pages 48–65, 2000.

[DFPV00] R. De Nicola, G. Ferrari, R. Pugliese, and B. Veneri. Types for access control. *Theoretical Computer Science*, 240(1) :215–254, 2000.

[FG96] C. Fournet and G. Gonthier. The reflexive chemical abstract machine and the join-calculus. In *Proceedings of the 23rd ACM Symposium on Principles of Programming Languages*, pages 372–385, St. Petersburg Beach, Florida, 1996. ACM.

[FGL+96] C. Fournet, G. Gonthier, JJ. Lévy, L. Maranget, and D. Rémy. A calculus of mobile agents. In *7th International Conference on Concurrency Theory (CONCUR'96)*, pages 406–421, Pisa, Italy, August 26-29 1996. Springer-Verlag. LNCS 1119.

[Fou98] C. Fournet. *The Join-Calculus : a Calculus for Distributed Mobile Programming.* PhD thesis, 1998.

[GCCC85] D. Gelernter, N. Carriero, S. Chandran, and S. Chang. Parallel programming in linda. In *Proceedings of the 1985 International Conference on Parallel Processing*, pages 255–263, University Park, Pennsylvania, August 1985. IEEE.

[Gir93] J.-Y. Girard. On the unity of logic. *Annals of Pure and Applied Logic*, 59 :201–217, 1993.

[HMR02] M. Hennessy, M. Merro, and J. Rathke. Towards a behavioral theory of access and mobility control in distributed systems. Non publié, 2002.

[HR98a] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. In Uwe Nestmann and Benjamin C. Pierce, editors, *HLCL '98 : High-Level Concurrent Languages (Nice, France, September 12, 1998)*, volume 16.3 of *ENTCS*, pages 3–17. Elsevier Science Publishers, 1998. Full version as CogSci Report 2/98, University of Sussex, Brighton.

[HR98b] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. Technical Report 2/98, School of Cognitive and Computer Sciences, University of Sussex, 1998.

[HT91] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In Pierre America, editor, *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, volume 512, pages 133–147, Berlin, Heidelberg, New York, Tokyo, 1991. Springer-Verlag.

[HT92] K. Honda and M. Tokoro. On asynchronous communication semantics. In Mario Tokoro, Oscar Nierstrasz, and Peter Wegner, editors, *Proceedings of the ECOOP'91 Workshop on Object-Based Concurrent Computing*, pages 21–51. Springer-Verlag, 1992.

[IK00] A. Igarashi and N. Kobayashi. Type reconstruction for linear pi-calculus with i/o subtyping. *Information and Computation*, pages 161–144, August 2000.

[JM93] Lalita Jategaonkar and John C. Mitchell. Type inference with extended pattern matching and subtypes. *Fundamenta Informaticae*, 19(1/2) :127–165, 1993.

[Kob98] N. Kobayashi. A partially deadlock-free typed process calculus. *ACM Transactions on Programming Languages and Systems*, 20(2) :436–482, 1998.

[KPT96] N. Kobayashi, B.C. Pierce, and D.N. Turner. Linearity and the π-calculus. In *Conference record of Symposium on Principles of Programming Languages*, pages 358–371, 1996.

[LBCF99] G. Lacoste, A. Bailly, S. Conchon, and F. Le Fessant. Projet MARVEL : Document d'analyse de machines virtuelles. Technical report, INRIA - France Télécom R& D, 1999.

[Lév97] JJ. Lévy. Some results in the join-calculus. In *TACS'97*, pages 233–249, 1997. LNCS 1281.

[Mil80] R. Milner. *A calculus of communicating systems*, volume 92. Springer-Verlag Inc., New York, NY, USA, 1980.

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

[Mil90] R. Milner. Functions as processes. In *ICALP : Annual International Colloquium on Automata, Languages and Programming*, 1990.

[Mil93] R. Milner. The polyadic π-calculus : a tutorial. *Logic and Algebra of Specication*, pages 203–246, 1993.

[Mil99] R. Milner. *Communicating and Mobile Systems : the π-Calculus.* Cambridge University Press, 1999.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts i and ii. *Information and Computation*, 100(1) :1–77, 1992.

[MS92] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In *Proceedings ICALP '92*, pages 685–695, Vienna, 1992. Springer-Verlag.

[MS98] M. Merro and D. Sangiorgi. On asynchrony in name-passing calculi. In *ICALP : Annual International Colloquium on Automata, Languages and Programming*, 1998.

[NP96] U. Nestmann and B.C. Pierce. Decoding choice encodings. In *International Conference on Concurrency Theory*, pages 179–194, 1996.

[Pal97] C. Palamidessi. Comparing the expressive power of the synchronous and the asynchronous pi-calculus. In *Symposium on Principles of Programming Languages*, pages 256–265, 1997.

[Pie98] B.C. Pierce. *Programming in the Pi-Calculus : A Tutorial Introduction to Pict (Pict Version 4.1).* Indiana University, March 1998.

[Pot96] François Pottier. Simplifying subtyping constraints. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming (ICFP '96)*, volume 31(6), pages 122–133, 1996.

[Pot01] Pottier. Simplifying subtyping constraints : A theory. *INFCTRL : Information and Computation (formerly Information and Control)*, 170, 2001.

[PS93] B.C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. In *Proceedings LICS'93*, 1993.

[PT98] B.C. Pierce and D.N. Turner. Pict : A programming language based on the pi-calculus. In Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language and Interaction : Essays in Honour of Robin Milner.* mit, 1998.

[Qua99] P. Quaglia. The π-calculus : Notes on labelled semantics. In *Bulletin of the EATCS*, number 68, Juin 1999. Version préliminaire BRICS Report LS-98-4.

[Rém93a] Didier Rémy. Syntactic theories and the algebra of record terms. Research Report 1869, Institut National de Recherche en Informatique et Automatisme, Rocquencourt, BP 105, 78 153 Le Chesnay Cedex, France, 1993.

[Rém93b] Didier Rémy. Type inference for records in a natural extension of ML. In Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects Of Object-Oriented Programming. Types, Semantics and Language Design.* MIT Press, 1993.

[RH98] J. Riely and M. Hennessy. Trust and partial typing in open systems of mobile agents. Technical Report 4, University of Sussex, 1998.

[Rob65] J. Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12 :227–234, 1965.

[San95] D. Sangiorgi. On the bisimulation proof method. In J. Wiedermann and P. Háiek, editors, *Proc. MFCS'95*, volume 969 of *Lecture Notes in Computer Science*, pages 479–488. Springer Verlag, 1995.

[San99] D. Sangiorgi. The name discipline of uniform receptiveness. *Theoretical Computer Science*, 221(1–2) :457–493, 1999.

[SM92] D. Sangiorgi and R. Milner. Techniques of «weak bisimulation up to». In *CONCUR 92*, number 630 in LNCS, 1992.

[SW01] D. Sangiorgi and D. Walker. *The Pi-Calculus : A Theory of Mobile Processes*. Cambridge University Press, 2001.

[SWP99] P. Sewell, P. Wojciechowski, and B.C. Pierce. Location-independent communication for mobile agents : a two-level architecture. Technical Report 462, Computer Laboratory, University of Cambridge, April 1999. A version of this appeared in Internet Programming Languages, Springer LNCS 1686.

[SY97] T. Sekiguchi and A. Yonezawa. A calculus with code mobility. In H. Bowman and J. Derrick, editors, *Proc. 2nd IFIP Workshop on Formal Methods for Open Object-Based Distributed Systems (FMOODS)*, pages 21–36, Canterbury, UK, 1997. Chapman and Hall, London.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and applications. *Pacific Journal of Mathematics*, 5 :285–309, 1955.

[Tur96] D. Turner. *The Polymorphic Pi-Calculus : Theory and Implementation*. PhD thesis, Dept. of Computer Science, University of Edinburgh, 1996.

[Uny01] A. Unyapoth. *Nomadic Pi Calculi : Expressing and Verifying Infrastructure for Mobile Computation*. PhD thesis, University of Cambridge, Computer Laboratory, June 2001. Appeared as Technical Report 514.

[Vas94] V.T. Vasconcelos. Typed concurrent objects. In M. Tokoro and R. Pareschi, editors, *Proceedings ECOOP'94*, pages 100–117, Bologna, Italy, 1994. Springer-Verlag.

[Wan87] M. Wand. Complete type inference for simple objects. In *Symposium on Logic in Computer Science, Ithaca, NY*, pages 37–44, 1987.