Logique spatiale pour le pi-calcul appliqué

Jules Villard — Rapport de stage de M2

Stage effectué au Laboratoire Spécification et Vérification de l'ENS Cachan,
sous la direction d'Étienne Lozes.

Fiche de synthèse

Le contexte général

On se place ici dans le cadre de l'étude logique des algèbres de processus, initiée par Matthew Hennessy et Robin Milner [HM85] dans le but d'obtenir des classes de processus pertinentes. Les travaux en ce sens ont continué d'une part par la définition de nouvelles logiques, comme les logiques *spatiales* [CC03], et d'autre part par la mise au point de *systèmes de types*, comme celui d'Emmanuel Beffara [Bef06] pour le pi-calcul. Les logiques spatiales permettent de raisonner de façon *modulaire*, en limitant les interférences entre les sous-systèmes étudiés, et sont ainsi particulièrement intéressantes pour l'étude des protocoles de communication en général, et des protocoles cryptographiques en particulier. Or, c'est là l'objet du pi-calcul appliqué, et c'est donc naturellement que nous avons tenté durant ce stage de définir une logique spatiale qui lui correspondrait.

Ce dernier, introduit par Martìn Abadi et Cédric Fournet [AF01], a vocation à représenter les protocoles cryptographiques de manière simple et concise, en prenant en compte de manière primitive les théories mathématiques sous-jacentes (par exemple, les mécanismes de *hachage* et de chiffrement). Le calcul possède de plus une construction nouvelle permettant de décrire de façon *statique* et *globale* une forme de connaissance commune. Il s'oppose en cela au pi-calcul dont il est issu, ou à d'autres algèbres de processus, qui ont en commun de ne prendre en compte que la *dynamique* des processus.

Le problème étudié

Qui dit nouveau calcul dit nouvelles classes d'équivalences pour les processus, et deux types d'équivalences, statique ou observationnelle, ont immédiatement été proposées. Plus récemment, une logique a été créée pour étudier la partie statique du calcul [HP07], mais sans travaux conséquents pour traiter de son interaction avec la partie dynamique. Nous nous sommes donc naturellement posé la question de ce que pourrait être une logique *spatiale* pour le pi-calcul appliqué, et de ce qu'elle pourrait exprimer sur ces processus. Plus précisément, on a cherché à retrouver (ou à invalider) les résultats connus pour ce type de logique sur les autres algèbres de processus [Loz04, Hir02, CL04], et donc à obtenir une caractérisation précise de l'équivalence logique, ainsi qu'une forme d'élimination des quantificateurs.

La contribution proposée

Nous avons donc ici adapté les opérateurs logiques au traitement du pi-calcul appliqué, et avons en particulier été amenés à séparer les effets

de l'opérateur de *composition spatiale* en deux, suivant qu'on l'applique à la partie dynamique d'un processus ou à sa partie statique. Ce choix s'est fait après de nombreux essais concernant la simplicité des formules et leur expressivité suivant les différentes sémantiques considérées. La logique a ainsi subi des modifications significatives, tout en restant dans l'esprit des logiques spatiales. La syntaxe et la sémantique du pi-calcul appliqué ont également subi quelques modifications, plus mineures et principalement pour les adapter à la logique, mais aussi pour mieux mesurer l'impact de certains choix techniques de leurs créateurs.

Dans le même temps, nous avons démontré le caractère intensionnel de cette logique, ainsi qu'un résultat d'élimination des quantificateurs.

Les arguments en faveur de sa validité

Quelques autres résultats d'expressivité viennent compléter cette étude, dont le plus remarquable est sans doute la capture par la logique des classes d'*équivalence statique*, propre au pi-calcul appliqué. En effet, une autre logique avait précédemment été développée à cette seule fin [HP07].

On conjecture de plus que tous nos résultats, à l'exception sans doute de l'élimination des quantificateurs, résistent à des changements mineurs sur la forme du calcul ou de la logique. Ainsi, les choix que nous fûmes amenés à faire, s'ils permettent de mieux comprendre les subtilités des deux composantes, semblent aisément réversibles, et les résultats adaptables.

Le bilan et les perspectives

Nous sommes ainsi parvenus à la définition d'une logique spatiale pour le pi-calcul appliqué qui semble naturelle, et y avons exprimé quelques formules et propriétés qui aident à mieux la comprendre. Ce faisant, on comprend également mieux le fonctionnement du pi-calcul appliqué lui-même qui, s'il paraît simple au premier abord pour qui est déjà familier avec le pi-calcul, comporte en fait de nombreuses subtilités, que nous avons pu prendre en compte dans son traitement logique. Il reste maintenant à étudier le fragment *extensionnel* de la logique pour valider ou non son appellation dans le cadre du pi-calcul appliqué, ainsi que l'impact des diverses variations possibles autour du calcul et de la logique. Enfin, et cela dépasse l'étude de ce cas particulier pour s'étendre de manière plus générale aux logiques spatiales pour les algèbres de processus dans leur ensemble, il serait souhaitable d'obtenir quelques résultats de *décidabilité*, par exemple sur les problèmes de *model checking* pour ces logiques, pour différents fragments représentatifs. De nouvelles perspectives s'ouvriraient alors pour la vérification automatisée de protocoles cryptographiques, déjà initiée dans le cadre du pi-calcul appliqué par Bruno Blanchet [Bla04].

Table des matières

Remerciements

Je tiens à remercier en premier lieu mon maître de stage, Étienne Lozes, pour sa disponibilité, sa gentillesse et son enthousiasme tout au long de ce stage.

Je remercie également Daniel Hirschkoff qui nous a fait nous rencontrer, et qui s'est tout de suite intéressé à nos travaux, ainsi que Cristiano Calcagno qui m'a fait découvrir le monde des logiques de séparation.

Je tiens également à remercier Steve Kremer, pierre de touche de nos tâtonnements en pi-calcul appliqué, et l'ensemble du LSV et plus particulièrement des habitants de la salle Renaudeau pour m'avoir si bien accueilli.

Je remercie enfin Jade Alglave, vecteur de réflexion de par ses questions incongrues, et toujours prête à répondre aux miennes.

1 Introduction

Les algèbres de processus sont nées du besoin de modéliser les programmes concurrents pour donner un cadre formel à l'étude des mécanismes complexes de synchronisation qui surviennent dès lors que l'on cherche à effectuer des calculs de manière parallèle. Elles ont comme particularité d'être facilement adaptables pour faire la lumière sur des aspects particuliers de ce vaste domaine, et furent ainsi déclinées de nombreuses façons. On en distingue trois variantes « principales », par ordre chronologique d'apparition : CCS tout d'abord [Mil82], puis le pi-calcul [MPW92], et enfin les ambients [CG98].

Comprendre comment ces processus interagissaient n'était cependant pas une mince affaire, et, à mesure que les variations sur ces calculs se multipliaient, de nombreuses façons de comparer les processus entre eux (au sein d'un même calcul) voyaient le jour. Au premier niveau de celles-ci, la *congruence structurelle*, qui trace les frontières de l'égalité syntaxique et immédiate entre processus, et identifie les processus qui ne diffèrent que par des jeux d'écriture, par exemple de parenthésage. Viennent ensuite une pléthore d'équivalences plus ou moins complexes reposant toutes sur la notion de *bisimulation*, qui joue avec les réductions internes et les communications de deux processus pour vérifier qu'ils se comportent de la même manière. Enfin, la dernière de ces façons, qui est celle qui nous intéresse et qui fut initiée par Matthew Hennessy et Robin Milner [HM85], repose sur la définition préalable d'une *logique*, et cherche alors à identifier les classes de processus qui vérifient une formule donnée, ou encore ce que signifie pour deux processus le fait de vérifier les mêmes formules logiques.

De l'intérêt pour les systèmes parallèles est ensuite né un intérêt pour les systèmes *distribués*, où l'on prend en compte la localisation des processus. De nouvelles logiques sont donc apparues pour traiter de ce nouveau caractère *spatial* [CC03] ; celles-ci seront au centre de notre étude. Elles permettent en effet, comme les logiques de séparation dont elles sont issues [Rey02], de raisonner de façon *modulaire*, en limitant les interférences entre les sous-systèmes étudiés. Elles sont ainsi particulièrement adaptées à l'étude des protocoles de communication en général, et des protocoles cryptographiques en particulier, qui sont les objets du pi-calcul appliqué.

Ce dernier, introduit par Martìn Abadi et Cédric Fournet [AF01], apparaît comme un pont à la frontière des algèbres de processus et de l'étude des protocoles cryptographiques. Il permet en effet de décrire ces derniers de manière simple et concise en prenant en compte de manière primitive les théories mathématiques sous-jacentes (par exemple, les mécanismes de *hachage* et de chiffrement). Le calcul possède de plus une construction nouvelle permettant de décrire de façon *statique* et *globale* une forme de connaissance commune : les substitutions actives. Il s'oppose en cela au pi-calcul dont il est issu, ou à d'autre algèbres de processus, qui ont en commun de

ne prendre en compte que la *dynamique* des processus.

Qui dit nouveau calcul dit nouvelles classes d'équivalences pour les processus, et dès l'article d'introduction, deux types d'équivalences, statique ou observationnelle, ont été proposées. D'autres ont suivi pour définir des propriétés spécifiques ayant trait aux protocoles cryptographiques (par exemple pour les protocoles de vote [KR05]). Une logique a également été définie pour étudier la partie statique du calcul [HP07], mais sans travaux conséquents pour traiter de son interaction avec la partie dynamique.

C'est donc naturellement que nous nous sommes posé la question de ce que pourrait être une logique *spatiale* pour le pi-calcul appliqué. Plus précisément, nous avons défini une nouvelle sémantique propre au pi-calcul appliqué, avant de comparer les propriétés de cette logique à celles des logiques spatiales précédentes. Ces dernières sont en effet connues pour être *intensionnelles* et pour autoriser une certaine forme d'élimination des quantificateurs au sein de la logique [CL04]. Nous nous sommes donc demandé quelle forme prenait l'équivalence logique dans notre cas, dans quelle mesure notre logique éliminait les quantificateurs, et enfin quelles propriétés plus orientées sur la modélisation de protocoles cryptographiques nous pouvions exprimer dans notre logique.

Nous commencerons ce rapport par une brève introduction à la logique spatiale dans le cadre du pi-calcul, avant de présenter un peu plus en détail le pi-calcul appliqué, et la logique spatiale telle que nous l'avons adaptée. Nous présenterons ensuite quelques résultats, d'abord sur la caractérisation de l'équivalence logique, puis sur l'élimination des quantifications sur les termes, et enfin sur l'application de la logique à l'étude des processus du pi-calcul appliqué. Enfin, l'on exposera quelques pistes de recherches futures avant de conclure.

2 Logique spatiale pour le pi-calcul

La logique spatiale fut originellement introduite par Luìs Caires et Luca Cardelli [CC03] pour étudier les systèmes distribués, modélisés sous la forme d'algèbres de processus. Les modèles de cette logique furent donc tout d'abord des processus du pi-calcul asynchrone, mais d'autres calculs ont également été étudiés, comme le calcul des ambients [GC00, HLS03] et le pi-calcul synchrone [HLS03].

Présentons ici cette logique dans le cadre du pi-calcul synchrone, dont on commence par rappeler brièvement la définition, et qui servira de base à l'étude du pi-calcul appliqué.

2.1 Le pi-calcul

Le pi-calcul, introduit par Robin Milner, Davide Sangiorgi et David Walker en 1992 [MPW92], a pour vocation première de modéliser des échanges de messages entre des programmes s'exécutant en parallèle. Les messages s'échangent sur des *canaux*, qui peuvent être publics ou privés, et consistent uniquement en d'autres noms de canaux. Formellement, les processus sont décrits par la grammaire suivante :

$P ::=$	processus
$\mathbf{0}$	processus nul
$P \mid P$	composition parallèle
$\nu n.\, P$	restriction de nom
$a(n).P$	réception de nom
$\bar{a}\langle n\rangle.P$	émission de noms
$!P$	réplication

L'ensemble des noms libres d'un processus est défini de manière usuelle (dans les processus $a(n).P$ et $\nu n.\, P$, le nom n est *lié*), et noté $fn(P)$. On appelle *contexte* un processus avec un trou.

On considère alors les processus modulo une relation de *congruence structurelle*, notée $\equiv$, qui est la plus petite relation d'équivalence close par contexte et α-conversion et vérifiant, pour tous processus P, Q et R et tous noms n et m :

$$P \equiv P \mid \mathbf{0} \qquad \nu n.\, \mathbf{0} \equiv \mathbf{0}$$
$$P \mid (Q \mid R) \equiv (P \mid Q) \mid R \qquad \nu m.\, \nu n.\, P \equiv \nu n.\, \nu m.\, P$$
$$P \mid Q \equiv Q \mid P \qquad P \mid \nu n.\, Q \equiv \nu n.\, (P \mid Q) \text{ si } n \notin fn(P)$$
$$!P \equiv !P \mid P$$

Enfin, la relation de réduction $\rightarrow$ est la plus petite relation close par contexte et par congruence structurelle vérifiant

$$\bar{a}\langle b\rangle.P \mid a(n)Q \rightarrow P \mid Q[n{\leftarrow}b]$$

quels que soient les processus P et Q et les noms a, b et n (on a noté $Q[n{\leftarrow}b]$ le processus Q où b remplace n).

2.2 La logique spatiale

Présentons maintenant la logique spatiale $\mathcal{L}^\pi$ qui servira de base à notre étude d'une logique pour le pi-calcul appliqué. Les opérateurs sont regroupés en deux ensembles qui ont en commun un fragment $\mathcal{L}^\pi_{\text{core}}$: un fragment $\mathcal{L}^\pi_{\text{int}}$ dit *intensionnel*, et un fragment $\mathcal{L}^\pi_{\text{ext}}$ dit *extensionnel*. La syntaxe de ces opérateurs est exposée figure 1. Les opérateurs booléens $\neg$ et $\wedge$ ont priorité sur les opérateurs spatiaux $\triangleright$ et $|$, eux-mêmes prioritaires sur la modalité $\Diamond$. Les opérateurs $И$ et $®$ ont quant à eux la priorité la plus faible. Ainsi, la formule $Иa.\, (\neg(\Diamond\top) \wedge \neg a \,®\, \top) \triangleright \neg\Diamond\top$ est implicitement parenthésée

$\mathit{N}a.\,(((\neg(\Diamond\top)) \wedge (\neg a \circledR \top)) \rhd (\neg(\Diamond\top)))$. Détaillons maintenant la sémantique des différents opérateurs, présentée formellement figure 2.

Figure 1 Syntaxe de la logique spatiale pour le pi-calcul

	$A ::=$	
$\mathcal{L}^{\pi}_{\text{core}}$	$\top$	vrai
	$\neg A$	négation
	$A \wedge A$	conjonction
	$\mathit{N}a.\,A$	quantification fraîche
	$\Diamond A$	modalité (forte)
	$A \rhd A$	garantie
$\mathcal{L}^{\pi}_{\text{int}}$	$\mathbf{0}$	processus nul
	$A \mid A$	composition séparante
	$n \circledR A$	révélation
$\mathcal{L}^{\pi}_{\text{ext}}$	$A \oslash n$	dissimulation

Figure 2 Relation de satisfaction pour le pi-calcul

$\mathcal{L}^{\pi}_{\text{core}}$	$P \vDash \top$	toujours
	$P \vDash \neg A$	ssi $P \nvDash A$
	$P \vDash A_1 \wedge A_2$	ssi $P \vDash A_1$ et $P \vDash A_2$
	$P \vDash \mathit{N}n.\,A$	ssi $\forall m \notin \mathit{fn}(P, A).\, P \vDash A[n \leftarrow m]$
	$P \vDash \Diamond A$	ssi $\exists Q.\, P \rightarrow Q$ et $Q \vDash A$
	$P \vDash A_1 \rhd A_2$	ssi $\forall Q.\, Q \vDash A_1$ implique $P \mid Q \vDash A_2$
$\mathcal{L}^{\pi}_{\text{int}}$	$P \vDash \mathbf{0}$	ssi $P \equiv \mathbf{0}$
	$P \vDash A_1 \mid A_2$	ssi $\exists P_1, P_2.\, P \equiv P_1 \mid P_2$, $P_1 \vDash A_1$ et $P_2 \vDash A_2$
	$P \vDash n \circledR A$	ssi $\exists Q.\, P \equiv \nu n.\, Q$ et $Q \vDash A$
$\mathcal{L}^{\pi}_{\text{ext}}$	$P \vDash A \oslash n$	ssi $\nu n.\, P \vDash A$

La définition des opérateurs $\top$, $\neg$ et $\wedge$ est classique. Le quantificateur N est un peu particulier, puisqu'il quantifie par rapport aux noms dits *frais* (c'est-à-dire qui n'apparaissent pas libres), à la fois pour le processus considéré et pour la suite de la formule. Cet opérateur a comme propriété d'être *auto-dual* : pour tout nom n et toute formule A, $\mathit{N}n.\,A$ est équivalent à $\neg\mathit{N}n.\,\neg A$. En d'autres termes, on peut le définir aussi bien par une quantification universelle sur les noms frais que par une quantification existentielle. En effet, ce qui est vrai d'un nom frais l'est de tout autre nom frais.

L'opérateur modal $\Diamond$ examine le processus après réduction. Ici, on a choisi une version *forte* de cet opérateur, c'est-à-dire qu'on se limite à *une seule* réduction à la fois. Par exemple, la formule $\Diamond\top$ est vérifiée par tout processus pouvant effectuer au moins un pas de réduction. On aurait pu

choisir une modalité *faible*, et une formule $\Diamond A$ aurait été vérifiée par tout processus qui vérifie A après un certain nombre non défini de pas de réduction, comme cela est discuté à la section 2.3.

Enfin, le dernier opérateur commun aux deux fragments, et nommé *garantie*, permet de tester le comportement d'un processus lorsqu'il est plongé dans certains types d'environnements : un processus P vérifie $A \rhd B$ si pour tout processus vérifiant A, la composition $P \mid Q$ des deux processus vérifie B[1]. On peut définir par exemple la formule $\mathsf{nil} \triangleq (\neg\Diamond\top) \rhd (\neg\Diamond\top)$ qui représente l'ensemble des processus *bloqués*, c'est-à-dire l'ensemble des processus qui ne peuvent interagir avec aucun autre processus. Cet opérateur a un *dual* naturel $\blacktriangleright$, qui lui quantifie *existentiellement* sur les contextes : $P \vDash A \blacktriangleright B$ *si et seulement si* il *existe* un processus Q vérifiant A tel que $P \mid Q$ vérifie B. Il est défini en posant $A \blacktriangleright B \triangleq \neg(A \rhd \neg B)$.

Logique intensionnelle. Cette partie de la logique permet d'exprimer des propriétés plus fines sur les processus, en particulier plus syntaxiques. On peut ainsi décrire précisément le processus nul par la formule $\mathbf{0}$, ou la composition parallèle de deux processus à l'aide de l'opérateur $|$. Ce dernier est à la base du caractère *spatial* de la logique : il permet de partitionner le système en deux sous-processus disjoints spatialement[2]. On peut par exemple décrire l'ensemble des processus composés d'un seul *thread* par la formule $\mathbf{1} \triangleq \neg\mathbf{0} \wedge \neg(\neg\mathbf{0} \mid \neg\mathbf{0})$: c'est l'ensemble des processus non-nuls qui ne peuvent être considérés comme la mise en parallèle de deux processus non-nuls.

Enfin, l'opérateur de *révélation* $\circledR$ permet de choisir un nom restreint du processus et de le révéler. On peut alors écrire des formules telles que $\circledC\, n \triangleq \neg n \circledR \top$ qui est vérifiée par tous les processus P où n apparaît libre (car l'on ne peut alors pas écrire P sous la forme $\nu n.\, Q$, quel que soit Q), ou encore $\mathsf{public} \triangleq \neg И n.\, n \circledR \circledC\, n$ qui est vraie de tout processus sans nom restreint (c'est-à-dire dont on ne peut pas choisir un nom n tel que n apparaisse libre dans le processus où n a été révélé).

Logique extensionnelle. Ce fragment ajoute simplement l'opérateur $\oslash$ à la logique $\mathcal{L}^{\pi}_{\text{core}}$. Celui-ci permet d'observer le résultat d'une restriction de nom sur un processus. Par exemple, la formule $\neg\mathsf{nil} \wedge (\mathsf{nil} \oslash a)$ caractérise les processus pouvant effectuer une communication sur a (et seulement sur a).

2.3 Résultats connus d'expressivité

Équivalence logique. Dans sa thèse [Loz04], Étienne Lozes a démontré que l'équivalence logique pour le fragment intensionnel coïncidait avec la

[1] Cela n'est pas sans rappeler les formalismes dits de *rely/guarantee* [Jon83].

[2] Comme dans les logiques de séparation [Rey02].

congruence structurelle. En d'autres termes, deux processus P et Q vérifient les mêmes formules de $\mathcal{L}^{\pi}_{\text{int}}$ *si et seulement si* $P \equiv Q$. Ce résultat reste vrai (à une subtilité près) si l'on choisit la version *faible* de $\diamondsuit$.

Un résultat similaire pour la logique extensionnelle a été découvert par Étienne Lozes et Daniel Hirschkoff [Hir02] : deux processus sont logiquement équivalents *si et seulement si* ils sont *bisimilaires*.

Élimination des quantificateurs et des modalités. Luìs Caires et Étienne Lozes ont de plus montré [CL04] que la logique $\mathcal{L}^{\pi}_{\text{int}}$ suffisait à exprimer la quantification sur les noms, c'est-à-dire les formules de la forme $\exists n.\, A$, et les modalités à la Hennessy-Milner, c'est-à-dire les formules de la forme $\langle a \rangle . A$ et $\langle \bar{a} \rangle . A$. Plus précisément, toute formule A écrite à l'aide de ces nouveaux connecteurs peut être transformée en une formule $[\![A]\!]_K$ telle que :

$$\forall P.\, P \vDash A \textit{ ssi } \pi_K(P) \vDash [\![A]\!]_K$$

Ici, K est un paramètre de la traduction qui borne la taille des processus (la taille représentant ici la profondeur d'un processus), et $\pi_K(P)$ la troncature du processus P en un processus de taille K.

Décidabilité. Cette logique, et plus précisément les jugements de *model checking* sur cette logique, est connue pour être *indécidable*, et ce à cause de plusieurs facteurs, notamment de l'opérateur de révélation [CG04]. On soupçonne les opérateurs $\triangleright$ et $\diamondsuit$ d'amener eux aussi de l'indécidabilité.

Remarquons de plus qu'en présence de l'opérateur $\triangleright$, la résolution du problème du *model checking* implique celle de la satisfiabilité d'une formule. Pour savoir si une formule A est satisfiable, il suffit en effet de résoudre $P \vDash A \blacktriangleright \top$ pour n'importe quel processus P.

3 Logique spatiale pour le pi-calcul appliqué

Le stage s'est articulé autour de l'idée d'adapter la logique spatiale que nous venons de voir à une variante du pi-calcul : le pi-calcul appliqué. Présentons donc d'abord celui-ci avant de dévoiler les mutations subies par la logique pour s'y adapter.

3.1 Le pi-calcul appliqué

Le pi-calcul appliqué fut introduit par Abadi et Fournet [AF01] pour modéliser les protocoles cryptographiques. Il descend directement du pi-calcul, avec deux évolutions principales : contrairement au pi-calcul, le pi-calcul appliqué inclut une grammaire de termes de manière primitive dans sa syntaxe, et il étend les processus du pi-calcul par une partie statique :

les substitutions actives. Celles-ci ont vocation à représenter la connaissance de l'environnement, concept souvent utilisé dans les modélisations formelles de protocoles cryptographiques.

Ce calcul a par la suite été utilisé entre autre pour définir des propriétés générales sur les protocoles cryptographiques [CRZ07] et les vérifier formellement sur des exemples [ABF04], parfois de manière automatique [BAF05, Bla04], ou encore pour modéliser des propriétés complexes sur des protocoles de vote [DKR06].

3.1.1 Termes, théorie équationnelle

Le pi-calcul appliqué repose sur la donnée préalable d'un ensemble de *termes* définis par une signature et liés entre eux par une *théorie équationnelle*. Ainsi, chacun est libre de définir la théorie qui lui convient le mieux pour décrire la nature des messages échangés, indépendamment des caractéristiques générales du calcul.

On suppose donc donnés :

- une *signature* Σ constituée d'un nombre fini de symboles de fonctions, chacune dotée d'une arité (on notera $ar(f)$ l'arité de f). Par exemple, une *constante* est alors une fonction d'arité nulle.
- un ensemble $\mathcal{V}$ de *variables* ;
- un ensemble $\mathcal{N}$ de *noms*.

On notera alors sans ambiguïté x, y, z des éléments de $\mathcal{V}$, a, b, c, m, n, s des éléments de $\mathcal{N}$ et u, v des « méta-variables » qui parcourent à la fois $\mathcal{V}$ et $\mathcal{N}$. On définit ensuite l'ensemble des termes à l'aide de la grammaire suivante (où f parcourt Σ) :

$M ::=$	termes
$x, y, z, \ldots$	variables
$a, b, c, m, n, s, \ldots$	canaux
$f(M_1, \ldots, M_{ar(f)})$	application de fonction

On suppose également ces termes équipés d'une *théorie équationnelle* $\mathcal{E}$, et on note $\mathcal{E} \vdash M = N$ (ou simplement $\vdash M = N$) lorsque les termes M et N sont identifiés par celle-ci. On impose à la relation $=$ ainsi définie sur les termes d'être une relation d'équivalence close par substitution de variables ou de noms ($\mathcal{E} \vdash M = N$ implique que, pour toute substitution $\sigma : \mathcal{V} \cup \mathcal{N} \to \mathcal{V} \cup \mathcal{N}$, $\mathcal{E} \vdash M\sigma = N\sigma$), et par application de contexte ($\mathcal{E} \vdash M = N$ implique $\mathcal{E} \vdash M'[x \leftarrow M] = M'[x \leftarrow N]$ pour tous termes M, N et M'). Ce sont là les seules contraintes demandées, et la théorie équationnelle $\mathcal{E}$ peut être définie indifféremment à l'aide d'un système de réécriture, d'un ensemble d'axiomes que l'on clôt, *etc.*

On peut par exemple définir un système de chiffrement à clé symétrique par $\Sigma = \{\mathrm{dec}(\cdot,\cdot), \mathrm{enc}(\cdot,\cdot)\}$, et par la théorie équationnelle dé-

finie comme la plus petite relation satisfaisant les contraintes ci-dessus et contenant l'égalité $\mathrm{dec}(\mathrm{enc}(x, y), y) = x$.

Dans la suite, on fera l'hypothèse supplémentaire qu'aucune des théories équationnelles considérée n'identifie tous les termes. Ainsi, pour deux variables ou noms distincts u et v, on aura toujours $\mathcal{E} \vdash u \neq v$ (sinon, par invariance de la théorie équationnelle par application de contexte, on en déduirait $M = N$ quels que soient les termes M et N).

3.1.2 Grammaire des processus

Les processus du pi-calcul appliqué reprennent et étendent ceux du pi-calcul, repris ici sous une forme plus adaptée à la manipulation de termes et sous le nom de *processus simples*. Les cinq premières lignes de la grammaire définissent ainsi exactement les processus du pi-calcul, à l'exception de la réplication, qui a été omise pour simplifier notre étude.

$P^s ::=$	processus (simples)
$\mathbf{0}$	processus nul
$P^s \mid P^s$	composition parallèle
$\nu a.\, P^s$	restriction de nom
$a^{\mathsf{ch}}(n).P^s$	réception de nom
$\bar{a}^{\mathsf{ch}}\langle n\rangle.P^s$	émission de nom
$a(x).P^s$	réception de terme
$\bar{a}\langle M\rangle.P^s$	émission de terme
$\mathsf{if}\ M = M\ \mathsf{then}\ P^s\ \mathsf{else}\ P^s$	conditionnelle

On distingue les communications de *noms* de celles de *termes*. En effet, le pi-calcul appliqué d'Abadi et Fournet [AF01] repose sur un système de types implicite pour les communications (voir la section 3.3), et on a préféré ici inclure ces distinctions explicitement dans la syntaxe des processus pour plus de clarté. En particulier, on verra par la suite qu'une communication de terme ne peut pas interagir avec une communication de nom. Par exemple, le processus $\bar{a}\langle b\rangle.\mathbf{0} \mid a^{\mathsf{ch}}(n).\bar{n}\langle M\rangle$ sera *bloqué*.

On étend ensuite les processus simples par des *substitutions actives* :

$P ::=$	processus étendus
P^s	processus simple
$\{^M/_x\}$	substitution active
$P \mid P$	composition parallèle
$\nu a.\, P$	restriction de nom
$\nu x.\, P$	restriction de variable

On impose de plus à tout processus étendu d'avoir au plus une substitution active par variable, *exactement une* si la variable est restreinte, et que les substitutions actives soient sans cycle. Par exemple, les processus $\{^m/_x\} \mid \{^n/_x\}$, $\nu x.\, \mathbf{0}$ et $\{^{f(y)}/_x\} \mid \{^x/_y\}$ seront considérés comme mal formés.

Notations. On définit de manière usuelle l'ensemble des variables et des noms libres d'un processus (notés respectivement $fv(P)$ et $fn(P)$ pour un processus P), en posant dans le cas des substitutions actives $fv(\{^M/_x\}) \triangleq \{x\} \cup fv(M)$ et $fn(\{^M/_x\}) \triangleq fn(M)$. L'ensemble $fv(P) \cup fn(P)$ est noté $fnv(P)$. Les opérateurs νu sont liants, et on abrégera une série (possiblement vide) $\nu u_1 \dots \nu u_n$ où les u_i sont deux à deux distincts en $\nu\tilde{u}$. On dit qu'un terme M est *clos* lorsque $fv(M) = \emptyset$.

La composition de substitutions actives $\{^{M_1}/_{x_1}\} \mid \cdots \mid \{^{M_n}/_{x_n}\}$ sera notée $\{^{M_1 \dots M_n}/_{x_1 \dots x_n}\}$. On abrégera $\bar{a}\langle M\rangle.\mathbf{0}$ par $\bar{a}\langle M\rangle$ et $\mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ \mathbf{0}$ par $\mathsf{if}\ M = N\ \mathsf{then} P$.

On notera $P[u \leftarrow N]$ la substitution qui remplace chaque occurrence *libre* de u dans P par N.

3.1.3 Sémantique opérationnelle

On appelle *contexte d'évaluation*, et on note $C[\,\cdot\,]$, un processus étendu dans lequel un sous-processus étendu a été remplacé par un trou. On appelle *contexte*, et on note de même $C[\,\cdot\,]$, un processus étendu dans lequel un sous-processus simple a été remplacé par un trou.

Congruence structurelle. On définit la congruence structurelle comme étant la plus petite relation d'équivalence sur les processus bien formés (notée $\equiv$) close par α-conversion (à la fois sur les noms et les variables) et par application de contextes, et qui satisfait aux règles de la figure 3.

Figure 3 Règles de congruence structurelle

PAR-**0**	$P \equiv P \mid \mathbf{0}$
PAR-A	$P \mid (Q \mid R) \equiv (P \mid Q) \mid R$
PAR-C	$P \mid Q \equiv Q \mid P$
NEW-**0**	$\nu n.\, \mathbf{0} \equiv \mathbf{0}$
NEW-C	$\nu u.\, \nu v.\, P \equiv \nu v.\, \nu u.\, P$
NEW-PAR	$P \mid \nu u.\, Q \equiv \nu u.\, (P \mid Q)$ si $u \notin fnv(P)$
ALIAS	$\nu x.\, \{^M/_x\} \equiv \mathbf{0}$
SUBST	$\{^M/_x\} \mid P^s \equiv \{^M/_x\} \mid P^s[x \leftarrow M]$
REWRITE	$\{^M/_x\} \equiv \{^N/_x\}$ si $\mathcal{E} \vdash M = N$
TEST-S	$\mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q \equiv \mathsf{if}\ N = M\ \mathsf{then}\ P\ \mathsf{else}\ Q$

On remarque que, si $\mathcal{E} \vdash M = N$, alors $\bar{a}\langle M\rangle \equiv \bar{a}\langle N\rangle$, grâce à la suite

de règles suivante :

$$\begin{array}{lll}
\bar{a}\langle M\rangle \equiv (\nu x.\, \{^{M}/_{x}\}) \mid \bar{a}\langle M\rangle & & \text{par ALIAS} \\
\equiv \nu x.\, \{^{M}/_{x}\} \mid \bar{a}\langle x\rangle & & \text{par NEW-PAR puis SUBST} \\
\equiv \nu x.\, \{^{N}/_{x}\} \mid \bar{a}\langle x\rangle & & \text{par REWRITE} \\
\equiv \nu x.\, \{^{N}/_{x}\} \mid \bar{a}\langle N\rangle & & \text{par SUBST} \\
\equiv \bar{a}\langle N\rangle & & \text{par NEW-PAR puis ALIAS}
\end{array}$$

Les règles de congruence pour les substitutions actives font que deux processus structurellement congruents peuvent ne pas avoir le même ensemble de variables ou de noms libres. Par exemple, pour $P = \bar{a}\langle x\rangle$ et $Q = \nu y.\, \{^{N}/_{y}\} \mid \bar{a}\langle x\rangle$, on a bien $P \equiv Q$, mais $fv(Q) = \{x, y\} \neq \{x\} = fv(P)$. On définit alors les ensembles suivants, qui eux sont stables par congruence structurelle :

$$\overline{fn}(P) \triangleq \bigcap_{Q \equiv P} fn(Q) \qquad \overline{fv}(P) \triangleq \bigcap_{Q \equiv P} fv(Q) \qquad \overline{fnv}(P) \triangleq \overline{fn}(P) \cup \overline{fv}(P)$$

Notons que, si P est un processus du pi-calcul, on a $\overline{fn}(P) = fn(P)$. On définit de manière analogue ces ensembles pour des termes :

$$\overline{fn}(M) \triangleq \bigcap_{N = M} fn(N) \qquad \overline{fv}(M) \triangleq \bigcap_{N = M} fv(N) \qquad \overline{fnv}(M) \triangleq \overline{fn}(M) \cup \overline{fv}(M)$$

Relation de réduction. On appelle *relation de réduction* et on note $\rightarrow$ la plus petite relation sur les processus étendus close par congruence structurelle et par application de contexte d'évaluation, et vérifiant :

$$\begin{array}{ll}
\text{COMM-T} & \bar{a}\langle x\rangle.P \mid a(x).Q \rightarrow P \mid Q \\
\text{COMM-C} & \bar{a}^{\mathsf{ch}}\langle m\rangle.P \mid a^{\mathsf{ch}}(n).Q \rightarrow P \mid Q[n \leftarrow m] \\
\text{THEN} & \mathsf{if}\ M = M\ \mathsf{then}\ P\ \mathsf{else}\ Q \rightarrow P \\
\text{ELSE} & \mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q \rightarrow Q \\
 & \qquad \text{si } M \text{ et } N \text{ sont clos, et que } \mathcal{E} \nvdash M = N
\end{array}$$

La règle COMM-T est un peu inhabituelle, puisqu'elle ne traite que de l'envoi d'une variable composé avec la réception de cette même variable. Il se trouve cependant que cette forme particulièrement simple est suffisante pour traiter de l'envoi de n'importe quel terme, grâce à l'emploi des substitutions actives. Étudions par exemple la réduction de $\bar{a}\langle f(y)\rangle \mid a(y).\bar{b}\langle y\rangle$:

$$\begin{array}{lll}
\bar{a}\langle f(y)\rangle \mid a(y).\bar{b}\langle y\rangle & & \\
\quad \equiv \bar{a}\langle f(y)\rangle \mid a(x).\bar{b}\langle x\rangle & & \text{par } \alpha\text{-conversion} \\
\quad \equiv \nu x.\, \{^{f(y)}/_{x}\} \mid \bar{a}\langle f(y)\rangle \mid a(x).\bar{b}\langle x\rangle & & \text{par ALIAS puis NEW-PAR} \\
\quad \equiv \nu x.\, \{^{f(y)}/_{x}\} \mid \bar{a}\langle x\rangle \mid a(x).\bar{b}\langle x\rangle & & \text{par SUBST} \\
\quad \rightarrow \nu x.\, \{^{f(y)}/_{x}\} \mid \mathbf{0} \mid \bar{b}\langle x\rangle & & \text{par COMM-T} \\
\quad \equiv \bar{b}\langle f(y)\rangle & & \text{par PAR-}\mathbf{0}\text{, SUBST et ALIAS}
\end{array}$$

3.1.4 Exemple : protocole à chiffrement asymétrique

Nous décrivons ici une implémentation relativement naïve (adaptée de l'exemple de l'article d'Abadi et Fournet [AF01]) d'une communication chiffrée à l'aide d'une paire de clés publique et privée sur un canal public. On notera $\mathrm{pk}(s)$ la clé publique obtenue à partir d'un *nonce* (c'est-à-dire un très grand nombre aléatoire) s, et $\mathrm{sk}(s)$ la clé privée correspondante. On se munit alors d'une fonction de chiffrement enc et d'une fonction de déchiffrement dec vérifiant, pour tout message M, $\mathrm{dec}(\mathrm{enc}(M, \mathrm{pk}(s)), \mathrm{sk}(s)) = M$.

Ce formalisme se traduit très simplement en pi-calcul appliqué sous la forme d'une signature $\Sigma = \{\mathrm{pk}, \mathrm{sk}, \mathrm{dec}, \mathrm{enc}\}$, et d'une théorie équationnelle $\mathcal{E}$ obtenue à partir de l'axiome $\mathrm{dec}(\mathrm{enc}(x, \mathrm{pk}(n)), \mathrm{sk}(n)) = x$.

On peut alors par exemple écrire un processus (à l'utilité non avérée) publiant sa clé publique sur un canal a et attendant un message chiffré à l'aide de cette clé sur le canal b. Il retourne ensuite le message déchiffré sur le canal c :

$$\nu s.\, (\bar{a}\langle \mathrm{pk}(s)\rangle \mid b(x).\bar{c}\langle \mathrm{dec}(x, \mathrm{sk}(s))\rangle)$$

Ce processus peut par exemple communiquer avec le processus publiant sur le canal b un message M, chiffré à l'aide de la clé publique préalablement récupérée sur le canal a, ce qui donne :

$$\begin{aligned}
&\nu s.\, (\bar{a}\langle \mathrm{pk}(s)\rangle \mid b(x).\bar{c}\langle \mathrm{dec}(x, \mathrm{sk}(s))\rangle) \mid a(x).\bar{b}\langle \mathrm{enc}(M, x)\rangle \\
&\qquad \to \nu y.\, \{^{\mathrm{pk}(s)}/_{y}\} \mid (\nu s.\, b(x).\bar{c}\langle \mathrm{dec}(x, \mathrm{sk}(s))\rangle) \mid \bar{b}\langle \mathrm{enc}(M, y)\rangle \\
&\qquad \to \nu s.\, \bar{c}\langle \mathrm{dec}(\mathrm{enc}(M, \mathrm{pk}(s)), \mathrm{sk}(s))\rangle \\
&\qquad \equiv \bar{c}\langle M\rangle
\end{aligned}$$

Avant de passer à la suite, remarquons la simplicité du formalisme et sa flexibilité : on peut par exemple aisément prendre en compte le fait que les fonctions enc et dec *commutent* (comme c'est par exemple le cas pour certains chiffrements RSA) en ajoutant à la théorie équationnelle l'équation $\mathrm{dec}(\mathrm{enc}(x, y), z) = \mathrm{enc}(\mathrm{dec}(x, z), y)$. Plus d'exemples d'utilisation, dont les implémentations sont rarement plus complexes que celle-ci, sont détaillés dans l'article original [AF01].

3.1.5 Cadres, équivalence statique

Un *cadre*[3] est un processus étendu obtenu à partir de $\mathbf{0}$, des substitutions actives, de la composition parallèle et des deux restrictions. On peut définir le cadre associé à un processus étendu P, noté $\varphi(P)$, comme étant P dans lequel chaque processus simple a été remplacé par $\mathbf{0}$. On obtient de même le processus simple associé à P, que l'on note $(P)^s$, en annulant ses substitutions actives *non restreintes*. Le *domaine* d'un cadre

[3] En anglais *frame*.

φ, noté $dom(\varphi)$, est l'ensemble des variables sur lesquelles agit le cadre : $dom(\nu\tilde{u}.\{^{M_1}/_{x_1}\} \mid \cdots \mid \{^{M_n}/_{x_n}\}) \triangleq \{x_1, \ldots, x_n\} \setminus \{\tilde{u}\}$. La notion de domaine s'étend alors naturellement aux processus étendus, en posant, pour tout processus P, $dom(P) \triangleq dom(\varphi(P))$.

Remarquons qu'à l'aide des règles de congruence structurelle, on peut écrire n'importe quel processus P sous la forme $\nu\tilde{n}.(\varphi \mid Q)$, où φ est un cadre, et Q est un processus simple. Pour accentuer cette séparation entre la partie statique et la partie simple d'un processus, on notera ce même processus $\nu\tilde{n}.(\varphi[Q])$. On a alors $\nu\tilde{n}.\varphi \equiv \varphi(P)$.

Lemme 3.1 *Si $P \equiv Q$, alors $\varphi(P) \equiv \varphi(Q)$.*

DÉMONSTRATION : Il suffit de reprendre la preuve de $P \equiv Q$ en remplaçant tous les processus simples qui y apparaissent par $\mathbf{0}$, et en supprimant les règles qui ne s'appliquent plus. □

Il est par contre faux que $P \equiv Q$ implique $(P)^s \equiv (Q)^s$, et ce à cause de l'action possible des substitutions actives sur les processus simples. Ainsi, $\{^y/_x\} \mid \bar{a}\langle x\rangle \equiv \{^y/_x\} \mid \bar{a}\langle y\rangle$, mais $\bar{a}\langle x\rangle \not\equiv \bar{a}\langle y\rangle$.

Définissons maintenant *l'équivalence statique*, qui identifie les cadres qui égalisent les mêmes termes.

Définition 3.2 *On dit qu'un cadre φ égalise deux termes M et N, ce que l'on note $\mathcal{E}, \varphi \vdash M = N$ (ou simplement $\varphi \vdash M = N$ lorsqu'il n'y a pas d'ambiguïté), si il existe des noms et variables $\tilde{u}$ et une composition parallèle de substitutions σ tels que $\varphi \equiv \nu\tilde{u}.\sigma$, $M\sigma = N\sigma$ et $\{\tilde{u}\} \cap fnv(M, N) = \emptyset$.*

On notera de la même manière $P \vdash M = N$ si $\varphi(P) \vdash M = N$.

Remarque 3.3 *Cette définition est équivalente à la suivante : $\varphi \vdash M = N$ si pour un nom frais a, on a $\varphi \mid \bar{a}\langle M\rangle \equiv \varphi \mid \bar{a}\langle N\rangle$.*

On a donc par exemple, pour une fonction h unaire et sans équation (on peut donc penser à h comme à une fonction de hachage parfaite), et pour $\varphi_1 = \nu s.\{^s/_x\} \mid \{^{h(s)}/_y\}$, $\varphi_1 \vdash h(x) = y$.

Définition 3.4 *On dit que deux cadres φ et ψ sont* statiquement équivalents, *et on note $\varphi \approx_s \psi$, si $dom(\varphi) = dom(\psi)$ et si, pour tous termes M et N, $\varphi \vdash M = N$ si et seulement si $\psi \vdash M = N$.*

De même, on dit que deux processus P et Q sont statiquement équivalents et on note $P \approx_s Q$ si $\varphi(P) \approx_s \varphi(Q)$.

Si l'on pose $\varphi_2 = \nu s.\{^{h(s)}/_x\} \mid \{^{h'(s)}/_y\}$ et $\varphi_3 = \nu s.\{^s/_x\} \mid \nu s'.\{^{s'}/_y\}$, où h et h' sont deux fonctions sans équation, on a ainsi $\varphi_2 \approx_s \varphi_3$. On ne peut en effet pas distinguer, du moins vu de l'extérieur, entre deux secrets indépendants, comme c'est le cas pour φ_3, et deux secrets générés à partir d'une même clé, par exemple par hachage, comme c'est le cas pour φ_2.

Par contre, on a $\varphi_1 \not\approx_s \varphi_2$, puisque $\varphi_2 \nvdash h(x) = y$, contrairement à φ_1.

3.2 Une logique spatiale adaptée

3.2.1 Deux opérateurs séparants

On pourrait facilement adapter la logique spatiale vue à la section 2.2 au traitement du pi-calcul appliqué, en gardant le même opérateur séparant. Sa sémantique, et celle de son adjoint $\triangleright$, serait alors la suivante :

$$P \vDash A_1 \mid A_2 \text{ ssi } \exists P_1, P_2.\, P \equiv P_1 \mid P_2 \text{ et } P_i \vDash A_i \ (i \in \{1,2\})$$
$$P \vDash A \triangleright B \text{ ssi } \forall Q \perp P.\, Q \vDash A \text{ implique } P \mid Q \vDash B$$

On a ici pris garde à ne pas générer de processus mal formés dans la sémantique de $\triangleright$, en ne quantifiant que sur les processus Q dont la mise en parallèle avec P est bien formée (on écrit $P \perp Q$ lorsque c'est le cas).

Ainsi défini, l'opérateur $\mid$ sépare à la fois la composante statique et la composante dynamique du processus ; les processus simples et les substitutions actives sont traitées indifféremment, et l'adjoint peut ajouter au processus étendu courant de la connaissance en même temps qu'un réel processus au sens du pi-calcul.

Une manière de faire plus naturelle pour traiter du pi-calcul appliqué, et qui a été retenue ici, est alors de définir deux opérateurs séparants sur les processus étendus, l'un séparant les processus simples à cadre constant et l'autre se chargeant de découper le cadre en deux sous-cadres disjoints, tout en conservant de part et d'autre le même processus simple. On peut ainsi d'une part se préoccuper de l'interaction de processus simples partageant la même connaissance, et d'autre part étudier l'effet de l'apport ou de la soustraction de connaissance sur un processus simple.

Deux nouvelles opérations sur les processus. Définissons d'abord les effets de telles séparations sur les processus étendus eux-mêmes. Nous noterons $P \nmid Q$ la composition des processus P et Q à cadre constant, c'est-à-dire lorsque $\varphi(P) = \varphi(Q)$. L'opération $P * Q$ représentera quant à elle la composition des cadres de P et de Q et, de la même manière, ne sera définie qu'à processus simples égaux.

Définition 3.5 *Soient $\varphi, P_1, P_2, \tilde{n}, \tilde{n}_1, \tilde{n}_2$, tels que :*
- $\tilde{n} = \tilde{n}_1\tilde{n}_2$
- $\{\tilde{n}_1\} \cap fn(P_2) = \{\tilde{n}_2\} \cap fn(P_1) = \emptyset$

On pose :
$$\nu\tilde{n}.\,(\varphi[P_1]) \nmid \nu\tilde{n}.\,(\varphi[P_2]) \triangleq \nu\tilde{n}.\,(\varphi[P_1 \mid P_2])$$

Définition 3.6 *Soient $\varphi_1, \varphi_2, P, \tilde{n}, \tilde{n}_1, \tilde{n}_2$, tels que :*
- $\tilde{n} = \tilde{n}_1\tilde{n}_2$
- $\{\tilde{n}_1\} \cap fn(\varphi_2) = \{\tilde{n}_2\} \cap fn(\varphi_1) = \emptyset$
- $\varphi_1 \perp \varphi_2$

On pose :

$$\nu\tilde{n}.\,(\varphi_1[P]) * \nu\tilde{n}.\,(\varphi_2[P]) \triangleq \nu\tilde{n}.\,((\varphi_1 \mid \varphi_2)[P])$$

La présence de noms restreints en tête des processus est rendue nécessaire par le fait que ces noms peuvent apparaîtres dans la partie du processus qui reste constante, comme c'est par exemple le cas pour le processus

$$\nu n, m.\,\{^{(n,m)}/_x\}[\bar{a}\langle n\rangle \mid \bar{b}\langle m\rangle] = \nu n, m.\,\{^{(n,m)}/_x\}[\bar{a}\langle n\rangle] \nmid \nu n, m.\,\{^{(n,m)}/_x\}[\bar{b}\langle m\rangle]$$

Ces deux définitions sont incomplètes : on aimerait par exemple pouvoir définir $P \nmid Q$ ou $P * Q$ dès lors que P et Q vérifient les contraintes nécessaires à réécriture près. Le lemme suivant, dont la démonstration (relativement technique) est fournie à l'annexe A.2, va nous permettre d'étendre ainsi la définition :

Lemme 3.7 *Pour tous processus P, Q, P', Q', si $P \equiv P'$, $Q \equiv Q'$, et si $P \nmid Q$ (resp. $P * Q$) et $P' \nmid Q'$ (resp. $P' * Q'$) sont définis, alors $P \nmid Q \equiv P' \nmid Q'$ (resp. $P * Q \equiv P' * Q'$).*

On pose alors, pour des processus P et Q pour lesquels il existe $P' \equiv P$ et $Q' \equiv Q$ tels que $P' \nmid Q'$ (resp. $P' * Q'$) soit défini, $P \nmid Q = P' \nmid Q'$ (resp. $P * Q = P' * Q'$). Le lemme précédent nous garantie alors que ces définitions sont uniques à congruence structurelle près.

On note $P \nmid Q\downarrow$ et $P * Q\downarrow$ lorsque ces opérations sont ainsi définies. Nous pouvons maintenant définir notre version de la logique spatiale pour le pi-calcul appliqué.

3.2.2 Syntaxe et sémantique

Comme pour le pi-calcul, on distingue deux fragments dans la logique (notée cette fois $\mathcal{L}$) : l'un intensionnel (noté $\mathcal{L}_{\text{int}}$) et l'autre extensionnel (noté $\mathcal{L}_{\text{ext}}$), tous deux partageant un même sous-fragment $\mathcal{L}_{\text{core}}$. On ne s'intéressera dans la suite qu'au fragment intensionnel dont on étudiera l'expressivité de diverses manières. La syntaxe des différents opérateurs est décrite figure 4. On notera la présence de deux opérateurs de révélation et de dissimulation, et de deux quantifications fraîches. Dans les trois cas, on a distingué suivant que l'opérateur s'appliquait à un nom ou une variable[4]. On a de plus inclus dans la logique les jugements d'égalité entre termes relativement au cadre du processus (comme défini à la section 3.1.5).

La relation de satisfaction, décrite à la figure 5, s'exprime alors presque aussi simplement que précédemment grâce à la définition des opérations $\nmid$ et $*$ sur les processus. On notera que certaines restrictions sont apparues pour s'assurer que l'on ne considère que des processus bien formés.

[4]On commet un abus de notation en notant chaque couple d'opérateurs de la même manière, mais on lève l'ambiguïté grâce à l'opérande utilisée (par exemple x pour une variable et n pour un nom).

Figure 4 Syntaxe de la logique spatiale

	$A ::=$	
$\mathcal{L}_{\text{core}}$	$M = N$	comparaison de termes
	$\neg A$	négation
	$A \wedge A$	conjonction
	$\mathit{V}x.\, A$	quantification fraîche sur les variables
	$\mathit{V}a.\, A$	quantification fraîche sur les noms
	$\Diamond A$	modalité (forte)
	$A \rhd A$	garantie (processus simples)
	$A \mathrel{-\!\!*} A$	garantie (processus étendus)
$\mathcal{L}_{\text{int}}$	$\mathbf{0}$	processus nul
	$\emptyset$	cadre nul
	$A \mid A$	composition séparante (processus simples)
	$A * A$	conjonction séparante (processus étendus)
	$x \circledR A$	révélation de variable
	$a \circledR A$	révélation de nom
$\mathcal{L}_{\text{ext}}$	$A \oslash x$	dissimulation de variable
	$A \oslash a$	dissimulation de nom

Figure 5 Relation de satisfaction

$\mathcal{L}_{\text{core}}$	$P \vDash M = N$	ssi $P \vdash M = N$
	$P \vDash \neg A$	ssi $P \nvDash A$
	$P \vDash A_1 \wedge A_2$	ssi $P \vDash A_1$ et $P \vDash A_2$
	$P \vDash \mathit{V}x.\, A$	ssi $\forall y \notin \mathit{fv}(P, A).\, P \vDash A[x \leftarrow y]$
	$P \vDash \mathit{V}a.\, A$	ssi $\forall b \notin \mathit{fn}(P, A).\, P \vDash A[a \leftarrow b]$
	$P \vDash \Diamond A$	ssi $\exists Q.\, P \rightarrow Q$ et $Q \vDash A$
	$P \vDash A_1 \rhd A_2$	ssi $\forall Q.\, (P \mid Q\!\downarrow$ et $Q \vDash A_1)$ implique $P \mid Q \vDash A_2$
	$P \vDash A_1 \mathrel{-\!\!*} A_2$	ssi $\forall Q.\, (P * Q\!\downarrow$ et $Q \vDash A_1)$ implique $P * Q \vDash A_2$
$\mathcal{L}_{\text{int}}$	$P \vDash \mathbf{0}$	ssi $(P)^s \equiv \mathbf{0}$
	$P \vDash \emptyset$	ssi $\varphi(P) \equiv \mathbf{0}$
	$P \vDash A_1 \mid A_2$	ssi $\exists P_1, P_2.\, P \equiv P_1 \mid P_2,\ P_1 \vDash A_1$ et $P_2 \vDash A_2$
	$P \vDash A_1 * A_2$	ssi $\exists P_1, P_2.\, P \equiv P_1 * P_2,\ P_1 \vDash A_1$ et $P_2 \vDash A_2$
	$P \vDash x \circledR A$	ssi $\exists Q.\, P \equiv \nu x.\, Q,\ x \in \mathit{dom}(Q)$ et $Q \vDash A$
	$P \vDash a \circledR A$	ssi $\exists Q.\, P \equiv \nu a.\, Q$ et $Q \vDash A$
$\mathcal{L}_{\text{ext}}$	$P \vDash A \oslash x$	ssi $x \in \mathit{dom}(P)$ et $\nu x.\, P \vDash A$
	$P \vDash A \oslash a$	ssi $\nu a.\, P \vDash A$

Comme dans le cas du pi-calcul, la sémantique des deux quantifications fraîches peut être définie indifféremment de manière existentielle ou de manière universelle, comme l'exprime ce lemme, à la démonstration immédiate :

Lemme 3.8 *Pour tout processus P, tout nom ou variable u et toute formule A, $P \vDash И u.\, A$ ssi $P \vDash \neg И u.\, \neg A$.*

Nous pouvons alors montrer que la relation de satisfaction est invariante par congruence structurelle :

Lemme 3.9 *Pour tous processus P et Q et pour toute formule A, si $P \vDash A$ et $P \equiv Q$, alors $Q \vDash A$.*

DÉMONSTRATION : Par induction structurelle sur A, le seul cas délicat étant celui du quantificateur $И$: si $P \vDash И u.\, A$, alors, par définition, $\forall u' \notin fv(P, A).\, P \vDash A[u \leftarrow u']$, donc en particulier, pour $v \notin fnv(P) \cup fnv(Q)$, $P \vDash A[u \leftarrow v]$. Par hypothèse d'induction, $Q \vDash A[u \leftarrow v]$, donc $\exists u' \notin fv(P, A).\, Q \vDash A[u \leftarrow u']$ soit $Q \vDash \neg И u.\, \neg A$, et on conclut par le lemme 3.8. □

3.2.3 Échauffement

Définissons quelques formules simples dans le fragment intensionnel, qui nous seront utiles par la suite, regroupées figure 6. On retrouve ici certaines formules définies pour le pi-calcul à la section 2.2.

Les preuves correspondantes sont regroupées à l'annexe B.1, et il est à noter que celle de la formule public nécessite de prouver un lemme relativement technique sur la stabilité des ensembles $\widetilde{fn}(P)$ par substitution d'un nom du processus par un nom frais (énoncé à l'annexe A.1). Cette formule n'a d'ailleurs pas d'équivalent sur les variables, ou du moins la formule $\neg И x.\, x \mathbin{®©} x$ n'est-elle vérifiée par aucun processus. En effet, d'après les règles de bonne formation des processus étendus, révéler une variable x fait toujours apparaître une substitution active sur x, donc x apparaîtra libre dans le processus résultant.

3.3 Choix techniques

3.3.1 Typage des canaux

Les définitions de [AF01] sont plus libérales en ce qui concerne l'utilisation des variables dans la grammaire des processus, mais s'appuient en contrepartie sur des restrictions de typage, gardées plus ou moins implicites dans l'article en question. Ainsi, les entrées et sorties peuvent s'écrire de manière plus générale $u(x).P$ et $\bar{u}\langle M \rangle.P$, en s'assurant ensuite que les communications sont bien typées (par exemple, qu'on n'envoie pas un entier là où on attend un nom de canal).

Figure 6 Quelques formules et leur caractérisation

$$\top \triangleq \mathbf{0} \vee \neg\mathbf{0} \qquad \copyright\, u \triangleq \neg u \,®\, \top \qquad \mathsf{public} \triangleq \neg \mathcal{U} n.\, n \,®\, \copyright\, n$$

$$\mathbf{1} \triangleq \neg\mathbf{0} \wedge \neg(\neg\mathbf{0} \mid \neg\mathbf{0}) \qquad \mathbb{I} \triangleq \neg\emptyset \wedge \neg(\neg\emptyset * \neg\emptyset) \qquad \mathsf{single} \triangleq \mathbf{1} \wedge \emptyset \wedge \mathsf{public}$$

$$A[B] \triangleq (A \wedge \mathbf{0}) \mid ((B \wedge \emptyset) * \top) \qquad A \blacktriangleright B \triangleq \neg(A \rhd \neg B)$$

$$A \mathrel{-\!\!\!\overrightarrow{*}} B \triangleq \neg(A \mathrel{-\!\!*} \neg B)$$

$P \vDash \top$	toujours
$P \vDash \copyright\, x$	*ssi* $x \in \overline{fv}(P)$
$P \vDash \copyright\, n$	*ssi* $n \in \overline{fn}(P)$
$P \vDash \mathsf{public}$	*ssi* $\forall n, Q.\, P \equiv \nu n.\, Q$ implique $n \notin \overline{fn}(Q)$
	ssi $\forall n, Q.\, P \equiv \nu n.\, Q$ implique $P \equiv Q$
$P \vDash \mathbf{1}$	*ssi* $(P)^s \not\equiv \mathbf{0}$ et $(P)^s \equiv P_1 \mid P_2$ implique $\exists i \in \{1,2\}.\, P_i \equiv \mathbf{0}$
$P \vDash \mathbb{I}$	*ssi* $\varphi(P) \not\equiv \mathbf{0}$ et $\varphi(P) \equiv P_1 \mid P_2$ implique $\exists i \in \{1,2\}.\, P_i \equiv \mathbf{0}$
$P \vDash \mathsf{single}$	*ssi* P est une communication ou une conditionnelle
$P \vDash A[B]$	*ssi* $\varphi(P) \vDash A$ et $\exists Q \equiv P.\, (Q)^s \vDash B$
$P \vDash A \blacktriangleright B$	*ssi* $\exists Q.\, P \mid Q\!\downarrow,\ Q \vDash A$ et $P \mid Q \vDash B$
$P \vDash A \mathrel{-\!\!\!\overrightarrow{*}} B$	*ssi* $\exists Q.\, P * Q\!\downarrow,\ Q \vDash A$ et $P * Q \vDash B$

Un des résultats du-dit article est, étant donnés une relation d'équivalence observationnelle et un système de transitions étiquetées tous deux définis dans l'article, que l'équivalence observationnelle correspond à la bisimulation étiquetée. Or, il est nécessaire pour obtenir ce résultat (et ceci n'est pas mentionné explicitement dans l'article), d'interdire l'écriture d'un processus étendu où une substitution active par un nom de canal restreint (donc de la forme $\nu n.\, \{^n/_x\}$) se fait sur une variable non restreinte.

En effet, le processus $\nu a.\, (\{^a/_x\} \mid \bar{a}\langle a\rangle)$ n'est pas observationnellement équivalent à $\mathbf{0}$ (sa mise en parallèle avec $x(y)$ donne le processus $\nu a.\, (\{^a/_x\} \mid \bar{a}\langle a\rangle \mid a(y))$ qui peut se réduire en $\nu a.\, \{^a/_x\}$), mais aucune transition étiquetée ne lui correspond (un tel comportement n'étant en fait pas souhaitable). Cette restriction est néanmoins peu naturelle, et on a préféré ici faire rentrer le système de types dans la syntaxe, pour plus de simplicité. Il n'y a alors pas d'autres restrictions que celles qui ont été mentionnées à la section 3.1.2.

3.3.2 Congruence structurelle

Application des substitutions actives. À l'origine, les substitutions actives s'appliquaient aussi bien aux processus simples qu'aux autres substitutions actives ; la règle de congruence structurelle SUBST était alors la suivante :

$$\textsc{Subst}' \quad \{^M/_x\} \mid P \equiv \{^M/_x\} \mid P[x \leftarrow M]$$

Cette règle est néanmoins problématique dès que l'on tente de séparer un processus étendu en deux (ou même seulement son cadre, comme c'est le cas pour l'opération $*$). En effet, comment décider quelles substitutions appliquer (sens direct de la règle de congruence structurelle) ou factoriser (sens indirect) avant le découpage ? Nous avons ici fait le choix de limiter le champ d'action des substitutions actives aux seuls processus simples. Ce choix ne change ni la relation d'équivalence statique, ni l'ensemble des états accessibles d'un processus. Il suffit pour montrer cela de remarquer que les substitutions ont la même action sur les processus simples que l'on utilise la règle SUBST ou la règle SUBST'. Comme de plus les réductions se font uniquement au sein des processus simples, la relation de réduction est inchangée. On conclut de même de la remarque 3.3 que la relation d'équivalence statique est inchangée.

Le seul désavantage de ce choix est alors qu'on ne peut pas toujours écrire un processus sous la forme $\nu\tilde{n}.\, P$, où P n'a aucune variable liée : on ne peut pas se débarasser de la substitution sur x à l'aide des nouvelles règles de congruence structurelle dans le cadre $\nu x.\, \{^z/_x\} \mid \{^x/_y\}$.

Tests conditionnels. La règle TEST-S a été ajoutée uniquement pour simplifier l'énoncé du théorème 4.1 qui caractérise l'équivalence logique, et n'a aucune autre conséquence sur le comportement du calcul.

Invariance par contexte. Toujours par souci de faire correspondre de la manière la plus naturelle possible le résultat du théorème 4.1 avec la relation de congruence structurelle, on a ici fait le choix (au demeurant bénin) de rendre cette dernière invariante par contexte, au lieu d'être simplement invariante par contexte *d'évaluation* comme c'était le cas dans l'article d'Abadi et Fournet [AF01]. Cette définition semble de plus plus naturelle : elle identifie par exemple les processus $\bar{a}\langle M\rangle.(P \mid Q)$ et $\bar{a}\langle M\rangle.(Q \mid P)$ contrairement à précédemment.

4 Équivalence logique

Comme nous allons maintenant le voir, la logique $\mathcal{L}_{\text{int}}$ est, à l'instar du fragment $\mathcal{L}^{\pi}_{\text{int}}$ pour le pi-calcul, *intensionnelle* ; on retrouve le même résultat que celui obtenu pour cette dernière :

Théorème 4.1 *Pour tout processus étendu P, il existe une formule $\mathbf{F}_P$ telle que, pour tout processus Q :*

$$Q \vDash \mathbf{F}_P \textit{ si et seulement si } Q \equiv P$$

Corollaire 4.2 *Pour tous processus étendus P et Q, P et Q satisfont les mêmes formules de $\mathcal{L}_{int}$* si et seulement si $P \equiv Q$.

Le reste de cette section est dévolu à la démonstration de ce théorème, par la définition de la formule $\mathbf{F}_P$. On ne donnera la plupart du temps que les intuitions nécessaires à la compréhension des formules, les démonstrations complètes pouvant être trouvées à l'annexe B.

4.1 Caractérisation logique des substitution actives

Commençons par trouver une formule $\mathsf{Subst}(x = M)$ qui caractérise précisément, pour une variable x et un terme M donnés, les processus congruents à la substitution active $\{^{M}/_{x}\}$. On donne figure 7 les quelques formules intermédiaires nécessaires à la définition de $\mathsf{Subst}(x = M)$.

Figure 7 Formules caractéristiques pour les substitutions actives

$$\mathsf{Substs} \triangleq \mathbf{0} \wedge \mathsf{public} \wedge \neg \mathcal{И}x.\, x \,®(\neg\emptyset \wedge ©\, x) * (\neg\emptyset \wedge ©\, x)$$

$$\mathsf{Subst} \triangleq \mathsf{Substs} \wedge \mathbb{I} \qquad \mathsf{Subst}(x) \triangleq \mathsf{Subst} \wedge \mathcal{И}n.\, (\mathsf{Subst} \wedge x = n) \mathrel{-\!\!*} \neg\top$$

$$\mathsf{Subst}(x = M) \triangleq \mathsf{Subst}(x) \wedge x = M$$

Ces formules sont relativement simples, notamment du fait de la formule $\mathbb{I}$: on commence par définir les processus formés d'un nombre quelconque de substitutions (formule Substs), puis d'une seule (formule Subst). On prend garde à ne pas inclure de cadres où apparaissent des noms restreints grâce à public, ou des variables restreintes grâce à $\mathit{И}x.\, x\,®(\neg\emptyset \wedge ©\, x) * (\neg\emptyset \wedge ©\, x)$.

On caractérise ensuite le fait de contenir x dans son domaine en utilisant le fait qu'alors la substitution $\{^n/_x\}$ ne fait pas partie des processus avec lesquels le processus peut se composer. Or, pour un nom frais n, un processus vérifiant $\mathsf{Subst} \wedge x = n$ ne peut être que cette substitution-ci, donc un processus P ne vérifie $\mathit{И}n.\, (\mathsf{Subst} \wedge x = n) \mathrel{-\!\!*} \neg\top$ que si x fait partie de son domaine, puisqu'alors l'ensemble des processus Q tels que $P * Q$ est défini et vérifiant $\mathsf{Subst} \wedge x = n$ est vide, donc par propriété de l'ensemble vide, chacun de ces processus est tel que $P * Q$ vérifie la formule fausse $\neg\top$. La formule $\mathsf{Subst}(x)$ caractérise donc bien exactement toutes les substitutions sur x.

Enfin, pour un terme M différent de x (au sens $\mathcal{E} \nvdash M = x$), la formule $\mathsf{Subst}(x = M)$ caractérise les substitutions actives de la forme $\{^N/_x\}$ pour un certain N telles que $(x = M)[x{\leftarrow}N]$. On a alors forcément $\mathcal{E} \vdash N = M$ car x n'apparaît pas dans N (sinon la substitution serait mal formée), et la seule égalité « rajoutée » par $\{^N/_x\}$ est $x = N$.

Notations. On notera maintenant $\mathsf{Subst}(x_1, \ldots, x_n)$ la formule $\mathsf{Subst}(x_1) * \cdots * \mathsf{Subst}(x_n)$, et $\mathsf{Subst}(x_1 = M_1, \ldots, x_n = M_n)$ la formule $\mathsf{Subst}(x_1 = M_1) * \cdots * \mathsf{Subst}(x_n = M_n)$ quels que soient les variables $x_1, \ldots, x_n$ et les termes $M_1, \ldots, M_n$. On s'autorisera également à mélanger les deux notations, en écrivant par exemple $\mathsf{Subst}(x, y = M)$ pour caractériser les processus de la forme $\{^N/_x\} \mid \{^M/_y\}$ pour un certain N.

4.2 Caractérisation logique des communications

Détaillons maintenant les formules $\mathsf{in}(a, x).A$ et $\mathsf{out}(a, M).A$, qui caractérisent respectivement les processus de la forme $a(x).Q$ et $\bar{a}\langle M\rangle.Q$ où Q vérifie à son tour la formule A (voir figure 8). On définira ensuite des formules $\mathsf{in}^{\mathsf{ch}}(a, b).A$ et $\mathsf{out}^{\mathsf{ch}}(a, b)$ qui caractériseront de même les processus $a^{\mathsf{ch}}(b).Q$ et $\bar{a}^{\mathsf{ch}}\langle b\rangle.Q$ où Q vérifie A.

Pour cela, on va tout d'abord caractériser, pour un certain a et un certain x, les processus de la forme $\bar{a}\langle x\rangle$, ce qui nous permettra ensuite de définir les processus de la forme $a(x).Q$ (en les faisant interagir avec les processus de la forme $\bar{a}\langle x\rangle$) où Q vérifie une certaine formule A, puis les processus de la forme $\bar{a}\langle M\rangle.Q$ pour n'importe quel terme M.

On définit préalablement la formule $©^{\mathsf{s}}_{=}\, x$, qui caractérise les processus dont la partie simple ne contient pas de sur-terme *strict* de x, c'est-à-dire dont on peut réécrire la partie simple de manière à ce qu'aucun terme

Figure 8 Formules caractéristiques pour les communications de termes

$$\overset{\mathsf{s}}{\underset{=}{\copyright}}\, x \triangleq \top[\copyright\, x] \wedge \neg \mathcal{U} z.\, (\mathsf{Subst}(z) \wedge z \neq x \wedge \copyright\, x)[\top] \mathrel{-\!\!\!\overrightarrow{*}} \top[\copyright\, z]$$

$$\mathsf{out}(a, x) \triangleq \mathrm{single} \wedge \copyright\, a \wedge \overset{\mathsf{s}}{\underset{=}{\copyright}}\, x \wedge (\mathrm{single} \blacktriangleright \Diamond \mathbf{0})$$

$$\mathsf{in}(a, x).A \triangleq \mathrm{single} \wedge \neg \Diamond \top \wedge \mathcal{U} x.\, \mathsf{out}(a, x) \rhd \Diamond A$$

$$\mathsf{out}(a, M).A \triangleq \mathcal{U} x.\, x \circledR \mathsf{Subst}(x = M)[\mathcal{U} b.\, \mathsf{in}(a, y).\mathsf{out}(b, y) \rhd \Diamond(\mathsf{out}(b, x) \nmid A \wedge \neg \copyright\, x)]$$

$M(x)$ tel que $x \in fv(M)$ et $\mathcal{E} \not\vdash M(x) = x$ n'y apparaisse. En suivant la formule, c'est l'ensemble des processus P dont la partie simple vérifie $\copyright\, x$ ($P \vDash \top[\copyright\, x]$), et tels qu'aucune substitution $\{^{M(x)}/_{z}\}$ pour une variable z fraîche et un terme $M(x) \neq x$ où x apparaît (c'est-à-dire aucune substitution vérifiant $\mathsf{Subst}(z) \wedge z \neq x \wedge \copyright\, x$) ne puisse se factoriser dans le processus simple ($\{^{M(x)}/_{z}\} \mid P \not\vDash \top[\copyright\, z]$, donc on ne peut pas faire apparaître z dans le processus simple de P).

Pour un nom a et une variable x donnés, le processus $\bar{a}\langle x\rangle$ est alors caractérisé par la formule $\mathsf{out}(a, x)$. En effet, tout processus P vérifiant cette formule est soit une communication, soit une conditionnelle (car il doit vérifier single), et il existe un processus non-nul (qui sera le processus $a(y)$) qui, composé avec P, peut se réduire en le processus nul. On a ici affaire à deux processus non nul se réduisant en le processus nul en *une* étape. Ce sont donc deux communications compatibles, chacune suivie du processus nul. Comme de plus P vérifie $\copyright\, a$ et $\overset{\mathsf{s}}{\underset{=}{\copyright}}\, x$, P est nécessairement une émission de *terme*, soit de la forme $\bar{a}\langle x\rangle$, soit de la forme $\bar{b}\langle M(a, x)\rangle$ où a et x apparaissent libres dans $M(a, x)$. Or dans ce dernier cas, $M(a, x)$ serait un sur-terme strict de x, ce que la formule $\overset{\mathsf{s}}{\underset{=}{\copyright}}\, x$ interdit. La formule $\overset{\mathsf{s}}{\underset{=}{\copyright}}\, x$ nous assure ainsi que c'est bien uniquement la *variable* x qui est envoyée, et pas seulement un terme où x apparaîtrait. Réciproquement, le processus $\bar{a}\langle x\rangle$ vérifie clairement $\mathsf{out}(a, x)$.

On peut maintenant facilement caractériser, étant donnée une formule A et un nom de canal a, les processus de la forme $a(x).Q$ où $Q \vDash A$ à l'aide de la formule $\mathsf{in}(a, x).A$ (on note qu'ici x est *lié* dans A) : c'est l'ensemble des processus vérifiant single, et dont la composition avec $\bar{a}\langle x\rangle$ se réduit en un processus vérifiant A. On demande de plus au processus d'origine de ne pouvoir se réduire de lui-même, pour être sûr que la réduction provient d'une interaction avec $\bar{a}\langle x\rangle$ (sinon le processus pourrait être une conditionnelle).

Enfin, la formule caractérisant les processus $\bar{a}\langle M\rangle.Q$ où Q vérifie à son

tour une formule A se révèle plus complexe, puisqu'il faut tester le terme envoyé pour s'assurer que c'est bien M. L'idée est de réécrire un tel processus sous la forme $\nu x.\, \{^M/_x\} \mid \bar{a}\langle M\rangle.Q$ (où x est fraîche pour $\bar{a}\langle M\rangle.Q$), à l'aide des règles de congruence structurelle, puis sous la forme $\{^M/_x\} \mid \bar{a}\langle x\rangle.Q$ en révélant x et en factorisant la substitution active. On écarte ensuite cette substitution pour composer le processus simple obtenu avec $a(y).\bar{b}\langle y\rangle$, pour un nom b frais. On obtient alors $a(y).\bar{b}\langle y\rangle \mid \bar{a}\langle x\rangle.Q$, qui se réduit en $\bar{b}\langle x\rangle \mid Q$. On vérifie ensuite qu'on a effectivement $\bar{b}\langle x\rangle$ d'un côté (à l'aide de la formule $\mathsf{out}(b,x)$), ce qui nous permet d'affirmer que le terme envoyé était bien M (à égalité près). L'autre partie du processus, qui est alors Q, doit elle vérifier A (puisque $\bar{b}\langle x\rangle \mid Q \vDash \mathsf{out}(b,x) \mathbin{¦} A$ et qu'on avait choisi b frais pour Q).

Remarque. Ces formules présentent deux manières différentes de composer un processus P avec une substitution active : l'une qui joue avec les règles de congruence structurelle pour réécrire le processus en $\nu x.\, \{^M/_x\} \mid P$ pour un certain M, et qui révèle ensuite la variable x, et l'autre qui utilise la baguette magique $-\!\!*$ (ou ici son dual $-\!\!\!\!\rightarrow\!\!\!\!*$) pour augmenter explicitement le cadre du processus. On notera que dans cette deuxième méthode, la formule utilisée à gauche de $-\!\!\!\!\rightarrow\!\!\!\!*$ est de la forme $\cdot\,[\top]$, puisque l'opérateur $*$ ne peut ajouter que des processus ayant la même partie simple que le processus d'origine. Ainsi, la composition exacte qui a lieu est par exemple $\{^M/_x\}[P] * P$.

Les formules de la figure 9 définissent de manière assez similaire les formules $\mathsf{in}^{\mathsf{ch}}(a,b).A$ et $\mathsf{out}^{\mathsf{ch}}(a,b).A$, la seule différence notable résidant dans la définition de $\mathsf{test}^{\mathsf{ch}}(a,b)$, qui joue ici le rôle que jouait la formule $\mathsf{out}(a,x)$ précédemment.

Figure 9 Formules caractéristiques pour les communications de noms

$$\mathsf{test}^{\mathsf{ch}}(a,b) \triangleq \copyright\, a \wedge \copyright\, b \wedge (\mathrm{single} \blacktriangleright \Diamond \mathbf{0}) \wedge \neg \mathit{И} x.\, \mathsf{Subst}(x)[\top] \mathbin{-\!\!\!\!\rightarrow\!\!\!\!*} \top[\copyright\, x]$$

$$\mathsf{in}^{\mathsf{ch}}(a,b).A \triangleq \mathrm{single} \wedge \neg\Diamond\top \wedge \mathit{И} b.\, \mathsf{test}^{\mathsf{ch}}(a,b) \blacktriangleright \Diamond A$$

$$\mathsf{out}^{\mathsf{ch}}(a,b).A \triangleq \mathrm{single} \wedge \mathit{И} b'.\, (\mathsf{in}^{\mathsf{ch}}(a,c).\mathsf{test}^{\mathsf{ch}}(b',c) \rhd \Diamond(\mathsf{test}^{\mathsf{ch}}(b',c) \mathbin{¦} A))$$

En effet, $\mathsf{test}^{\mathsf{ch}}(a,b)$ est vérifiée par les processus de la forme $\bar{a}^{\mathsf{ch}}\langle b\rangle$ ou $\bar{b}^{\mathsf{ch}}\langle a\rangle$: ce sont les processus qui, comme précédemment, peuvent se réduire à $\mathbf{0}$ lorsqu'il sont composés avec un certain autre processus vérifiant single, et où a et b apparaissent libres. Pour s'assurer qu'on a bien affaire à une communication de *nom* (et non par exemple à la communication d'un

terme contenant à la fois a et b), on requiert de plus qu'aucune substitution active ajoutée en parallèle ne puisse se factoriser : si le processus d'origine consistait en l'envoi d'un terme M avec par exemple $a \in \overline{fn}(M)$, la substitution $\{^a/_x\}$ pourrait, en se factorisant, faire apparaître x en tant que variable libre du processus.

4.3 Caractérisation logique des conditionnelles

Il ne nous reste plus qu'à décrire les formules définissant les processus de la forme $\mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q$. Les formules nécessaires sont regroupées figure 10. Le but est cette fois ci de parvenir à une formule $\mathsf{if}(M = N, A_1, A_2)$ où A_1 et A_2 sont des formules, telle que, pour tout processus P :

$$P \vDash \mathsf{if}(M = N, A_1, A_2) \textit{ ssi } P \equiv \mathsf{if}\ M = N\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$$

pour certains processus P_1 et P_2 tels que $P_1 \vDash A_1$ et $P_2 \vDash A_2$.

Figure 10 Formules caractéristiques pour les conditionnelles

$$\mathsf{if} \triangleq \mathsf{single} \wedge \mathit{И}x.\, x \circledR \mathsf{Subst}(x)[\mathsf{Subst}(x)[\top] \mathrel{-\!\!\!\overrightarrow{*}} \Diamond\top] \qquad \overset{\mathsf{br}}{\copyright}\, x \triangleq$$

$$\mathsf{if} \wedge \mathit{И}z, z'.\, z \circledR z' \circledR \mathsf{Subst}(z, z')[(\mathsf{Subst}(z, z') \wedge \neg \copyright\, x)[\top] \multimap \top[\Diamond \copyright\, x]]$$

$$\mathsf{if}(M = N, A, B) \triangleq \mathsf{if} \wedge \mathit{И}x, y.\, \mathsf{Subst}(x = M, y = N) \multimap \top \left[\begin{array}{l} \copyright^{\mathsf{s}}_{=} x \wedge \copyright^{\mathsf{s}}_{=} y \wedge \neg \copyright^{\mathsf{br}} x \wedge \neg \copyright^{\mathsf{br}} y \\ \wedge \mathsf{Subst}(x = y) \multimap \top[\Diamond A] \\ \wedge \mathit{И}s, s'.\, \mathsf{Subst}(x = s, y = s') \multimap \top[\Diamond B] \end{array} \right]$$

La première de ces formules, if, caractérise tous les processus conditionnels, quels que soient la nature de leur test et les sous-processus de chaque branche. En effet, tout processus vérifiant if est forcément de la forme $\mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q$ puisqu'il vérifie single et que, grâce au seul ajout de substitutions actives que nous détaillons plus bas, il est capable de se réduire. Étant mono-threadé, c'est donc bien une conditionnelle.

Réciproquement, soit un processus $P \equiv \mathsf{if}\ M = N\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$. On fait apparaître, comme dans la formule $\mathsf{out}(a, M).A$, une substitution sur x (pour une variable x fraîche) en révélant cette variable grâce à la règle de congruence structurelle ALIAS. On peut en particulier choisir ici cette substitution comme étant $\{^M/_x\}$, et on a $P \mid \{^M/_x\} \equiv \mathsf{if}\ x = N\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \mid \{^M/_x\}$. On met ensuite cette substitution de côté (à l'aide de la construction $\mathsf{Subst}(x)[\,\cdot\,]$), pour la remplacer par $\{^N/_x\}$, et on obtient le processus $\mathsf{if}\ x = N\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \mid \{^N/_x\}$, lui-même congruent à $\mathsf{if}\ x = x\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \mid \{^N/_x\}$ qui se réduit en $P_1 \mid \{^N/_x\}$.

Ainsi, par le truchement des règles d'application des substitutions actives, l'on est parvenu à remplacer le test d'origine par un autre trivialement vrai ($x = x$) pour obtenir un processus dont on est sûr qu'il se réduit.

La formule $©^{\mathsf{br}}\, x$ caractérise elle les processus conditionnels $\mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q$ où x apparaît dans P ou dans Q. On ne détaillera pas ici son fonctionnement, pour se concentrer sur celui de la formule $\mathsf{if}(M = N, A, B)$.

Celle-ci part d'un processus de la forme $\mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q$, qui est transformé en $\{^{M}/_{x}\} \mid \{^{N}/_{y}\} \mid \mathsf{if}\ x = y\ \mathsf{then}\ P\ \mathsf{else}\ Q$ en révélant les variables fraîches x et y et en faisant jouer les règles d'application. En effet, le processus simple obtenu doit vérifier $©^{\mathsf{test}}_{=}\, x \wedge ©^{\mathsf{test}}_{=}\, y$ donc le test, à égalité et symétrie près, est $x = y$, et $\neg\, ©^{\mathsf{br}}\, x \wedge \neg\, ©^{\mathsf{br}}\, y$, donc les substitutions n'ont pas agi sur les sous-processus. On explore ensuite chacune des branches en donnant successivement une valeur vraie puis fausse au test, respectivement à l'aide de $\{^{y}/_{x}\}$ qui permet d'égaliser x et y, et de $\{^{s}/_{x}\} \mid \{^{s'}/_{y}\}$ pour deux noms frais s et s', qui permet de *clore* le test $x = y$ en ayant $x \neq y$.

4.4 Tous ensemble

Nous pouvons maintenant démontrer le théorème 4.1 en construisant, pour un processus P, la formule $\mathbf{F}_P$ en question par induction structurelle sur P. Les formules correspondantes sont présentées figure 11, et on remarque que la construction $P_1 \mid P_2$ a été remplacée par les deux cas : $P_1 \nmid P_2$ et $P_1 * P_2$. En effet, à congruence structurelle près, on peut toujours écrire un processus en utilisant les opérations $\nmid$ et $*$ à la place de $\mid$.

Figure 11 Définition de $\mathbf{F}_P$

$$\mathbf{F}_{\mathbf{0}} \triangleq \mathbf{0} \qquad \mathbf{F}_{\{^{M}/_{x}\}} \triangleq \mathsf{Subst}(x = M) \qquad \mathbf{F}_{P_1 \nmid P_2} \triangleq \mathbf{F}_{P_1} \nmid \mathbf{F}_{P_2}$$

$$\mathbf{F}_{P_1 * P_2} \triangleq \mathbf{F}_{P_1} * \mathbf{F}_{P_2} \qquad \mathbf{F}_{\nu u.\, P} \triangleq u\, ® \mathbf{F}_P \qquad \mathbf{F}_{a^{\mathsf{ch}}(n).P} \triangleq \mathsf{in}^{\mathsf{ch}}(a, n).\mathbf{F}_P$$

$$\mathbf{F}_{\bar{a}^{\mathsf{ch}}\langle n\rangle.P} \triangleq \mathsf{out}^{\mathsf{ch}}(a, n).\mathbf{F}_P \qquad \mathbf{F}_{a(x).P} \triangleq \mathsf{in}(a, x).\mathbf{F}_P \qquad \mathbf{F}_{\bar{a}\langle M\rangle.P} \triangleq \mathsf{out}(a, M).\mathbf{F}_P$$

$$\mathbf{F}_{\mathsf{if}\ M=N\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2} \triangleq \mathsf{if}(M = N, \mathbf{F}_{P_1}, \mathbf{F}_{P_2})$$

5 Élimination des quantificateurs

On va montrer ici qu'on peut quantifier sur l'ensemble des termes, et ce indépendamment de la signature et la théorie équationnelle utilisées pour définir ces derniers, sans changer l'expressivité de la logique. Plus précisément, considérons la logique $\mathcal{L}^{\exists t}_{\mathsf{int}}$ qui étend la grammaire du fragment

intensionnel $\mathcal{L}_{\text{int}}$ par la construction $\exists t.\, A$, où t est une *variable de terme*. Celle dernière ne peut alors être utilisée que dans des jugements d'égalité de termes. La sémantique formelle de cet opérateur est la suivante :

$P \vDash \exists t.\, A$ *si et seulement si* il existe un terme M tel que $P \vDash A[t \leftarrow M]$

On peut par exemple écrire la formule $\exists t_1, t_2.\, \mathsf{Subst}(x = t_1)[\bar{a}\langle t_2 \rangle]$ qui caractérise les processus de la forme $\{^{M}/_{x}\} \mid \bar{a}\langle N \rangle$, où M et N sont deux termes. Le résultat que nous présentons dans cette partie est le suivant :

Théorème 5.1 *Soit A une formule de $\mathcal{L}_{int}^{\exists t}$. Il existe une formule $[\![A]\!]$ de $\mathcal{L}_{int}$ telle que, pour tout processus P,*

$$P \vDash A \text{ ssi } P \vDash [\![A]\!]$$

5.1 Intuitions

L'idée de la preuve est de remplacer les quantifications sur les termes de la forme $\exists t.\, A$ par des formules de la forme $\mathit{M}x.\, \mathsf{Subst}(x)[\top] \mathrel{-\!\!\!\overline{*}} [\![A]\!]$, où on a remplacé les occurrences de t par x dans la formule A, afin de laisser à l'opérateur $\mathrel{-\!\!\!\overline{*}}$ le choix du terme à substituer à x. Ce faisant, on augmente le cadre du processus d'une nouvelle substitution active pour chaque quantification rencontrée, et les processus considérés sont alors constitués d'une part du processus de départ P, et d'autre part d'une *valuation* V. Celle-ci est constituée des substitutions actives rajoutées, et doit être prise en compte dans la suite de la formule. Par exemple, la formule $\exists t.\, \emptyset$ ne sera pas convertie en $\mathit{M}x.\, \mathsf{Subst}(x)[\top] \mathrel{-\!\!\!\overline{*}} \emptyset$ mais en $\mathit{M}x.\, \mathsf{Subst}(x)[\top] \mathrel{-\!\!\!\overline{*}} \mathsf{Subst}(x)$. On a donc besoin, lors de l'exploration par induction de la formule de départ, de garder en mémoire les substitutions actives qui ont été rajoutées lors de la traversée d'existentielles. On note ainsi $[\![A]\!]_{\{x_1 \leftarrow t_1, \ldots\ldots, x_n \leftarrow t_n\}}$ pour signifier qu'on a du encadrer le processus de départ par $\{^{M_1 \ldots M_n}/_{x_1 \ldots x_n}\}$ (pour certains termes M_i non connus à l'avance) lors de la traversée des quantifications $\exists t_1, \ldots, \exists t_n$. On nommera également « valuation » l'ensemble $v = \{x_1 \leftarrow t_1, \ldots\ldots, x_n \leftarrow t_n\}$.

De plus, pour que la valuation ainsi construite puisse avoir l'effet souhaité, il faut ensuite remplacer chaque occurrence de variable de terme par la variable qui lui correspond dans la valuation. Les seuls endroits où ces variables peuvent apparaître étant les tests d'égalités entre termes, la formule correspondant à $M = N$ sera donc de la forme :

$$[\![M = N]\!]_v \triangleq (M = N)[t_1 \leftarrow x_1] \cdots [t_n \leftarrow x_n]$$

On aura également besoin de pouvoir séparer ce qui concerne le processus étudié de la valuation rajoutée lors de la transformation de la formule,

et donc de caractériser la présence d'une valuation au sein d'un processus. Pour $v = \{x_1 \leftarrow t_1, \ldots ..., x_n \leftarrow t_n\}$, on définit :

$$\Phi_\emptyset \triangleq \emptyset \wedge \mathbf{0} \qquad \Phi_v \triangleq \mathsf{Subst}(x_1, \ldots, x_n) \qquad \mathsf{ProcVal}(v) \triangleq (\Phi_v * \top)[\top]$$

Les deux premières formules caractérisent exactement la valuation (les variables $x_1, \ldots, x_n$ ayant été choisies fraîches, il ne peut pas y avoir de confusion avec le cadre du processus original), et la troisième un processus sous l'influence d'une valuation $v' \supseteq v$. Pour les besoins de la preuve par induction, on imposera systématiquement aux processus de vérifier $\mathsf{ProcVal}(v)$. On posera donc par exemple :

$$[\![M = N]\!]_v \triangleq \mathsf{ProcVal}(v) \wedge (M = N)[t_1 \leftarrow x_1] \cdots [t_n \leftarrow x_n]$$

Énonçons maintenant la propriété exacte d'induction que nous allons montrer, qui tient compte de la propagation de la valuation.

Propriété 5.2 *Pour toute valuation $v = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ et toute formule A de $\mathcal{L}_{int}^{\exists t}$, il existe une formule $[\![A]\!]_v$ de $\mathcal{L}_{int}$ telle que, pour tout processus P,*

$$P \vDash [\![A]\!]_v \textit{ ssi } \exists Q, V.\, P \equiv Q \mid V,\ Q \vDash \exists t_1, \ldots, t_n.\, A,\ V \vDash \Phi_v \textit{ et } fv(Q) \cap \{\tilde{x}\} = \emptyset$$

De plus, si $V = \{^{M_1 \ldots M_n}/_{x_1 \ldots x_n}\}$, les termes témoins des quantifications dans le jugement $Q \vDash \exists t_1, \ldots, t_n.\, A$ sont $M_1, \ldots, M_n$ et, pour tout i de $[\![1, n]\!]$, on a $fv(M_i) \cap \{\tilde{x}\} = \emptyset$.

Le théorème 5.1 suit alors en prenant $v = \emptyset$. La condition $fv(Q) \cap \{\tilde{x}\} = \emptyset$ est nécessaire pour éviter les phénomènes de capture lors de l'induction.

La propriété exacte d'induction est donc relativement complexe, et on va seulement donner ici les traductions des différents opérateurs sur lesquelles la preuve repose. La principale difficulté réside dans les opérateurs qui modifient le cadre du processus, à savoir $*$ et $-\!\!*$, puisqu'il faut alors, dans le premier cas, recopier le processus de valuation de part et d'autre de $*$, et dans le deuxième cas exposer le cadre rajouté (à gauche de $-\!\!*$) à la même valuation que celle du processus. Pour tous les opérateurs restants, l'opération $[\![\cdot]\!]_v$ est homomorphique : par exemple, $[\![A_1 \nmid A_2]\!]v$ est défini comme valant $[\![A_1]\!]_v \nmid [\![A_2]\!]_v$ puisque l'opération $\nmid$ s'effectue à *cadre constant*. Étudions maintenant de plus près la traduction des opérateurs $\exists$, $*$ et $-\!\!*$.

5.2 Les cas difficiles

La quantification sur les termes. La forme exacte de $[\![\exists t.\, A]\!]_v$ est la suivante :

$$[\![\exists t.\, A]\!]_v \triangleq \mathsf{ProcVal}(v) \wedge \mathit{И} x_{n+1}.\, (\mathsf{Subst}(x_{n+1}) \wedge \bigwedge_{i=1}^{n} \neg \copyright\, x_i)[\top] \mathrel{-\!\!\!\overrightarrow{*}} [\![A]\!]_{v \cup \{x_{n+1} \leftarrow t\}}$$

On retrouve la formule $\mathsf{ProcVal}(v)$, et l'ajout d'une substitution $\{^{M}/_{x_{n+1}}\}$ au cadre de valuation. On met alors à jour la valuation dans la suite de la traduction, qui devient $[\![A]\!]_{v \cup \{x_{n+1} \leftarrow t\}}$. On impose de plus à la substitution rajoutée (et donc au témoin M associé) de ne pas dépendre des autres variables de la valuation, et ce afin de garantir que M corresponde bien au témoin de la formule $\exists t.\, A$.

En effet, si ce n'était pas le cas, on pourrait par exemple traduire $P \vDash \exists t_1, t_2.\, t_1 = t_2$ par $\{^{M_1}/_{x_1}\} \mid \{^{x_1}/_{x_2}\} \mid P \vDash x_1 = x_2$, mais dire que le témoin de la quantification existentielle en t_2 est x_1 n'aurait guère de sens… On préfère donc dans ce cas la traduction en $\{^{M_1}/_{x_1}\} \mid \{^{M_1}/_{x_2}\} \mid P \vDash x_1 = x_2$, où les deux témoins sont M_1.

Les opérations sur le cadre. Lors de la séparation du cadre d'un processus en deux à l'aide de l'opérateur $*$, on doit prendre garde à recopier le cadre de valuation de manière à avoir une copie de chaque côté lors du découpage. Comme deux substitutions actives sur la même variable, fussent-elles égales, ne peuvent coexister, on crée un nouveau cadre V', dont on s'assure qu'il est l'exacte copie de V. Cela s'effectue de la manière suivante :

$$\begin{aligned}[\![A_1 * A_2]\!]_v \triangleq \mathsf{ProcVal}(v) \wedge \mathit{M} x'_1, \ldots, x'_n.\, (\Phi_{v'} \wedge \textstyle\bigwedge_{i=1}^{n} \neg \copyright\, x_i) \mathrel{-\!\!\!\overrightarrow{*}} \\ (*_{i=1}^{n} (\mathsf{Subst}(x_i, x'_i) \wedge x_i = x'_i) * \top)[\top] \wedge [\![A_1]\!]_v * [\![A_2]\!]_{v'}\end{aligned}$$

Le but est ici d'ajouter un processus de valuation $V' = \{^{M_1 \ldots M_n}/_{x'_1 \ldots x'_n}\}$ au cadre avant de procéder au découpage par $*$. On commence par allouer des variables fraîches $x'_1, \ldots, x'_n$ en nombre égal à la taille de la valuation v. On ajoute ensuite au cadre du processus déjà existant un processus V' dont on sait qu'il est un processus de valuation puisqu'il vérifie $\Phi_{v'}$ (où v' est une valuation sur les variables $x'_1, \ldots, x'_n$). De plus, les substitutions qui le composent ne sont pas autorisées à dépendre des variables de la valuation d'origine, et, une fois celles-ci ajoutées, on doit pouvoir trouver dans la cadre n paires de substitutions de la forme $\{^{M_i M'_i}/_{x_i x'_i}\}$ vérifiant chacune $x_i = x'_i$. Nous sommes donc assurés que, pour tout i, $M_i = M'_i$, donc que V' est bien de la forme voulue. On sépare alors le processus $P \mid V \mid V'$ en $P_1 \mid V \vDash [\![A_1]\!]_v$ et $P_2 \mid V' \vDash [\![A_2]\!]_{v'}$, où $P \equiv P_1 * P_2$ (et donc $P \mid V \mid V' \equiv (P_1 \mid V) * (P_2 \mid V')$).

Le cas de l'opérateur $-\!\!*$ se traite de manière similaire.

5.3 Robustesse

De la bonne application des substitutions. Ce résultat dépend fortement de la forme de la règle SUBST. En effet, si on permet aux substitutions de s'appliquer aux autres substitutions comme c'est le cas dans l'article d'introduction au pi-calcul [AF01], c'est-à-dire si la règle correspondante prend

la forme $\{^M/_x\} \mid P \equiv \{^M/_x\} \mid P[x \leftarrow M]$ (appelons cette nouvelle règle SUBST') au lieu de $\{^M/_x\} \mid P^s \equiv \{^M/_x\} \mid P^s[x \leftarrow M]$, la méthode énoncée ici est inopérante. Pour s'en convaincre, prenons comme exemple la formule $\varphi = \exists t.\, (y = t * y = t) \wedge (y = t * y \neq t)$, et un processus $P = \{^M/_y\}$. On a alors $P \not\models \varphi$. En effet, lorsqu'on découpe P en deux parties disjointes, on obtient d'un côté P et de l'autre le processus nul, et le seul terme N qui vérifie à la fois $P \vDash y = N$ et $\mathbf{0} \vDash y = N$ est y. Mais alors, l'autre partie de la formule n'est pas vérifiée, puisqu'on aura jamais $y \neq y$.

Il se trouve cependant que $P \vDash [\![\varphi]\!]_\emptyset$. En effet, après traduction de φ, on doit prouver d'une part (en choisissant y comme témoin de l'existentielle) $P \mid \{^y/_x\} \mid \{^y/_{x'}\} \vDash y = x * y = x'$ et d'autre part $P \mid \{^y/_x\} \mid \{^y/_{x'}\} \vDash y = x * y \neq x'$. Or les deux sont vrais, il suffit d'effectuer les découpages suivants :

$$\{^M/_y\} \mid \{^y/_x\} \mid \{^y/_{x'}\} \equiv (\{^M/_y\} \mid \{^y/_x\}) * \{^y/_{x'}\}$$

$$\begin{aligned} \text{et} \quad \{^M/_y\} \mid \{^y/_x\} \mid \{^y/_{x'}\} &\equiv \{^M/_y\} \mid \{^y/_x\} \mid \{^M/_{x'}\} \qquad \text{(par SUBST')} \\ &\equiv (\{^M/_y\} \mid \{^y/_x\}) * \{^M/_{x'}\} \end{aligned}$$

Le premier cas nous donne $P \mid \{^y/_x\} \mid \{^y/_{x'}\} \vDash y = x * y = x'$ et le second $P \mid \{^y/_x\} \mid \{^y/_{x'}\} \vDash y = x * y \neq x'$. Tout le problème vient ici du fait qu'on n'est pas parvenu à assurer la stricte égalité des deux copies du processus de valuation au moment du découpage du cadre, et qu'un choix supplémentaire indésirable s'est donc fait à ce niveau.

Choix des opérateurs séparants. Le résultat tient cependant si on remplace les deux opérateurs spatiaux $*$ et $\restriction$ par $|$. Il faut alors systématiquement recopier la valuation lors des découpages par $|$.

6 Expressivité de la logique

Intéressons-nous maintenant à un aspect plus pragmatique de l'expressivité de cette logique spatiale, en donnant des formules caractérisant diverses propriétés.

6.1 Équivalence statique

Dans leur article d'introduction au pi-calcul appliqué [AF01], Abadi et Fournet introduisent deux relations d'équivalence sur les processus : une « équivalence observationnelle » et une bisimulation étiquetée, dont ils démontrent qu'elles coïncident. La première nécessite d'observer le processus sous l'influence de n'importe quel contexte, alors que la seconde ne repose que sur l'équivalence statique et les réductions dans le système de transitions étiquetées que le processus est capable d'effectuer. Utiliser la

bisimulation étiquetée est donc bien plus avantageux pour prouver que deux processus sont observationnellement équivalents que la méthode directe, et l'étude de l'équivalence statique joue par conséquent un rôle crucial. Ainsi, Hans Hüttel and Michael D. Pedersen ont-ils développé une logique [HP07] dans laquelle on peut écrire, étant donné un cadre φ, une formule caractérisant la classe d'équivalence statique de φ (comme le travail qui a été présenté à la section 4 pour la congruence structurelle). La formule obtenue ne permet cependant pas d'affirmer, comme cela est requis par la définition de l'équivalence statique, que deux cadres ont le même domaine, et c'est donc là une condition qui doit être vérifiée au préalable.

Nous allons voir ici que notre logique spatiale permet elle aussi la définition d'une telle formule.

Soit donc φ un cadre, dont on souhaite caractériser la classe d'équivalence statique : on veut obtenir une formule $\mathbf{F}_{\varphi}^{\approx_s}$ telle que, pour tout cadre ψ, $\psi \vDash \mathbf{F}_{\varphi}^{\approx_s}$ *si et seulement si* $\psi \approx_s \varphi$, pour une version *faible* de l'équivalence statique qui ne prend pas en compte les domaines des cadres considérés. On s'autorise pour cela à utiliser dans les formules la quantification existentielle sur les termes $\exists t$ introduite à la section précédente, ainsi que la quantification universelle que l'on déduit de celle-ci, puisque l'on a vu que cela ne changeait pas l'expressivité de la logique. Rappelons que pour qu'un cadre ψ soit statiquement équivalent à un autre cadre φ, il faut d'une part que φ et ψ vérifient les mêmes égalités de termes, et d'autre part que les deux cadres aient le même domaine.

Domaine. Nous laissons ici cette dernière condition de côté, puisqu'elle est étonnamment difficile à caractériser par une formule. Pour nous en convaincre, observons tout d'abord que, si h est une fonction sans équation et $s_1 s_2$ représente la concaténation de s_1 et s_2,

$$\nu s.\, \{^{h(s)}/_{x}\} \approx_s \nu s_1, s_2.\, \{^{h(s_1 s_2)}/_{x}\} \approx_s \cdots \approx_s \nu s_1 \ldots s_n.\, \{^{h(s_1 \ldots s_n)}/_{x}\} \approx_s \cdots$$

On peut ainsi, au sein d'une même classe d'équivalence statique, trouver des processus comportant un nombre arbitrairement grand de restrictions de nom.

De plus, définir précisément le domaine d'un processus nécessite d'observer soit directement la structure de celui-ci, ce qui demande de connaître cette structure à l'avance, notamment de savoir combien de noms sont restreints, soit les termes sur lesquels il agit, ce qui demande de placer ces derniers sous les restrictions de noms en provenance du cadre, et donc ici encore d'en révéler un nombre non connu à l'avance, soit enfin les substitutions actives que l'on peut y ajouter (car si on ne peut pas rajouter de substitutions actives sur une certaine variable, c'est que celle-ci fait déjà partie du domaine du cadre). Cette dernière méthode nécessite de trouver un processus qui ne peut être mis en parallèle avec celui d'origine (par exemple,

une substitution active sur une variable qui est dans le domaine de celui-ci), sans pouvoir caractériser précisément celui-ci. On doit donc quantifier, à l'aide de $-\!\!*$, sur une classe de processus (par exemple l'ensemble des substitutions actives) qui inclura des processus dont la composition avec le processus considéré est valide (c'est du moins le constat général auquel nous sommes ici parvenus en cherchant de telles formules).

On peut par contre facilement imposer au domaine d'un processus de contenir au moins une certaine variable x à l'aide de la formule $\mathrm{dom}^+(x)$ ci dessous, dont on déduit ensuite la formule $\mathrm{dom}^+(\varphi)$ qui caractérise les processus dont le domaine est au moins $\mathrm{dom}(\varphi)$ (que l'on considère ici comme valant $\{x_1, \ldots, x_n\}$).

$$\mathrm{dom}^+(x) \triangleq \mathsf{Subst}(x) -\!\!* \neg\top \qquad \mathrm{dom}^+(\varphi) \triangleq \mathrm{dom}^+(x_1) \wedge \ldots \wedge \mathrm{dom}^+(x_n)$$

Équivalence statique faible. On peut par contre caractériser une version *faible* de l'équivalence statique, où l'on tient seulement compte des égalités de termes vérifiées, et non des domaines des cadres. Pour ce faire, on teste pour chaque couple de termes (M, N) l'égalité $M = N$, puis on remplace le cadre d'origine par le cadre de référence φ à l'aide des opérateurs $*$ et $-\!\!*$ avant de tester à nouveau l'égalité $M = N$. Tout ceci s'exprime par la formule suivante, où $\mathbf{F}_\varphi$ est la formule caractéristique de φ telle que définie à la section 4.4 :

$$\forall t_1, t_2.\, t_1 = t_2 \Leftrightarrow (\top[\mathbf{F}_\varphi \mathrel{-\!\!\!\rightarrow\!\!*} t_1 = t_2])$$

On peut généraliser cette formule à tous les processus étendus (et non seulement aux cadres), simplement en précisant qu'elle ne s'applique qu'au cadre des-dits processus, et on obtient la formule caractéristique de l'équivalence statique faible à φ suivante :

$$\mathbf{F}_\varphi^{\approx_s} \triangleq (\forall t_1, t_2.\, t_1 = t_2 \Leftrightarrow (\top[\mathbf{F}_\varphi \mathrel{-\!\!\!\rightarrow\!\!*} t_1 = t_2]))[\top]$$

6.2 Propriétés des protocoles cryptographiques

Forts de cette caractérisation de l'équivalence statique faible, nous pouvons maintenant décrire certaines propriétés que l'on peut demander aux protocoles cryptographiques de vérifier à l'aide de la logique spatiale. On suppose pour cela que la première étape d'une telle analyse de protocole, à savoir sa modélisation par des processus du pi-calcul appliqué, a déjà été effectuée.

Secret fort. Véronique Cortier *et al.* ont par exemple montré [CRZ07] que la propriété dite de *secret fort* s'exprime très simplement en termes d'équivalence sur des processus. En cryptographie, l'on dit qu'un *secret* s est *fortement secret* pour un protocole donné si un attaquant ne peut distinguer

entre deux exécutions du protocole où seule la valeur de s a changé. En pi-calcul appliqué, l'on dira que s est fortement secret dans $\nu s.\, P$ si :

$$P[s \leftarrow M] \approx P[s \leftarrow N]$$

Résistance aux guessing attacks. En cryptographie, cette propriété signifie qu'un attaquant potentiel ne peut pas amasser suffisamment d'informations lors du déroulement d'un protocole pour pouvoir découvrir un secret « hors-ligne », c'est-à-dire sans avoir besoin de rejouer le protocole pour tester chaque valeur possible du secret. Un tel attaquant pourrait alors aisément effectuer par exemple une attaque par dictionnaire de manière locale, découvrir le secret, puis s'en servir pour jouer le vrai protocole. De même que le secret fort, la résistance aux guessing attacks s'énonce simplement en pi-calcul appliqué, cette fois comme une équivalence statique sur les cadres, comme l'ont montré Steve Kremer et Mark Ryan [KR05]. Ainsi, un cadre φ est résistant aux guessing attacks s'il vérifie

$$\nu p.\, \varphi \mid \{^{p}/_{x}\} \approx_s \nu p, p'.\, \varphi \mid \{^{p'}/_{x}\}$$

En effet, on ne peut alors pas distinguer entre le cadre où on a (intuitivement) donné le « bon » secret p et celui où on a donné le « mauvais » p'. Si on a représenté tout un protocole par un processus $\nu p.\, P$, la propriété recherchée devient :

$$\forall w.\, \forall P'.\, \nu p.\, P \xrightarrow{w} \nu p.\, P' \text{ implique } \nu p.\, \varphi(P) \mid \{^{p}/_{x}\} \approx_s \nu p, p'.\, \varphi(P') \mid \{^{p'}/_{x}\}$$

Ainsi, à aucun moment du déroulement du protocole l'attaquant n'a-t-il assez d'informations pour déterminer de lui-même si un secret est le bon ou non. On a noté ici $\xrightarrow{w}$ une série de transitions étiquetée par le mot w. Ces dernières, contrairement aux réductions internes, sont susceptibles de modifier le cadre du processus.

Ces deux propriétés utilisent l'équivalence statique comme moyen de comparaison, soit implicitement, comme pour le secret fort où on demande une bisimulation, laquelle se prouve en utilisant entre autre l'équivalence statique, soit directement comme dans le cas de la résistance aux guessing attacks. Un traitement logique est donc possible, et fait partie des pistes à explorer dans le futur. On remarque au passage que, dans les deux cas, les deux processus à comparer sont déjà assurés d'avoir les mêmes domaines ; il n'y a donc aucune différence entre l'équivalence statique recherchée et notre équivalence statique faible.

Protocoles sans reçu. Les propriétés des protocoles de votes dépendent elles d'équivalences plus complexes, du moins dans leur forme énoncée

par Steve Kremer et Mark Ryan [DKR06] pour le pi-calcul appliqué. Par exemple, pour caractériser les protocoles de votes dits « sans reçu », c'est-à-dire ceux pour lesquels les votant ne peuvent pas prouver leur vote à un tiers, on a besoin d'observer l'effet de modifications structurelles profondes sur le processus de vote, avant de vérifier une classique bisimulation. Cela ne sort donc pas encore complètement de notre cadre, puisqu'il suffirait pour caractériser la même propriété à l'aide de la logique d'appliquer ces modifications « à la main » avant de vérifier l'équivalence par la logique.

Une autre propriété intéressante à vérifier pour un protocole de vote est le fait d'être « résistant à la cœrcition ». Cela signifie qu'un votant ne peut pas prouver son vote à un tiers, comme précédemment, à la différence que le tiers en question peut maintenant forcer le votant à envoyer certains messages (autres que le vote lui-même). Dans la caractérisation obtenue par Steve Kremer et Mark Ryan, on a cette fois-ci besoin d'une toute autre équivalence : la « bisimulation adaptative », qu'il faudrait alors caractériser à l'aide de la logique sans pouvoir nous aider des résultats que nous avons obtenus ici.

7 Conclusion

7.1 Pour résumer

Nous avons donc défini une logique pour le pi-calcul appliqué, puis démontré que, comme dans le cas du pi-calcul, l'équivalence logique correspondait à la congruence structurelle (théorème 4.1), et que la logique éliminait les quantifications sur les termes (théorème 5.1), résultat spécifique au pi-calcul appliqué, mais qui utilise une technique qui rappelle celle utilisée dans le cadre de l'élimination des quantifications de noms du pi-calcul [CL04]. Enfin, nous avons montré que notre logique était suffisamment riche pour que l'on puisse y exprimer l'équivalence statique faible.

7.2 Pour aller plus loin

De la logique. Cette nouvelle logique mérite sans doute une étude plus approfondie, d'une part car on a montré ici qu'elle s'adaptait plutôt bien au pi-calcul appliqué et pouvait être suffisamment expressive, et d'autre part parce que les logiques spatiales en général pourraient en bénéficier.

Variations en $\Diamond$. Un premier point à étudier serait le remplacement de la modalité *forte* par une modalité *faible*. L'opérateur $\Diamond$ serait alors doté d'une nouvelle sémantique :

$$P \vDash \Diamond A \textit{ ssi } \exists Q.\, P \rightarrow^* Q \text{ et } Q \vDash A$$

Pour le pi-calcul par exemple, ce changement a une influence subtile (mais mineure) dans la caractérisation de l'équivalence logique [Loz04]. Il serait intéressant de mener une étude similaire dans le cas du pi-calcul appliqué, notamment car certaines des propriétés (par exemple cryptographiques) que l'on souhaiterait exprimer par un biais logique impliquent des séquences de réductions de taille indéterminée.

Fragments. Un autre sujet dont l'étude s'impose est celui de l'expressivité de divers fragments de la logique. Par exemple, peut-on conserver les résultats qu'on a obtenus ici si on remplace l'opérateur de révélation (à la fois pour les noms et pour les variables) par © ? En effet, ce dernier n'introduit a priori pas d'indécidabilité, contrairement à ®.

L'étude du fragment extensionnel reste également à faire, notamment au niveau de l'équivalence logique qu'il induit en présence des opérateurs $\triangleright$ et $-\!\!*$. Quelques formules ont déjà été étudiées dans le cadre de ce stage, en particulier pour les substitutions actives, que l'on peut caractériser précisément par la formule :

$$x = M \wedge (\mathsf{nil} \oslash x)$$

pour un certain terme M et une variable x, et la formule nil telle que définie à la section 2.2.

Décidabilité. Enfin, on pourrait étudier l'influence des différents opérateurs sur la décidabilité de la logique, et même si cela s'applique aux logiques spatiales en général, mener cette étude dans le cadre du pi-calcul appliqué fournirait sans doute quelques résultats intéressants, notamment du fait de la présence des substitutions actives et des opérateurs $*$ et $-\!\!*$ associés.

Du calcul. Le pi-calcul appliqué mériterait lui aussi une étude théorique plus approfondie : quelle est par exemple l'influence exacte des substitutions actives et de la façon dont elles agissent sur le calcul ? On pourrait imaginer de considérer leur application non plus comme une règle de congruence structurelle mais comme un pas de calcul, et de leur imposer un comportement plus proche de celui d'une mémoire partagée. Ce faisant, l'on s'éloignerait certainement des applications cryptographiques auxquelles ce calcul est dédié, tout en se rapprochant du traitement de la mémoire par les logiques de séparation et des récents travaux de David Pym et Chris Tofts sur ce thème [PT06][5].

[5] Bien qu'ils n'aient pas considéré le pi-calcul appliqué, mais une algèbre de processus avec ressources partagées. Le formalisme logique obtenu ne manque cependant pas de points commun avec le notre, notamment au niveau des opérateurs spatiaux.

7.3 Pour conclure

Nous avons essayé ici de définir une logique pour le pi-calcul appliqué qui serait à la fois proche des logiques spatiales déjà étudiées entre autre pour le pi-calcul, tout en prenant en compte les spécificités de ce nouveau calcul. Cela ne s'est pas fait sans beaucoup de tâtonnements, à la fois pour comprendre les possibilités du pi-calcul appliqué et pour trouver des sémantiques naturelles pour les opérateurs logiques. Nous sommes finalement parvenus à une logique qui permet de bien séparer les aspects statiques et dynamiques du calcul, ce qui montre sa bonne adaptation au pi-calcul appliqué, tout en restant très riche, comme l'ont montré les divers résultats d'expressivité qu'on a obtenu, et qui sont caractéristiques des logiques spatiales. L'étude n'en est cependant qu'à ses débuts, et de nombreuses questions restent ouvertes, notamment sur l'influence de certains choix que nous avons écartés dans un premier temps. Nous espérons ainsi avoir posé des questions intéressantes, et avoir répondu à certaines, sur les logiques spatiales aussi bien que sur le pi-calcul appliqué.

Références

[ABF04] Martín Abadi, Bruno Blanchet, and Cédric Fournet. Just Fast Keying in the Pi Calculus. In David Schmidt, editor, *Programming Languages and Systems : Proceedings of the 13th European Symposium on Programming (ESOP'04)*, volume 2986 of *Lecture Notes on Computer Science*, pages 340–354, Barcelona, Spain, March 2004. Springer Verlag.

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, 2001.

[BAF05] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005)*, pages 331–340, Chicago, IL, June 2005. IEEE Computer Society.

[Bef06] Emmanuel Beffara. A concurrent model for linear logic. In *Proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI)*, volume 155 of *Electronic Notes in Theoretical Computer Science*, pages 147–168. Elsevier, May 2006.

[Bla04] Bruno Blanchet. Automatic Proof of Strong Secrecy for Security Protocols. In *IEEE Symposium on Security and Privacy*, pages 86–100, Oakland, California, May 2004.

[CC03] Luís Caires and Luca Cardelli. A spatial logic for concurrency (part i). *Journal of Information and Computation 186(2)*, 186(2), 2003.

[CG98] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In *FoSSaCS '98 : Proceedings of the First International Conference on Foundations of Software Science and Computation Structure*, pages 140–155, London, UK, 1998. Springer-Verlag.

[CG04] G. Conforti and G. Ghelli. Decidability of freshness, undecidability of revelation. *Proc. of Foundations of Software Science and Computation Structures*, 2004.

[CL04] Luis Caires and Étienne Lozes. Elimination of quantifiers and undecidability in spatial logics for concurrency. In Philippa Gardner and Nobuko Yoshida, editors, *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR'04)*, volume 3170 of *Lecture Notes in Computer Science*, pages 240–257, London, UK, August 2004. Springer.

[CRZ07] Véronique Cortier, Michael Rusinowitch, and Eugen Zalinescu. Relating two standard notions of secrecy. *Logical Methods in Computer Science*, July 2007. Volume 3, Issue 3, Paper 2.

[DKR06] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 28–39, Venice, Italy, July 2006. IEEE Computer Society Press.

[GC00] Andrew Gordon and Luca Cardelli. Anytime, anywhere : Modal logics for mobile ambients. In ACM Press, editor, *Proceedings of the 27th ACM Symposium on Principles of Programming Languages*, pages 365–377, 2000.

[Hir02] Daniel Hirschkoff. An extensional spatial logic for mobile processes. In *Proceedings of CONCUR'02*, volume 3252 of *Lecture Notes on Computer Science*. Springer Verlag, 2002.

[HLS03] Daniel Hirschkoff, Étienne Lozes, and Davide Sangiorgi. Minimality results for spatial logics. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *Proceedings of the 23rd Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, volume 2914 of *Lecture Notes in Computer Science*, pages 252–264, Mumbai, India, December 2003. Springer.

[HM85] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1) :137–161, 1985.

[HP07] Hans Hüttel and Michael D. Pedersen. A logical characterisation of static equivalence. *Electron. Notes Theor. Comput. Sci.*, 173 :139–157, 2007.

[Jon83] Cliff B. Jones. Specification and design of (parallel) programs. In *Proceedings of IFIP'83*, pages 321 – 332, 1983.

[KR05] Steve Kremer and Mark D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In Mooly Sagiv, editor, *Programming Languages and Systems — Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, Edinburgh, U.K., April 2005. Springer.

[Loz04] Étienne Lozes. *Expressivité des logiques spatiales*. Thèse de doctorat, Laboratoire de l'Informatique du Parallélisme, ENS Lyon, France, November 2004.

[Mil82] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, i. *Inf. Comput.*, 100(1) :1–40, 1992.

[PT06] D. Pym and C. Tofts. A Calculus and logic of resources and processes. *Formal Aspects of Computing*, 18(4) :495–517, 2006.

[Rey02] John C. Reynolds. Separation logic : A logic for shared mutable data structures. In *Proceedings Seventeenth Annual IEEE Symposium on Logic in Computer Science*, pages 55–74, Los Alamitos, California, 2002. IEEE Computer Society.

A Où les titres commencent par « De »

On donne ici les preuves formelles de divers lemmes et propriétés utiles pour la définition et l'étude de la logique spatiale pour le pi-calcul appliqué.

A.1 De la stabilité des ensembles $\overline{fn}(P)$ et $\overline{fv}(P)$

Lemme A.1 *Pour tout u, pour tout M, si $u \in \overline{fnv}(M)$, alors pour tout $u' \notin fnv(M)$, $u' \in \overline{fnv}(M[u \leftarrow u'])$.*

DÉMONSTRATION : Cela découle directement des propriétés de stabilité de la théorie équationnelle par échange de noms. □

Lemme A.2 *Pour tout u, pour tout P, si $u \in \overline{fnv}(P)$, alors pour tout $u' \notin fnv(P)$, $u' \in \overline{fnv}(P[u \leftarrow u'])$.*

DÉMONSTRATION : Commençons par le cas d'un nom $n \in \overline{fn}(P)$, qu'on veut remplacer par un nom $n' \notin fn(P)$.

Si P contient une émission ou une réception (non restreinte) sur n, on trouve la même réception, cette fois-ci sur n', dans $P[n \leftarrow n']$. Celle-ci ne pouvant être modifiée par les règles de congruence structurelle, on a $n' \in \overline{fn}(P[n \leftarrow n'])$.

Sinon, si P contient une substitution active $\{^{M(n)}/_{x}\}$ (où $n \in \overline{fn}(M(n))$ et x et n ne sont pas restreints), alors en substituant n' à n, et par le lemme A.1, n' apparaîtra libre dans tous les termes $N = M(n')$, et comme la substitution active considérée apparaît dans tout $Q \equiv P[n \leftarrow n']$ sous la forme $\{^{N}/_{x}\}$ avec $N = M(n)$, on a bien $n' \in \overline{fn}(P[n \leftarrow n'])$.

S'il n'y a pas de telle substitution, on cherche un sous-terme $\bar{m}^{\mathsf{ch}}\langle M(n)\rangle$ avec $n \in \overline{fn}(M(n))$ et on obtient le même résultat.

Le dernier cas possible est celui d'une substitution *restreinte* $\{^{M(n)}/_{x}\}$, où $n \in \overline{fn}(M(n))$, et où x apparaît dans un terme sous la portée de la restriction. Si ce terme est lui-même l'image d'une substitution restreinte, cette substitution devra à son tour voir son domaine présent dans un autre terme sous sa portée, et ainsi de suite jusqu'à arriver dans un terme qui n'est pas dans l'image d'une substitution restreinte. En effet, on pourrait sinon, étant donné que les cas précédents ne sont pas vérifiés, trouver $Q \equiv P$ tel que $n \notin fn(Q)$. Le terme en question (ou d'autres termes qui lui sont égaux) apparaîtra alors dans tous les processus congruents à P, et donc on a $n' \in \overline{fn}(P[n \leftarrow n'])$.

Le cas d'une variable $x \in \overline{fv}(P)$ est analogue, à la différence que les cas d'émissions ou de réception sur n ci-dessus sont remplacés par le cas d'une substitution active sur x. □

A.2 De la bonne définition des opérateurs séparants

Donnons maintenant la preuve du lemme 3.7 de la page 3.7, qu'on découpe en deux sous-lemmes.

Lemme A.3 *Si $P \equiv P'$, $Q \equiv Q'$, et si $P \upharpoonright Q$ et $P' \upharpoonright Q'$ sont définis, alors $P \upharpoonright Q \equiv P' \upharpoonright Q'$.*

DÉMONSTRATION : Soient $P = \nu\tilde{n}.\,(\varphi[P_1]) \equiv \nu\tilde{m}.\,(\psi[P_2]) = P'$ et $Q = \nu\tilde{n}.\,(\varphi[Q_1]) \equiv \nu\tilde{m}.\,(\psi[Q_2]) = Q'$, $\tilde{n} = \tilde{n}_1\tilde{n}_2$ et $\tilde{m} = \tilde{m}_1\tilde{m}_2$ vérifiant les contraintes de l'énoncé. On va reproduire les transformations effectuées pour passer de P à P' et de Q à Q' sur $P \upharpoonright Q = \nu\tilde{n}.\,(\varphi[P_1 \mid Q_1])$ pour arriver à $P' \upharpoonright Q' = \nu\tilde{m}.\,(\psi[P_2 \mid Q_2])$.

On peut pour cela supposer sans perte de généralité que les premières règles appliquées lors de ces deux transformations sont les extrusions de noms, puis les α-conversions qui amènent $\nu\tilde{m}$ en tête du processus.

On applique ensuite les mêmes transformations qui transforment P en P' à $P \upharpoonright Q$ (par induction sur l'arbre de preuve de $P \equiv Q$), en adaptant les extrusions de noms de l'étape mentionnée ci-dessus pour les faire traverser également Q_1 (ce qui est possible puisque les noms concernés sont soit parmi $\{\tilde{n}_1\}$, qui est disjoint des variables libres de Q_1, soit parmi $\{\tilde{m}_1\}$, et on peut transformer Q_1 pour obtenir un processus dont l'ensemble des variables libres est disjoint de $\{\tilde{m}_1\}$, quitte à empiéter sur les étapes de la transformation de Q à Q'). On effectue ensuite les α-conversions qui font passer de Q à Q' (ce qui n'affecte pas P_2), et on obtient alors un processus de la forme $\nu\tilde{m}.\,(\psi[P_2 \mid Q_1'])$ où Q_1' est Q_1 où l'on a effectué les α-conversions et les extrusions de noms de $Q \equiv Q'$. Soit φ' le cadre φ qui a subit ces mêmes transformations. On a alors $\psi \equiv \varphi$, puis $\nu\tilde{m}.\,(\psi[P_2 \mid Q_1']) \equiv \nu\tilde{m}.\,(\varphi'[Q_1' \mid P_2])$.

On reprend alors les transformations menant de Q à Q', lesquelles ne concernent maintenant plus que $\varphi' \mid Q_1' \equiv \psi \mid Q_2$, et finalement $P \upharpoonright Q \equiv \nu\tilde{m}.\,(\psi[Q_2 \mid P_2]) \equiv P' \upharpoonright Q'$. □

Lemme A.4 *Si $P \equiv P'$, $Q \equiv Q'$, et si $P * Q$ et $P' * Q'$ sont définis, alors $P * Q \equiv P' * Q'$.*

DÉMONSTRATION : Soient $P_1 = \nu\tilde{n}.\,\varphi_1[P] \equiv \nu\tilde{m}.\,\psi_1[Q] = Q_1$ et $P_2 = \nu\tilde{n}.\,(\varphi_2[P]) \equiv \nu\tilde{m}.\,(\psi_2[Q]) = Q_2$ vérifiant les contraintes de l'énoncé. Par le lemme 3.1, on a $\nu\tilde{n}.\,\varphi_i \equiv \nu\tilde{m}.\,\psi_i$, $i \in \{1, 2\}$.

En appliquant les α-conversion et extrusions de noms nécessaires, on se ramène à $\nu\tilde{m}.\,\varphi_i'[P'] \equiv \nu\tilde{m}.\,\psi_i[Q]$, et $\varphi_i' \equiv \psi_i$.

On a aussi $P' \equiv Q$: si ce n'était pas le cas, ce serait à cause d'une interaction avec le cadre, par la règle d'application des substitutions, par exemple entre φ_1' et P', mais alors celle-ci ne pourrait pas se produire entre φ_2' et P' les domaines des deux cadres étant disjoints, et on ne pourrait pas avoir $P_2 \equiv Q_2$.

On a donc $P_1 * P_2 = \nu\tilde{n}.\,(\varphi_1 \mid \varphi_2[P]) \equiv \nu\tilde{m}.\,(\varphi'_1 \mid \varphi'_2[P']) \equiv \nu\tilde{m}.\,(\psi_1 \mid \varphi'_2[P']) \equiv \nu\tilde{m}.\,(\psi_1 \mid \psi_2[P']) \equiv \nu\tilde{m}.\,(\psi_1 \mid \psi_2[Q]) = Q_1 * Q_2$ □

B Où l'on refait les choses formellement

On a regroupé ici les caractérisations en termes de classes de processus des formules définies à la section 4.

B.1 Échauffement

Commençons par les formules de la figure 6 page 17.

Propriété B.1 *Pour tout processus P, $P \vDash \top$.*

Propriété B.2 *Pour tout processus P, pour toute variable ou nom u, $P \vDash ©\,u$* ssi *$u \in \overline{fnv}(P)$.*

DÉMONSTRATION : Supposons $P \vDash ©\,u$. Pour tout Q, $P \equiv \nu u.\,Q$ implique $Q \vDash \neg\top$. Supposons par l'absurde $\exists P' \equiv P.\,u \notin fnv(P')$. On a alors $P \equiv P' \equiv \nu u.\,P'$ ce qui contredit la formule énoncée plus haut.

Réciproquement, si $\forall P' \equiv P.\,u \in fnv(P')$, alors par définition, on ne peut pas écrire P sous la forme $\nu u.\,Q$. □

Propriété B.3 *Pour tout processus P, $P \vDash$ public* ssi *pour tout processus Q et tout nom n, $P \equiv \nu n.\,Q$ implique $n \notin \overline{fn}(Q)$.*

DÉMONSTRATION : Supposons $P \vDash \mathsf{public}$, et qu'il existe m et Q tels que $P \equiv \nu m.\,Q$ et $m \in \overline{fn}(Q)$. Soit alors $n' \notin fn(P) \cup \{n\}$. On a $P \equiv \nu m.\,Q \equiv \nu n'.\,Q[n{\leftarrow}n']$, et $n' \in \overline{fn}(Q[n{\leftarrow}n'])$ (par le lemme A.2), donc $P \vDash n'\,®\,©\,n'$, et $P \nvDash \mathsf{public}$.

Réciproquement, si $P \equiv \nu n.\,Q$ implique $n \notin \overline{fn}(Q)$, alors, quel que soit $n' \notin fn(P) \cup \{n\}$, $P \nvDash n'\,®\,©\,n'$, donc $P \vDash \mathsf{public}$. □

Propriété B.4 *Pour tout processus P, pour toutes formules A et B, $P \vDash A[B]$* ssi *$\varphi(P) \vDash A$ et il existe un processus $Q \equiv P$ tel que $(Q)^s \vDash B$.*

DÉMONSTRATION : Soit $P \vDash A[B]$. On a $P \equiv \nu\tilde{n}.\,\varphi[P_1 \mid P_2]$, avec $\nu\tilde{n}.\,\varphi[P_1] \vDash A \wedge \mathbf{0}$, donc en particulier $\nu\tilde{n}.\,P_1 \equiv 0$, donc $P_1 \equiv 0$. On a ainsi $\nu\tilde{n}.\,\varphi[\mathbf{0}] \vDash A$, soit $\varphi(P) \vDash A$.

De plus, $\nu\tilde{n}.\,\varphi[P_2] \vDash (B \wedge \emptyset) * \top$, et comme $P_1 \equiv \mathbf{0}$, $P \equiv \nu\tilde{n}.\,\varphi[P_2]$, donc $P \equiv \nu\tilde{m}.\,(\varphi_1 \mid \varphi_2)[P']$, avec $\nu\tilde{m}.\,\varphi_1[P'] \vDash B \wedge \emptyset$, donc en particulier $\nu\tilde{m}.\,\varphi_1 \equiv \mathbf{0}$, donc $\varphi_1 \equiv \mathbf{0}$, et $P \equiv \nu\tilde{m}.\,\varphi_2[P']$, et finalement $(\nu\tilde{m}.\,\varphi_2[P'])^s = \nu\tilde{m}.\,P' \vDash B$, donc on a bien prouvé $\exists Q \equiv P.\,(Q)^s \vDash B$.

Réciproquement, si $\varphi(P) \vDash A$ et $\exists Q \equiv P.\,(Q)^s \vDash B$, alors $P \equiv Q \equiv \nu\tilde{n}.\,(\varphi \mid \mathbf{0})[Q' \mid \mathbf{0}]$ avec $(Q)^s = \nu\tilde{n}.\,Q'$. On a alors $\varphi(P) \equiv \nu\tilde{n}.\,\varphi \vDash A \wedge \mathbf{0}$ (par les lemmes 3.1 et 3.9), et $\nu\tilde{n}.\,Q' \vDash B \wedge \emptyset$. Il est alors immédiat que $P \vDash (A \wedge \mathbf{0}) \mathbin{¦} ((B \wedge \emptyset) * \top)$. □

Propriété B.5 *Pour tout processus P, $P \vDash \mathbf{1}$* ssi *$(P)^s \not\equiv \mathbf{0}$ et $\forall Q_1, Q_2.\ (P)^s \equiv Q_1 \mid Q_2$ implique soit $Q_1 \equiv \mathbf{0}$, soit $Q_2 \equiv \mathbf{0}$.*

DÉMONSTRATION :

$$\begin{aligned} P \vDash \mathbf{1} &\iff (P)^s \not\equiv \mathbf{0} \text{ et } P \equiv \nu\tilde{n}.\,\varphi[Q_1 \mid Q_2] \Rightarrow \begin{array}{r} \nu\tilde{n}.\,\varphi[Q_1] \vDash \mathbf{0} \\ \text{ou } \nu\tilde{n}.\,\varphi[Q_2] \vDash \mathbf{0} \end{array} \\ &\iff (P)^s \not\equiv \mathbf{0} \text{ et } (P)^s \equiv \nu\tilde{n}.\,Q_1 \mid Q_2 \Rightarrow \begin{array}{r} \nu\tilde{n}.\,Q_1 \equiv \mathbf{0} \\ \text{ou } \nu\tilde{n}.\,Q_2 \equiv \mathbf{0} \end{array} \\ &\iff (P)^s \not\equiv \mathbf{0} \text{ et } (P)^s \equiv \nu\tilde{n}_1.\,Q_1 \mid \nu\tilde{n}_2.\,Q_2 \Rightarrow \begin{array}{r} \nu\tilde{n}_1.\,Q_1 \equiv \mathbf{0} \\ \text{ou } \nu\tilde{n}_2.\,Q_2 \equiv \mathbf{0} \end{array} \end{aligned}$$

□

Propriété B.6 *Pour tout processus P, $P \vDash \mathbb{I}$* ssi *$\varphi(P) \not\equiv \mathbf{0}$ et $\forall Q_1, Q_2.\ \varphi(P) \equiv Q_1 \mid Q_2$ implique soit $Q_1 \equiv \mathbf{0}$, soit $Q_2 \equiv \mathbf{0}$.*

DÉMONSTRATION : *idem.* □

B.2 Caractérisation logique des substitution actives

On se réfère ici aux formules de la figure 7 page 19.

Propriété B.7 $P \vDash \mathit{Substs}$ ssi $\exists x_1, \ldots, x_k, M_1, \ldots, M_k.\ P \equiv \{^{M_1}/_{x_1}\} \mid \cdots \mid \{^{M_k}/_{x_k}\}$.

DÉMONSTRATION : Il est évident que si $P \equiv \{^{M_1}/_{x_1}\} \mid \cdots \mid \{^{M_k}/_{x_k}\}$, alors $P \vDash \mathsf{Substs}$.

Réciproquement, si $P \vDash \mathsf{Substs}$, $P \equiv \nu\tilde{n}.\,\varphi[P']$, $P' \equiv \mathbf{0}$, donc $P \equiv \nu\tilde{m}.\,\nu\tilde{x}.\,\{^{M_1}/_{x_1}\} \mid \cdots \mid \{^{M_k}/_{x_k}\}$, avec $\{\tilde{x}\} \subseteq \{x_1, \ldots, x_k\}$. Comme $P \vDash \mathsf{public}$, $P \equiv \nu\tilde{x}.\,\{^{M_1}/_{x_1}\} \mid \cdots \mid \{^{M_k}/_{x_k}\}$, et enfin, comme $P \vDash \mathit{И}x.\,x\,®(\neg\emptyset \wedge ©\,x) * (\neg\emptyset \wedge ©\,x)$, chaque x de $\tilde{x}$ ne peut apparaître que dans la substitution active sur x correspondante, et on peut donc appliquer la règle ALIAS autant de fois qu'il y a de tels x pour obtenir $P \equiv \{^{M'_1}/_{x'_1}\} \mid \cdots \mid \{^{M'_k}/_{x'_k}\}$, où l'on retrouve seulement une partie des substitutions $\{^{M_i}/_{x_i}\}$ ci-dessus. □

Propriété B.8 $P \vDash \mathit{Subst}$ ssi $\exists x, M.\ P \equiv \{^{M}/_{x}\}$.

DÉMONSTRATION : Évident d'après la propriété précédente. □

Propriété B.9 $P \vDash \mathit{Subst}(x)$ ssi $\exists M.\ P \equiv \{^{M}/_{x}\}$.

DÉMONSTRATION : Commençons par remarquer que $Q \vDash \mathsf{Subst} \wedge x = n$ *ssi* $Q \equiv \{^{n}/_{x}\}$. En effet, $Q \equiv \{^{M}/_{y}\}$, et, pour tout nom a frais, $\{^{M}/_{y}\} \mid \bar{a}\langle x\rangle \equiv \{^{M}/_{y}\} \mid \bar{a}\langle n\rangle$. Comme de plus $\Sigma \nvdash x = n$ (car il est interdit d'identifier deux noms dans la théorie équationnelle), on a $y = x$ et $M = n$ ou $M = x$ et $y = n$. Ce dernier étant impossible, on a bien $Q \equiv \{^{n}/_{x}\}$.

La formule nous dit alors que ¬(pour tout m frais, pour tout Q tel que $Q \perp P$ et $Q[\mathbf{0}] \equiv \{^{m}/_{x}\}$, $P \mid Q \vDash \top$), soit il existe m frais tel que l'ensemble

des $Q \perp P$ tels que $Q \equiv \{^m/_x\}$ est vide, soit encore $\forall Q \perp P. x \in dom(Q)$, donc $x \in dom(P)$, et comme P ne représente qu'une seule substitution, on a bien $\exists M. P \equiv \{^M/_x\}$.

La réciproque est évidente. □

Propriété B.10 *Pour tout x, M avec $M \neq x$, $P \vDash \mathit{Subst}(x = M)$* ssi *$P \equiv \{^M/_x\}$.*

DÉMONSTRATION : Évident. □

B.3 Caractérisation logique des communications

On donne ici les caractérisations exactes des formules pour les communications, en commençant par celles de la figure 8 page 21 qui concernent les communications de termes, ainsi que les démonstrations correspondantes.

Propriété B.11 *Pour tout processus P et toute variable x, $P \vDash \textcircled{c}^s_= x$* ssi *$x \in \overline{fv}((P)^s)$, il n'existe pas de sur-terme* strict *$M(x)$ de x (c'est-à-dire de terme $M(x)$ tel que $x \in \overline{fv}(M(x))$ et $\mathcal{E} \vdash M(x) = x$) tel que $M(x)$ apparaisse dans $(P)^s$ et que, pour une variable z fraîche, $z \in \overline{fv}((P)^s[M(x) \leftarrow z])$.*

DÉMONSTRATION : Soit $P \vDash \textcircled{c}^s_= x$. En particulier, $P \vDash \top[\textcircled{c}\, x]$ donc on a bien $x \in fnv((P)^s)$. Soit maintenant une variable z fraîche. La suite de la formule nous indique qu'il n'existe pas de sur-terme strict $M(x)$ de x tel que la substitution $\{^{M(x)}/_z\}$ puisse se factoriser dans P, c'est-à-dire telle que $z \in \overline{fv}((P)^s[M(x) \leftarrow z])$.

Il est clair que la réciproque est également vraie. □

Propriété B.12 *Pour tout processus P, tout nom a et toute variable x, $P \vDash \mathit{out}(a, x)$* ssi *$P \equiv \bar{a}\langle x \rangle$.*

DÉMONSTRATION : Soit $P = \bar{a}\langle x \rangle$. On a bien $P \vDash \text{single} \wedge \textcircled{c}\, a \wedge \textcircled{c}^s_= x$. De plus, en posant $Q = a(y)$, on a bien $Q \vDash \text{single}$, et $P \mid Q = \bar{a}\langle M(x) \rangle \mid a(y) \rightarrow \mathbf{0}$, donc $P \mid Q \vDash \Diamond\mathbf{0}$, et donc $P \vDash \mathsf{out}(a, x)$.

Réciproquement, soit $P \vDash \mathsf{out}(a, x)$. Comme $P \vDash \text{single}$, P est soit une conditionnelle, soit une communication sur un canal suivie d'un autre processus. Étant capable de communiquer avec un processus Q possédant les mêmes caractéristiques ($P \vDash \text{single} \blacktriangleright \Diamond\mathbf{0}$ et les cadres des deux processus sont communs, donc vides), c'est forcément une communication sur un canal, à son tour nécessairement suivie de $\mathbf{0}$ puisque $P \mid Q \vDash \Diamond\mathbf{0}$. Finalement, comme P a au moins une variable libre ($P \vDash \textcircled{c}^s_= x$), P est forcément une émission de terme, donc Q est une réception, qui se fait sur a (car $Q \vDash \textcircled{c}\, a$), on a donc $P \equiv \bar{a}\langle M(x) \rangle$ et finalement, comme $P \vDash \textcircled{c}^s_= x$, $P \equiv \bar{a}\langle x \rangle$. □

Propriété B.13 *Pour tout processus P, tout nom a, toute variable x et toute formule A, $P \vDash in(a,x).A$ ssi $P \equiv a(x).Q$ où $Q \vDash A$.*

DÉMONSTRATION : Soit $P \equiv a(x).Q$ avec $Q \vDash A$, et soit $R \vDash \mathsf{out}(a,x)$ (x est alors fraîche pour P). P vérifie bien single $\wedge \neg\Diamond\top$, et d'après la propriété précédente, $R \equiv \bar{a}\langle x\rangle$, donc$P \mid R \to Q$. Ce dernier vérifiant A, P vérifie bien la formule demandée.

Réciproquement, soit $P \vDash \mathsf{in}(a,x).A$. Comme $\bar{a}\langle x\rangle \vDash \mathsf{out}(a,x)$, il existe Q tel que $P \mid \bar{a}\langle x\rangle \to Q$ et $Q \vDash A$. Or, cette réduction vient soit d'une réduction interne à P, soit d'une réduction interne à $\bar{a}\langle x\rangle$, soit d'une interaction de ces deux processus. Comme P et $\bar{a}\langle x\rangle$ ne peuvent se réduire seuls, c'est qu'il y a interaction, donc P est de la forme $a(y).P'$, et $P'[y{\leftarrow}x] \vDash A$. Finalement, $P \equiv a(x).Q$ et $Q \vDash A$. □

Propriété B.14 *Pour tout processus P, tout nom a, tout terme M et toute formule A, $P \vDash out(a,M).A$ ssi $P \equiv \bar{a}\langle M\rangle.Q$ où $Q \vDash A$.*

DÉMONSTRATION : Soit $P \equiv \bar{a}\langle M\rangle.Q$ avec $Q \vDash A$. En appliquant les règles ALIAS et SUBST, on obtient $P \equiv \nu x.\{^M/_x\}[\bar{a}\langle x\rangle.Q]$. De plus, pour un nom b frais pour Q, on a $a(y).\bar{b}\langle y\rangle \mid \bar{a}\langle x\rangle.Q \to \bar{b}\langle x\rangle \nmid Q$, et $\bar{b}\langle x\rangle \vDash \mathsf{out}(b,x)$ et $Q \vDash A \wedge \neg \copyright\, x$. P vérifie donc bien la formule demandée.

Soit maintenant $P \vDash \mathsf{out}(a,M).A$. D'après la formule, il existe P' tel que $P \equiv \nu x.\{^M/_x\}[P']$, et $P' \vDash \mathcal{U}b.\,\mathsf{in}(a,y).\mathsf{out}(b,y) \rhd \Diamond(\mathsf{out}(b,x) \nmid A \wedge \neg \copyright\, x)$. Soit dont b libre pour P', et $R = a(y).\bar{b}\langle y\rangle$. Les deux propriétés précédentes nous permettent d'affirmer que $R \vDash \mathsf{in}(a,y).\mathsf{out}(b,y)$. De plus, $P' \mid R$ se réduit vers un processus composé d'une part de $\bar{b}\langle x\rangle$, et d'autre part d'un processus vérifiant $A \wedge \neg \copyright\, x$. Comme b a été choisi libre pour P', cela signifie que P' et R ont interagi, donc que P' est de la forme $\bar{a}\langle N\rangle.Q$, où $Q \vDash A \wedge \neg \copyright\, x$. On déduit du fait que la réduction a donné $\bar{b}\langle x\rangle$ que $N = x$, et donc la substitution $\{^M/_x\}$ s'est factorisée dans P pour donner P'. Comme de plus Q n'a pas été affecté par cette factorisation (car x n'apparaît pas dans Q), c'est que $P \equiv \bar{a}\langle M\rangle.Q$, où $Q \vDash A$. □

Montrons maintenant les résultats correspondants pour les communications de noms (figure 9 page 22).

Propriété B.15 *Pour tout processus P et tous noms a et b, $P \vDash test^{ch}(a,b)$ ssi $P \equiv \bar{a}^{ch}\langle b\rangle$ ou $P \equiv \bar{b}^{ch}\langle a\rangle$.*

DÉMONSTRATION : Soit $P \vDash \mathsf{test}^{\mathsf{ch}}(a,b)$. Comme $P \vDash$ single $\blacktriangleright$ **0**, P est une communication suivie de **0**, et comme P a deux noms libres, P est nécessairement une communication sortante, de la forme $\bar{c}\langle M\rangle$ pour un certain c et un certain M, ou $\bar{a}^{\mathsf{ch}}\langle b\rangle$ ou $\bar{b}^{\mathsf{ch}}\langle a\rangle$. Si P était de la première forme, on aurait, pour une variable fraîche x, $\{^M/_x\} \mid P \equiv \{^M/_x\}[\bar{c}\langle x\rangle]$ qui vérifie $\top[\copyright\, x]$, ce que contredirait la formule.

La réciproque est évidente. □

Les deux dernières preuves sont similaires à celles qu'on a faites pour les communications de termes.

B.4 Caractérisation logique des conditionnelles

Terminons par les preuves correspondant aux formules de la figure 10 page 23. On utilise ici une formule intermédiaire supplémentaire :

$$\mathsf{if}(M = N) \triangleq \mathsf{if} \wedge \mathcal{U}x, y.\, \mathsf{Subst}(x = M, y = N)[\top] \mathrel{-\!\!*} \top[©^{s}_{=} x \wedge ©^{s}_{=} y \wedge \neg\, ©^{\mathsf{br}} x \wedge \neg\, ©^{\mathsf{br}} y]$$

Propriété B.16 *Pour tout processus P, $P \vDash \mathit{if}$ ssi il existe des termes M, N et des processus simples Q et R tels que $P \equiv \mathit{if}\ M = N\ \mathit{then}\ Q\ \mathit{else}\ R$.*

DÉMONSTRATION : On a $P \vDash \mathbf{1} \wedge \mathsf{public}$, et $P \equiv \nu x.\, \{^{M}/_{x}\}[\mathsf{if}\ x = N\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]$, et $(\{^{N}/_{x}\}[\mathsf{if}\ x = N\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2]) * \mathsf{if}\ x = N\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \equiv \{^{N}/_{x}\}[\mathsf{if}\ N = N\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2] \to \{^{N}/_{x}\}[P_1]$. On en déduit que $\mathsf{if}\ x = N\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \vDash \mathsf{Subst}(x)[\top] \mathrel{-\!\!\!\!\!*} \Diamond\top$, et finalement $P \vDash \mathsf{if}$.

Réciproquement, si $P \vDash \mathsf{if}$, alors, comme $P \vDash \mathbf{1} \wedge \mathsf{public}$, P est soit un processus préfixé par une communication, soit une conditionnelle. Comme de plus il existe x, M, N et P' tels que $P \equiv \nu x.\, \{^{M}/_{x}\} \mid P'$ et $\{^{N}/_{x}\} \mid P' \vDash \Diamond\top$, P ne peut être qu'une conditionnelle (une communication seule ne pourra en effet jamais se réduire). □

Propriété B.17 *Pour tout processus P et toute variable x, $P \vDash ©^{\mathsf{br}} x$ si P est une conditionnelle où x apparaît libre dans une des branches, c'est-à-dire si il existe des termes M et N et des processus Q et R tels que $P \equiv \mathit{if}\ M = N\ \mathit{then}\ Q\ \mathit{else}\ R$, et que $Q \vDash © x$ ou $R \vDash © x$.*

DÉMONSTRATION : Supposons $P \equiv \mathsf{if}\ M = N\ \mathsf{then}\ Q\ \mathsf{else}\ R$. D'après la propriété précédente, $P \vDash \mathsf{if}$. Soient maintenant deux variables z et z' fraîches. On a $\{^{M}/_{z}\} \mid \{^{N}/_{z'}\} \mid P \vDash \mathsf{Subst}(z, z')[\top]$, et $\{^{M}/_{z}\} \mid \{^{N}/_{z'}\} \mid P \equiv \{^{M}/_{z}\} \mid \{^{N}/_{z'}\} \mid \mathsf{if}\ z = z'\ \mathsf{then}\ Q\ \mathsf{else}\ R$.

Si $Q \vDash © x$, on a ensuite $\{^{z'}/_{z}\} \mid \{^{s}/_{z'}\} \mid \mathsf{if}\ z = z'\ \mathsf{then}\ Q\ \mathsf{else}\ R \vDash (\mathsf{Subst}(z, z') \wedge \neg\, © x)[\top]$ pour un nom frais s quelconque, et ce même processus est congruent à $\{^{z'}/_{z}\} \mid \{^{s}/_{z'}\} \mid \mathsf{if}\ z' = z'\ \mathsf{then}\ Q\ \mathsf{else}\ R$ puis $\mathsf{if}\ z' = z'\ \mathsf{then}\ Q\ \mathsf{else}\ R$ se réduit en Q qui vérifie $© x$. On en conclut que $\mathsf{if}\ z = z'\ \mathsf{then}\ Q\ \mathsf{else}\ R \vDash (\mathsf{Subst}(z, z') \wedge \neg\, © x)[\top] \mathrel{-\!\!*} \top[\Diamond\, © x]$, et finalement $P \vDash ©^{\mathsf{br}} x$.

Si au contraire $R \vDash © x$, on choisit comme substitutions en z et z' $\{^{s}/_{z}\}$ et $\{^{s'}/_{z'}\}$ pour deux noms frais s et s', et on a de même $\{^{s}/_{z}\} \mid \{^{s'}/_{z'}\} \mid \mathsf{if}\ z = z'\ \mathsf{then}\ Q\ \mathsf{else}\ R \vDash (\mathsf{Subst}(z, z') \wedge \neg\, © x)[\top]$, mais cette fois-ci le processus obtenu est congruent à $\{^{s}/_{z}\} \mid \{^{s'}/_{z'}\} \mid \mathsf{if}\ s = s'\ \mathsf{then}\ Q\ \mathsf{else}\ R$, qui se réduit en R lequel vérifie $© x$. On conclut alors comme précédemment.

Réciproquement, soit $P \vDash ©^{\mathsf{br}} x$. Grâce à la propriété précédente, on sait qu'il existe M, N, Q, R tels que $P \equiv \mathsf{if}\ M = N\ \mathsf{then}\ Q\ \mathsf{else}\ R$. De plus, il existe des variables fraîches z et z' et des termes M', N', M'', N'' avec

$x \notin \overline{fv}(M'', N'')$ tels que $\{^{M'}/_{z}\} \mid \{^{N'}/_{z'}\}[P] \equiv \{^{M'}/_{z}\} \mid \{^{N'}/_{z'}\}[P']$, et $\{^{M''}/_{z}\} \mid \{^{N''}/_{z'}\}[P'] \equiv \{^{M''}/_{z}\} \mid \{^{N''}/_{z'}\}[P'']$ où P'' se réduit en un processus où x est libre. Ici, P' est P où certaines occurrences de M' ou de N' ont été remplacées par z (ou respectivement z'), et P'' est P' où certaines occurrences de z ou de z' ont été remplacées par M'' (ou respectivement N''). Aucune de ces opérations n'introduisant d'occurrences libres de x, et x apparaissant libre dans le processus final, on avait forcément soit $Q \vDash \copyright\, x$, soit $R \vDash \copyright\, x$. □

Propriété B.18 *Pour tout processus P et tous termes M, N, $P \vDash \mathit{if}(M = N)$ ssi il existe des processus simples Q et R tels que $P \equiv \mathit{if}\ M = N\ \mathit{then}\ Q\ \mathit{else}\ R$.*

DÉMONSTRATION : Soit $P \vDash \mathsf{if}(M_1 = N_1)$. Comme $P \vDash \mathsf{if}$, il existe M, N, Q, R tel que $P \equiv \mathsf{if}\ M = N\ \mathsf{then}\ Q\ \mathsf{else}\ R$. On a de plus, pour x et y frais $\{^{M_1}/_{x}\} \mid \{^{N_1}/_{y}\} \mid P \equiv \{^{M_1}/_{x}\} \mid \{^{N_1}/_{y}\}[P']$, où $P' \vDash \copyright^{s}_{=} x \wedge \copyright^{s}_{=} y \wedge \neg\, \copyright^{\mathsf{br}} x \wedge \neg\, \copyright^{\mathsf{br}} y$. Notons $P' \equiv \mathsf{if}\ M' = N'\ \mathsf{then}\ Q'\ \mathsf{else}\ R'$ On sait alors que x et y n'apparaissent libres ni dans Q' ni dans R' (car $P' \vDash \neg\, \copyright^{\mathsf{br}} x \wedge \neg\, \copyright^{\mathsf{br}} y$), de quoi l'on déduit que $\{x, y\} \subseteq \overline{fv}(M', N')$. De plus, il n'existe pas de sur-terme strict $M(x) \neq x$ pouvant apparaître dans $M' = N'$, et idem pour y, donc nécessairement $M' = x$ et $N' = y$ ou $M' = y$ et $N' = x$. Ainsi, $M = M_1$ et $N = N_1$, ou $M = N_1$ et $N = M_1$, et donc $P \equiv \mathsf{if}\ M = N\ \mathsf{then}\ Q\ \mathsf{else}\ R$.

Réciproquement, si $P \equiv \mathsf{if}\ M = N\ \mathsf{then}\ Q\ \mathsf{else}\ R$, alors pour x et y fraîches, $\{^{M}/_{x}\} \mid \{^{N}/_{y}\} \mid P \equiv \{^{M}/_{x}\} \mid \{^{N}/_{y}\}[\mathsf{if}\ x = y\ \mathsf{then}\ Q\ \mathsf{else}\ R]$ et $\mathsf{if}\ x = y\ \mathsf{then}\ Q\ \mathsf{else}\ R$ vérifie bien $\copyright^{s}_{=} x \wedge \copyright^{s}_{=} y \wedge \neg\, \copyright^{\mathsf{br}} x \wedge \neg\, \copyright^{\mathsf{br}} y$. □

Propriété B.19 *Pour tout processus P, tous termes M, N et toutes formules A et B, $P \vDash \mathit{if}(M = N, A, B)$ ssi il existe des processus simples Q et R tels que $P \equiv \mathit{if}\ M = N\ \mathit{then}\ Q\ \mathit{else}\ R$, $Q \vDash A$ et $R \vDash B$.*

DÉMONSTRATION : Soit $P \vDash \mathsf{if}(M = N, A, B)$. D'après la propriété précédente, et puisqu'il est évident que $\mathsf{if}(M = N, A, B)$ implique $\mathsf{if}(M = N)$, donc il existe Q, R tels que $P \equiv \mathsf{if}M = NQR$. En reprenant la démonstration précédente, on a de plus, en notant que $\{x, y, s, s'\} \cap \overline{fnv}(Q, R) = \emptyset$:

$$\mathsf{if}\ x = y\ \mathsf{then}\ Q\ \mathsf{else}\ R \vDash \mathsf{Subst}(x = y) \multimap \top[\Diamond A] \\ \wedge \forall s, s'.\, \mathsf{Subst}(x = s, y = s') \multimap \top[\Diamond B]$$

Ainsi, $\{^{y}/_{x}\} \mid \mathsf{if}\ x = y\ \mathsf{then}\ Q\ \mathsf{else}\ R$ est congruent à un processus dont la partie simple se réduit pour vérifier A, donc nécessairement, $Q \vDash A$, et de même, $R \vDash B$.

Réciproquement, pour $P \equiv \mathsf{if}\ M = N\ \mathsf{then}\ Q\ \mathsf{else}\ R$ avec $Q \vDash A$ et $R \vDash B$, on vérifie de manière directe que P vérifie bien $\mathsf{if}(M = N, A, B)$. □