Calculs de processus: observations et inspections

Daniel Hirschkoff

N°d'ordre : 013

SOUTENANCE
en vue d'obtenir

L'HABILITATION A DIRIGER DES RECHERCHES
délivrée par école normale supérieure de Lyon

présentée et soutenue publiquement le 08/12/09

par **Monsieur HIRSCHKOFF Daniel**

Titre :

CALCULS DE PROCESSUS : OBSERVATIONS ET INSPECTIONS.

Après avis de :

Monsieur ACETO Luca
Monsieur CURIEN Pierre-Louis
Madame DEZANI Mariangiola

Devant la Commission d'examen formée de :

Madame DEZANI Mariangiola
Monsieur BOUDOL Gérard
Monsieur CURIEN Pierre-Louis
Monsieur HENNESSY Matthew
Monsieur LESCANNE Pierre
Monsieur SANGIORGI Davide

Calculs de processus : observations et inspections

Daniel Hirschkoff

Mémoire d'habilitation à diriger les recherches

Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon

7 décembre 2009

Table des matières

Introduction

Sources de ce mémoire.

Ce mémoire présente une sélection des travaux menés au sein du LIP (ENS Lyon) depuis ma thèse. Tous ont trait à la théorie des systèmes concurrents, et, plus précisément, à l'étude des calculs de processus.

Pour organiser la présentation, certains résultats ont été laissés de côté – les articles correspondants sont mentionnés à l'annexe B. On se focalise ici surtout sur les études menées durant la thèse d'Étienne Lozes (2001-2004), et sur des travaux plus récents, en collaboration avec Damien Pous. L'objectif en faisant ce choix est de mettre en perspective ces deux axes de recherche, avec l'espoir que se dégage, ce faisant, un certain angle sur l'étude des calculs de processus. La description des résultats ne suit d'ailleurs pas l'ordre chronologique dans lequel ils ont été obtenus.

Il va sans dire les travaux qui ne sont pas abordés, s'ils sont en arrière-plan de ce document, ne sont pas moins importants pour moi. Ils s'agit principalement des travaux menés en collaboration avec David Teller durant sa thèse (2001-2004), qui a porté sur des systèmes de types pour le contrôle des ressources, et de l'étude de systèmes de types pour la terminaison dans un contexte concurrent, que nous menons avec Romain Demangeon dans le cadre de sa thèse (2006-présent), en cours. On peut d'ailleurs se dire qu'il aurait également été possible de 'raconter une histoire' décrivant un cheminement suivant ces deux lignes de travaux. N'oublions pas non plus un travail effectué avec Arnaud Carayol (stage de M2), qui a été l'occasion d'une collaboration très agréable.

Contenu : observation et inspection, l'oreille et la main.

Les travaux que l'on décrit ici s'inscrivent dans le domaine de la théorie des systèmes concurrents, et, plus précisément, de l'étude des calculs de processus. Le point de vue est opérationnel : on décrit des systèmes concurrents, au sens où ils peuvent interagir avec d'autres entités, en se donnant une syntaxe, et en munissant cette syntaxe d'une sémantique opérationnelle décrivant comment les interactions sont mises en œuvre.

L'étude qui est présentée porte, de manière très générale, sur la *connaissance* des processus : de quelle manière, et dans quelle mesure, on peut établir des propriétés portant sur un processus donné. Pour ce faire, on exploite et l'on met en relation des informations de deux sortes sur le processus, que l'on peut illustrer par le biais d'une analogie sensorielle. On se trouve dans une pièce obscure en présence d'une (ou plusieurs ?) créature mystérieuse, on discute avec elle et on la touche. À partir de ses réponses et de ce que l'on sent sous ses doigts, on se fabrique une idée de la créature.

S'adresser à la créature et écouter ce qu'elle a à dire, c'est communiquer, interagir. On peut bâtir des expériences permettant de détecter ce qu'elle est capable d'exprimer — on parlera d'observation, en référence notamment aux équivalences observationnelles usuellement étudiées en sémantique de la concurrence. Toucher, c'est acquérir des informations d'un autre type sur la créature, d'ordre anatomique : si l'on est face à une seule créature ou à un troupeau, si certains éléments peuvent être isolés, etc. On parlera alors d'inspection.

Ces deux formes d'acquisition de l'information s'entremêlent au cours du temps, au fur et à mesure que l'on construit la connaissance du processus que l'on a devant soi. On organise ensuite la connaissance que l'on acquiert en recoupant les informations, en classifiant. Une partie importante des résultats que l'on présente dans ce document porte sur l'étude des classes de processus que l'on fabrique ainsi : qu'obtient-on si l'on s'abstient de toucher ? peut-on faire le tour de la question en un seul test, ou bien a-t-on besoin de multiples expériences ? est-il possible d'écrire des équations qui caractérisent les classes que l'on a obtenues ?

Organisation : plan du mémoire.

La présentation des travaux est structurée en quatre parties, après cette partie introductive.

1. On décrit sommairement le cadre général, en introduisant les notions qui seront utiles pour développer le discours dans la suite : processus (CCS et le π-calcul), comportements, équivalences, logiques, axiomatisations.
2. On se focalise d'abord sur la notion d'équivalence comportementale, qui exploite des observations des processus. On présente deux caractérisations de la bisimilarité pour le π-calcul sans opérateur de choix. La première est établie à l'aide d'une logique modale, la seconde est équationnelle.

3. On s'intéresse ensuite à des observations plus fines, portant sur la structure des processus. Après avoir décrit comment on définit ces inspections, on s'intéresse aux équivalences que celles-ci induisent, en les comparant notamment aux équivalences comportementales.
4. Dans la partie 4, on présente des travaux qui s'inscrivent dans le prolongement de ce qui précède, mais qui portent sur des calculs de processus plus riches, dans la mesure où ceux-ci comportent une notion de localité explicite.

Comme il a été dit plus haut, l'ordre dans lequel les résultats sont exposés ne correspond pas toujours à la chronologie ; en particulier, une fraction importante de la quatrième partie décrit des travaux qui ont précédé les autres. Le style adopté pour la présentation n'est pas formel ; on fournira des pointeurs vers les articles correspondants, où sont donnés des développements rigoureux. On pourra aussi se reporter à [86] et [113] s'agissant de CCS et du π-calcul, respectivement.

1 Interactions, observations

1.1 Des processus aux comportements

Algèbres de processus canal historique : transitions étiquetées. L'approche traditionnelle pour la spécification des processus consiste à considérer qu'un système parallèle est décrit par son *système de transitions étiquetées* (LTS, pour *Labelled Transition System*). Il s'agit d'un graphe où les nœuds dénotent les états qui peuvent être atteints, et un arc correspond à une transition entre états. Les transitions sont des triplets, notés $P \xrightarrow{\mu} P'$, pour signifier que le système peut passer de l'état P à l'état P' en effectuant l'*action* μ. Lorsque l'on spécifie des processus pouvant interagir entre eux, μ représente une offre d'interaction : en CCS ou en π-calcul, une émission ou une réception sur un canal.

Lorsque $P \xrightarrow{\mu} P'$, un observateur peut susciter la transition menant de P à P' (tout du moins lorsque μ représente effectivement une interaction avec le contexte — on revient sur ce point plus bas). Dès lors, dans une situation où P offre une interaction μ que ne peut fournir Q, l'observateur a le pouvoir de distinguer ces deux processus, et il paraît alors naturel de considérer que P et Q n'expriment pas le même comportement.

Autour de ce scénario élémentaire, une grande variété d'équivalences comportementales peut être définie, chaque équivalence correspondant à un ensemble de conditions selon lesquelles deux processus définissent le même comportement. Les règles du jeu définissant ces équivalences sont fonction de ce que l'on juge comme observable, comment les processus jouent le jeu de l'équivalence, à quel moment chaque processus joue, etc. (cf. notamment les travaux de van Glabbeek [120, 121], qui font référence à des machines diverses, munies de voyants et autres boutons, pour se faire l'écho du bestiaire d'équivalences existant dans la littérature).

Nous nous intéressons ici principalement à la *bisimilarité*, qui est, parmi les équivalences comportementales généralement étudiées, la plus discriminante, et qui bénéficie de bonnes propriétés mathématiques.

Définition 1 (Bisimulation, bisimilarité — version étiquetée) *Une relation $\mathcal{R}$ entre processus est une bisimulation ssi dès que $P\mathcal{R}Q$ et $P \xrightarrow{\mu} P'$, il existe Q' tel que $Q \xrightarrow{\mu} Q'$ et $P'\mathcal{R}Q'$, et symétriquement pour les transitions de Q :*

$$\begin{array}{ccc} P & \mathcal{R} & Q \\ \mu\downarrow & & \downarrow\mu \\ P' & \mathcal{R} & Q' \end{array}$$

La relation de bisimilarité, notée $\sim$, est la plus grande bisimulation.

La bisimilarité est en particulier adoptée dans l'ouvrage de référence sur CCS [86] pour décrire, spécifier et analyser les systèmes concurrents. Suivant que les observations $P \xrightarrow{\mu} P'$ se font modulo les transitions internes (qui ne font pas intervenir le contexte) ou pas, on définit de la sorte les versions faible et forte de la bisimilarité. [111] présente une mise en perspective historique de la bisimilarité et de la coinduction, ainsi que de leurs utilisations en informatique.

Définir un nouveau calcul de processus, afin par exemple de modéliser des éléments issus d'un domaine d'application, c'est donner une syntaxe et la sémantique opérationnelle associée [40]. Ce que dénote un processus donné découle le plus souvent aussi d'une définition opérationnelle : on associe à un processus sa classe pour l'équivalence comportementale associée. La bisimilarité permet donc de déduire de manière systématique une notion de comportement à partir d'une définition de LTS.

Logiques modales et observations partielles. L'équivalence comportementale est définie par une règle du jeu, qui permet de distinguer deux processus (s'il existe une partie qui se termine sur une offre à laquelle on ne peut répondre) ou de les identifier (si toutes les parties mènent à l'impossibilité de distinguer).

Plutôt que de comparer un processus que l'on veut analyser à un autre processus, qui joue ainsi, en quelque sorte, le rôle de spécification du premier, on peut choisir de faire passer des tests au processus que l'on examine. C'est l'approche des logiques modales, et en particulier de la logique de Hennessy-Milner [59], que nous évoquons ici. Les tests sont décrits par des *formules* de la logique, et on note $P \vDash \mathcal{A}$ pour dire que le processus P satisfait la formule $\mathcal{A}$ (ou passe le test $\mathcal{A}$).

Les formules de la logique de Hennessy-Milner sont données par la grammaire suivante :

$$\mathcal{A} \quad ::= \quad \top \mid \neg\mathcal{A} \mid \mathcal{A}_1 \wedge \mathcal{A}_2 \mid \langle\mu\rangle\mathcal{A} \ .$$

On a toujours $P \vDash \top$; $P \vDash \neg\mathcal{A}$ est vérifié si $P \vDash \mathcal{A}$ ne l'est pas, P satisfait $\mathcal{A}_1 \wedge \mathcal{A}_2$ s'il satisfait $\mathcal{A}_1$ et $\mathcal{A}_2$, et on a

$$P \vDash \langle\mu\rangle\mathcal{A} \quad \text{si et seulement si} \quad P \xrightarrow{\mu} \vDash \mathcal{A} \ ,$$

autrement dit, s'il existe P' tel que $P \xrightarrow{\mu} P'$ et $P' \vDash \mathcal{A}$. On pose $[\mu]\mathcal{A} \stackrel{\text{def}}{=} \neg\langle\mu\rangle\neg\mathcal{A}$: on a $P \vDash [\mu]\mathcal{A}$ si et seulement si pour toute transition de P suivant μ, le processus résultant de la transition satisfait $\mathcal{A}$.

On remplace de la sorte des observations globales ($P \sim Q$ signifie que P exprime *exactement* le même comportement que Q) par des observations partielles ($P \vDash \mathcal{A}$ ne garantit que l'*une* des propriétés de P, celle qu'exprime $\mathcal{A}$). Par exemple, on sait d'un processus satisfaisant la formule $\langle a\rangle\top \wedge [b]\bot$ (où $\bot \stackrel{\text{def}}{=} \neg\top$) qu'il peut faire une transition suivant a et ne peut en faire suivant b.

Le résultat fondamental sur la logique de Hennessy-Milner dit qu'en rassemblant tous les tests exprimés par la logique, on atteint le même pouvoir discriminant que l'équivalence comportementale :

Théorème 2 *On note $P =_{\mathrm{HM}} Q$ si pour tout $\mathcal{A}$, $P \vDash \mathcal{A}$ ssi $Q \vDash \mathcal{A}$.*
Alors, si la relation $\xrightarrow{\mu}$ est à branchement fini pour tout μ, on a $P =_{\mathrm{HM}} Q$ si et seulement si $P \sim Q$.

Écrire les processus – Axiomatisations. Dans ce qui a été présenté jusqu'ici, on se fonde sur un LTS pour la spécification d'un système : le LTS détermine la définition à la fois de l'équivalence comportementale ($\sim$), et des tests de la logique de Hennessy-Milner.

L'équivalence comportementale peut être mise en équations. Il faut pour cela que les processus (les états du LTS) soient décrits par une syntaxe. On s'intéresse alors à déterminer quelles lois entre processus sont valides pour l'équivalence comportementale, et, au-delà, si cette dernière peut être caractérisée par un ensemble d'équations.

Le *Calculus of Communicating Systems* (CCS, [86]) est un calcul de processus que l'on peut décrire de la manière suivante. On se donne un ensemble infini de *noms*, qui seront notés par des lettres minuscules.

La grammaire des processus, notés avec des lettres majuscules, est donnée par :

$$P \quad ::= \quad \mathbf{0} \mid P_1|P_2 \mid (\boldsymbol{\nu} a)\, P \mid P_1 + P_2 \mid a.P \mid \overline{a}.P \ .$$

Dans le LTS associé à cet ensemble de processus, chaque nom a détermine deux actions, la réception sur a, notée aussi a, et l'émission sur a, notée $\overline{a}$. Une action spéciale, appelée τ, est utilisée pour dénoter la synchronisation interne (le préfixe τ sera ajouté à la syntaxe, lorsque l'on écrira le théorème d'expansion). On utilise μ comme métavariable pour les actions.

Les règles de sémantique opérationnelle sont les suivantes (on omet les versions symétriques des règles faisant intervenir les opérateurs | et +) :

$$\frac{}{a.P \xrightarrow{a} P} \qquad \frac{}{\overline{a}.P \xrightarrow{\overline{a}} P} \qquad \frac{P \xrightarrow{\mu} P'}{P+Q \xrightarrow{\mu} P'} \qquad \frac{P \xrightarrow{\mu} P'}{P|Q \xrightarrow{\mu} P'|Q} \qquad \frac{P \xrightarrow{\mu} P'}{(\boldsymbol{\nu} a)\, P \xrightarrow{\mu} (\boldsymbol{\nu} a)\, P'}\ \mu \notin \{a, \overline{a}\}$$

$$\frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

On est alors en mesure de présenter une axiomatisation de la bisimilarité sur CCS, dont l'élément central est le *théorème d'expansion* (parfois appelé *loi* d'expansion[1]), qui est présenté sur la figure 1 (on utilise la notation $\sum$ pour une somme n-aire, la somme 0-aire étant égale par définition au processus $\mathbf{0}$).

Si $P = \sum_i \alpha_i.P_i$ et $Q = \sum_j \beta_j.Q_j$, alors

$$P|Q \quad = \quad \sum_i \alpha_i.(P_i|Q) \ + \ \sum_j \alpha_i.(P|Q_j) \ + \ \sum_{i,j.\ \alpha_i = \overline{\beta_j}} \tau.(P_i|Q_j)$$

FIG. 1 – Le théorème d'expansion pour CCS

Proposition 3 (Axiomatisation sur CCS**)** *Le théorème d'expansion, associé aux lois ci-dessous pour la somme et la restriction, fournit une axiomatisation complète de la bisimilarité forte sur* CCS.

$$P+Q = Q+P \qquad P+(Q+R) = (P+Q)+R \qquad P+\mathbf{0} = P \qquad P+P = P$$

$$(\boldsymbol{\nu} a)\,(P|Q) = P \,|\, (\boldsymbol{\nu} a)\, Q \ \textit{si}\ a \notin \mathrm{fn}(P) \qquad (\boldsymbol{\nu} a)(\boldsymbol{\nu} b)\, P = (\boldsymbol{\nu} b)(\boldsymbol{\nu} a)\, P \qquad (\boldsymbol{\nu} a)\, P = (\boldsymbol{\nu} b) P[b/a] \ \textit{si}\ b\ \textit{frais}$$

$$(\boldsymbol{\nu} a)\, \mu.P = \mathbf{0} \ \textit{si}\ \mu \in \{a, \overline{a}\} \qquad (\boldsymbol{\nu} a)\, \mu.P = \mu.(\boldsymbol{\nu} a)\, P \ \textit{si}\ \mu \notin \{a, \overline{a}\} \qquad (\boldsymbol{\nu} a)\,(P+Q) = (\boldsymbol{\nu} a)\, P + (\boldsymbol{\nu} a)\, Q$$

Le théorème d'expansion, lu de gauche à droite, permet d'éliminer l'opérateur de composition parallèle en réécrivant un processus comme une somme, autrement dit, en en explicitant le comportement. Ce faisant, on réduit la concurrence au non-déterminisme. Cette transformation induit une notion de forme normale des processus, qui est au cœur de la preuve de complétude de l'axiomatisation.

Une instance simple de la loi d'expansion est donnée par $a \,|\, b = a.b + b.a$, la devise inscrite au fronton de la sémantique à entrelacement : la composition parallèle agit intuitivement comme un ordonnanceur, qui ne permet de faire qu'une seule action à la fois. Dès lors que l'on remplace a par $\overline{b}$ (par exemple), le terme de gauche peut faire une synchronisation que ne permet pas le terme de droite, et cette égalité n'est plus valide. Nous revenons sur ce point à la section 2.2.

1.2 Sémantique réductionnelle et contextes

Ce qui a été présenté définit la *méthode* des algèbres de processus : on décrit un calcul et sa sémantique opérationnelle par un LTS, qui détermine une notion d'interaction. Ces définitions s'accompagnent généralement de la description de liens escomptés avec un domaine d'application/de spécification (pour CCS : des systèmes parallèles, où les entités se synchronisent ; pour le π-calcul : on introduit la mobilité, sous forme de liens dont la connectivité évolue au cours du calcul ; pour les Mobile Ambients et d'autres calculs avec des localités explicites : on représente la mobilité de code, qui s'appuie sur une structuration de l'espace).

[1] Il est plus approprié de parler de *théorème* d'expansion, dans la mesure où l'on décrit une famille d'égalités. On prendra néanmoins la liberté de parler de *loi* d'expansion, et on fera de même plus loin pour la loi de distribution.

Dans les années 90, une manière alternative de présenter la sémantique opérationnelle des calculs de processus s'est affirmée, fondée sur une présentation à deux étages [9]. L'idée est de s'appuyer sur une notion relâchée d'*état* du systeme (exprimée par la relation de congruence structurelle) pour définir une relation de *réduction* entre 'états relâchés'. Suivant cette approche, on abandonne le LTS et sa manière de décrire les constituants de l'interaction par le biais d'étiquettes 'visibles', au profit d'un système de réécriture de processus, défini par les règles de réduction (qui correspondent, via ce qu'on appelle généralement le *lemme d'harmonie*, aux étiquettes τ du LTS). La définition de la congruence structurelle, $\equiv$, et de la réduction, $\longrightarrow$, sont standard dans le cas de CCS et ne sont pas rappelées ici.

Pour définir une notion d'équivalence comportementale dans ce cadre, on s'appuie alors sur la notion de *barbelé* (on opte parfois pour la terminologie des *barbes*), qui donne lieu à l'introduction de la *congruence barbelée*.

Définition 4 (Congruence barbelée en CCS**)**
Le processus P exhibe le barbelé *a si $P \equiv (\boldsymbol{\nu}\widetilde{b})\,(a.P_1 \mid P_2)$ pour certains $\widetilde{b}, P_1, P_2$, avec $a \notin \widetilde{b}$. On note alors $P \downarrow a$. On définit similairement $P \downarrow \overline{a}$.*
On dit qu'une relation $\mathcal{R}$ entre processus

1. préserve les barbelés *si, dès que $P\mathcal{R}Q$, on a ($P \downarrow a$ ssi $Q \downarrow a$) et ($P \downarrow \overline{a}$ ssi $Q \downarrow \overline{a}$) pour tout a ;*
2. *est* close par réduction *ssi, dès que $P\mathcal{R}Q$ et $P \longrightarrow P'$, on a $Q \longrightarrow Q'$ et $P'\mathcal{R}Q'$ pour un certain Q', et symétriquement pour les réduits de Q.*
3. *est* close par contexte parallèle *ssi $P\mathcal{R}Q$ implique $P|T\ \mathcal{R}\ Q|T$ pour tout processus T.*

$\simeq$, la congruence barbelée, est la plus grande relation sur les processus CCS *satisfaisant les trois conditions ci-dessus.*

À noter que la troisième clause est généralement énoncée de manière plus générale, puisqu'on préfère opter pour une clôture *par tout contexte*.

En CCS, il y a coïncidence entre les présentations étiquetée et barbelée de l'équivalence :

Théorème 5 (Correspondance LTS - sémantique réductionnelle)
Pour tous processus P, Q de CCS *$P \sim Q$ si et seulement si $P \simeq Q$.*

Origines et contexte. Plusieurs observations permettent de mettre en perspective l'émergence de la sémantique barbelée pour les algèbres de processus, qui remonte à [93].

Parentés.
Les ingrédients sur lesquels se fonde la sémantique barbelée de [93] peuvent être trouvés dans des travaux antérieurs. La sémantique chimique développée dans [9] met en avant une relation de réduction (à l'image de la β-réduction en λ-calcul) pour la présentation de la sémantique opérationnelle. Les relations de réchauffement et de refroidissement jouent dans ce travail le rôle de la congruence structurelle. La bisimulation applicative, développée par Abramsky pour le λ-calcul dans [3], exploite une relation de réduction non étiquetée. Enfin, on peut faire le rapprochement entre les barbelés et les diverses notions d'observation élémentaire mises en avant pour définir les équivalences de test (voir à ce sujet les travaux de van Glabbeek [120] mentionnés plus haut).

Des modèles plus compliqués, des étiquettes plus compliquées.
Alors que la sémantique opérationnelle de CCS s'exprime naturellement à l'aide d'un LTS, le π-calcul a vu le jour dans [91, 92] avec *deux* LTS, appelés précoce et tardif, sans que l'on sache s'il y en a un 'qui est le bon' (les sémantiques précoce et tardive se distinguent essentiellement par le traitement de la substitution, en lien avec le passage de noms, qui est l'ingrédient en plus en π-calcul par rapport à CCS). La présentation barbelée de la bisimilarité, apparue plus tard, semble arbitrer en faveur de la sémantique précoce, avec laquelle elle coïncide (théorème 2.2.29 dans [113]). D'un autre côté, une troisième version de la sémantique,

dite ouverte, s'avère plus intéressante pour définir des méthodes pour la vérification de la bisimulation. La situation est donc moins claire pour le π-calcul que pour CCS, où un LTS fait référence.

L'introduction de la sémantique barbelée a été motivée par l'étude du π-calcul d'ordre supérieur [109], où les substitutions font intervenir le remplacement d'une variable par un processus. Dans ce formalisme, les étiquettes pour l'émission doivent a priori contenir des processus, ce qui rend la notion d'un LTS 'canonique' d'autant plus difficile à cerner.

Dans le même ordre d'idées, après le π-calcul et le π-calcul d'ordre supérieur, un certain nombre de calculs de processus offrant des primitives de haut niveau très expressives (typiquement, pour rendre compte de la distribution, de la mobilité de code) ont été proposés. Dans ces travaux, la présentation barbelée a pris le pas sur la version étiquetée ; une raison en est sans doute que dans ces formalismes riches, où il est question de localités, de localités imbriquées, de code actif en mouvement, il peut être difficile de formuler d'emblée des transitions étiquetées qui induisent une notion sensée d'équivalence entre processus.

L'exemple le plus frappant en la matière est donné par les Mobile Ambients de Cardelli et Gordon [34] : alors que les opérations élémentaires d'entrée, sortie, et ouverture d'un ambient se spécifient facilement à l'aide de règles de réduction (cf. partie 4.2), il a fallu attendre quelques années avant de trouver un système de transitions étiquetées qui aille de pair avec la sémantique réductionnelle [85]. Les étiquettes du LTS correspondant sont d'ailleurs loin d'être simples. De fait, contrairement au cas de CCS, il ne paraît pas raisonnable de voir le LTS pour les Mobile Ambients comme une *définition* opérationnelle du calcul ; il s'agit plutôt d'un outil technique en vue de la construction de preuves d'équivalences, selon une approche coinductive.

Vers la canonicité. La présentation réductionnelle des calculs de processus a été volontiers adoptée depuis son introduction, en particulier du fait de la facilité d'exposition qu'elle permet. Divers aspects liés à cette approche ont fait l'objet de questionnements, avec notamment l'objectif de dégager une méthode, que l'on souhaite aussi systématique que possible, pour la présentation et l'étude de nouveaux formalismes.

Un ensemble de travaux se focalise sur la manière dont on définit l'équivalence comportementale. En particulier, le choix des barbelés, observations statiques élémentaires (clause 1 de la définition 4), paraît a priori assez arbitraire. Dans [74], Honda et Yoshida mettent en avant la notion de *processus insensible*, qui, par sa généralité, a vocation à remplacer l'observation des barbelés. Dans le même ordre d'idées, Rathke, Sassone et Sobocínski décrivent dans [104] une méthode, fondée sur la biorthogonalité, où l'on *dérive* les barbelés à partir des règles de réduction du calcul. Par ailleurs, plus ponctuellement, Sangiorgi dans [109] (dans le cas fort), puis Amadio dans [6] (dans le cas faible) préconisent de réduire la notion de barbelé à l'observation de la convergence d'un processus.

Le traitement de la clôture par contextes a également été questionné. La définition originelle de [93] introduit l'*équivalence barbelée*, qui est la plus grande congruence incluse dans la plus grande relation qui satisfait les clauses 1 et 2 de la définition 4. Autrement dit, alors que dans $\simeq$ (qui vient de [74]), la clôture a lieu à tous les pas du jeu de bisimulation, on plonge ici les processus que l'on compare dans un même contexte avant de tenter de les séparer en jouant sur les barbelés et sur la réduction. [52] et [112] ont étudié en détail les différences entre ces définitions, dans le cas du π-calcul et du π-calcul asynchrone. Il se trouve que dans le cas des équivalences fortes sur CCS, qui nous préoccupe ici, ces deux notions coïncident.

Un autre axe d'étude a pour objectif de systématiser l'obtention de résultats tels que le théorème 5. L'objectif, formulé dans un cadre catégorique, est de déduire le LTS à partir de la relation de réduction sur les processus, de manière à ce que les deux équivalences comportementales induites coïncident. Intuitivement, les étiquettes du LTS ainsi construit correspondent à des fragments de contexte minimaux susceptibles de provoquer une transition. Initiée par P. Sewell [115], cette direction d'étude a été développée notamment par R. Milner, J. Leifer, P. Sobocínski, et V. Sassone [79, 114, 77]. Les bigraphes [88] s'inscrivent dans cette démarche, et ont vocation à fournir un formalisme générique pour la définition de calculs tels que CCS, le λ-calcul, le π-calcul, ou les Mobile Ambients. L'approche de [106] poursuit des objectifs similaires, suivant une méthode davantage axée sur la description opérationnelle des calculs.

En arrière-plan de ces études se trouve aussi la volonté de revisiter la théorie des calculs de processus, d'une part afin de mieux en comprendre les fondements, et, d'autre part, en réaction à une tendance qui s'est manifestée dans les années 90, où l'on a vu apparaître un foisonnement de calculs nouveaux, traitant de problématiques similaires, et pour lesquels la question de la légitimité était difficile ne serait-ce qu'à formuler.

Remarque 6 (Terminologie) *On a mentionné ci-dessus les deux approches pour clore par les contextes dans la définition de* $\simeq$*. La définition originelle [93] est en deux étapes : on requiert la clôture* après *avoir défini la plus grande relation qui préserve les barbelés et soit préservée par réduction. Dans la seconde version, appelée* dynamique *dans [104] et* close par réduction *dans [78], la clôture par contextes est incluse d'emblée dans la définition de l'équivalence comportementale, et peut être faite 'à chaque étape' (Sangiorgi et Walker utilisent 'open barbed bisimilarity' pour la version dynamique de la barbed bisimilarity dans [112]).*

La définition 4, telle qu'elle est formulée, entre en conflit avec certains usages. D'une part, on ne s'intéresse à la clôture que par composition parallèle, et non par tous les contextes : cela suffit pour CCS*, et il en sera de même pour* π_0*, le calcul étudié à la partie 2. Dans le cas général,* $\simeq$ *est définie en fermant par tous les contextes du langage étudié. D'autre part, 'barbed congruence' est utilisé dans [113] pour une définition en deux étapes, alors que la nôtre adopte le style dynamique. Ce problème se retrouve dans la littérature, dans la mesure où l'appellation barbed congruence est utilisée dans [52] ou [105] comme nous le faisons ici, alors que barbed equivalence est utilisé pour l'approche en deux étapes.*

2 Équivalences comportementales : caractérisation logique, axiomatisation et congruence

Cette partie porte sur le π-calcul fini, que l'on note π_0 :

Définition 7 (Calcul π_0) *π_0 est le fragment du π-calcul défini par la grammaire suivante :*

$$P \quad ::= \quad \mathbf{0} \mid P_1|P_2 \mid (\boldsymbol{\nu} n)\, P \mid m(n).P \mid \overline{m}\langle n\rangle.P \ .$$

On s'appuie sur la présentation réductionnelle de π_0, que l'on ne rappelle pas – [113], par exemple, peut servir de référence.

On a exposé à la partie 1.1 trois points de vue sur la bisimilarité : sa définition fondée sur la sémantique étiquetée, suivie par ses caractérisations logique et algébrique. Lorsque l'on remplace les transitions étiquetées par la présentation réductionnelle de la sémantique opérationnelle (partie 1.2), l'équivalence comportementale que l'on obtient est la congruence barbelée, qui offre une nouvelle caractérisation de $\sim$. C'est le théorème 5 vu plus haut, qui se décline ainsi pour π_0 ($\sim$ désigne la bisimilarité *précoce* – voir aussi la note de bas de page placée avant l'énoncé du théorème 10) :

Théorème 8 *Pour tous processus P, Q de π_0 $P \sim Q$ si et seulement si $P \simeq Q$.*

Dans la présentation réductionnelle, la caractérisation équationnelle de l'équivalence est la même, puisque celle-ci ne fait pas référence à la définition de la sémantique opérationnelle. La situation est moins claire en revanche pour ce qui est de la caractérisation logique (théorème 2), dans la mesure où la logique de Hennessy-Milner est définie à partir des transitions du LTS, et 'colle' à la sémantique étiquetée. La partie 2.1 expose ce qui peut être vu comme le pendant, du côté de la sémantique réductionnelle, du théorème 2.

Une autre question, celle de la propriété de congruence pour la bisimilarité en π-calcul, est traitée dans la partie 2.2. Les résultats que l'on y expose soulignent l'influence de la substitution sur les équivalences entre processus (du π-calcul et de CCS).

2.1 $\mathcal{L}_{\mathrm{E}}$, une logique extensionnelle [61]

La contribution de [61] se situe dans le domaine des logiques modales pour la concurrence, et plus précisément des logiques spatiales. Les processus de π_0 sont analysés par l'intermédiaire de la logique $\mathcal{L}_{\mathrm{E}}$, définie comme suit.

Définition 9 (Logique extensionnelle) *Les formules de la* logique extensionnelle, $\mathcal{L}_{\mathrm{E}}$, *sont définies par la grammaire suivante :*

$$\mathcal{A} \quad ::= \quad \top \mid \neg\mathcal{A} \mid \mathcal{A}_1 \wedge \mathcal{A}_2 \mid \Diamond\mathcal{A} \mid \mathcal{A}_1 \triangleright \mathcal{A}_2 \mid \mathcal{A} \oslash n \mid \mathit{M}n.\,\mathcal{A} \ .$$

On note $=_{\mathrm{E}}$ *l'équivalence logique induite par* $\mathcal{L}_{\mathrm{E}}$ *:* $P =_{\mathrm{E}} Q$ *si et seulement si pour toute formule* $\mathcal{A}$ *de* $\mathcal{L}_{\mathrm{E}}$, $P \vDash \mathcal{A}$ *ssi* $Q \vDash \mathcal{A}$.

La relation de satisfaction pour $\mathcal{L}_{\mathrm{E}}$ s'appuie sur la sémantique réductionnelle du π-calcul. En premier lieu, on a

$$P \vDash \Diamond\mathcal{A} \quad \text{ si et seulement si } \quad P \longrightarrow \vDash \mathcal{A}$$

(à noter que, pour ne pas surcharger la présentation, on réutilise ici l'opérateur $\vDash$, comme pour la satisfaction des formules de la logique de Hennessy-Milner).

On voit que $\Diamond\mathcal{A}$ joue le rôle du $\langle\tau\rangle\mathcal{A}$ de la logique de Hennessy-Milner (on abordera aussi plus loin l'interprétation *faible* de l'opérateur $\Diamond$, où l'on requiert $P \longrightarrow^* \vDash \mathcal{A}$, en notant $\longrightarrow^*$ la clôture réflexive et transitive de $\longrightarrow$). Les constructions $\triangleright$, $\oslash$ et M sont moins classiques ; elles ont été mises en avant dans les travaux sur les logiques spatiales, dont il sera question en détail plus loin. La relation de satisfaction est définie de la manière suivante pour ces connecteurs :

- $P \vDash \mathcal{A}_1 \triangleright \mathcal{A}_2$ si et seulement si pour tout Q tel que $Q \vDash \mathcal{A}_1$, on a $P|Q \vDash \mathcal{A}_2$;
- $P \vDash \mathcal{A} \oslash n$ si et seulement si $(\boldsymbol{\nu} n)\, P \vDash \mathcal{A}$;
- $P \vDash \mathit{M}n.\,\mathcal{A}$ si et seulement si pour tout nom m qui est *frais* à la fois pour P et pour $\mathcal{A}$, $P \vDash \mathcal{A}_{[m/n]}$.

Le rôle de $\triangleright$ et $\oslash$ est de plonger le processus que l'on examine dans un contexte, en appliquant respectivement la composition parallèle et la restriction. $\triangleright$ exprime une requête de type fonctionnel sur un processus : satisfaire $\mathcal{A}_2$ en présence d'un processus satisfaisant $\mathcal{A}_1$. $\oslash$ est l'opérateur de *dissimulation*, puisqu'il a pour effet de cacher un nom sous une restriction. L'idée est d'utiliser ces constructions pour mettre en place un 'contexte expérimental' dans lequel observer un processus. Le quantificateur frais, M, sert à éviter les conflits de noms lorsque l'on construit de la sorte une expérience (on dit que m est frais pour P et $\mathcal{A}$ si m n'apparaît ni dans P ni dans $\mathcal{A}$). Il est issu des travaux de Gabbay et Pitts [54] sur la liaison. À noter que la présence de M n'est pas strictement nécessaire pour obtenir les résultats de [61], mais simplifie grandement la présentation.

Nous revenons à la partie 3 sur ces opérateurs et sur leurs origines. On peut constater que les seules constructions syntaxiques auxquelles il est fait référence dans la définition de la relation de satisfaction pour $\mathcal{L}_{\mathrm{E}}$ sont $|$ et $\boldsymbol{\nu}$, pour les opérateurs $\triangleright$ et $\oslash$ respectivement. Un langage d'assertions comme $\mathcal{L}_{\mathrm{E}}$ peut par conséquent être utilisé comme point de départ pour raisonner sur une algèbre de processus plus ou moins arbitraire, du moment qu'elle fournit composition parallèle et restriction : un tel point de vue est adopté à la partie 3.2.1.

Par ailleurs, on peut remarquer que $\mathcal{L}_{\mathrm{E}}$ ne fait pas référence aux actions d'émission et de réception comme c'est le cas dans la logique de Hennessy-Milner. On retrouve en revanche dans $\mathcal{L}_{\mathrm{E}}$ (et dans les autres logiques spatiales qui sont examinées aux parties 3 et 4) certains éléments qui font écho à une définition comme celle de $\simeq$ (définition 4) :

(*i*) La clause 2, sur la clôture par réduction, a sa contrepartie dans $\mathcal{L}_{\mathrm{E}}$ avec l'opérateur $\Diamond$.

(*ii*) La clause 3, de clôture par contexte, évoque l'opérateur $\triangleright$: une spécification de la forme $\mathcal{A}_1 \triangleright \mathcal{A}_2$ peut être utilisée pour placer les deux processus que l'on compare en parallèle avec un processus testeur (satisfaisant $\mathcal{A}_1$).

En revanche, rien a priori ne permet de faire le lien entre la clause 1 (sur les barbelés) et $\mathcal{L}_{\mathrm{E}}$.

En lien avec le point (*ii*) ci-dessus, on peut revenir sur la comparaison entre les transitions étiquetées et la présentation réductionnelle de la sémantique. Les étiquettes décrivent les interactions entre le processus observé et son contexte, qui est en quelque sorte 'hors champ'. Au contraire, en version réductionnelle, la

troisième clause de la définition de $\simeq$ est utilisée pour *réifier* le contexte — le faire entrer en scène, et, ce faisant, susciter des réductions. Ainsi, le point de vue de $\simeq$ (et de $\mathcal{L}_{\mathrm{E}}$) décompose l'observation d'une transition en, d'une part, la possibilité d'une interaction (le barbelé), et, d'autre part, l'évolution du système en interne. Si le premier aspect n'est pas présent dans les constructeurs de $\mathcal{L}_{\mathrm{E}}$, qui n'autorise pas directement d'observation des potentialités d'interaction, cette articulation entre observations statiques et dynamiques est au cœur des logiques spatiales pour la concurrence ; elle les distingue des formalismes comme la logique de Hennessy-Milner.

Le principal résultat qui est établi dans [61] est le suivant[2] :

Théorème 10 (Extensionalité – [61, Théorèmes 3 et 4])
Pour tous P et Q de π_0, $P =_{\mathrm{E}} Q$ si et seulement si $P \sim Q$.

Plutôt que de décrire fidèlement la preuve telle qu'elle est faite dans [61], on présente ici une approche légèrement différente et plus intuitive, suggérée par P.-L. Curien. D'un point de vue technique, le cœur de la preuve repose sur des raisonnements qui sont exposés à la partie 3.2.2, et qui sont inchangés entre la présentation que l'on fait ci-dessous et la preuve de [61].

L'idée générale est de montrer que l'équivalence logique coïncide avec la congruence barbelée, qui à son tour, d'après le théorème 8, coïncide avec $\sim$, la bisimilarité précoce.

On commence par établir que deux processus équivalents pour $\simeq$ satisfont les mêmes formules. Ceci est une conséquence immédiate, pour les opérateurs $\Diamond$ et $\triangleright$, des clauses de fermeture par réduction et par contexte parallèle définissant $\simeq$. La clôture par restriction, qui correspond à l'opérateur $\oslash$, fait également partie de la définition standard de $\simeq$ (comme on l'a indiqué plus haut, elle est omise de la définition 4 pour simplifier l'exposé, et peut s'obtenir, dans les cas qui nous préoccupent ici, comme conséquence de la définition). On obtient ainsi

$$\simeq \subseteq =_{\mathrm{E}} \quad .$$

On souhaite ensuite établir l'inclusion inverse. L'enjeu de cette partie de la démonstration est d'utiliser les connecteurs logiques pour exprimer les observations qui définissent $\simeq$. Comme on l'a dit, la clause 2 de la définition 4, sur la clôture par réduction, a sa contrepartie dans $\mathcal{L}_{\mathrm{E}}$ avec l'opérateur $\Diamond$. Manquent en revanche les observables (clause 1), puisque $\mathcal{L}_{\mathrm{E}}$ n'a pas de construction qui permette de détecter statiquement une propriété des processus ayant trait à l'interaction.

Quant à la clause 3 de clôture par contexte, l'opérateur $\triangleright$ s'en rapproche : une spécification de la forme $\mathcal{A}_1 \triangleright \mathcal{A}_2$ peut être utilisée pour placer les deux processus que l'on compare dans un même contexte, donné par un processus placé en parallèle. Intuitivement, on exploite pour cela les opérateurs $|$ et $\boldsymbol{\nu}$, afin de spécifier des 'contextes d'expérience', suivant une stratégie qui est exploitée intensivement dans les parties 3 et 4.1 : on insère le processus dans un contexte avec lequel il est sensé interagir, on fait mijoter (via $\Diamond$), puis on observe le résultat.

Dans ses grandes lignes, cette partie de la démonstration est à rapprocher des preuves de caractérisation de $\sim$ via une équivalence barbelée, pour le sens $\simeq \subseteq \sim$: il s'agit de montrer que les contextes sont suffisamment discriminants pour 'provoquer' des transitions au sens du LTS. Cependant, une spécifité importante du cadre dans lequel on se place est que dans une formule de la forme $\mathcal{A}_1 \triangleright \mathcal{A}_2$, $\mathcal{A}_1$ ne donne a priori qu'une description *partielle* du processus testeur, alors que dans la définition 4 on a la possibilité de choisir *individuellement* le processus testeur. En ce sens, les possibilités d'observation des processus sur lesquelles se fonde $\mathcal{L}_{\mathrm{E}}$ sont une forme relâchée de ce que prescrit la définition 4, et l'enjeu est de montrer que la logique est suffisamment précise pour pouvoir décrire des scenarii de tests à même de jouer le rôle de la clause de clôture par contextes (on revient sur ce point ci-dessous).

Dans [61], on montre l'inclusion $=_{\mathrm{E}} \subseteq \simeq^p$, où $\simeq^p$ est définie en replaçant la clause 3 de la définition 4 par

$3'$. $P \mathcal{R} Q$ implique que pour tous T, T' tels que T est public et $T \sim T'$, $P|T \mathcal{R} Q|T'$.

[2] Dans l'énoncé de ce théorème, $\sim$ désigne la bisimilarité dans sa version *précoce* pour π_0. Mais le théorème 11 ci-dessous a pour conséquence que les bisimilarités précoce, tardive et *ground* coïncident sur π_0 : ceci nous dispense de préciser davantage la définition de $\sim$ pour π_0.

(un processus est dit *public* s'il s'écrit sans occurrence de l'opérateur de restriction — à noter qu'un processus bisimilaire à un processus public n'est pas nécessairement public). Il est aisé de montrer que $\simeq = \simeq^p$.

On s'appuie alors sur la définition de formules logiques qui sont des *formules caractéristiques* vis-à-vis de $\sim$ pour les processus publics : si T est un processus public, on définit une formule $\mathcal{F}_T$ telle que $(P \vDash \mathcal{F}_T)$ si et seulement si $(P \sim T)$. Ceci permet de montrer $=_{\mathrm{E}} \subseteq \simeq^p$. Comme $\simeq^p \subseteq \simeq$, on aboutit finalement à $=_{\mathrm{E}} = \simeq = \sim$.

On conclut cette exposition volontairement sommaire par quelques remarques.

- *Équivalence et congruence barbelées.* L'exposé qui vient d'être fait ne correspond pas complètement à la preuve qui est présentée dans [61], dans la mesure où l'on y travaille avec l'équivalence barbelée au lieu de la congruence barbelée (voir la discussion avant la remarque 6). Ces deux relations coïncident sur π_0 ; $\simeq$ a été privilégiée ici car elle paraît plus naturelle pour la comparaison avec $\mathcal{L}_{\mathrm{E}}$.
- *Contextes : comment clore ?* On peut constater que la correspondance entre $\mathcal{L}_{\mathrm{E}}$ et la définition de $\simeq$ est moins étroite que dans le cas de la logique de Hennessy-Milner : outre l'absence de formules primitives pour observer les barbelés, la clôture par contextes parallèles (clause 3) se fait de manière moins précise dans $\mathcal{L}_{\mathrm{E}}$.
  Comme on l'a vu, alors que, selon $\simeq$, l'expérience à laquelle a accès l'observateur consiste à faire interagir les deux processus que l'on compare avec un même processus testeur, on ne peut a priori choisir individuellement dans $\mathcal{L}_{\mathrm{E}}$ un terme à ajouter en parallèle : on a accès avec la même précision aux processus testeurs et aux processus testés. Cette constatation suggère une variante de la définition de l'équivalence comportementale, $\overset{\circ}{\simeq}$, que l'on définirait comme suit :

  > *Soit $\overset{\circ}{\simeq}$ la plus grande relation symétrique $\mathcal{R}$ entre processus satisfaisant les conditions suivantes, dès que $P\mathcal{R}Q$:*
  >
  > 1. *$P \downarrow a$ implique $Q \downarrow a$ et $P \downarrow \overline{a}$ implique $Q \downarrow \overline{a}$;*
  > 2. *$P \longrightarrow P'$ implique $Q \longrightarrow Q'$ et $P'\mathcal{R}Q'$ pour un certain Q', et symétriquement lorsque $Q \longrightarrow Q'$;*
  > 3. *pour tout T, il existe T' tel que $T\mathcal{R}T'$ et $P|T \ \mathcal{R} \ Q|T'$.*

  L'intuition est ici que l'on choisit T et T' 'à $\mathcal{R}$ près' pour observer le comportement de P et Q plongés dans un contexte. La propriété de clôture par contexte parallèle que l'on impose ainsi se rapproche davantage du type d'observations que permet une logique telle que $\mathcal{L}_{\mathrm{E}}$.
  Comme on l'a dit plus haut, on établit dans [61] l'existence de formules caractéristiques pour $\sim$, ce qui nous permet de prendre T et T' dans la même classe de bisimilarité dans la clause 3 ci-dessus, et d'obtenir la coïncidence de $\overset{\circ}{\simeq}$ avec $\simeq$. Il serait intéressant d'étudier l'existence de conditions garantissant qu'une définition comme celle de $\overset{\circ}{\simeq}$ induit une équivalence connue.
- *La logique sans $\oslash$.* Si l'on supprimait l'opérateur $\oslash$ de la logique (et donc aussi V, qui n'a du coup plus de raison d'être), l'équivalence induite serait moins précise que $\sim$. Celle-ci serait alors *anonyme*, au sens où elle ne pourrait distinguer deux processus qui diffèrent d'une substitution injective sur les noms : par exemple, on ne saurait distinguer $a(x).\overline{b}\langle n\rangle.\mathbf{0}$ de $u(x).\overline{v}\langle n\rangle.\mathbf{0}$, mais on saurait distinguer ce deux processus de $a(x).\overline{a}\langle n\rangle.\mathbf{0}$. On peut conjecturer que l'équivalence logique induite serait donnée par la relation $\sim$ quotientée par l'application de substitutions injectives sur les noms.
- *Équivalences faibles.* Il paraît naturel de chercher à obtenir une caractérisation de la bisimilarité faible (où l'on traite les transitions $\xrightarrow{\tau}$ comme non observables) à l'aide d'une logique proche de $\mathcal{L}_{\mathrm{E}}$. Si l'on voulait pour cela adapter l'approche de [61], la difficulté principale résiderait en la définition de formules suffisamment précises, qui, dans le contexte d'une sémantique faible, sont plus difficiles à dériver (voir à ce sujet la partie 4.2).

2.2 La substitution comme observation [68]

2.2.1 Congruence de $\sim$ en π-calcul

Les résultats établis dans [68] (initialement parus dans [67]) mettent en avant l'influence de la substitution de noms sur le comportement. On appelle substitution une application des noms vers les noms, les substitutions étant notées avec σ. On notera $P\sigma$ le résultat de l'application de σ au processus P en respectant l'alpha conversion (σ n'agit que sur les noms libres de P, et ne rend pas lié un nom qui est libre).

Si l'on sait $P \sim Q$, a-t-on $P\sigma \sim Q\sigma$? Il apparaît que lorsque l'on enrichit le jeu comportemental habituel de la bisimilarité avec la possibilité de remplacer certains noms par d'autres dans les processus examinés, la notion d'équivalence peut s'en trouver modifiée. Cette question est centrale dans le cas du π-calcul, puisque la synchronisation entre deux processus s'accompagne d'une substitution de noms. Du fait de la présence du préfixe de réception, pour que l'équivalence comportementale soit une congruence en π-calcul, il faut qu'elle soit close par substitution. Ainsi, les équivalences *précoce* et *tardive* initialement introduites pour le π-calcul ne sont pas des congruences, alors que la bisimilarité *ouverte*, par son traitement particulier de la substitution, satisfait cette propriété (cf. [113]).

Alors que c'est le préfixe de réception du π-calcul qui introduit la substitution, c'est le préfixe d'émission qui semble être au cœur de la propriété de congruence de l'équivalence comportementale. En effet, si l'on bride le préfixe d'émission $\overline{m}\langle n\rangle.P$, les trois équivalences susmentionnées (précoce, tardive, ouverte) coïncident et sont des congruences. C'est le cas pour les sous-calculs $A\pi$ et $P\pi$, les π-calculs *asynchrone* et *privé* respectivement : en $A\pi$, la seule continuation possible à l'émission est $\mathbf{0}$, tandis qu'en $P\pi$, on n'émet que des noms frais (privés) [113].

[68] établit le résultat suivant pour le calcul π_0 (introduit à la définition 7).

Théorème 11 *$\sim$ est une congruence sur π_0.*

Ce résultat clôt un problème mentionné comme ouvert dans [113] (et existant au moins depuis [11]). Il met en avant π_0 comme le seul fragment connu du π-calcul contenant le préfixe d'émission dans toute sa splendeur pour lequel $\sim$ est une congruence. À noter qu'il n'est pas plus fort que les résultats connus sur $A\pi$ et $P\pi$, puisqu'il porte sur un calcul *fini* – on discute de l'extension au cas infini à la partie 2.2.4.

2.2.2 Preuve : utilisation d'une axiomatisation

Pour établir le théorème 11, on montre que la relation $\sim$ est close par substitution dans π_0. On s'appuie pour cela dans [68] sur une preuve de la même propriété dans un fragment de CCS, un résultat de transfert permettant ensuite de conclure pour π_0.

Appliquer des substitutions de noms aux processus n'est pas très naturel en CCS, et, à notre connaissance, la propriété de clôture par substitution de $\sim$ n'est pas mentionnée dans les travaux existants sur CCS. La raison en est que la substitution n'intervient pas dans le mécanisme de simple synchronisation qui régit ce modèle (en CCS avec passage de valeurs [86], une substitution est à l'œuvre, mais elle n'influe pas directement sur l'interaction entre les processus).

On se focalise sur un sous-calcul minimal de CCS, nommé μCCS, qui ne comporte que préfixe, composition parallèle, et communication :

Définition 12 (μCCS) *Les processus de μCCS sont définis par la grammaire suivante :*

$$P \quad ::= \quad \mathbf{0} \mid \eta.P \mid P_1|P_2 \qquad\qquad \eta \quad ::= \quad a \mid \overline{a}$$

Axiomatisation. On présente dans [68] une axiomatisation de $\sim$ sur μCCS. On établit que $\sim$ est caractérisée par les lois de monoïde commutatif pour $|$ et la loi suivante, appelée *loi de distribution* :

$$\eta.(P \mid \eta.P \mid \ldots \mid \eta.P) \quad = \quad \eta.P \mid \eta.P \mid \ldots \mid \eta.P$$

η désigne ici un préfixe, de la forme a ou $\overline{a}$, et il y a autant d'occurrences de η à gauche de l'égalité qu'à droite.

La loi de distribution suffit ainsi à exprimer les égalités comportementales sur μCCS. Par le biais des trois petits points, ce schéma équationnel décrit en réalité une infinité de lois. On montre dans [68] qu'il n'existe pas d'axiomatisation finie de $\sim$. Pour comprendre pourquoi il en est ainsi, on peut s'intéresser à l'instance de la loi de distribution au rang six, qui s'écrit $\eta.(P \,|\, (\eta.P)^5) = (\eta.P)^6$ — la notation Q^k, désigne la composition parallèle de k copies du processus Q. En partant du membre droit de cette égalité, on peut effectuer la suite d'étapes de réécriture suivantes :

$$\begin{array}{rcll} (\eta.P)^6 & = & (\eta.(P|\eta.P))^3 & \text{3 fois la loi de distribution au rang 2} \\ & = & \eta.\big(\, P|\eta.P \mid (\eta.(P|\eta.P))^2 \,\big) & \text{la loi de distribution au rang 3} \\ & = & \eta.(P \,|\, (\eta.P)^5) & \text{2 fois la loi de distribution au rang 2} \end{array}$$

Ainsi, la loi de distribution au rang six peut se dériver à partir des lois aux rangs deux et trois. Mais la seule manière d'avoir la loi de distribution à un rang premier est de l'inclure dans l'axiomatisation, ce qui permet de conclure sur le caractère non fini de l'axiomatisation. Une formulation détaillée de ces considérations est donnée dans [68, Théorème 3.2].

La loi de distribution, lue de droite à gauche, exprime une sorte d'*ordonnancement élémentaire* pour une composition parallèle de plusieurs copies d'un processus préfixé : la première chose que ce processus peut faire est offrir l'interaction correspondant au préfixe en question. C'est cependant plutôt dans le sens inverse, celui d'une distribution, que l'on oriente cette loi dans [68], ce qui permet de définir une notion de 'forme normale distribuée' pour les processus de μCCS.

Une fois qu'un processus est réécrit en sa forme normale, il est 'parallèle au possible', comme l'exprime la propriété suivante de *décomposition en facteurs premiers*, due à Milner et Moller :

Proposition 13 (Décomposition unique en processus premiers [90])
Un processus $P \nsim \mathbf{0}$ est dit premier *si $P \sim Q_1|Q_2$ implique $Q_1 \sim \mathbf{0}$ ou $Q_2 \sim \mathbf{0}$. On dit que $(P_1, \ldots, P_k)$ est une décomposition première de P si les P_i sont premiers et $P \sim P_1| \ldots |P_k$.*

Pour tout processus P de μCCS, il existe une décomposition première $(P_1, \ldots, P_k)$ de P. De plus, pour toute autre décomposition première $(Q_1, \ldots, Q_{k'})$ de P, on a $k' = k$ et $P_i \sim Q_{\rho(i)}$, où ρ est une permutation de l'intervalle $[1, .., k]$.

La caractérisation que l'on a obtenue explicite ce résultat de Milner et Moller dans [90], où l'existence d'une décomposition en facteurs premiers est établie, mais la preuve ne débouche pas sur une notion de forme normale obtenue par réécriture de termes (voir aussi [42], dont la présentation est plus proche de notre étude, sans toutefois que la loi de distribution n'apparaisse explicitement).

Il a été montré par la suite dans [78] que la loi de distribution permet également d'axiomatiser $\sim$ sur le π-calcul d'ordre supérieur sans restriction.

Transfert de μCCS à π_0. On peut remarquer que la loi de distribution est close par substitution, ce qui entraîne, du fait de l'axiomatisation décrite plus haut :

Corollaire 14 *Dans μCCS, si $P \sim Q$, alors pour toute substitution σ, $P\sigma \sim Q\sigma$.*

Pour déduire la même propriété pour π_0 (résultat qui à son tour mène à la congruence de $\sim$ dans π_0), on procède dans [68] par l'absurde, en montrant que si $\sim$ n'était pas close par substitution dans π_0, alors il en serait de même dans μCCS. Cela se fait en exploitant une propriété de transfert, fondée sur la 'projection' d'un processus de π_0 en un processus de μCCS (Proposition 5.5 dans [68] – nous n'entrons pas plus dans les détails ici).

Cela conduit finalement au théorème 11 (congruence de $\sim$ sur π_0).

2.2.3 Observer les substitutions, le rôle de la composition parallèle

Nous présentons ici un certain nombre de remarques sur l'approche qui a été suivie dans [68], notamment en lien avec le traitement que l'on y fait de l'opérateur de composition parallèle.

En π-calcul, la clôture de la bisimilarité vis-à-vis des substitutions survient comme condition nécessaire pour que cette relation soit une congruence. On peut vouloir s'intéresser à cette propriété de clôture en tant que telle, en cherchant à caractériser la plus grande relation contenue dans $\sim$ qui soit close par substitution, et ceci également pour des formalismes différents du π-calcul.

L'intérêt d'une telle approche est avant tout théorique : c'est un angle pour évaluer la robustesse d'une équivalence, et mieux en comprendre l'expressivité. L'idée ce faisant est de traiter l'application d'une substitution comme une opération à laquelle a accès l'observateur, pour faire éventuellement émerger des différences entre des processus qu'il compare. On peut s'interroger sur le sens qu'a, en π-calcul par exemple, le fait d'analyser les effets d'une substitution que l'on applique sur un processus *actif*. La substitution apparaît en π-calcul en lien avec l'opération de réception : au moment où $a(x).P$ reçoit un message sur a, x est substitué avant que la continuation P ne commence à s'exécuter. Demander à ce que l'équivalence comportementale soit close par substitution revient à s'autoriser, dans un scénario de test du type de celui de la congruence barbelée, à remplacer des noms par d'autres, ce que l'on peut voir comme une manière pour l'observateur de rediriger certaines interactions : ce qui passait par le canal a passe désormais par le canal a', ou encore, telle ressource qui était accessible en c doit maintenant être interrogée en d. Ainsi, une forme de 'mise à jour du routage' pourrait être le pendant intuitif, concret, de l'application d'une substitution, au même titre que l'on associe une idée d'expérience aux autres constituants de la congruence barbelée : la sensibilité à un barbelé, le plongement dans un contexte, ou l'évolution par réduction.

Demander à ce que l'équivalence soit stable par substitution impose une forme de robustesse en lien avec les transitions internes ($\xrightarrow{\tau}$) des processus. En identifiant deux noms, une substitution peut en effet rendre possibles certaines interactions qui ne l'étaient pas avant : c'est le cas par exemple en CCS pour $a.P \,|\, \overline{b}.Q$ si la substitution associe la même image à a et b. Plus généralement, dans les calculs de processus dont il a été question jusqu'ici (CCS, π-calcul), la composition parallèle est en quelque sorte *le lieu de l'interaction*. On peut d'ailleurs observer que dans la présentation réductionnelle de ces calculs, les règles qui ne sont pas des règles de congruence (vis-à-vis des contextes, ou de $\equiv$) mentionnent l'opérateur $|$. En se propageant sur $P|Q$, une substitution peut donc *déclencher* des transitions résultant d'interactions entre P et Q. Il est ainsi pertinent, dans cette optique, de s'intéresser aux équivalences 'local-global' de la forme $P = Q_1|Q_2$, où P ne s'écrit pas comme une composition parallèle : la question est alors de savoir si $Q_1\sigma \,|\, Q_2\sigma$ peut faire des transitions que ne peut offrir $P\sigma$.

Ces observations nous amènent à comparer le théorème d'expansion (partie 1.1) et la loi de distribution (partie 2.2.2). Ces deux schémas permettent de dériver des équivalences 'local-global'. D'un côté, pour la loi de distribution, la substitution est anodine, car tous les processus commencent par le même préfixe dans $\eta.P|\ldots|\eta.P$. De l'autre, en revanche, le théorème d'expansion permet de dériver par exemple $a.\overline{b}+\overline{b}.a \sim a|\overline{b}$, équivalence qui cesse d'être vraie dès que l'on remplace b par a (dit autrement, l'ensemble d'indices i, j pour lesquels $\alpha_i = \overline{\beta_j}$, qui intervient dans la troisième somme de la figure 1, n'est pas invariant par substitution des α_i, β_j). Ainsi, puisque la loi de distribution caractérise la bisimilarité sur μCCS, la relation $\sim$ est close par substitution sur ce calcul ; elle ne l'est pas sur CCS, qui contient l'opérateur $+$, et où la loi d'expansion tient[3].

De plus, le théorème d'expansion et la loi de distribution vont 'à l'envers' l'une de l'autre pour ce qui est du traitement de la composition parallèle. Comme indiqué plus haut, la loi de distribution est utilisée dans [68] pour calculer une forme normale qui exprime le degré maximal de parallélisme que peut manifester un processus. À l'inverse (cf. partie 1.1), en présence de l'opérateur de somme, la composition parallèle est éliminée, au profit de $+$, via le théorème d'expansion. En se référant à la métaphore chimique de [9], on peut

[3] On peut en passant mentionner [71], qui présente une étude très complète des situations où l'on a équivalence entre un terme préfixé et une composition parallèle, dans le cadre de l'algèbre de processus PA (PA offre une forme généralisée de composition séquentielle, de la forme $P;Q$). Une égalité proche de la loi de distribution apparaît déjà dans ce travail.

dire qu'on obtient ainsi une forme 'desséchée' des processus, sur laquelle sont lisibles tous les entrelacements possibles de transitions.

La littérature est riche de travaux proposant une caractérisation algébrique de $\sim$, sur différents langages, et où le cœur du problème consiste à décomposer la composition parallèle. L'article [4] dresse un panorama des résultats existants ; dans son résumé, il est notamment dit

> *"The paper also highlights the role that auxiliary operators, such as Bergstra and Klop's left and communication merge and Hennessy's merge operator, play in the search for a finite, equational axiomatization of parallel composition both for classic process algebras and for their real-time extensions."*

À l'inverse, l'objectif serait dans notre cas, pour prolonger l'étude de [68], de s'intéresser précisément à des formalismes où la composition parallèle ne peut être réduite de la sorte. On pourrait ainsi mieux comprendre les propriétés, notamment algébriques, de cet opérateur, que l'on ne voit alors pas comme un outil pour composer des comportements, mais plutôt comme une construction 'spatiale', ou 'structurelle' (point de vue qui est amplement développé dans la suite de ce document). On présente ci-dessous les premières investigations que l'on a entreprises dans cette direction.

Remarquons enfin que les travaux sur l'encodage du choix en π-calcul [96, 100, 95] partent d'un point de vue qui rejoint également ces observations, puisque ceux-ci étudient dans quelle mesure le choix peut être vu comme un opérateur non primitif. Ceci va à rebours du théorème d'expansion (Proposition 3) : alors que traditionnellement on *exprime* $|$ à l'aide de $+$ en CCS, on a plutôt tendance à vouloir *programmer* $+$ à l'aide de $|$ en π-calcul.

2.2.4 Prolongements [69]

La question de la clôture de $\sim$ vis-à-vis des substitutions, qui a été entamée dans [68], peut être approfondie de plusieurs manières.

Compléter le tableau : propriétés algébriques de la réplication. Comme on l'a indiqué plus haut, un angle que l'on souhaite continuer à développer est celui de l'étude des propriétés algébriques de la bisimilarité, dans des formalismes qui n'incluent pas l'opérateur de choix. Ceci devrait en particulier permettre d'analyser la question de la clôture vis-à-vis des substitutions de $\sim$. On a vu en effet que dans les cas qui ont été étudiés, la bisimilarité n'est pas close par substitution en présence de $+$.

En π-calcul, depuis [87], les comportements infinis s'obtiennent volontiers par l'intermédiaire de l'opérateur de réplication, noté !, qui fournit une forme de 'récursion spatiale'. Il est régi par les lois de congruence structurelle suivantes :

$$!P \equiv !P \mid P \qquad !(P \mid Q) \equiv !P \mid !Q \qquad !!P \equiv !P \qquad !\mathbf{0} \equiv \mathbf{0}$$

On peut résumer comme suit les propriétés connues s'agissant de la clôture vis-à-vis des substitutions en π-calcul (et donc de la congruence de la bisimilarité), dès lors que l'on inclut le préfixe d'émission :

restriction sans réplication : oui [68] réplication sans restriction : ? [69]	restriction et réplication : non [11]

FIG. 2 – Congruence de $\sim$ en présence d'émissions synchrones

Dans les calculs sans opérateur de choix, la clôture par substitution échoue dès que l'on inclut les opérateurs de restriction et de réplication [11]. Pour le π-calcul, on l'a dit, on peut rétablir cette propriété en restreignant l'opérateur d'émission, dans les versions *asynchrone* et *localisée* du calcul [113, calculs Aπ et Lπ – à noter que dans le cas de Lπ on a affaire à une classe restreinte de substitutions]. On a montré ci-dessus que lorsque l'émission synchrone est incluse dans le calcul, la clôture par substitutions tient en l'absence de réplication.

La question se pose donc pour le cas d'un calcul sans restriction, mais avec des comportements infinis. On choisit d'inclure l'opérateur de réplication plutôt que des définitions récursives, avec l'idée qu'il est plus simple d'exprimer des propriétés algébriques dans ce cadre. De plus, la réplication apparaît comme un choix naturel, dans la mesure où elle peut être vue comme une version infinitaire de la composition parallèle, opérateur qui est au cœur de notre étude.

Nous avons commencé à analyser les propriétés algébriques de la bisimilarité lorsque l'on ajoute la réplication à μCCS. Dans [69], on étudie le cas où les réplications ne sont appliquées que sur des préfixes, et sont *actives*, au sens où elles n'apparaissent pas sous des préfixes. La première limitation est anodine, puisque l'on travaille dans un calcul sans restriction : les lois mentionnées ci-dessus pour $\equiv$ pourraient être ajoutées sans difficulté afin de la lever. La seconde est plus déterminante, car, comme on l'évoque plus bas, le traitement des *réplications enfouies* (apparaissant sous un préfixe) introduit des difficultés supplémentaires. Le calcul avec lequel on travaille est donc défini par une grammaire à deux étages, pour les processus finis et les processus (Figure 3 — comme précédemment, on se focalise sur un fragment de CCS, pour ensuite étendre les résultats au π-calcul).

$$F \quad ::= \quad \mathbf{0} \;\big|\; \eta.F \;\big|\; F|F \qquad\qquad P \quad ::= \quad !\eta.F \;\big|\; F \;\big|\; P|P$$

FIG. 3 – Extension de μCCS avec réplications actives

Dans ce calcul, la loi de distribution vue plus haut est bien sûr toujours valide, et doit être prise en compte pour caractériser $\sim$. Il s'agit, au-delà de cette observation, d'exprimer les propriétés de la réplication vis-à-vis de la bisimilarité. Un point de départ possible, dans cette optique, est donné par les lois classiques suivantes :

$$!\eta.P \,|\, !\eta.P = !\eta.P \qquad\qquad !\eta.P \,|\, \eta.P = !\eta.P \ .$$

On montre dans [69] que la première loi ci-dessus peut être adoptée telle quelle, alors qu'il faut généraliser la seconde pour capturer $\sim$. Une première généralisation est donnée par

$$!\eta.P \,|\, C[\eta.P] = !\eta.P \,|\, C[\mathbf{0}] \ .$$

Ici $C[]$ désigne un contexte, qui est défini comme un processus avec un trou ; $C[Q]$ est le processus que l'on obtient en insérant Q à la place du trou dans $C[]$. Intuitivement, cette loi dit qu'un processus répliqué peut absorber une copie de lui-même non seulement en position active (autrement dit, en parallèle), mais aussi n'importe où dans un terme. On a par exemple $!a.b \,|\, d.(e|a.b) \ \sim !a.b \,|\, d.e$ (mais $!a.b \,|\, d.a.b.c \ \not\sim !a.b \,|\, d.c$).

Cela n'est toutefois pas suffisant : une première remarque est qu'un processus peut se répliquer à l'intérieur de lui-même : $!\eta.C[\eta.C[\mathbf{0}]] = !\eta.C[\mathbf{0}]$ (par exemple, $!a.(b|a.b) \sim !a.b$). Plus généralement, k processus répliqués peuvent se recopier les uns dans les autres 'simultanément', sans que cela résulte d'une séquence d'applications des lois que l'on a mentionnées jusqu'ici. On a par exemple l'équivalence

$$!a.a' \,|\, !b.b' \,|\, !c.c' \quad \sim \quad !a.(b.b'|a') \;|\; !b.(c.c'|b') \;|\; !c.(a.a'|c') \ ,$$

que l'on obtient en recopiant chaque processus répliqué chez son voisin de gauche, circulairement.

Nous nous contentons ici de cette esquisse des propriétés algébriques de $\sim$ dans le calcul avec réplications actives. Intuitivement, il suffit, pour caractériser $\sim$, de rendre compte du type d'égalités que l'on vient d'évoquer. Il paraît hors d'atteinte d'écrire une loi qui décrive de manière raisonnablement concise et lisible le phénomène de 'copies mutuelles' que l'on vient d'évoquer. Dans [69], on définit une procédure de réécriture des processus s'appuyant sur les lois ci-dessus, dont on démontre qu'elle induit une notion de forme normale pour la bisimilarité. Plus précisément, cette procédure permet de calculer, étant donné un processus P, un processus S de taille minimale tel que $P \sim S$ (la taille étant définie comme le nombre de préfixes d'un terme). Ce procédé de normalisation des processus par réécriture a donné lieu à une implémentation prototype [44].

Cette caractérisation permet de conclure à la clôture par substitutions de la bisimilarité sur le sous-calcul de CCS sans restriction et avec réplications actives :

Théorème 15 ([69]) *Soient P, Q deux processus du sous-calcul de* CCS *calcul décrit à la Figure 3. Si $P \sim Q$, alors pour toute substitution σ, $P\sigma \sim Q\sigma$.*

On s'appuie sur ce résultat pour établir une nouvelle propriété de congruence pour la bisimilarité en π-calcul, pour le calcul correspondant :

Théorème 16 ([69, Corollary 36]) *La bisimilarité forte, définie sur le sous-calcul du π-calcul induit par la syntaxe de la Figure 3, où η fait référence aux préfixes du π-calcul (émission $a(x)$ et réception $\overline{a}\langle v\rangle$), est une congruence.*

L'étape suivante est de s'intéresser à un calcul avec réplications enfouies. On commence pour ce faire par travailler dans le cadre d'un calcul où l'on autorise les réplications enfouies, mais on interdit les réplications imbriquées (pas de réplication apparaissant sous une réplication). On voit alors que la loi

$$\eta.(!\eta.F \mid F) = !\eta.F$$

fait son apparition ; cette égalité peut être vue comme une version infinitaire de la loi de distribution.

La notion de forme canonique des processus pour $\sim$ est assez subtile à définir dans ce cadre. Ainsi, si l'on reprend l'idée de la forme canonique comme processus bisimilaire de taille minimale, on s'attend à ce que la forme normale de $!a.b \mid c.(!b.a \mid d.b.a)$ soit $!a.b \mid c.(!b|d)$. Intuitivement, lorsque le préfixe c est consommé, $!a.b$ et $!b.a$ se retrouvent en parallèle, ce qui leur permet de se 'simplifier' en $!a|!b$, ce qui déclenche de nouvelles simplifications — simplifications qui ne sont valables que lorsque c a été consommé, ce qui justifie le fait que l'on garde $!a.b$ en position active dans la forme canonique : on ne peut pas encore effacer le second préfixe b. On ne sait pas dire pour le moment si la loi ci-dessus est suffisante pour caractériser $\sim$ en présence de réplications enfouies.

Plus généralement, il apparaît que plusieurs étapes restent encore à franchir avant d'avoir une vision détaillée des propriétés algébriques de $|$ (et de !) qui entrent en jeu dans la caractérisation de la bisimilarité forte en l'absence d'opérateur de choix. En particulier, le cas des réplications imbriquées ne semble pas pouvoir se traiter de manière immédiate dans le prolongement de l'étude sur les réplications enfouies non imbriquées. Il serait intéressant, au fur et à mesure que se dessine une compréhension plus détaillée de ces questions, de dégager une vision générale des lois régissant $\sim$, avec l'ambition de pouvoir traiter, à partir de cette connaissance, d'autres formalismes, plus riches que les calculs que l'on a évoqués jusqu'ici.

Équivalences plus fines. Une autre manière d'analyser la question de la côture vis-à-vis des substitutions est de jouer sur l'équivalence comportementale que l'on adopte : lorsque $\sim$ n'est pas assez discriminante, comment raffiner cette relation pour obtenir la clôture vis-à-vis des substitutions ? Un résultat de ce type est établi dans [68] pour le calcul μCCS$^+$, dont la syntaxe est la suivante :

$$S ::= \mathbf{0} \mid \eta.P \mid S_1 + S_2 \qquad\qquad P ::= S \mid P_1|P_2$$

(η désigne ci-dessus un préfixe CCS, de la forme a, $\overline{a}$ ou τ). μCCS$^+$ est l'extension de μCCS avec des sommes de termes préfixés. On note $\sim_{\mathrm{d}}$ l'adaptation à μCCS$^+$ de la *bisimilarité distribuée* de [39]. Informellement, les coups dans le jeu de bisimulation pour $\sim_{\mathrm{d}}$ sont des transitions de la forme $P \xrightarrow{\mu} \langle P_1, P_2\rangle$, où P_2 est le processus résultant de l'évolution de P selon μ, et P_1 correspond au résidu de la composante qui a bougé dans P ; par exemple, $a.T \mid \overline{b}.U \xrightarrow{\overline{b}} \langle a.T|U, U\rangle$. Sans entrer davantage dans les détails techniques de la définition de $\sim_{\mathrm{d}}$, il s'agit d'une équivalence strictement plus discriminante que $\sim$, qui prend en compte les phénomènes de causalité ($\sim_{\mathrm{d}}$ est *non entrelaçante* – voir aussi à ce sujet le début de la partie 4.1). En particulier, $a.a$ et $a \mid a$ sont distincts pour $\sim_{\mathrm{d}}$. On a alors :

Proposition 17 (Proposition 4.10 dans [68]) *$\sim_{\mathrm{d}}$ est close par substitution sur μCCS$^+$.*

On peut remarquer d'une part que $\sim$ n'est bien sûr pas close par substitution en μCCS^+ (puisque le théorème d'expansion vaut pour $\sim$), alors que $\sim$ et $\sim_{\text{d}}$ le sont en μCCS.

Une équivalence comme $\sim_{\text{d}}$ met en avant la structure des processus, en étant plus fine que la bisimilarité. Ce point de vue est développé abondamment dans les parties qui suivent, dans la mesure où les logiques spatiales étudiées se fondent sur des la prise en compte de propriétés *intensionnelles*, qui rejoignent la définition de $\sim_{\text{d}}$.

3 Inspections et équivalences intensionnelles

Les développements précédents ont porté sur l'analyse du comportement des processus, et de la notion d'équivalence qui lui est associée. Dans cette partie, nous nous intéressons à des propriétés de la structure des processus. Outre la possibilité de reconnaître un comportement, on veut également être sensible à comment celui-ci est 'codé' — à partir de quels composants il s'obtient. Pour prendre une métaphore théâtrale, on ne s'intéresse pas uniquement à l'intrigue de la pièce, mais aussi à sa mise en scène. Ce type d'analyse a une connotation *intensionnelle* : on obtient des informations portant sur la constitution des processus, plus que sur leur comportement (auquel on attache le caractère extensionnel). L'articulation entre intensionalité et extensionalité pour les logiques spatiales est discutée, au vu des résultats que l'on a pu obtenir, à la partie 3.2.3.

Un rôle central dans ce qui suit est joué par l'opérateur de *conjonction spatiale*, que nous présentons pour commencer. On verra comment cet opérateur peut être employé dans divers scenarii, qui ne se cantonnent pas à l'analyse des processus concurrents ; on verra aussi comment on lui adjoint, suivant les cas, d'autres mécanismes d'analyse pour donner sens à l'exploration de ce que l'on nomme de manière générique une *structure*. On s'attachera ensuite à confronter inspections et interactions, autrement dit, à établir une comparaison entre le pouvoir expressif et discriminant fourni par les analyses spatiales (les inspections) et les observations que permet la bisimilarité.

3.1 Observations spatiales

3.1.1 Conjonction spatiale : découper

Décomposer les processus

Pour illustrer le point de vue que l'on souhaite développer dans cette partie, on peut partir de la question suivante : est-ce qu'un comportement que l'on observe peut s'obtenir comme combinaison de deux comportements plus simples, ou bien est-il premier ? En ce sens, le processus $P = a.b.a$ est premier, puisque la seule manière de le décomposer est d'écrire quelque chose comme $P \,|\, \mathbf{0}$, qui peut être considéré comme une décomposition triviale, au même titre que lorsque l'on écrit $19 = 19 \times 1$ (cette idée est à rapprocher de la proposition 13).

On introduit pour cela la *conjonction spatiale*, notée $|$, comme la composition parallèle. Cette observation va en quelque sorte à rebours de la composition parallèle : alors que $|$ (sur les processus) combine deux processus, $|$ (sur les formules ou les observations) décompose un processus.

La conjonction spatiale correspond à une perception statique, de l'*état* que définit un terme : on se fonde pour ce faire sur une vision plus 'textuelle' que ce que propose la bisimilarité, tout en s'éloignant de l'arbre syntaxique. La congruence structurelle correspond généralement à cette notion d'état, ce qui conduit à la définition suivante pour la satisfaction d'une conjonction spatiale :

$$P \vDash \mathcal{A}_1|\mathcal{A}_2 \quad \text{si et seulement si} \quad \exists P_1, P_2.\, P \equiv P_1|P_2 \text{ et } P_i \vDash A_i,\ i = 1, 2\ .$$

On peut faire trois remarques à propos de cette définition.

- *Combinatoire.* Etant donné que la satisfaction de $\mathcal{A}_1|\mathcal{A}_2$ est définie modulo $\equiv$, on est immédiatement confronté à une difficulté combinatoire si l'on s'intéresse à la question du *model checking*, c'est-à-dire si l'on cherche à savoir si un processus P satisfait $\mathcal{A}_1|\mathcal{A}_2$. Cet aspect combinatoire se ressent

en réfléchissant à l'interprétation de l'opérateur dit *dual* de $|$, noté $||$, introduit dans [35], et défini par $\mathcal{A} \,||\, \mathcal{B} \stackrel{\text{def}}{=} \neg(\neg\mathcal{A} \,|\, \neg\mathcal{B})$ — on laisse au lecteur le soin de déduire cette interprétation[4]. Comme le montre [7], l'un des enjeux dans la conception d'un logiciel comme Smallfoot est de maîtriser les utilisations de la conjonction spatiale, afin de limiter l'explosion combinatoire. L'algorithme de model checking décrit dans [17] est lui relativement naïf, dans la mesure où il traite la satisfaction d'une formule $\mathcal{A}_1|\mathcal{A}_2$ dans le cas général, en parcourant toutes les décompositions possibles d'un processus (Lemme 4.1 de [17]).

– *Origines.* L'idée de concevoir des logiques modales permettant d'inspecter la structure des configurations dans des calculs de processus comme le π-calcul ou les Mobile Ambients apparaît dans [24, 35]. Pour autant, la volonté de s'appuyer sur la composition parallèle pour raisonner de manière compositionnelle sur les systèmes concurrents est plus ancienne. La partie 3.2.4 rassemble des références (pour la plupart datant des années 80) dans lesquelles on trouve des idées qui sont mises en avant plus tard dans les travaux sur les logiques spatiales pour la concurrence.
– *Une observation élémentaire.* La conjonction spatiale est une observation de *séparation statique*. Il est nécessaire de l'associer à d'autres formes d'analyse des processus, afin de découvrir les propriétés des processus résultant de cette inspection (on a en particulier vu ci-dessus qu'un processus P peut toujours être découpé, de façon triviale, en $P|\mathbf{0}$). Ce dernier aspect est développé dans la suite de cette partie.

3.1.2 Observer ce qui a été séparé

La conjonction spatiale apparaît dans divers travaux, qui n'ont pas uniquement trait à la théorie de la concurrence. Elle est associée à différentes formes d'observations élémentaires, qui varient suivant les formalismes, et que nous décrivons ci-dessous.

Observations statiques.
Le néant.

L'observation 'radicalement élémentaire' consiste à détecter la présence (ou l'absence) de l'espace observé. L'assertion correspondante est incluse dans la logique de séparation (dont il est question ci-dessous), et aussi dans les logiques spatiales pour la concurrence [33, 20]. La formule 0 dénote l'*absence d'espace* :

$$P \vDash 0 \quad \text{si et seulement si} \quad P \equiv \mathbf{0}$$

(ne pas confondre le processus inactif, $\mathbf{0}$, et la formule 0).

Puisque l'on peut écrire tout processus P comme P composé avec un nombre quelconque de copies de $\mathbf{0}$, il n'est a priori pas très intéressant d'utiliser la formule 0 en conjonction avec $|$. En revanche, la négation de 0 permet de compter les composantes parallèles d'un processus, de la manière suivante :

$$\lceil 1 \rceil \stackrel{\text{def}}{=} \neg 0 \wedge \neg(\neg 0 | \neg 0) \qquad\qquad \lceil k \rceil \stackrel{\text{def}}{=} \underbrace{\lceil 1 \rceil | \ldots | \lceil 1 \rceil}_{k \text{ fois}}, \quad k \geq 2$$

Un processus insécable est un processus qui n'est pas rien et qui n'est pas séparable en deux processus qui ne sont pas rien : c'est la formule $\lceil 1 \rceil$, à l'aide de laquelle on peut exprimer le fait qu'un processus se décompose en k sous-processus insécables.

Les opérateurs des logiques spatiales permettent de 'compter' dans les structures que les logiques observent : intuitivement, on peut décrire des ensembles d'entiers naturels, $|$ servant à faire des sommes, et l'adjoint $\triangleright$ (cf. partie 3.1.3) servant à faire des soustractions. Ce point de vue est développé en particulier dans [45], où les auteurs étudient le fragment *statique* de la logique des Ambients ($\mathcal{L}_{\mathrm{A}}$, partie 4.2), fragment qui peut être vu comme une logique pour raisonner sur des arbres (par exemple, des données au format XML). Une contribution importante de ce travail est de montrer que le problème du *model checking* pour cette logique peut être mis en relation avec la satisfaction de contraintes sur des vecteurs d'entiers, exprimées sous forme de contraintes dans l'arithmétique de Presburger.

[4] Réponse : $P \vDash \mathcal{A} \,||\, B$ si et seulement si pour toute écriture $P \equiv P_1|P_2$, soit $P_1 \vDash \mathcal{A}$, soit $P_2 \vDash \mathcal{B}$.

Programmes manipulant des pointeurs.

La *logique de séparation* (*Separation Logic*, voir [107]) sert à raisonner sur les programmes manipulant des pointeurs. Un tel programme, en s'exécutant, lit et modifie l'état de la mémoire. En logique de séparation, une observation élémentaire (autre que 0, qui dit qu'aucune cellule n'est allouée dans la mémoire) est de la forme $x \hookrightarrow 19$, pour dénoter que la variable x du programme pointe vers une cellule mémoire contenant la valeur 19. La logique permet en particulier d'exprimer des propriétés liées aux phénomènes de partage dans la mémoire. Ainsi, la formule $x \hookrightarrow 27 \wedge y \hookrightarrow 27$ dit que les variables x et y pointent vers une case mémoire contenant la valeur 27, mais elle n'empêche pas x et y d'être des alias. Au contraire, la formule $x \hookrightarrow 27 \mid y \hookrightarrow 27$, qui utilise la conjonction spatiale, impose que les deux sous-formules soient vérifiées par deux portions disjointes de la mémoire, ce qui force x et y à pointer vers deux cases distinctes (en logique de séparation, l'opérateur $\mid$ est généralement noté $*$).

En logique de séparation, les assertions exprimant des propriétés de l'état de la mémoire sont utilisées, dans le prolongement de la logique de Floyd-Hoare, pour spécifier des pré- et post-conditions à l'exécution des programmes. On s'appuie sur la conjonction spatiale pour établir qu'un programme n'est pas sujet à des phénomènes d'interférence.

La logique de séparation se prête naturellement à l'étude de programmes śequentiels de bas niveau. Elle a donné lieu à des mécanisations pour raisonner sur machine, que ce soit via des méthodes complètement automatiques, comme dans l'outil Smallfoot [8], ou dans le cadre d'assistants de preuve — voir par exemple [73], qui présente une mise en œuvre de la logique de séparation concurrente.

Une extension de la logique de séparation pour raisonner sur des programmes concurrents a en effet été proposée [98]. Le tas reste le même, et plusieurs tâches y accèdent simultanément, selon le modèle classique de la mémoire partagée. Dans cette approche, les assertions sur le tas rendent compte de l'accès exclusif à une ressource partagée pour une tâche donnée. On a donc affaire à un scénario où données et programmes sont distingués ; en particulier, les assertions ne portent que sur les données, et les formules de la logique sont utilisées pour découper le tas, par le biais de la conjonction spatiale, en des 'zones d'influence' sur lesquelles les tâches agissent en l'absence d'interférences. Contrairement au cas des calculs de processus, la structure sur laquelle on raisonne à l'aide des formules de la logique de séparation est complètement passive.

L'article [72] définit une logique spatiale, apparentée à la logique de séparation, pour les processus CSP. En un certain sens, cette approche se situe à la convergence entre les logiques pour la mémoire, issues de la logique de séparation, et les logiques pour les processus, dont il est question dans ce document. Il serait particulièrement intéressant, dans la lignée de cette proposition, de pouvoir appliquer les méthodes qui ont été définies et implémentées dans le cadre de la logique de séparation (règles d'inférence, vérification automatisée) afin de raisonner sur les processus. L'article [123] représente une avancée prometteuse dans cette direction : les techniques de la logique de séparation sont à l'œuvre pour raisonner sur des processus interagissant par passage de messages, les assertions sur le tas étant utilisées pour décrire le transfert des messages. Un prolongement naturel de ce travail serait de faire la transition entre la granularité fine que met en avant la logique de séparation dans son analyse de la concurrence et des modèles de plus haut niveau, où certains aspects de l'exécution des processus peuvent être abstraits.

La restriction (et la fraîcheur).

L'opérateur de restriction, par le biais de la notion de portée qui lui est associée, a une connotation spatiale : une restriction sur un nom a détermine une zone où a est connu et peut être utilisé, ce nom étant inconnu du reste du processus (et du contexte). L'article [36] étudie comment définir une forme d'inspection qui puisse être associée à l'opérateur de restriction. Ceci se fait par l'intermédiaire d'un opérateur de *révélation*, noté Ⓡ, dont la satisfaction est définie comme suit :

$$P \vDash n\text{Ⓡ}\mathcal{A} \quad \text{si et seulement si} \quad \exists P'.\, P \equiv (\boldsymbol{\nu} n)\, P' \ \wedge \ P' \vDash \mathcal{A} \ .$$

Puisque la satisfaction d'une formule de la forme $n\text{Ⓡ}\mathcal{A}$ se fait modulo $\equiv$, il n'y a pas moyen de spécifier *quelle* restriction est révélée lors de cette inspection, car les restrictions 'flottent' dans un processus, notamment du fait de la loi d'extrusion pour $\equiv$. Cette forme d'incertitude quant au nom que l'on révèle rappelle

la combinatoire liée à la satisfaction d'une formule de la forme $\mathcal{A}_1|\mathcal{A}_2$, pour laquelle on ne sait 'où on coupe' dans le processus. On peut ainsi en particulier révéler une restriction inutile, c'est-à-dire qui ne porte pas sur un nom utilisé dans le processus qui est observé. La seule contrainte est que $n®\mathcal{A}$ ne peut être satisfaite par un processus où le nom n a une occurrence libre — de ce fait, la formule $\neg(n®\top)$ exprime que le nom n a une occurrence libre dans le processus observé.

L'opérateur ® seul n'est pas suffisant pour traduire dans la logique le comportement de l'opérateur de restriction : il faut pouvoir exprimer le fait que le nom que l'on révèle au moment de l'ouverture de la restriction est inconnu (ou *frais*) vis-à-vis de tout observateur. Ceci se fait par l'intermédiaire du quantificateur de fraîcheur M, qui a été présenté à la partie 2.1 :

$$P \vDash \mathsf{M}n.\mathcal{A} \quad \text{si et seulement si pour } m \text{ frais pour } P \text{ et } \mathcal{A}, \quad P \vDash \mathcal{A}_{\{m/n\}} \ .$$

Comme il a déjà été mentionné, la définition (et la notation) de l'opérateur M est issue des travaux de A. Pitts et J. Gabbay sur les lieurs [54]. La *quantification sur les noms cachés*, qui permet d'exprimer une propriété portant sur un processus que l'on découvre en passant sous une restriction, peut alors être introduite de la manière suivante : $\mathsf{H}n.\mathcal{A} \stackrel{\text{def}}{=} \mathsf{M}n.\, n®\mathcal{A}$. Cet opérateur dérivé H, pendant du côté logique de $\boldsymbol{\nu}$, est mis en avant dans [33] ; il est utilisé dans des travaux ultérieurs comme [20, 84].

Localités.

Comme on le verra à la partie 4.2, les logiques spatiales sont volontiers utilisées pour raisonner sur des calculs de processus comportant une notion primitive de localité. À une telle construction correspond naturellement une inspection élementaire, qui consiste à entrer dans une localité pour observer le processus qui s'y exécute. Les localités jouent en quelque sorte le rôle des adresses en mémoire dans la logique séparante, à ceci près qu'alors que des valeurs sont stockées en mémoire dans les programmes manipulant des pointeurs, c'est un processus, c'est-à-dire une entité active, qui s'exécute en une localité.

Observations dynamiques. La conjonction spatiale, de même que les inspections dont il vient d'être question, observent une structure 'au repos'. Ce qui est propre aux logiques spatiales pour la concurrence est le caractère dynamique du système observé : les observations spatiales et temporelles interagissent et coopèrent pour analyser les évolutions des processus.

Le temps.

On a déjà vu plus haut le connecteur $\Diamond$, qui sert à faire évoluer le processus observé. Si $\rightsquigarrow$ désigne la relation de réduction des processus, on définit

$$P \vDash \Diamond\mathcal{A} \quad \text{si et seulement si} \quad P \rightsquigarrow\vDash \mathcal{A} \ .$$

En pratique, la relation $\rightsquigarrow$ est instanciée par $\longrightarrow$, ou par $\longrightarrow^*$, la clôture réflexive et transitive de $\longrightarrow$, ce qui détermine les interprétations forte et faible, respectivement, de l'opérateur $\Diamond$. La formule $\Diamond 0$, par exemple, suivant que l'on se place dans l'interprétation forte ou faible, a des significations différentes. Dans le cas fort, un processus satisfaisant $\Diamond 0$ est un processus qui peut se réduire en un pas pour devenir le processus $\mathbf{0}$: en μCCS, cette formule caractérise les processus de la forme $a|\overline{a}$ (à $\equiv$ près) pour un certain a.

Avec l'interprétation faible de $\Diamond$, la formule exprime que le processus *peut* se réduire (en zéro ou plusieurs pas) vers le processus $\mathbf{0}$. C'est le cas, en μCCS, par exemple pour $a.b \mid a.c \mid \overline{a} \mid \overline{c}.(\overline{a} \mid \overline{b})$, processus qui satisfait également la formule $\Diamond\neg 0$. La formule $\neg\Diamond\neg 0$, souvent notée $\Box 0$, exprime le fait qu'un processus ne peut qu'évoluer vers un processus satisfaisant 0 : toujours avec l'interprétation faible, ce n'est même pas le cas pour $a|\overline{a}$, qui peut faire zéro transition vers lui-même ! L'interprétation de la formule $\Box\Diamond 0$ paraît plus sensée : quelle que soit l'évolution du processus observé, on pourra toujours continuer à se réduire vers un processus satisfaisant 0.

Interactions : les modalités à la Hennessy-Milner.

L'introduction de modalités à la Hennessy-Milner (voir partie 1.1) dans la logique fournit a priori davantage de pouvoir d'observation au contexte que lorsqu'on se contente de l'opérateur $\Diamond$ pour observer l'évolution des processus. La logique de [17], sur laquelle on revient plus bas, inclut de telles modalités, afin de permettre la combinaison d'observations comportementales et spatiales.

La logique de Hennessy-Milner, étendue avec des inspections spatiales, induit une équivalence plus fine que la bisimilarité. Si l'on ajoute 0 et $|$, on obtient un pouvoir discriminant très fort, et on s'attend à ce que l'équivalence logique coïncide avec la congruence structurelle (ceci est notamment suggéré par les résultats exposés à la partie 4.2, ainsi que par [17]).

Si l'on n'ajoute que la conjonction spatiale, la situation est a priori moins claire. Lorsque le calcul sous-jacent ne comporte pas l'opérateur de restriction, les observations permises par la logique semblent se rapprocher fortement de la définition de la bisimilarité distribuée (qui a été évoquée à la partie 2.2.4). On peut alors conjecturer que l'équivalence logique coïncide avec cette équivalence (qui à son tour devrait être égale à $\equiv$). Il paraît plus difficile d'énoncer un résultat de ce type lorsque le calcul contient la restriction — la notion même de bisimilarité distribuée en présence de restriction n'est pas totalement claire, comme l'indique [38].

Les *transitions localisées* de [12] sont également à mettre en relation avec ces considérations : des modalités de la forme $\langle a \rangle_t \mathcal{A}$, où t est une localité, permettent de définir une logique correspondant dans certains cas à la bisimilarité distribuée.

3.1.3 Opérateurs adjoints

Garantie.

À l'opérateur $|$, qui est une forme de conjonction, on associe l'opérateur d'*implication spatiale*, appelé souvent *opérateur de garantie*, et noté $\triangleright$. Alors que $|$ inspecte le processus en en décomposant la structure, $\triangleright$'s'en éloigne' en plongeant le processus dans un contexte :

$$P \vDash \mathcal{A}_1 \triangleright \mathcal{A}_2 \quad \text{ssi pour tout } T \text{ tel que } T \vDash \mathcal{A}_1, \quad \text{on a} \quad P|T \vDash \mathcal{A}_2 \ .$$

Tout processus satisfaisant $(\mathcal{A}_1|\mathcal{A}_2) \triangleright \mathcal{B}$ satisfait $\mathcal{A}_1 \triangleright (\mathcal{A}_2 \triangleright \mathcal{B})$, et inversement : en ce sens, $|$ et $\triangleright$ sont des opérateurs adjoints. On utilise souvent dans les preuves l'opérateur dérivé $\blacktriangleright$, que l'on définit en posant $\mathcal{A}_1 \blacktriangleright \mathcal{A}_2 \ = \ \neg(\mathcal{A}_1 \triangleright \neg\mathcal{A}_2)$: on exprime l'*existence* d'un processus satisfaisant $\mathcal{A}_1$ en présence duquel $\mathcal{A}_2$ est satisfaite.

Le processus T est un testeur : en présence d'un testeur (ou *intrus*) satisfaisant $\mathcal{A}_1$, on garantit le comportement $\mathcal{A}_2$ (la terminologie *"guarantee"* est aussi utilisée en référence aux spécifications *assume-guarantee* [76] – voir aussi la partie 3.2.4 à ce sujet). En d'autres termes, $\triangleright$ permet à l'observateur d'interagir avec le processus, en lui donnant la possibilité d''incarner' une partie du contexte.

La quantification universelle que contient la définition de $\vDash$ pour $\triangleright$ confère un très grand pouvoir expressif à cet opérateur. Par exemple, on peut réduire la validité au model checking : la formule $\mathcal{A} \triangleright \bot$ exprime que la formule $\mathcal{A}$ n'a pas de modèle, et, si l'on sait décider si $\mathbf{0} \vDash \mathcal{A} \triangleright \bot$, on sait décider si $\mathcal{A}$ a un modèle. Du fait de ce grand pouvoir expressif, les outils existants pour les logiques spatiales n'incluent pas $\triangleright$: c'est le cas de l'implémentation prototype d'un model checker pour la logique spatiale décrite dans [122] (en amont, la logique spatiale pour le π-calcul décrite dans [17] inclut des modalités à la Hennessy-Milner, mais pas $\triangleright$, afin d'avoir la décidabilité du model checking). C'est également le cas de Smallfoot [8, 7], un outil pour la vérification de programmes manipulant des pointeurs, qui repose sur une version de la logique de séparation dont (l'opérateur correspondant à) $\triangleright$ est exclu.

Dissimulation.

Sur le modèle de $\triangleright$ pour $|$, on introduit l'opérateur $\oslash$, qui est la contrepartie de ®® :

$$P \vDash \mathcal{A} \oslash n \quad \text{si et seulement si} \quad (\boldsymbol{\nu} n)\, P \vDash \mathcal{A} \ .$$

Il est facile d'associer une intution à l'utilisation de $\oslash$: l'observateur renonce à interagir sur un nom donné. Dans $\mathcal{L}_{\mathrm{E}}$, en particulier, $\oslash$ est le seul opérateur de la logique qui mentionne les noms, et c'est grâce à lui que l'on peut définir des formules distinguant par exemple $a(x).\mathbf{0}$ de $b(x).\mathbf{0}$.

Situation.

Similairement, à l'opérateur $n[\cdot]$ vu plus haut correspond $\cdot@n$:

$$P \vDash \mathcal{A}@n \quad \text{ si et seulement si } \quad n[P] \vDash \mathcal{A} \ .$$

3.1.4 Logiques pour les processus : propositions

On évoque ici plusieurs logiques spatiales pour la concurrence ayant été étudiées dans la littérature. Ces formalismes combinent les opérateurs que l'on vient de décrire, pour raisonner sur divers calculs de processus.

La logique spatiale pour les Mobile Ambients (voir partie suivante) est décrite dans [35] par L. Cardelli et A. Gordon. Certaines des idées qui sous-tendent ce formalisme étaient déjà présentes dans la logique proposée par L. Caires et L. Monteiro dans [24] pour raisonner sur une extension du π-calcul avec des idées liées à la programmation logique. Caires et Cardelli ont développé ensuite une logique spatiale pour le π-calcul, présentée dans [20, 21] : celle-ci hérite des idées de [35], en y ajoutant des opérateurs de point fixe pour exprimer des propriétés définies de manière (co)inductive sur les processus.

[17] analyse une logique apparentée pour le π-calcul, pour laquelle un résultat de décidabilité du model-checking est établi. Un point essentiel pour obtenir ce résultat est que l'implication spatiale ($\triangleright$) n'est pas incluse dans la logique. En plus des opérateurs permettant des inspections de la structure des processus (0, $|$, $®$), des modalités à la Hennessy-Milner primitives permettent de spécifier des propriétés comportementales des processus.

À la suite de ces travaux, d'autres propositions ont vu le jour pour raisonner sur des formalismes apparentés. [41] présente une logique pour exprimer des propriétés *statiques* sur les bigraphes [88]. Cette logique comporte deux opérateurs, qui correspondent aux deux formes de composition présentes dans les bigraphes : l'une consiste à mettre deux bigraphes côte â côte, l'autre à insérer un bigraphe dans un autre. À chacun de ces opérateurs de composition sont associées deux formes d'implication (qui, pour la seconde forme de composition, collapsent en une seule lorsque cette opération est commutative).

É. Lozes et D. Villard étudient dans [84] une logique pour le π-calcul appliqué [1], une version du π-calcul où les processus s'échangent des *termes*. Un processus est composé d'une part de sa partie active (le 'code'), et d'autre part d'un environnement consistant en un ensemble de substitutions actives, qui associent des termes à des noms de variables. La logique spatiale de [84] comporte deux couples d'opérateurs de conjonction et implication spatiale, pour inspecter et analyser chacune des composantes, le processus et l'environnement.

Les idées sous-tendant les logiques spatiales ont filtré vers les systèmes de types, suscitant l'introduction de systèmes de *types spatiaux*. C'est le cas notamment dans l'article [19], qui présente un calcul d'objets distribués muni d'un systèmes de types permettant de contrôler l'accès aux ressources : l'opérateur de conjonction spatiale (défini sur les types) exprime la non interférence entre accès aux ressources.

Enfin, on peut mentionner l'article [18] de L. Caires, qui présente un survol des logiques spatiales pour la concurrence, ainsi que la thèse d'Étienne Lozes [82], où sont présentés de nombreux résultats sur ces logiques (dont une partie est exposée à la partie 4.2).

3.2 Logiques spatiales pour la concurrence : expressivité, pouvoir séparant

3.2.1 $\mathcal{L}_{\mathrm{G}}$, une logique spatiale générique [65] – capacités

L'article [65] analyse le pouvoir expressif de $\mathcal{L}_{\mathrm{G}}$, un langage que l'on peut voir comme le 'cœur' des logiques spatiales pour la concurrence (logique des Ambients, logiques spatiales pour le π-calcul).

Les formules de la logique $\mathcal{L}_{\mathrm{G}}$ sont décrites par la grammaire suivante :

$$\mathcal{A} \quad ::= \quad \mathcal{A}_1 \wedge \mathcal{A}_2 \mid \neg\mathcal{A} \mid \Diamond\mathcal{A} \mid \mathcal{N}n.\mathcal{A} \mid 0 \mid \mathcal{A}_1|\mathcal{A}_2 \mid n®\mathcal{A} \mid \mathcal{A}_1 \triangleright \mathcal{A}_2 \ .$$

Le constructeur $\Diamond$ est utilisé pour les observations temporelles. Le quantificateur frais est également inclus dans la logique. Les observations spatiales dont on dispose sont restreintes, puisque l'on ne présuppose l'existence que de la composition parallèle et de la restriction. En ce sens, $\mathcal{L}_{\mathrm{G}}$ représente le socle commun à plusieurs logiques ; ses formules peuvent en effet être interprétées par des ensembles de processus du π-calcul (synchrone ou asynchrone), ou de CCS, mais aussi par des Mobile Ambients. Ce qui différencie les modèles sur lesquels on peut raisonner en utilisant cette logique, c'est la forme d'interaction : communication synchrone ou asynchrone en π-calcul, mouvement pour les Ambients.

À noter que l'adjoint de l'opérateur de révélation ($\oslash$) n'est pas inclus dans $\mathcal{L}_{\mathrm{G}}$, car il n'est pas nécessaire pour obtenir les résultats exposés dans l'article.

La principale contribution de [65] est d'établir que sur différents modèles (π-calcul, π-calcul asynchrone, Mobile Ambients), certaines 'briques élémentaires d'interaction' peuvent être exprimées à l'aide des primitives d'inspection fournies par $\mathcal{L}_{\mathrm{G}}$, ce que l'on peut résumer par le slogan *"spatial implique comportemental"*. Puisque par ailleurs la logique $\mathcal{L}_{\mathrm{G}}$ permet d'inspecter les processus, et en particulier de compter (cf. section 3.1.2, partie "le néant"), on peut être plus précis que des modalités à la Hennessy-Milner : $\mathcal{L}_{\mathrm{G}}$ permet de dériver des formules caractérisant les *capacités* propres à un calcul donné — en l'occurrence, les préfixes qui servent à programmer l'interaction.

L'article [65] établit les résultats suivants pour le π-calcul dans ses versions synchrone et asynchrone (auxquelles correspondent les relations de satisfaction $\vDash_{\mathrm{S}}$ et $\vDash_{\mathrm{A}}$, respectivement) :

Théorème 18 ([65], Thm. 1, π-calcul synchrone) *Pour toute formule $\mathcal{A}$ de $\mathcal{L}_{\mathrm{G}}$ et pour tous noms m, n, avec $m \neq n$, il existe une formule $\mathbf{in}_{\mathrm{S}}(m,n).\mathcal{A}$ telle que $P \vDash_{\mathrm{S}} \mathbf{in}_{\mathrm{S}}(m,n).\mathcal{A}$ ssi $P \equiv m(n').P'$ pour un certain P' et un certain $n' \notin \mathrm{fn}(\mathcal{A})$, avec $P' \vDash_{\mathrm{S}} \mathcal{A}[n'/n]$.*

Pour toute formule $\mathcal{A}$ de $\mathcal{L}_{\mathrm{G}}$ et pour tous noms m, n, il existe une formule $\mathbf{out}_{\mathrm{S}}(m,n).\mathcal{A}$ telle que $P \vDash_{\mathrm{S}} \mathbf{out}_{\mathrm{S}}(m,n).\mathcal{A}$ ssi $P \equiv \overline{m}\langle n\rangle.P'$ pour un certain P', avec $P' \vDash_{\mathrm{S}} \mathcal{A}$.

Théorème 19 ([65], Thm. 3, π-calcul asynchrone) *Pour toute formule $\mathcal{A}$ de $\mathcal{L}_{\mathrm{G}}$ et pour tous noms m, n, avec $m \neq n$, il existe une formule $\mathbf{in}_{\mathrm{A}}(m,n).\mathcal{A}$ telle que $P \vDash_{\mathrm{A}} \mathbf{in}_{\mathrm{A}}(m,n).\mathcal{A}$ ssi $P \equiv m(n').P'$ pour un certain P' et un certain $n' \notin \mathrm{fn}(\mathcal{A})$, avec $P' \vDash_{\mathrm{A}} \mathcal{A}[n'/n]$.*

Pour tous noms m, n, il existe une formule $\mathbf{out}_{\mathrm{A}}(m,n)$ telle que $P \vDash_{\mathrm{A}} \mathbf{out}_{\mathrm{A}}(m,n)$ ssi $P \equiv \overline{m}\langle n\rangle$.

Il faut remarquer que bien que les π-calculs synchrone et asynchrone soient assez proches (le second s'obtient à partir du premier en imposant la continuation $\mathbf{0}$ après les émissions), les démonstrations des théorèmes 18 et 19 sont indépendantes, puisque l'interprétation d'une formule est fortement liée au modèle sous-jacent. Ceci vaut particulièrement pour les formules qui comportent $\triangleright$, puisque le pouvoir discriminant des contextes change suivant le calcul. Les preuves des théorèmes 18 et 19 sont présentées succintement ci-dessous, à la partie 3.2.2.

Les résultats énoncés ci-dessus sont néanmoins relativement robustes, dans la mesure où l'on peut les adapter (à travers des preuves plus complexes) au cas *faible* (théorèmes 2 et 4 de [65]). On peut également établir des résultats analogues en appliquant la logique $\mathcal{L}_{\mathrm{G}}$ au calcul des Mobile Ambients, modulo l'ajout d'une construction supplémentaire, l'adjoint de la localité (@, cf. section 4.2 — il est fait référence ci-dessous aux capacités du calcul des Ambients, qui sont également introduites plus loin ; ici $\vDash_{\mathrm{M}}$ désigne la relation de satisfaction pour les Mobile Ambients).

Théorème 20 ([65], Thm. 5, Mobile Ambients) *Pour toute formule $\mathcal{A}$, et pour tout nom n :*

- *il existe une formule $\mathbf{amb}(n).\mathcal{A}$ telle que $P \vDash_{\mathrm{M}} \mathbf{amb}(n).\mathcal{A}$ ssi $P \equiv n[P']$ pour un certain P', avec $P' \vDash_{\mathrm{M}} \mathcal{A}$;*
- *il existe une formule $\mathbf{open}(n).\mathcal{A}$ telle que $P \vDash_{\mathrm{M}} \mathbf{open}(n).\mathcal{A}$ ssi $P \equiv \mathsf{open}\, n.P'$ pour un certain P', avec $P' \vDash_{\mathrm{M}} \mathcal{A}$;*
- *il existe une formule $\mathbf{in}(n).\mathcal{A}$ telle que $P \vDash_{\mathrm{M}} \mathbf{in}(n).\mathcal{A}$ ssi $P \equiv \mathsf{in}\, n.P'$ pour un certain P', avec $P' \vDash_{\mathrm{M}} \mathcal{A}$;*
- *il existe une formule $\mathbf{out}(n).\mathcal{A}$ telle que $P \vDash_{\mathrm{M}} \mathbf{out}(n).\mathcal{A}$ ssi $P \equiv \mathsf{out}\, n.P'$ pour un certain P', avec $P' \vDash_{\mathrm{M}} \mathcal{A}$.*

3.2.2 Raisonnements spatiaux – construire des schémas d'expérience

L'analyse des logiques spatiales pour la concurrence telle qu'elle a été menée dans [65, 61] met en évidence des propriétés d'expressivité de ces logiques. En décrivant la méthode qui sous-tend les preuves de ces résultats (méthode que l'on retrouve d'ailleurs à l'œuvre plus loin, à la partie 4.2), on peut illustrer cette expressivité, et donner une idée du rôle des diverses constructions propres à ces logiques. Tout n'est pas explicité dans les preuves que l'on décrit ci-dessous, et certains aspects sont simplifiés pour faciliter l'exposé. On se réfèrera à [65, 61] pour davantage de détails.

On commence par s'intéresser à la construction de la formule $\mathbf{in}_{\mathrm{S}}(m,n).\mathcal{A}$, dont il est question au théorème 18. Celle-ci est donnée par

$$\mathbf{in}_{\mathrm{S}}(m,n).\mathcal{A} \quad \stackrel{\text{def}}{=} \quad \mathbf{single} \ \wedge \ \mathit{\Nu} n.\,(\mathbf{test}(m,n) \ \blacktriangleright \Diamond\mathcal{A})$$

La sous-formule **single** exprime le fait que le processus observé commence par un préfixe : son rôle est de restreindre l'espace des possibles. La seconde sous-formule est plus intéressante. La formule $\mathbf{test}(m,n)$ est satisfaite par exactement deux processus, $\overline{m}\langle n\rangle$ et $\overline{n}\langle m\rangle$ (on omet à chaque fois la continuation $\mathbf{0}$; l'interprétation de **single** et $\mathbf{test}(m,n)$ se fait bien entendu à $\equiv$ près, par définition de $\models_{\mathrm{S}}$). Le processus observé est mis en présence (construction $\blacktriangleright$) d'un processus de cette forme, où de plus le nom n est frais (quantification $\mathit{\Nu} n$). On demande que le système résultant évolue en un pas (construction $\Diamond$) vers un processus satisfaisant $\mathcal{A}$.

Pour que le scénario ainsi décrit puisse se concrétiser, il faut nécessairement que le processus initialement observé commence par une réception, car il doit pouvoir réagir en présence d'une simple émission. De plus, le processus qu'on lui adjoint en parallèle est de la forme $\overline{m}\langle n\rangle$: puisque n est frais, l'interaction avec $\overline{n}\langle m\rangle$ est impossible. Lorsque l'interaction entre le processus inital et $\overline{m}\langle n\rangle$ a eu lieu, on est passé à la continuation du processus initial, à laquelle il est demandé de satisfaire $\mathcal{A}$, comme spécifié au théorème 18.

Reste à comprendre comment sont définies les formules **single** et $\mathbf{test}(m,n)$. Pour **single**, de simples propriétés spatiales suffisent : sans entrer dans le détail, on exprime que le processus ne contient pas de restriction non triviale (i.e., portant sur un nom ayant des occurrences libres dans le processus) en tête, qu'il ne peut être décomposé en composantes parallèles non triviales, et qu'il est distinct de $\mathbf{0}$. Seuls les processus préfixés satisfont ces propriétés.

Pour isoler les processus qui satisfont $\mathbf{test}(m,n)$ (rappelons qu'il s'agit de $\overline{n}\langle m\rangle$ et $\overline{m}\langle n\rangle$), on s'appuie sur un scenario du même acabit que ce qui a été exposé ci-dessus. On peut commencer par remarquer que la formule

$$\mathbf{single} \wedge (\mathbf{single} \blacktriangleright \Diamond 0)$$

est satisfaite par tous les processus qui comportent un seul préfixe ; de plus, le préfixe doit mettre en œuvre une interaction sur un nom qui n'est pas restreint, afin de rendre la réduction possible. C'est le cas notamment de $a(x).\mathbf{0}$, $\overline{m}\langle n\rangle.\mathbf{0}$ et $(\boldsymbol{\nu} n)\,\overline{m}\langle n\rangle.\mathbf{0}$. La formule $\mathbf{test}(m,n)$ s'appuie sur un raffinement de cette idée. Informellement, les processus qui satisfont cette formule sont astreints à avoir des occurrences libres de m et n, et à admettre l'existence d'un processus satisfaisant **single** en présence duquel ils peuvent se réduire en un pas en $\mathbf{0}$ (on se réfèrera à [65] pour les détails – en particulier, la construction de $\mathbf{test}(m,n)$ n'est valide que lorsque $n \neq m$).

L'approche que l'on vient d'évoquer est déclinée, à quelques variations près, pour établir les théorèmes de [65] cités plus haut.

Les formules pour $\mathcal{L}_{\mathrm{E}}$ [61].

La caractérisation du pouvoir séparant de la logique $\mathcal{L}_{\mathrm{E}}$ exposée dans [61] utilise une démarche similaire à ce qui vient d'être présenté. Il est à noter cependant que la logique $\mathcal{L}_{\mathrm{E}}$ est moins riche que $\mathcal{L}_{\mathrm{G}}$, ce qui rend la tâche a priori plus difficile. La question qui se pose dans $\mathcal{L}_{\mathrm{E}}$ est de définir des scenarii de test en disposant d'un pouvoir discriminant plus faible.

Comme il a été dit à la partie 2.1, la difficulté pour établir que $=_{\mathrm{E}}$, l'équivalence induite par $\mathcal{L}_{\mathrm{E}}$ sur π_0, coïncide avec $\sim$, réside dans la preuve de l'inclusion $=_{\mathrm{E}} \subseteq \sim$. Pour ce faire, on définit des formules caractérisant des modalités à la Hennessy-Milner dans $\mathcal{L}_{\mathrm{E}}$. Une première idée pour cela est de définir une formule $\langle\mu\rangle\mathcal{A}$ par

$$\overline{\mu} \triangleright \Diamond\mathcal{A} \ ,$$

à condition d'avoir pu définir une formule $\overline{\mu}$ caractérisant les processus qui offrent l'interaction complémentaire de μ. Cette proposition n'est pas satisfaisante, car une réduction peut très bien avoir lieu sans qu'il y ait interaction avec le processus qui offre $\overline{\mu}$, et du coup on ne teste pas nécessairement la possibilité d'interagir selon μ. En effet, contrairement au cas des formules pour les capacités, où l'on commence par isoler une composante insécable du processus observé, on observe ici le processus dans sa totalité.

Envisageons donc une définition plus complexe. Si l'on avait droit à la conjonction spatiale, on pourrait définir $\langle\mu\rangle\mathcal{A}$ comme suit :

$$\mathcal{N}\eta.\,\big(\overline{\mu}.\eta \triangleright \Diamond(\eta \mid \mathcal{A})\big) \ .$$

L'idée serait ici

- d'ajouter en parallèle un processus proposant l'interaction conjuguée de μ,
- de détecter le déclenchement de cette interaction à l'aide d'un marqueur, susceptible de faire une interaction 'inédite' η, puis
- d'identifier le marqueur après que la réaction a eu lieu en le séparant du reste du processus (auquel il est demandé de satisfaire la formule $\mathcal{A}$).

Comme plus haut, il nous faut pouvoir définir une formule caractérisant les processus qui font la séquence de transitions $\overline{\mu}.\eta$. Le marqueur sert alors à détecter que c'est *la bonne* interaction qui a eu lieu, et non une autre réaction, interne au processus observé.

Un scenario de ce type est utilisé pour définir la formule $\mathbf{out}_{\mathrm{S}}(m,n).\mathcal{A}$ du théorème 18. Dans $\mathcal{L}_{\mathrm{E}}$, toutefois, puisque la conjonction spatiale n'est pas présente, il faut trouver un moyen de marquer différemment le processus testeur que l'on ajoute en parallèle, de manière à s'assurer à la fois que la bonne réaction ait lieu, et que la continuation du processus observé satisfasse $\mathcal{A}$. Dans [61], on remplace l'observation statique rendue possible par la conjonction spatiale par des formules pour les barbelés. On définit $\langle\mu\rangle\mathcal{A}$ par

$$\mathcal{N}\eta.\,\overline{\mu}.\eta \triangleright \Diamond\big(\eta^{\downarrow} \wedge \Diamond(\neg\,\eta^{\downarrow} \wedge \mathcal{A})\big) \ .$$

La formule $\eta^{\downarrow}$ signifie que le processus observé offre l'interaction η *immédiatement* : c'est une observation statique, qui correspond à la notion de barbelé (définition 4). Le scénario ci-dessus impose au processus observé d'effectuer deux pas de réduction, lorsqu'il est mis en présence de $\overline{\mu}.\eta$: le premier pour révéler le barbelé η (ainsi on sait qu'une transition selon μ a été effectuée), le second pour éliminer le barbelé η (ainsi l'artefact utilisé pour le test ne 'pollue' plus l'observation du processus).

On a donc résolu le problème consistant à définir $\langle\mu\rangle\mathcal{A}$ sans utiliser la conjonction spatiale, à condition d'avoir une formule pour $\eta^{\downarrow}$. L'idée utilisée dans [61] pour détecter la possibilité d'interagir immédiatement suivant η, qui intervient aussi dans les travaux de L. Caires et É. Lozes [22, 23], est plutôt technique, et exploite de manière déterminante le fait que l'on travaille avec une sémantique forte pour $\Diamond$ (intuitivement, on bâtit un scénario où la consommation du barbelé annule la possibilité d'effectuer une séquence 'très longue' de réductions).

D'un point de vue technique, on peut remarquer que le fait de caractériser $=_{\mathrm{E}}$ en exploitant la possibilité qu'a un processus de *renoncer* à un certain comportement est un point de passage classique. On retrouve en particulier cette idée dans les analyses des équivalences barbelées (voir à ce sujet les remarques qui suivent le théorème 2.2.9 dans [113]).

Les éléments que l'on a esquissés ci-dessus sont utilisés pour construire de la sorte des formules de $\mathcal{L}_{\mathrm{E}}$ dont on montre qu'elles s'interprètent comme des modalités de Hennessy-Milner. Plus généralement, l'ensemble des constructions que l'on a décrites indiquent l'importance, dans les logiques spatiales pour la concurrence, du mécanisme d'observation des processus mis en œuvre à l'aide du couple d'opérateurs ($\triangleright$,$\Diamond$).

3.2.3 Intensionalité, extensionalité

En relation avec ce qui vient d'être présenté, on peut revenir sur le résultat principal de [61], qui a été introduit à la section 2.1. La logique $\mathcal{L}_{\mathrm{E}}$ s'obtient à partir de $\mathcal{L}_{\mathrm{G}}$ en supprimant les opérateurs d'inspection 0, $|$ et ®, et en ajoutant $\oslash$. Comme il a déjà été mentionné, $\oslash$ pourrait être inclus dans $\mathcal{L}_{\mathrm{G}}$ sans affecter les résultats de [65].

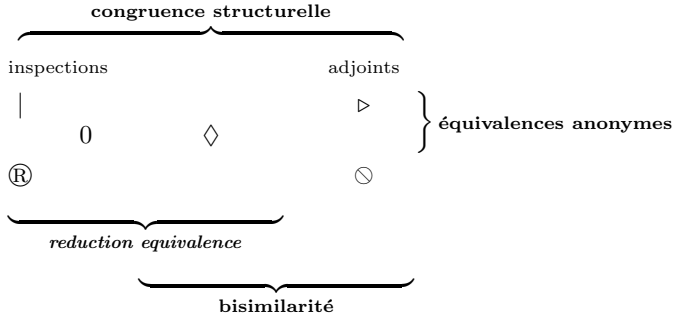
On a vu ci-dessus que $\mathcal{L}_{\mathrm{G}}$ permet d'exprimer les capacités ; on peut aussi y représenter les modalités à la Hennessy-Milner (du moins pour le π-calcul), ce qui a pour conséquence que la logique est au moins aussi discriminante que l'équivalence comportementale. S'agissant de $\mathcal{L}_{\mathrm{E}}$, le résultat de [61] montre que dans le cas du π-calcul synchrone, on obtient *exactement* le pouvoir discriminant associé aux modalités à la Hennessy-Milner en renonçant aux opérateurs d'inspection (puisqu'alors l'équivalence logique coïncide avec la bisimilarité). Ceci suggère de factoriser les opérateurs propres aux logiques spatiales en un fragment extensionnel, donné par les opérateurs adjoints $\triangleright$ et $\oslash$, et un fragment intensionnel, correspondant aux opérateurs d'inspection.

Le fragment purement intensionnel, correspondant à la logique $\mathcal{L}_{\mathrm{G}}$ privée de $\triangleright$, ne paraît pas très intéressant. L'équivalence induite est un raffinement de la *reduction bisimulation* de [93], qui correspond à la relation que l'on obtient en omettant la clause 2 dans la définition 4. Elle est intuitivement bien moins discriminante que la congruence barbelée, du fait de l'absence de clôture par contextes. Contrairement la *reduction bisimulation*, cette équivalence distingue $\mathbf{0}$ de $a.\mathbf{0}$ en CCS, mais, comme elle, elle ne distingue pas $a.P$ de $b.Q$, quels que soient P et Q.

Lorsque l'on ajoute les opérateurs d'inspection à $\mathcal{L}_{\mathrm{E}}$, cela donne $\mathcal{L}_{\mathrm{G}}$, et on obtient le pouvoir discriminant de $\equiv$, la congruence structurelle : c'est le résultat qui est prouvé pour la logique des Ambients dans [64] (cf. partie 4.2). Il semble fort probable que le même résultat soit valable pour la logique $\mathcal{L}_{\mathrm{G}}$ dans le cas des π-calculs synchrone et asynchrone.

Ainsi, les logiques spatiales sont intensionnelles, au sens où elles permettent d'observer la structure des processus de la manière la plus fine possible : puisque la satisfaction est définie modulo $\equiv$, on ne peut espérer avoir une équivalence induite plus discriminante que la congruence structurelle.

Le diagramme suivant rassemble les considérations qui ont été faites ci-dessus :



Rappelons que les logiques *anonymes* sont obtenues en éliminant les opérateurs qui mentionnent les noms (voir à ce sujet les commentaires à la fin de la partie 2.1).

Les logiques *statiques* sont obtenues en éliminant l'opérateur $\Diamond$. Il s'agit en particulier des logiques permettant de décrire des arbres, ou des graphes. É. Lozes établit dans sa thèse [82] des propriétés d'expressivité pour certaines logiques statiques (élimination des adjoints, notamment).

Remarquons pour finir que la distinction entre opérateurs d'inspection, qui apportent le caractère intensionnel de la logique, et adjoints, dont le rôle est de rendre possible des observations dont le caractère est perçu comme étant extensionnel, n'est toutefois pas toujours si nette. Elle est remise en cause dès lors que l'équivalence comportementale est sensible à des propriétés que l'on peut constater en inspectant les processus. Afin d'illustrer cela, Caires et Vieira étudient dans [25] une extension de CCS avec des localités explicites (mais anonymes). Dans ce formalisme, les processus peuvent migrer entre sites, et les localités peuvent être

sujettes à des pannes. Le résultat principal de cet article est une caractérisation de l'équivalence observationnelle à l'aide d'une logique spatiale. Alors que l'opérateur de conjonction spatiale est présent dans la logique, l'équivalence comportementale est définie sans qu'il soit fait référence à la décomposition des processus observés. La coïncidence entre les deux équivalences dit donc que les contextes ont un pouvoir d'observation suffisamment grand pour pouvoir déceler la structure des processus. En ce sens, les inspections permises par la conjonction spatiale sont, dans ce cas précis, extensionnelles.

3.2.4 Bibliographie : au sujet de | et ▷

Nous présentons ici quelques travaux, antérieurs à ceux que l'on a mentionnés plus haut, où figurent des idées que l'on retrouve à l'œuvre dans les logiques spatiales. La filiation n'est pas immédiate, au point que certains de ces travaux ne sont pas cités dans les articles sur les logiques spatiales. Il apparaît néanmoins que, de manière assez naturelle, certaines idées mises en avant dans ces logiques ont déjà fait l'objet d'études antérieures.

On ne s'attarde pas ici sur les logiques relevantes, et BI, la logique des *"bunched implications"* [99]. Ces logiques ont sous-tendu le développement de la logique de séparation, dont il a été question plus haut. L'article [107] représente un point de convergence dans l'évolution de la logique BI vers les langages d'assertions pour raisonner sur les pointeurs. Tout en étant porteuses d'idées similaires, il semble que les logiques pour la mémoire et les logiques pour les processus se soient développées de manière assez indépendante. Nous approfondissons ici les origines du côté des systèmes concurrents (nous avons évoqué plus haut les points de ralliement potentiels entre les deux familles de logiques spatiales).

L'idée commune aux approches que nous présentons est la *compositionalité* : on souhaite pouvoir raisonner de manière modulaire sur les systèmes parallèles, et pouvoir mettre ensemble des raisonnements que l'on a tenus sur des éléments distincts d'un système.

Les spécifications *assume-guarantee* [2].

M. Abadi et G. Plotkin s'intéressent dans [2] à la vérification de propriétés de sûreté, en se plaçant dans un cadre abstrait où les systèmes sont décrits par des ensembles de transitions entre états. L'objectif de ce travail est de voir les principes importants dans l'approche *assume-guarantee* [75] que sont la composition et le raffinement de spécifications sous un angle logique.

Les formules dénotent des ensembles de *comportements* ou de *processus* : un processus est un ensemble de séquences de transitions $s \to s'$, où s et s' font référence à un état global du système ; un comportement requiert de plus que les transitions s'enchaînent : l'état initial d'une transition termine la transition précédente ; de plus, un nom d'agent décore chaque transition dans le cas d'un comportement. Des propriétés supplémentaires sont imposées sur les dénotations des formules, et l'article [2] s'attache à étudier les opérations entre spécifications que l'on peut faire émerger dans ce cadre.

Dans le cas des comportements, on a affaire à une logique intuitionniste, qui valide un principe de composition de la forme

$$(M_2 \to M_1) \wedge (M_1 \to M_2) \vdash M_1 \wedge M_2 \ ,$$

à condition que les formules M_1 et M_2 fassent référence à des ensembles d'agents μ_1 et μ_2 disjoints (en un sens qui est défini formellement). Les auteurs signalent que l'on souhaite typiquement raisonner avec seulement deux agents, le système qui est observé et son environnement.

La propriété de séparation que l'on décèle ainsi apparaît plutôt comme un 'effet de bord' de l'étude. Dans ce cadre, d'ailleurs, la conjonction ne permet pas de séparer les processus : la composition parallèle de deux processus satisfaisant M_1 et M_2 respectivement satisfait $M_1 \wedge M_2$, mais un processus satisfaisant ces deux propriétés à la fois satisfait également $M_1 \wedge M_2$.

En plus des connecteurs intuitionnistes, on peut également exprimer des connecteurs linéaires dans le cas des processus. On définit sur les processus une opération de composition parallèle, notée $||$, qui entremêle les transitions de ses arguments, et qui va de pair avec la conjonction multiplicative, notée $\otimes$: un processus

satisfait $M_1 \otimes M_2$ si et seulement s'il résulte de la composition parallèle de deux processus satisfaisant M_1 et M_2 respectivement. À noter qu'on ne peut pas vraiment associer de connotation spatiale à $\otimes$ dans ce cadre.

L'implication associée, notée $\multimap$, vérifie

$$M_1 \multimap M_2 \;=\; \{u \mid (\{u\}||M_1) \subseteq M_2\} \ .$$

Les auteurs remarquent cependant que ce connecteur ne se prête pas bien à l'expression de conditions de type *assume-guarantee*, car ces dernières portent sur un système dans sa totalité, alors que la propriété ci-dessus permet un raisonnement localisé, où l'environnement n'est pas complètement spécifié : vu des logiques spatiales, ce point de vue est singulier, le fait de raisonner localement étant plutôt considéré comme avantageux. Ceci amène Abadi et Plotkin à introduire une forme dérivée d'implication, notée $M_1 \multimap\!\diamond M_2$, où l'on impose que le processus satisfaisant M_1 'complète' le processus observé, donnant lieu via la composition à un processus dont l'exécution se déroule sans intervention de l'environnement. Des principes de composition peuvent alors être dérivés.

Dans l'ensemble, on voit que l'idée d'associer une construction logique à la composition parallèle des processus est présente dans [2], en lien avec les interprétations logiques des processus et des comportements. Cependant, contrairement à l'approche des logiques spatiales, la décomposition des processus se fait suivant une approche guidée par le comportement des processus plutôt que par leur structure.

Calculs de processus, logiques modales — associer une formule à la composition parallèle.

La question de la compositionalité a également été étudiée dans la tradition de calculs de processus comme CCS et de la logique de Hennessy-Milner, dans un contexte où l'on se fonde sur la syntaxe d'un calcul de processus pour décrire la manière dont les systèmes observés sont constitués.

Axiomatiser la relation de satisfaction [116].

Dans [116], C. Stirling étudie l'axiomatisation de logiques modales issues de la logique de Hennessy-Milner. En particulier, au-delà de l'axiomatisation de la validité, qui est établie pour un LTS quelconque, on s'intéresse à l'axiomatisation de la relation de satisfaction en tirant parti de la structure des processus. On souhaite, autrement dit, fournir des règles d'inférence avec lesquelles dériver des jugements de la forme $P \vDash \mathcal{A}$, où P est un processus et $\mathcal{A}$ une formule — les cas de CCS et de sa version synchrone, SCCS, sont traités dans cette étude.

Dès l'introduction de l'article, l'auteur signale

> *"there is little hope of finding a uniform binary function $*$ on arbitrary formulas which will licence a rule of the form*
>
> $$\frac{P \vDash \mathcal{A} \qquad Q \vDash \mathcal{B}}{P|Q \vDash \mathcal{A} * \mathcal{B}} \quad "$$

(on adapte très légèrement les notations de [116] pour privilégier la cohérence avec le reste de l'exposé), et fait le lien entre cette difficulté et la complexité inhérente au raisonnement compositionnel dans un cadre concurrent.

La solution qu'apporte [116], pour le cas de CCS, adopte la règle d'inférence suivante

$$\frac{P \vDash \mathcal{A} \qquad \mathcal{A}, \mathcal{B} \vdash \mathcal{C} \qquad Q \vDash \mathcal{B}}{P|Q \vDash \mathcal{C}} \ ,$$

où $\mathcal{A}, \mathcal{B} \vdash \mathcal{C}$ fait référence à son tour à l'axiomatisation d'une relation ternaire entre formules, notée $\mathcal{A}, \mathcal{B} \vDash \mathcal{C}$, et définie par

$$\mathcal{A}, \mathcal{B} \vDash \mathcal{C} \text{ si et seulement si, pour tous } P, Q \text{ tels que } P \vDash \mathcal{A} \text{ et } Q \vDash \mathcal{B}, \text{ on a } P|Q \vDash \mathcal{C}.$$

On est en face d'une forme de 'séparation modulo $\vdash$' : la séparation s'appuie sur la dérivation d'un jugement, plutôt que de se faire à l'aide d'un opérateur.

À noter que l'opérateur de restriction est également pris en compte dans [116] : l'axiomatisation mentionnée ci-dessus exploite en réalité une version *relativisée* des relations susmentionnées : si c est un nom, $P \vDash_c \mathcal{A}$ désigne ce que l'on noterait $P \vDash \mathcal{A} \oslash c$, et l'on s'intéresse à l'axiomatisation de la relation $\vDash_{\widetilde{c}}$, où $\widetilde{c}$ désigne un ensemble de noms.

Mêler observations statiques et dynamiques [46].

L'article [46] fonde son approche sur la volonté d'utiliser les connecteurs des logiques linéaire et relevante pour rendre compte de la structure des processus. On s'appuie pour ce faire sur une utilisation de la composition parallèle qui permet d'interprêter l'implication comme on le fait pour $\triangleright$ dans les logiques spatiales :

$$P \vDash \mathcal{A} \rightsquigarrow \mathcal{B} \text{ si et seulement si pour tout } Q \text{ tel que } Q \vDash \mathcal{A},\ P|Q \vDash \mathcal{B}.$$

Ici $\rightsquigarrow$ désigne l'implication. M. Dam s'appuie sur la propriété ($\vdash A \rightsquigarrow \mathcal{B} \rightsquigarrow \mathcal{C}$ ssi $\vdash \mathcal{A} \circ \mathcal{B} \rightsquigarrow \mathcal{C}$), pour associer une interprétation à l'opérateur de conjonction correspondant à $\rightsquigarrow$, noté $\circ$. Cette approche est appliquée à des calculs de processus, en ajoutant à ces opérateurs structurels des connecteurs (en particulier, des modalités à la Hennessy-Milner) pour analyser les évolutions des processus.

Alors que les ingrédients essentiels des logiques spatiales telles que nous les avons présentées semblent tous réunis, le cadre présenté dans [46] s'en éloigne. La raison essentielle est la volonté de l'auteur de retomber sur une logique extensionnelle, c'est-à-dire pour laquelle l'équivalence logique coïncide avec la bisimilarité. Pour ce faire, la satisfaction d'une formule construite à l'aide de la conjonction spatiale est définie *modulo bisimilarité* : $P \vDash \mathcal{A}_1 \circ \mathcal{A}_2$ si et seulement si $P \sim P_1|P_2$ avec $P_i \vDash \mathcal{A}_i$, pour $i = 1, 2$. On renonce ainsi à l'inspection directe de la structure d'un terme.

Un autre aspect propre à [46] est le fait que l'on étudie SCCS, une version synchrone de CCS. Ce choix convient bien à l'approche, dans la mesure où l'interprétation qui est faite de l'opérateur de composition parallèle se marie bien avec une évolution synchrone des processus. L'extension à CCS est mentionnée comme piste pour les travaux futurs dans [46].

Les travaux que l'on vient d'évoquer ont en commun de s'intéresser à la décomposition des comportements, en s'appuyant éventuellement sur la composition parallèle pour ce faire. Le pas franchi par les logiques spatiales est d''oser' décomposer la structure des processus, en donnant une connotation spatiale à cette décomposition — autrement dit, pour reprendre la terminologie employée dans l'introduction de ce mémoire, on passe d'observation à inspection.

4 Calculs avec localités

4.1 Interaction localisée, mouvement

Des sémantiques localisantes aux syntaxes localisées. Plusieurs approches ont été développées durant les années 80 pour rendre compte, dans la sémantique opérationnelle des processus, de phénomènes de localisation ; [38] expose un panorama de l'ensemble de ces travaux. On a évoqué, dans ce cadre, la bisimulation distribuée de [39] à la partie 2.2. Cette vision mène généralement à des sémantiques plus fines que les présentations habituelles, car non entrelaçantes : dans le cas de CCS, par exemple, on distingue les processus $a \,|\, a$ et $a.a$, car le premier offre davantage de parallélisme que le second. D'autres outils permettent de rendre compte de phénomènes de causalité dans les modèles du parallélisme. C'est le cas notamment des *structures d'événements* [97], qui ont récemment été utilisées pour analyser le π-calcul [43]. Les calculs de processus réversibles, développés par J. Krivine et V. Danos [47], rejoignent également ce point de vue, dans la mesure où l'analyse de la reversibilité permet de rendre manifeste la causalité entre actions. On peut d'ailleurs constater, comme le met en avant [124], que la pertinence des sémantiques non entrelaçantes s'est vue renouvelée récemment, en liaison avec différents domaines d'application (le calcul réparti, mais aussi les

modèles pour la biologie, l'analyse des transactions en réseau, et le raisonnement sur des modèles de mémoire faiblement cohérents [13]).

Les équivalences telles que la bisimulation distribuée ont été développées autour de CCS. Elles permettent surtout d'expliquer comment associer une sémantique localisée à des processus 'usuels', ou, en d'autres termes, comment une spécification concurrente peut s'implanter dans un contexte distribué.

Avec le développement de l'étude du π-calcul dans les années 90, des calculs de processus dérivés de manière plus ou moins directe de ce formalisme ont vu le jour, qui adoptent un traitement explicite de la distribution. Pour ce faire, on dispose d'une notion de localité, et l'on note le plus souvent $a[P]$ un terme représentant le processus P qui s'exécute en la localité a (suivant les cas, on utilise deux ensembles distincts de noms pour les localités et pour les canaux, ou on ne fait pas de distinction). Ajouter $a[P]$ à la grammaire des processus du π-calcul (ou d'un calcul dérivé, comme le π-calcul asynchrone pour $D\pi$ [58], ou le join calcul pour la version distribuée de join [53]) permet d'écrire des termes tels que

$$a[P] \mid b[Q] \ ,$$

où l'on peut voir P et Q comme des processus s'exécutant en des localités a et b, potentiellement distantes. On peut aussi, dans certains formalismes, autoriser les localités à contenir des sous-localités. Ainsi, dans

$$a\big[\, b[P] \mid c[Q] \mid R \,\big] \ \mid \ d\big[T\big] \ ,$$

b et c sont contenues dans a. À noter que rien n'impose, lorsque l'on écrit ce terme, que les processus P et Q s'exécutent physiquement sur la même machine. On peut penser cependant que P et Q partagent la ressource R, offerte par a. De son côté, T tourne ailleurs, dans d.

Interprétations de la localité. On peut lire dans le développement de ces syntaxes localisées, qui a eu lieu vers la fin des années 90, l'accroissement de l'attention portée aux phénomènes de distribution et de mobilité, et, plus généralement, le souci de décrire un calcul qui se déploie sur des sites distants, ou entre agents situés dans des espaces distincts (pour des raisons liées à la distance physique, à la sécurité, etc.).

Comme illustré par exemple dans [30], la construction de localité peut être employée pour interpréter de nombreux objets que l'on trouve dans les développements de l'informatique à grande échelle et en réseau : machine distante, sous-réseau, agent mobile, etc. Une autre interprétation naturelle est celle d'entité de calcul (typiquement, un module) dans un langage pour la programmation distribuée — on revient sur ce point à la partie 4.3.

Structurer et nommer l'espace, pour agir dessus. L'ajout de localités explicites dans le calcul a pour effet de structurer l'espace entre les processus. En π-calcul, sans localités, c'est la connectivité qui détermine les potentialités d'interaction, au sens où si deux sous-termes connaissent un canal c, on peut les estimer reliés via c, et proches en ce sens. En revanche, l'espace qui sépare les processus est morne, sans structure : plusieurs processus préfixés peuvent être mis en parallèle, sans que cela dise si certains sont plus à même d'interagir que d'autres. Remarquons qu'il est possible d'attribuer à l'opérateur de restriction une connotation 'spatiale', dans la mesure où $\boldsymbol{\nu}$ détermine un dedans et un dehors, via sa portée. Mais l'extrusion de noms, qui permet à la portée d'évoluer (tout en respectant la liaison), rend cette notion de frontière volatile.

Au contraire, la construction $a[P]$ est spatiale, et définit une frontière : elle distingue ce qui est l'espace propre à P de ce qui lui est barbare. Ainsi, comme en π-calcul on nomme les canaux, la présence de localités permet de nommer l'espace, et d'exercer une forme de contrôle sur les communications. On se donne ainsi la possibilité d'introduire des mécanismes de haut niveau, que l'on peut voir comme des structures de contrôle pour manipuler l'espace, et qui ont pour rôle d'agir sur la manière dont l'interaction a lieu, en fonction de la façon dont les localités sont organisées : on 'programme la distribution'.

Parmi les nombreux travaux sur les calculs de processus qui empruntent cette voie, [94, 5, 108] jouent un rôle précurseur. En particulier, le calcul $D\pi$ [58], développé par M. Hennessy et ses collaborateurs, propose

une combinaison assez simple de primitives. En $D\pi$, la communication de noms est *locale* : la synchronisation sur c ne peut avoir lieu dans la configuration

$$m[\overline{c}\langle v\rangle.P] \mid n[c(x).Q] \ ,$$

car émetteur et récepteur sont situés dans des espaces distincts. Une instruction de migration `goto` peut être utilisée par un processus pour se déplacer vers une localité donnée, suivant la règle de réduction

$$m[\texttt{goto}\ n.P] \quad \longrightarrow \quad n[P]$$

(on 'triche' un peu ici avec $D\pi$, en adaptant la présentation au cadre de cet exposé – le point essentiel ici est d'illustrer des mécanismes pour l''interaction localisée', plutôt que de rendre compte exactement d'un formalisme en particulier). À noter que les termes $n[P]\,|\,n[Q]$ et $n[P|Q]$ sont équivalents en $D\pi$.

En $D\pi$, la mobilité se traduit syntaxiquement par le renommage d'une localité. C'est le cas également en Nomadic Pict [119], une extension du langage de programmation Pict [102], fondé sur le π-calcul. Parmi les calculs de processus avec localités explicites, ces deux formalismes ont fait l'objet des études les plus approfondies : la théorie de $D\pi$, présentée dans [58], est la plus fouillée (équivalences comportementales, systèmes de types) ; les projets menés à partir des versions distribuées de Join et de Pict (DJoin, Nomadic Pict, puis Acute) ont donné lieu aux implantations probablement les plus abouties.

Au-delà des calculs de processus que l'on vient de mentionner, une autre famille de formalismes se fonde sur une structure spatiale où les localités peuvent être imbriquées. Les Ambients [34], les Seals [37], Djoin [53] d'une part, Homer [60] et les Kells [10] d'autre part, permettent ainsi aux processus des manipulations plus puissantes des localités : migration au sein d'une hiérarchie de sites pour les premiers, passivation et duplication de processus actifs pour les seconds (voir à ce sujet les parties 4.2 et 4.3).

En particulier, le point de vue mis en avant dans les Mobile Ambients est d'une certaine manière le plus épuré, dans la mesure où il consiste à mettre la notion de mouvement au cœur même du formalisme, plutôt que d'adjoindre des primitives permettant de faire migrer des processus à un calcul fondé sur le passage de messages. C'est sur cette approche, où la dimension 'spatiale' du calcul est mise en avant de manière prépondérante, que nous nous penchons ci-dessous, dans le contexte des logiques spatiales pour la concurrence.

Mentionnons pour finir les bigraphes, introduits par R. Milner [89]. Les bigraphes sont un formalisme fondé sur des outils catégoriques, dont l'un des objectifs est de faire la synthèse de cette tour de Babel de calculs de processus que l'on vient d'évoquer. Comme son préfixe l'indique, ce modèle est constitué de deux composantes : une moitié du formalisme traite de la connectivité, telle qu'elle est à l'œuvre en π-calcul, et l'autre moitié s'articule autour de la notion d'inclusion spatiale (le pendant des localités structurées). Les bigraphes se veulent un cadre formel général, propre à la représentation de formes très diversifiées de systèmes interagissant à l'intérieur d'un espace structuré — il ne s'agit plus uniquement de programmation distribuée, mais aussi, potentiellement, d'humains qui interagissent, ou de cellules biologiques hébergeant des réactions chimiques.

4.2 Expressivité et pouvoir séparant de la logique des Ambients

Calcul des Mobile Ambients.

On s'intéresse à une version simplifiée du calcul des Mobile Ambients, décrite par la syntaxe suivante :

$$P \quad ::= \quad \mathbf{0} \;\big|\; P_1|P_2 \;\big|\; a[P] \;\big|\; !P \;\big|\; \mathsf{cap}.P \qquad\qquad \mathsf{cap} \quad ::= \quad \mathsf{in}\,a \;\big|\; \mathsf{out}\,a \;\big|\; \mathsf{open}\,a$$

Dans le calcul que nous présentons, les lois pour $\equiv$ sont les mêmes qu'en π-calcul (en particulier, il n'y a pas d'égalité faisant intervenir les processus préfixés — ni d'ailleurs les processus localisés, du fait de l'absence de restriction). La figure 4 donne les règles de réduction définissant la sémantique opérationnelle.

Grâce à la réplication, le calcul peut exprimer des comportements infinis – la partie 6 de [83] présente un encodage des machines de Turing dans un sous-calcul des Mobile Ambients.

$$a[\,\mathsf{in}\,b.P \mid Q\,] \mid b[R] \quad\longrightarrow\quad b[\,a[\,P \mid Q\,] \mid R\,] \qquad a[\,b[\,\mathsf{out}\,a.P \mid Q\,] \mid R\,] \quad\longrightarrow\quad a[R] \mid b[\,P \mid Q\,]$$

$$\mathsf{open}\,a.P \mid a[Q] \quad\longrightarrow\quad P \mid Q \qquad \frac{P \longrightarrow P'}{a[P] \longrightarrow a[P']} \qquad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$$

$$\frac{P_1 \equiv P \qquad P \longrightarrow P' \qquad P' \equiv P_1'}{P_1 \longrightarrow P_1'}$$

FIG. 4 – Mobile Ambients : réduction

Ni restriction, ni communication. Il est à noter que l'on s'appuie ici sur deux simplifications majeures du calcul présenté dans [34]. La première simplification est l'omission de l'opérateur de restriction : la logique des Ambients a vu le jour [35] pour un calcul *public*, sans cet opérateur. L'article [36] étudie comment définir des constructions logiques correspondant aux propriétés de l'opérateur de restriction. Dès lors que le calcul sous-jacent inclut les opérateurs de réplication et de restriction, l'étude de la logique se complique considérablement (nous revenons sur ce point plus bas).

La seconde simplification consiste à omettre les communications, qui en Mobile Ambients sont locales à un site. Dans

$$a[\,\langle m\rangle \mid (x)P\,] \quad | \quad b[\,Q \mid \langle m'\rangle\,]\ ,$$

le processus récepteur $(x)P$ est susceptible d'accepter le message $\langle m\rangle$, mais n'a pas accès au message $\langle m'\rangle$, qui est local à l'ambient b. Les communications locales sont prises en compte dans [64, 66, 83], et leur présence n'affecte pas les résultats qui sont discutés ci-dessous. On les omet pour simplifier la présentation.

Le calcul étudié se résume donc à un ensemble restreint de primitives, qui sont centrées sur l'idée de mouvement comme unique forme d'interaction. Il n'est pas clair qu'une simplification si drastique rende ce calcul crédible en tant que modèle formel pour la programmation répartie ou les échanges en réseau. C'est néanmoins un bon cadre pour étudier une forme dépouillée de 'raisonnement spatial', tel que la logique des Ambients le permet, en particulier parce que la configuration spatiale des processus définit la notion d'état en Ambients. Comme on le voit dans ce qui suit, les idées que l'on a exposées plus haut pour les logiques $\mathcal{L}_{\mathrm{E}}$ ou $\mathcal{L}_{\mathrm{G}}$ se retrouvent dans la logique spatiale pour les Ambients. Elles sous-tendent aussi l'introduction d'autres logiques pour raisonner sur des structures arborescentes, ou plus complexes, qui généralisent la logique des Ambients. C'est le cas, par exemple, du langage de requètes pour les documents XML de [32], de la logique des graphes de [31], ainsi que de la logique des contextes de [26, 27]

Logique des Ambients, $\mathcal{L}_{\mathrm{A}}$.

Pour analyser les processus des Mobile Ambients, et raisonner sur leurs propriétés, plusieurs approches ont été envisagées. Un certain nombre de systèmes de types ont été définis ; la plupart d'entre eux visent à garantir que certaines situations ne se présentent pas (tel ambient ne sera jamais ouvert, tel autre est immobile, etc.). La logique des Ambients permet d'exprimer un éventail plus large de spécifications, portant en particulier sur des questions de distribution ou de non-interférence. Des travaux postérieurs à l'introduction de cette logique, sur l'équivalence comportementale en Mobile Ambients, et sur sa caractérisation coinductive, ont montré que la théorie comportementale est relativement pauvre : peu de lois algébriques non triviales ont pu être montrées[5]. La raison intuitive de cet état de fait est que le contexte a trop de pouvoir pour observer les processus, et il est pratiquement impossible de 'programmer' un comportement donné en se mettant à l'abri des interférences dans le modèle originel des Mobile Ambients. Ceci peut être vu comme un élément 'à charge' s'agissant du calcul des Mobile Ambients : il est très difficile de garantir des propriétés sur le comportement des processus, du fait, essentiellement, du caractère fortement asynchrone de l'interaction.

[5] À l'exception d'une loi de *pare-feu parfait* : $(\boldsymbol{\nu} n)\, n[P]$ est équivalent à $\mathbf{0}$ si P ne mentionne pas le nom n.

En première approche tout du moins, cette constatation suggère que pour analyser les propriétés des Ambients, des observations partielles, telles que la logique les permet, sont plus pertinentes qu'un point de vue fondé sur l'équivalence comportementale. Comme il apparaît plus loin, cependant, la logique permet dans certains cas d'exprimer en une formule une classe d'équivalence pour l'équivalence induite.

La syntaxe de $\mathcal{L}_{\mathrm{A}}$, la logique des Ambients, telle qu'elle est introduite dans [35], est la suivante :

$$\mathcal{A} \quad ::= \quad \mathcal{A}_1 \wedge \mathcal{A}_2 \mid \neg\mathcal{A} \mid \Diamond\mathcal{A} \mid \forall n.\mathcal{A} \mid 0 \mid \mathcal{A}_1|\mathcal{A}_2 \mid n[\mathcal{A}] \mid \mathcal{A}_1 \triangleright \mathcal{A}_2 \mid \mathcal{A}@n \ .$$

On remarque la présence d'opérateurs de *localité* $n[\cdot]$ et de *situation* @, pour lesquels la relation de satisfaction est étendue comme suit :

$$P \vDash n[\mathcal{A}] \quad \text{ssi} \quad P \equiv n[P'] \text{ et } P' \vDash \mathcal{A} \quad \text{pour un certain } P' \ ;$$

$$P \vDash \mathcal{A}@n \quad \text{ssi} \quad n[P] \vDash \mathcal{A} \ .$$

L'opérateur de localité est un opérateur permettant d'inspecter la structure d'un Ambient ("la vérité d'une formule spatiale dépend de sa localisation", suivant la formule de L. Cardelli [29]). L'adjoint de cet opérateur est @, qui permet de *localiser* un processus pour l'observer.

Dans $\mathcal{L}_{\mathrm{A}}$, on adopte une interprétation *faible* de l'opérateur $\Diamond$:

$$P \vDash \Diamond\mathcal{A} \quad \text{ssi} \quad P \longrightarrow^* \vDash \mathcal{A} \ .$$

Ce choix a été fait dès l'article [35]. Il s'avère, comme il est montré ci-dessous, qu'il n'empêche pas la logique d'inspecter les processus de manière très fine. Lorsque l'on passe des Ambients finis de [35] à un calcul avec réplication, où peuvent être programmés des comportements infinis, l'interprétation faible de $\Diamond$ est d'autant plus pertinente, car elle permet d'observer un futur arbitrairement lointain d'un processus.

Enfin, la quantification sur les noms est universelle, l'utilisation de la quantification fraîche (avec $\mathcal{N}$) ayant été adoptée dans des travaux postérieurs à [35]. La plupart des résultats que l'on obtient pour la logique $\mathcal{L}_{\mathrm{A}}$ sont valables en remplaçant $\forall$ par $\mathcal{N}$ (c'est notamment le cas pour les formules caractéristiques évoquées plus loin).

La suite de cette partie 4.2 est consacrée à un exposé des résultats sur la logique des Ambients. À l'image de ce qui a été présenté à la partie 3.2, ces résultats se déclinent en propriétés d'expressivité et de séparabilité. Dans leur ensemble, ces preuves suivent une approche similaire à ce qui a été présenté précédemment, bien que la logique qui est à l'œuvre, $\mathcal{L}_{\mathrm{A}}$, soit plus riche que $\mathcal{L}_{\mathrm{G}}$ ou $\mathcal{L}_{\mathrm{E}}$, et que les modèles sous-jacents soient différents (les Ambients remplacent le π-calcul).

4.2.1 Résultats d'expressivité [66]

L'étude de l'expressivité de $\mathcal{L}_{\mathrm{A}}$ met en évidence un certain nombre de spécificités par rapport à ce qui a été exposé pour $\mathcal{L}_{\mathrm{E}}$ et $\mathcal{L}_{\mathrm{G}}$. Du fait notamment de la possibilité de programmer des réductions infinies dans le calcul, et de l'interprétation faible du connecteur $\Diamond$ du côté des formules, on s'attend a priori à ce que l'analyse des processus soit moins fine en $\mathcal{L}_{\mathrm{A}}$ que dans les situations examinées plus haut. Tel n'est pas le cas, puisque la logique offre un regard précis sur les processus, comme on l'illustre dans ce qui suit.

La finitude.

Le caractère non fini du calcul étudié constitue une originalité par rapport au cadre envisagé à la partie 3.2. Il s'avère que $\mathcal{L}_{\mathrm{A}}$ permet de rendre compte de la (non-)finitude.

Un processus répliqué est en effet *persistant* : quels que soient P, Q, R, si $!P \,|\, Q \longrightarrow^* R$, alors $R \equiv !P \,|\, P'$ pour un certain P'. Cette propriété peut être mise à profit pour capturer la finitude des processus à l'aide de $\mathcal{L}_{\mathrm{A}}$: on dit que P est fini s'il existe P' n'ayant pas d'occurrence de ! dans son écriture syntaxique tel que $P \equiv P'$. La formule

$$\exists x.\, \big(\top \blacktriangleright \big(\top \blacktriangleright \Diamond 0\,\big)@\, x\big)$$

caractérise alors les processus finis. Suivant le scenario décrit par cette formule, P est fini si on peut exhiber deux testeurs Q et R, et un nom n tels que $Q \mid n[\,P|R\,]$ puisse se réduire à $\mathbf{0}$. Autrement dit, un processus fini est un processus que le contexte peut consommer complètement.

Il est à noter que cette formule ne peut pas être adaptée facilement lorsque la restriction est présente dans le calcul : intuitivement, la restriction permet de protéger certains processus vis-à-vis du contexte, qui ne peut les consommer. Il n'est a priori pas clair jusqu'à quel point les résultats essentiels de [66, 83] (formules caractéristiques, caractérisation du pouvoir séparant de la logique) tiendraient en présence des opérateurs de réplication et restriction dans le calcul — c'est essentiellement la possibilité d'engendrer une infinité de noms frais, autrement dit les occurrences de la restriction sous une réplication, qui pose problème.

Capacités : bégaiements.

On peut établir pour $\mathcal{L}_{\mathrm{A}}$ des formules analogues à celles que l'on a décrites à la partie 3.2.1 pour $\mathcal{L}_{\mathrm{G}}$, afin d'isoler les capacités du langage, ici interprétées comme les préfixes du calcul des Ambients (in, out et open). On observe alors que les formules que l'on peut définir pour les capacités de mouvement (entrée, sortie), tirées de [110], s'interprètent *à bégaiement près*. Ainsi, un processus qui satisfait la formule $\mathbf{in}(n).\mathcal{A}$, qui a pour rôle d'isoler la capacité d'entrée, est bien susceptible de faire entrer la localité où il se trouve dans n, mais celle-ci peut auparavant entrer et sortir de n un certain nombre de fois. Une interprétation similaire s'applique dans le cas de la capacité de sortie.

Cette perte de précision dans l'observation des capacités est due à l'interprétation faible de l'opérateur $\Diamond$: lorsque l'on bâtit un 'scénario d'expérience' du type de ceux qui ont été esquissés à la partie 3.2.1, on perd la faculté d'observer le processus *à l'instant suivant*.

"Concentrer l'expressivité" : formules caractéristiques.

Le pouvoir expressif de la logique $\mathcal{L}_{\mathrm{A}}$ est mis en évidence de manière frappante par la possibilité de définir des formules caractéristiques : pour tout processus P, il est possible de définir une formule qui capture la classe d'équivalence de P vis-à-vis de l'équivalence induite par la logique — on note $=_{\mathrm{L}}$ l'équivalence induite par $\mathcal{L}_{\mathrm{A}}$ sur le calcul des Mobile Ambients. Du fait de l'interprétation faible de $\Diamond$, ce résultat n'est valable que sur un sous-ensemble du calcul des Ambients, appelé $\mathrm{MA}_{\mathrm{IF}}$ dans [66].

La définition de $\mathrm{MA}_{\mathrm{IF}}$ repose sur un certain nombre de notations. On écrit $P \overset{\mathsf{cap}}{\to} P'$ lorsque $P \equiv \mathsf{cap}.P_1|P_2$ et $P' = P_1|P_2$, où cap est une capacité des Ambients (entrée, sortie, ouverture). Similairement, on écrit $P \overset{\mathsf{cap}}{\Longrightarrow} P'$ lorsque $P \longrightarrow^* \overset{\mathsf{cap}}{\to} \longrightarrow^* P'$. À tout terme de la forme $\mathsf{cap}.P'$ des Mobile Ambients, on associe un ensemble $E\langle \mathsf{cap}.P' \rangle \overset{\text{def}}{=} \{P''.\, P' \overset{\mathsf{cap}}{\Longrightarrow} P''\}$.

Un processus P est dit *à branchement fini* si, pour tout sous-terme de P de la forme $\mathsf{cap}.P'$, l'ensemble $E\langle \mathsf{cap}.P' \rangle$, quotienté par $=_{\mathrm{L}}$, est fini[6]. $\mathrm{MA}_{\mathrm{IF}}$ est alors défini comme l'ensemble des processus des Mobile Ambients à branchement fini.

Un exemple de processus qui n'est pas dans $\mathrm{MA}_{\mathrm{IF}}$ est donné par

$$T \quad \overset{\text{def}}{=} \quad \mathsf{in}\, n.!\big(\, n[\mathbf{0}] \mid \mathsf{open}\, n.a[\mathbf{0}] \,\big) \ .$$

T a en effet une transition (au sens de $\overset{\mathsf{in}\, n}{\Longrightarrow}$) vers tout processus de la forme $T \mid \underbrace{a[\mathbf{0}] \mid \ldots \mid a[\mathbf{0}]}_{k \text{ fois}}$, pour $k \geq 1$.

La définition de $\mathrm{MA}_{\mathrm{IF}}$ ne fournit pas une méthode permettant de savoir si un processus appartient à $\mathrm{MA}_{\mathrm{IF}}$. [83] propose un autre sous-calcul, dont la définition repose sur une restriction syntaxique, et qui ne contient que des processus à branchement fini.

Théorème 21 (Formules caractéristiques, Corollary 5.4 dans [66]) *Pour tout processus P de MA_{IF}, il existe une formule $\mathcal{F}_P$ telle que pour tout Q dans MA_{IF}, $(Q \vDash \mathcal{F}_P)$ ssi $(P =_{\mathrm{L}} Q)$.*

[6] En réalité, on quotiente dans [66] par une autre relation, la *bisimilarité intensionnelle*, pour définir cette notion. On montre que cette relation coïncide avec $=_{\mathrm{L}}$, l'équivalence logique, sur $\mathrm{MA}_{\mathrm{IF}}$.

Comme il a été dit plus haut, la preuve du théorème 21 repose sur des résultats d'expressivité de la logique similaires dans l'esprit aux constructions qui sont utilisées pour raisonner sur $\mathcal{L}_{\mathrm{E}}$ et $\mathcal{L}_{\mathrm{G}}$. Pour définir $\mathcal{F}_P$, on montre qu'il est possible d'associer une construction logique à chaque opérateur du calcul. Les opérateurs $\mathbf{0}$, $n[\cdot]$ et $|$ étant présents d'emblée dans la logique, il s'agit donc d'exprimer les capacités (préfixes), et le pendant de la réplication au niveau logique. Ce dernier aspect est le plus complexe, techniquement, la définition et l'interprétation d'une formule $!\mathcal{A}$ n'étant possible que sous certaines conditions sur $\mathcal{A}$. À noter que la contrainte de branchement fini apparaît dans la construction des formules pour les capacités, intuitivement parce qu'il faut pouvoir décrire l'ensemble des continuations possibles à l'aide d'une formule. On renvoie à [66] pour le détail des constructions.

Extension aux communications.
Comme il a été indiqué plus haut, le calcul originel des Mobile Ambients comprend des primitives permettant des communications à l'intérieur d'une localité. Il est possible, au prix de constructions du type de celles qui ont été évoquées à la partie 3.2, de définir des formules qui expriment les capacités d'émission et de réception. Comme c'est le cas pour les capacités de mouvement, on est confronté à un phénomène de bégaiement pour le préfixe de réception : la formule correspondante ne permet pas de distinguer un processus qui reçoit un message du processus qui reçoit, émet à nouveau, et ceci un certain nombre de fois, jusqu'à finalement recevoir ledit message.

4.2.2 Pouvoir séparant [83]

L'article [83] présente une étude de $=_{\mathrm{L}}$, l'équivalence induite par $\mathcal{L}_{\mathrm{A}}$ sur les Mobile Ambients. Cette relation est caractérisée par le biais d'une *bisimilarité intensionnelle*, dont la définition repose, en plus des transitions correspondant aux capacités du calcul (mouvement, ouverture, communications), sur des clauses imitant les observations structurelles permises par la logique. Cette approche pour l'analyse des logiques spatiales pour la concurrence a été introduite dans [110].

Plus précisément, pour qu'une relation $\mathcal{R}$ soit une bisimulation intensionnelle, il faut *en particulier* qu'elle satisfasse, dès que $P\mathcal{R}Q$, les propriétés suivantes :
- si $P \equiv P_1|P_2$, alors il existe Q_1, Q_2 tels que $Q \equiv Q_1|Q_2$ et $P_i\mathcal{R}Q_i$ pour $i = 1, 2$;
- si $P \equiv \mathbf{0}$, alors $Q \equiv \mathbf{0}$;
- si $P \equiv n[P']$, alors $Q \equiv n[Q']$ avec $P'\mathcal{R}Q'$.

La bisimilarité intensionnelle, notée $\simeq_{\mathrm{int}}$, est la plus grande bisimulation intensionnelle.

L'équivalence $\simeq_{\mathrm{int}}$ est correcte vis-à-vis de $=_{\mathrm{L}}$: ceci s'établit aisément, par induction sur la formule de $\mathcal{L}_{\mathrm{A}}$. La propriété réciproque, de complétude, est prouvée en s'appuyant sur une stratification de $\simeq_{\mathrm{int}}$. Dans le cas général, traité dans [64, 83], où l'on prend en compte l'opérateur de réplication, on ne peut s'appuyer sur la propriété de branchement fini, qui est usuellement nécessaire, et qui est vérifiée dans [110]. On s'appuie alors sur une caractérisation inductive de $\simeq_{\mathrm{int}}$, à partir de laquelle on peut établir l'existence de *formules caractéristiques locales* pour $\mathcal{L}_{\mathrm{A}}$: $\mathcal{A}$ est une formule caractéristique (locale) pour P sur un ensemble $\mathcal{E}$ de processus si $P \vDash \mathcal{A}$ et, pour tout $Q \in \mathcal{E}$, on a $P \simeq_{\mathrm{int}} Q$ dès que $Q \vDash \mathcal{A}$. Les propriétés d'expressivité de $\mathcal{L}_{\mathrm{A}}$ évoquées plus haut sont utilisées pour établir l'existence de telles formules. On établit alors la complétude de la bisimilarité intensionnelle en supposant $P \not\simeq_{\mathrm{int}} Q$ et en raisonnant sur une formule caractéristique locale pour un ensemble bien choisi contenant Q.

4.3 Espace avec ressources, la modularité

On expose ici des travaux portant sur l'utilisation de calculs de processus avec localités pour l'étude de langages de programmation distribuée. La hiérarchie arborescente des localités représente, dans cette approche, une organisation à la fois logicielle (modules et sous-modules) et matérielle (un module, ainsi éventuellement que ses sous-modules, peut être associé à une localité physique) du code. On analyse des mécanismes ayant trait à la manipulation des modules au moment de l'exécution ; plus précisément, on s'intéresse à l'opérateur de *passivation*, qui permet de figer l'exécution d'un processus localisé. Les travaux dont il est question ici

portent avant tout sur des questions liées à l'expressivité du formalisme et à l'implémentation distribuée d'un langage fondé sur un calcul de processus avec localités.

Modules et passivation. Dans les formalismes qui ont été évoqués plus haut, les capacités agissant sur les processus ont pour rôle de mettre en œuvre des formes diverses de mobilité.

Ainsi, comme on l'a vu, la mobilité (de code) en $D\pi$ se manifeste à travers le déplacement d'une tâche, par le biais d'une réduction de la forme

$$m[\texttt{goto}\ n.P] \mid m[Q] \ \longrightarrow\ n[P] \mid m[Q] \ ,$$

alors qu'en Ambients, c'est une localité entière qui se déplace :

$$m[\texttt{in}\ \ n.P \mid Q] \mid n[R] \ \longrightarrow\ n[R \mid m[P \mid Q]] \ .$$

Dans les deux cas, l'évolution fait intervenir une forme de linéarité, dans la mesure où le processus qui bouge n'est ni dupliqué, ni effacé. En outre, le code est actif immédiatement après le mouvement. La situation est légèrement différente dans le π-calcul d'ordre supérieur : rien n'empêche de manipuler les processus qui sont transmis de manière non linéaire, à ceci près que ces processus sont constitués de code figé, dont l'exécution n'a pas encore démarré.

Dans le contexte de l'étude de certains aspects de la programmation distribuée, nous nous sommes intéressés à la *passivation* des localités. Cette opération permet d'agir sur des processus localisés (vus comme des modules d'un langage de programmation) au cours de leur exécution.

La passivation, notée $n\{X\} \triangleright P$, est une opération locale : il faut être au même endroit qu'une localité n pour passiver n. L'étape de réduction correspondante s'écrit alors

$$n\{X\} \triangleright P \mid n[Q] \quad \longrightarrow \quad P[Q/X]$$

(ici $n[Q]$ est un processus localisé, alors que $P[Q/X]$ désigne le résultat de l'application d'une substitution à P).

Passiver une localité peut permettre d'en changer le nom

$$n\{X\} \triangleright m[X] \ ,$$

ou bien de dupliquer le code qui s'y exécute

$$n\{X\} \triangleright \big(n[X] \mid (\boldsymbol{\nu} p)\, (p[X] \mid \overline{c}\langle p \rangle) \big)$$

(on relance l'exécution en n, tout en renvoyant l'adresse de la nouvelle localité sur c), de changer la hiérarchie des modules dans le code

$$n\{X\} \triangleright m\{Y\} \triangleright n[\, X \mid m[Y]\,] \quad \text{ou} \quad n\{X\} \triangleright m\{Y\} \triangleright p[X \mid Y] \ ,$$

ou encore de transmettre le code du module, figé, dans un message :

$$n\{X\} \triangleright \overline{a}\langle X \rangle \ .$$

La passivation apparaît donc comme un mécanisme élémentaire expressif, qui permet de mettre en œuvre différentes formes de ce que l'on a appelé dans [62] la *modularité dynamique* : mettre à jour dynamiquement un module, dupliquer un module auquel plusieurs processus font référence, interrompre l'exécution d'un module pour le transmettre vers un autre espace d'exécution, faire disparaître un module. Ce type de mécanisme est volontiers à l'œuvre dans la *programmation à base de composants*, qui se trouve en arrière-plan des travaux dont il est question ici.

On constate sur les exemples ci-dessus que lorsqu'un module a été passivé, rien n'impose d'utiliser le résultat de la passivation de manière linéaire. Le langage Homer (pour *Higher-Order Mobile Embedded Resources*) contient un mécanisme semblable à la passivation, auquel il est fait référence dans [16] sous le nom de *'non-linear active process mobility'*. L'opérateur de passivation est également présent dans le langage des Kells [10, 63], qui a servi de point de départ aux travaux dont il est question ici.

Ajouter la passivation au π-calcul – implémentation, machines [62].

Nous avons introduit dans [62] (qui prolonge [63]) une extension du π-calcul avec des localités et l'opérateur de passivation. Dans ce calcul, nommé kπ, la communication sur les canaux se fait de manière transparente à travers les localités, mais en même temps l'extrusion de noms est limitée par les localités. Ainsi, en kπ, un processus localisé est vu comme un module, et les restrictions de noms sont interprétées comme des ressources hébergées par le module dans lequel elles ont été créées. Passiver un module signifie donc récupérer le contenu d'une localité, qui est constitué d'un processus (potentiellement composé de plusieurs tâches s'exécutant en parallèle) et d'un certain nombre de ressources. Par rapport au cadre des calculs évoqués à la partie 4.1, on enrichit ainsi la notion de localité : au-delà de la notion d'espace, on y rattache l'idée de ressource, et de limitation de l'accès à une ressource.

On se focalise dans [62] sur des questions liées à l'utilisation de la passivation comme primitive de programmation : comment celle-ci interagit avec les constructions habituelles des calculs de processus (localités, passage de messages, restriction), et dans quelle mesure elle peut être implémentée dans un contexte distribué.

Depuis l'introduction du π-calcul, un certain nombre de travaux on analysé la question de l'implémentation (distribuée ou non) des primitives des calculs de processus pour la mobilité. Ainsi, le passage de noms, tel qu'il est à l'œuvre en π-calcul, ou dans des formalismes apparentés, a fait l'objet d'études diverses – on peut mentionner le langage Pict [102], Join [51], les Fusions explicites [55], et les forwarders linéaires [56]. L'introduction du calcul des Mobile Ambients, avec ses primitives de haut niveau pour le mouvement, a également suscité des travaux visant à comprendre dans quelle mesure celles-ci peuvent être implémentées de manière raisonnablement efficace et naturelle. Des machines abstraites pour l'exécution distribuée des Ambients, ou de calculs dérivés, ont été proposées et analysées dans [57, 101, 70].

On donne dans [62] la définition d'une machine abstraite distribuée qui implémente un mécanisme très général ayant pour but d'effectuer les opérations de passivation et d'envoi de messages de manière cohérente vis-à-vis de la sémantique opérationnelle de kπ. Ce mécanisme est décrit par le biais d'un ensemble relativement complexe de protocoles faisant intervenir des messages asynchrones de bas niveau, qui sont échangés entre les structures de données servant à implémenter les localités. Le parti pris, ce faisant, est d'adopter une approche très générale, dans laquelle aucune hypothèse n'est faite au sujet du déploiement des *localités logiques* (celles qui apparaissent dans la syntaxe des termes de kπ) sur les localités physiques. Ainsi, les communications sont toutes a priori distantes, et la passivation d'un module peut déclencher des passivations de sous-modules dont l'exécution a lieu sur une autre machine. Ceci entraîne la nécessité de prendre en compte des formes variées d'interférences entre les protocoles mettant en œuvre la communication et la passivation. En cela, l'étude de [62] se démarque de la machine abstraite pour les Kells proposée dans [10], qui est considérablement plus simple dans sa conception, car elle exploite une contrainte permettant à la passivation d'être atomique : on ne passive que des modules dont tous les sous-modules sont 'co-localisés', au sens où ils s'exécutent sur la même localité physique.

On renvoie le lecteur à [62] pour une description formelle de la machine abstraite pour kπ. Ce travail peut être vu comme une première étude de la mise en œuvre effective de l'opérateur de passivation. Il a permis en particulier de faire apparaître diverses questions en lien avec cette problématique.

Une première piste concerne la possibilité de garantir statiquement l'absence de 'communications fautives' : au-delà des questions standard ayant trait à l'utilisation cohérente des canaux de communication (pas d'envoi de triplets sur un canal où l'on s'attend à échanger des couples), on impose en kπ que les communications n'entraînent pas d'extrusion de noms au-delà des frontières de localités. Nous avons travaillé sur un système de types qui assure cette propriété, en exploitant des informations sur la localisation des restrictions au sein de la hiérarchie des processus. Ce type d'approche permet de remplacer des vérifications à l'exécution par une analyse statique en amont.

Une seconde question soulevée par l'étude de [62] est celle de la preuve de correction de la machine abstraite. Il s'agit d'établir une équivalence comportementale entre la machine et le calcul qu'elle est sensée implémenter. Dans [70], l'expérience acquise sur la validation d'une machine abstraite optimisée pour les *Safe Ambients* (les Safe Ambients sont un raffinement des Mobile Ambients visant à en rendre l'implémentation plus aisée et contrôlable) a mis en évidence la nécessité de disposer d'outils permettant d'affronter des preuves

de grande taille, notamment en les rendant modulaires — ceci a constitué le cœur de la thèse de Damien Pous [103], qui a porté sur l'étude des techniques modulo (ou *up-to*) pour la bisimulation. Il apparaît que la preuve de correction de la machine pour kπ est au moins un ordre de grandeur plus conséquente qu'une preuve telle que celle de [70]. Cette preuve n'a d'ailleurs pas été menée à bien dans [62]. L'analyse du comportement de la machine abstraite fait intervenir une notion d'*état bien formé* (intuitivement, un état bien formé est un état pouvant être atteint à partir d'une configuration initiale), et une étape de 'décompilation' permettant d'associer un processus kπ à un état bien formé. Ce travail est considérable, du fait du nombre important de règles définissant la machine, et des interférences que celles-ci engendrent. Pour plusieurs raisons, on rechigne à entreprendre une tâche si colossale : d'une part, le caractère fastidieux de la chose rend le développement de la preuve sur papier fragile, car des erreurs se glissent facilement dans la multitude des cas à traiter. D'autre part, une preuve de correction énoncée d'un trait est d'un intérêt limité. Il serait bien plus utile de pouvoir construire une preuve de manière modulaire, afin d'en rendre certaines parties réutilisables, pour l'analyse de modifications de la machine, ou la validation d'optimisations. Si des principes permettant d'aller dans ce sens sont disponibles s'agissant du développement d'une implémentation (en l'occurrence, la machine prototype pour kπ, développée en OCaml par D. Pous, est modulaire, et s'appuie sur une librairie générique pour l'exécution de processus distribués et mobiles), on ne dispose pas encore d'outils suffisamment puissants pour décomposer et rendre modulaire une preuve telle que celle de la machine de [70]. Qu'il s'agisse de preuve sur papier, ou sur machine (dans un assistant de preuve), il paraît intéressant et pertinent d'œuvrer à se doter de tels outils, afin de contribuer à ce que les techniques de programmation distribuée soient à portée de la preuve formelle. C'est dans cette direction que s'inscrivent les travaux récents de D. Pous et T. Braibant dans le système Coq [14, 15].

Équivalences et passivation.

La construction de passivation est source d'un pouvoir expressif considérable. Elle affecte en particulier de manière non triviale la théorie comportementale des processus, comme le montre [81]. Même dans des cas simples, les techniques employées pour l'étude du π-calcul d'ordre supérieur [109, 113], en particulier, ne peuvent être adaptées de façon immédiate.

La raison intuitive en est que la passivation donne accès à des observations inédites de la part du contexte. L'observateur peut désactiver certains composants du processus avec lequel il est en train d'interagir, ou même en stocker une copie, pour la réactiver plus tard. En un certain sens, la passivation donne au contexte le pouvoir d'inspecter la structure spatiale des processus de kπ— il semblerait d'ailleurs naturel qu'une logique modale extensionnelle pour ce formalisme contienne des opérateurs qui soient sensibles à la présence de localités.

Les travaux de S. Lenglet *et al.* [81, 80] proposent de premières contributions à l'étude des équivalences comportementales dans des calculs contenant l'opérateur de passivation, en étudiant la question de la caractérisation coinductive de la congruence barbelée.

5 Conclusion

Sur ce qui a été présenté.

Pour résumer de manière exagérément concise le contenu de ce mémoire, on peut dire que les résultats que l'on a présentés ont en commun de mettre en avant l'opérateur de composition parallèle. C'est sur la composition parallèle que l'on se focalise pour construire les axiomatisations de la partie 2.2, c'est $|$ qui est la brique élémentaire dans la structuration de l'espace telle qu'elle est examinée par le biais des logiques spatiales (parties 3 et 4.2). Par ailleurs, comme on l'a évoqué, on a vu s'accroître, ces dernières années, l'importance de la connotation spatiale de l'interaction dans le domaine des calculs de processus, en lien avec l'accent mis sur les phénomènes de mobilité et de distribution. Cette évolution se reflète dans une partie des travaux que l'on a présentés — notamment à la partie 4. Suivant ce point de vue, 'traverser la composition parallèle' (pour se synchroniser, pour échanger des messages, pour faire migrer du code) est loin d'être anodin, et ce d'autant plus lorsque des frontières, représentées sous forme de localités explicites, interviennent également.

La théorie de la concurrence, et en particulier l'étude des calculs de processus, est un domaine suscitant de nombreux questionnements, d'ordre méthodologique, en son sein : quelles sont les bonnes primitives pour un calcul, et comment s'en convainc-t-on ? préfère-t-on les sémantiques entrelaçantes à celles qui ne le sont pas ? l'équivalence de traces suffit-elle pour rendre compte des comportements concurrents, ou bien doit-on plutôt chérir la finesse de la bisimulation ? comment se formule une propriété d'expressivité (ou de non-expressivité) ? Le moins que l'on puisse dire est qu'il n'y a pas d'idéologie communément admise au sein de la communauté. De surcroît, les formes diverses, sans cesse renouvelées et étendues (en lien notamment avec les évolutions technologiques de l'informatique distribuée), que peut prendre la notion d'interaction dans un contexte concurrent sont source de stimulations pour le domaine. Cela est l'occasion de questionnements nouveaux, d'extensions, et de remises en cause de l'existant. Cela peut représenter, en même temps, un obstacle à la stabilisation et à la maturation du domaine autour d'un certain nombre de concepts fondamentaux. Il peut également arriver, comme on l'a montré dans ce qui précède, que certaines branches de la théorie soient 'redécouvertes'.

Face à de telles sources de déséquilibre et de fragilité, il importe de cultiver un certain recul. C'est, je l'espère, un souci que l'on peut lire dans les travaux que l'on a présentés. Le fait même d'étudier l'équivalence comportementale induite sur un calcul, afin de se mettre à distance vis-à-vis des aspects 'textuels', est une démarche habituelle, qui relève de cette approche. Essayer d'établir des résultats qui valent pour plusieurs calculs de processus, ou plusieurs variantes de logiques spatiales, ou, similairement, focaliser son attention sur des formalismes (calculs, logiques) minimaux, se rattache également à ce souci.

On peut s'attendre à ce que la théorie de la concurrence continue à progresser en maintenant un équilibre entre d'un côté l'inspiration renouvelée qu'apporte le foisonnement de nouveaux modes d'interaction concurrente, et de l'autre un souci d'assimiler et décanter les nouveautés. S'agissant du renforcement du second aspect, certaines évolutions relativement récentes sont encourageantes : on peut évoquer le développement d'approches pour une compréhension unifiée des calculs de processus, ou encore de multiples points de contact avec des domaines mathématiquement plus mûrs, comme le λ-calcul ou, plus récemment, la théorie de la démonstration, ce qui est l'occasion d'importer des notions et des techniques. Dans leur ensemble, ces travaux germent sur un terreau constitué de quelques notions clés, qui s'incarnent dans des objets comme CCS ou le π-calcul. Un certain nombre de pistes pour des développements futurs autour de ces notions ont été évoquées dans le corps du document. Au-delà de ces propositions, on espère plus généralement contribuer au développement et à la maturation de la théorie de la concurrence.

Étudiants.

Depuis que j'ai rejoint l'ENS Lyon, pratiquement tous les travaux de recherche que j'ai menés l'ont été en collaboration avec des étudiants (stagiaires de L3, M1, M2, et doctorants). Je les mentionne ici, pour les remercier.

Avec Luca Petrosino, nous avons étudié l'adaptation d'idées de H. Xi à la CuCh machine développée par Corrado Böhm et son équipe.

Romain Kervarc a implémenté un algorithme de décision pour la congruence structurelle dans le calcul des Mobile Ambients.

La thèse d'Étienne Lozes a porté sur les logiques spatiales, en se focalisant surtout sur des questions d'expressivité et de décidabilité.

David Teller a, durant sa thèse, développé des méthodes pour garantir des bornes sur l'usage de ressources dans un cadre concurrent.

Samuel Mimram a étudié des propriétés de la déduction pour une logique spatiale pour la concurrence (É. Lozes a co-encadré ce stage).

Stéphane Gimenez a développé un système de types pour le contrôle des ressources en π-calcul.

Céline Coma a étudié une implémentation de la logique de séparation.

Durant sa thèse, Damien Pous a développé des techniques de preuve pour la bisimulation, et les a appliquées à l'étude de machines abstraites distribuées.

Thomas Braibant a participé à un travail sur la formalisation d'une machine pour l'exécution distribuée d'un calcul de processus avec localités et passivation.

Romain Demangeon travaille durant sa thèse sur la terminaison des systèmes concurrents.

Ioana Cristescu, à l'occasion d'un stage d'été, a travaillé sur l'implémentation et la formalisation de procédures de décision pour la bisimilarité dans des sous-calculs de CCS, à partir de résultats obtenus avec D. Pous (qui a co-encadré le stage).

Enfin, c'est pour moi un privilège d'avoir travaillé durant ces années avec Davide Sangiorgi, dans une forme de collaboration où je me considère comme un étudiant de Davide. Grazie di tutto.

Références

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of POPL*, pages 104–115, 2001.

[2] M. Abadi and G. D. Plotkin. A Logical View of Composition and Refinement. In *Proc. of POPL*, pages 323–332. ACM, 1991.

[3] S. Abramsky. The lazy lambda calculus. In *Research Topics in Functional Programming*, pages 65–116. Addison-Wesley, 1990.

[4] L. Aceto and W.J. Fokkink. The Quest for Equational Axiomatizations of Parallel Composition : Status and Open Problems. Algebraic Process Calculi : The First Twenty Five Years and Beyond, 2005.

[5] R. M. Amadio and S. Prasad. Localities and Failures (Extended Abstract). In *Proc. of FSTTCS*, volume 880 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 1994.

[6] R.M. Amadio. On convergence sensitive bisimulation and the embedding of CCS in timed CCS. *ENTCS*, 2008. Proceedings of EXPRESS'08, to appear.

[7] J. Berdine, C. Calcagno, and P. W. O'Hearn. Smallfoot : Modular Automatic Assertion Checking with Separation Logic. In *Proc. of FMCO 2005, Revised Lectures*, volume 4111 of *Lecture Notes in Computer Science*, pages 115–137. Springer, 2005.

[8] J. Berdine, C. Calcagno, and P. W. O'Hearn. Symbolic Execution with Separation Logic. In *Proc. of APLAS'05*, volume 3780 of *LNCS*, pages 52–68. Springer, 2005.

[9] G. Berry and G. Boudol. The Chemical Abstract Machine. *Theor. Comput. Sci.*, 96(1) :217–248, 1992.

[10] P. Bidinger, A. Schmitt, and J.-B. Stefani. An Abstract Machine for the Kell Calculus. In *In Proc. FMOODS '05*, volume 3535 of *LNCS*, pages 31–46. Springer, 2005.

[11] M. Boreale and D. Sangiorgi. Some Congruence Properties for π-calculus Bisimilarities. *TCS*, 198 :159–176, 1998.

[12] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing Localities. *Theor. Comput. Sci.*, 114(1) :31–61, 1993.

[13] G. Boudol and G. Petri. Relaxed memory models : an operational approach. In *Proc. of POPL*, pages 392–403. ACM, 2009.

[14] T. Braibant and D. Pous. Algebraic Tools for Binary Relations in Coq. available from `http ://sardes.inrialpes.fr/~braibant/atbr/`, 2009.

[15] T. Braibant and D. Pous. A Tactic for Deciding Kleene algebras. In *Proc. of the first Coq Workshop*, 2009.

[16] M. Bundgaard and T. Hildebrandt. Bigraphical Semantics of Higher-Order Mobile Embedded Resources with Local Names. Technical Report TR-2005-70, IT University of Copenhagen, 2005.

[17] L. Caires. Behavioral and Spatial Observations in a Logic for the pi-calculus. In *Proc. of FOSSACS 2004*, volume 2987 of *Lecture Notes in Computer Science*, pages 72–89. Springer, 2004.

[18] L. Caires. Dynamic Spatial Logics : A Tutorial Survey. *Bulletin of the EATCS, Concurrency Column*, 94 :77–112, 2008.

[19] L. Caires. Spatial-Behavioral Types for Concurrency and Resource Control in Distributed Systems. *Theoretical Computer Science*, 402(2–3), 2008.

[20] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). *Inf. Comput.*, 186(2) :194–235, 2003.

[21] L. Caires and L. Cardelli. A spatial logic for concurrency (part II). *Theor. Comput. Sci.*, 322(3) :517–565, 2004.

[22] L. Caires and É. Lozes. Elimination of Quantifiers and Undecidability in Spatial Logics for Concurrency. In *Proc. of CONCUR*, volume 3170 of *Lecture Notes in Computer Science*, pages 240–257. Springer, 2004.

[23] L. Caires and É. Lozes. Elimination of quantifiers and undecidability in spatial logics for concurrency. *Theoretical Computer Science*, 358(2-3) :293–314, 2006.

[24] L. Caires and L. Monteiro. Verifiable and Executable Logic Specifications of Concurrent Objects in L_{pi}. In *Proc. of ESOP'98*, volume 1381 of *Lecture Notes in Computer Science*, pages 42–56. Springer, 1998.

[25] L. Caires and H. Torres Vieira. Extensionality of Spatial Observations in Distributed Systems. *Electr. Notes Theor. Comput. Sci.*, 175(3) :131–149, 2007.

[26] C. Calcagno, P. Gardner, and U. Zarfaty. Context logic and tree update. In *Proc. of POPL*, pages 271–282. ACM, 2005.

[27] C. Calcagno, P. Gardner, and U. Zarfaty. Context logic as modal logic : completeness and parametric inexpressivity. In *Proc. of POPL*, pages 123–134. ACM, 2007.

[28] A. Carayol, D. Hirschkoff, and D. Sangiorgi. On the representation of McCarthy's amb in the Pi-calculus. *Theor. Comput. Sci.*, 330(3) :439–473, 2005.

[29] L. Cardelli. Spatial Logics Page. `http://lucacardelli.name/SpatialLogics.htm`.

[30] L. Cardelli. Wide Area Computation. In *Proc. of ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 10–24. Springer, 1999.

[31] L. Cardelli, P. Gardner, and G. Ghelli. A Spatial Logic for Querying Graphs. In *Proc. of ICALP*, volume 2380 of *Lecture Notes in Computer Science*, pages 597–610. Springer, 2002.

[32] L. Cardelli and G. Ghelli. TQL : a query language for semistructured data based on the ambient logic. *Mathematical Structures in Computer Science*, 14(3) :285–327, 2004.

[33] L. Cardelli and A. Gordon. Ambient Logic. *Mathematical Structures in Computer Science*, 2006. to appear.

[34] L. Cardelli and A. D. Gordon. Mobile Ambients. In *Proc. of FoSSaCS'98*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 1998.

[35] L. Cardelli and A. D. Gordon. Anytime, Anywhere : Modal Logics for Mobile Ambients. In *Proc. of POPL'00*, pages 365–377. ACM, 2000.

[36] L. Cardelli and A. D. Gordon. Logical Properties of Name Restriction. In *Proc. of TLCA*, pages 46–60, 2001.

[37] Giuseppe Castagna, Jan Vitek, and Francesco Zappa Nardelli. The Seal Calculus. *Inf. Comput.*, 201(1) :1–54, 2005.

[38] I. Castellani. Process Algebras with Localities. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 945–1045. North-Holland, 2001.

[39] I. Castellani and M. Hennessy. Distributed bisimulations. *J. ACM*, 36(4) :887–911, 1989.

[40] G. Clinton and the P-Funk All Stars. T.A.P.O.A.F.O.M. (The Awesome Power of a Fully Operational Mothership). Sony, 1996.

[41] G. Conforti, D. Macedonio, and V. Sassone. Spatial Logics for Bigraphs. In *Proc. of ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 766–778. Springer, 2005.

[42] F. Corradini, R. Gorrieri, and D. Marchignoli. Towards parallelization of concurrent systems. *Informatique Théorique et Applications*, 32(4-6) :99–125, 1998.

[43] S. Crafa, D. Varacca, and N. Yoshida. Compositional Event Structure Semantics for the Internal *pi*-Calculus. In *Proc. of CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 317–332. Springer, 2007.

[44] I. Cristescu. Rewriting-based decision procedures for bisimilarity in subcalculi of CCS. Training period report, 2009.

[45] S. Dal-Zilio, D. Lugiez, and C. Meyssonnier. A logic you can count on. In *Proc. of POPL*, pages 135–146. ACM, 2004.

[46] M. Dam. Process-Algebraic Interpretations of Positive Linear and Relevant Logics. *J. Log. Comput.*, 4(6) :939–973, 1994.

[47] V. Danos and J. Krivine. Reversible Communicating Systems. In *Proc. of CONCUR*, volume 3170 of *Lecture Notes in Computer Science*, pages 292–307. Springer, 2004.

[48] R. Demangeon, D. Hirschkoff, N. Kobayashi, and D. Sangiorgi. On the Complexity of Termination Inference for Processes. In *Proc. of TGC'07*, volume 4912 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 2007.

[49] R. Demangeon, D. Hirschkoff, and D. Sangiorgi. Static and dynamic typing for the termination of mobile processes. In *Proc. of IFIP TCS*, volume 273 of *IFIP*, pages 413–427. Springer Verlag, 2008.

[50] R. Demangeon, D. Hirschkoff, and D. Sangiorgi. Termination in higher order concurrent calculi. *Journal of Logic and Algebraic Programming*, 2009. to appear.

[51] C. Fournet, F. Le Fessant, L. Maranget, and A. Schmitt. JoCaml : A Language for Concurrent Distributed and Mobile Programming. In *In Proc. Advanced Functional Programming 2002*, volume 2638 of *LNCS*, pages 129–158. Springer, 2002.

[52] C. Fournet and G. Gonthier. A hierarchy of equivalences for asynchronous calculi. In *Proc. of ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*. Springer, 1998.

[53] C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget, and D. Rémy. A Calculus of Mobile Agents. In *Proc. of CONCUR*, volume 1119 of *Lecture Notes in Computer Science*, pages 406–421. Springer, 1996.

[54] M. Gabbay and A. M. Pitts. A New Approach to Abstract Syntax with Variable Binding. *Formal Asp. Comput.*, 13(3-5) :341–363, 2002.

[55] P. Gardner, C. Laneve, and L. Wischik. The Fusion Machine (extended abstract). In *Proc. of CONCUR 2002*, volume 2421 of *Lecture Notes in Computer Science*, pages 418–433. Springer Verlag, 2002.

[56] P. Gardner, C. Laneve, and L. Wischik. Linear forwarders. *Inf. Comput.*, 205(10) :1526–1550, 2007.

[57] P. Giannini, D. Sangiorgi, and A. Valente. Safe Ambients : Abstract machine and distributed implementation. *Science of Computer Programming*, 59(3) :209–249, 2006.

[58] M. Hennessy. *A Distributed Pi-Calculus*. Cambridge University Press, 2007.

[59] M. Hennessy and R. Milner. Algebraic Laws for Nondeterminism and Concurrency. *J. ACM*, 32(1) :137–161, 1985.

[60] T. Hildebrandt, J.C. Godskesen, and M. Bundgaard. Bisimulation Congruences for Homer - a Calculus of Higher Order Mobile Embedded Resources. Technical Report TR-2004-52, IT University of Copenhagen, 2004.

[61] D. Hirschkoff. An Extensional Spatial Logic for Mobile Processes. In *Proc. of CONCUR*, volume 3170 of *Lecture Notes in Computer Science*, pages 325–339. Springer, 2004.

[62] D. Hirschkoff, T. Hirschowitz, S. Hym, A. Pardon, and D. Pous. Encapsulation and Dynamic Modularity in the pi-calculus. *Electr. Notes Theor. Comput. Sci.*, 241 :85–100, 2009.

[63] D. Hirschkoff, T. Hirschowitz, D. Pous, A. Schmitt, and J.-B. Stefani. Component-Oriented Programming with Sharing : Containment is not Ownership. In *Proc. of GPCE*, volume 3676 of *LNCS*, pages 389–404, 2005.

[64] D. Hirschkoff, É. Lozes, and D. Sangiorgi. Separability, Expressiveness, and Decidability in the Ambient Logic. In *Proc. of LICS'02*, pages 423–432. IEEE, 2002.

[65] D. Hirschkoff, É. Lozes, and D. Sangiorgi. Minimality Results for the Spatial Logics. In *Proc. of FSTTCS'03*, volume 2914 of *Lecture Notes in Computer Science*, pages 252–264. Springer, 2003.

[66] D. Hirschkoff, É. Lozes, and D. Sangiorgi. On the Expressiveness of the Ambient Logic. *Logical Methods in Computer Science*, 2(2), 2006.

[67] D. Hirschkoff and D. Pous. A Distribution Law for CCS and a New Congruence Result for the *pi*-Calculus. In *Proc. of FoSSaCS*, volume 4423 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 2007.

[68] D. Hirschkoff and D. Pous. A Distribution Law for CCS and a New Congruence Result for the *pi*-Calculus. *Logical Methods in Computer Science*, 4(2), 2008.

[69] D. Hirschkoff and D. Pous. On characterising strong bisimilarity in a fragment of CCS with replication. *CoRR*, abs/0810.2061, 2008.

[70] D. Hirschkoff, D. Pous, and D. Sangiorgi. An Efficient Abstract Machine for Safe Ambients. *Journal of Algebraic and Logic Programming*, 71(2) :114–149, 2007.

[71] Y. Hirshfeld and M. Jerrum. Bisimulation Equivalence is Decidable for Normed Process Algebra. Technical Report ECS-LFCS-98-386, LFCS, 1998.

[72] T. Hoare and P. W. O'Hearn. Separation Logic Semantics for Communicating Processes. *Electr. Notes Theor. Comput. Sci.*, 212 :3–25, 2008.

[73] A. Hobor, A. W. Appel, and F. Zappa Nardelli. Oracle Semantics for Concurrent Separation Logic. In *Proc. of ESOP*, volume 4960 of *Lecture Notes in Computer Science*, pages 353–367. Springer, 2008.

[74] K. Honda and N. Yoshida. On Reduction-Based Process Semantics. *Theor. Comput. Sci.*, 151(2) :437–486, 1995.

[75] C. B. Jones. Specification and Design of (Parallel) Programs. In *IFIP Congress*, pages 321–332, 1983.

[76] C. B. Jones. Tentative Steps Toward a Development Method for Interfering Programs. *ACM TOPLAS*, 5(4) :596–619, 1983.

[77] B. Klin, V. Sassone, and P. Sobocínski. Labels from reductions : towards a general theory. In *Proc. of 1st Calco*, volume 3629 of *LNCS*, pages 30–50. Springer, 2005.

[78] I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. On the Expressiveness and Decidability of Higher-Order Process Calculi. In *Proc. of LICS*, pages 145–155. IEEE Computer Society, 2008.

[79] J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In *Proc. of CONCUR'00*, volume 1877 of *LNCS*, pages 243–258. Springer, 2000.

[80] S. Lenglet, A. Schmitt, and J.-B. Stefani. Howe's Method for Calculi with Passivation. In *Proc. of CONCUR*, volume 5710 of *Lecture Notes in Computer Science*, pages 448–462. Springer, 2009.

[81] S. Lenglet, A. Schmitt, and J.-B. Stefani. Normal Bisimulations in Calculi with Passivation. In *Proc. of FOSSACS*, volume 5504 of *Lecture Notes in Computer Science*, pages 257–271. Springer, 2009.

[82] É. Lozes. *Expressivité des logiques spatiales*. PhD thesis, École Normale Supérieure de Lyon, 2004.

[83] É. Lozes, D. Hirschkoff, and D. Sangiorgi. Separability in the Ambient Logic. *Logical Methods in Computer Science*, 4(3 :4), 2008.

[84] E. Lozes and J. Villard. A Spatial Equational Logic for the Applied Pi-Calculus. In *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR'08)*, volume 5201 of *Lecture Notes in Computer Science*, pages 387–401. Springer, 2008.

[85] M. Merro and F. Zappa Nardelli. Behavioral theory for Mobile Ambients. *J. ACM*, 52(6) :961–1023, 2005.

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

[87] R. Milner. Functions as Processes. In *Proc. of Automata, Languages and Programming, 17th International Colloquium, ICALP90*, Lecture Notes in Computer Science, pages 167–180. Springer, 1990.

[88] R. Milner. Bigraphs and Their Algebra. *Electr. Notes Theor. Comput. Sci.*, 209 :5–19, 2008.

[89] R. Milner. *The Space and Motion of communicating agents*. Cambridge University Press, 2009.

[90] R. Milner and F. Moller. Unique Decomposition of Processes. *TCS*, 107(2) :357–363, 1993.

[91] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, I. *Inf. Comput.*, 100(1) :1–40, 1992.

[92] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, II. *Inf. Comput.*, 100(1) :41–77, 1992.

[93] R. Milner and D. Sangiorgi. Barbed Bisimulation. In *Proc. of ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer, 1992.

[94] D. Murphy. Observing Located Concurrency. In *Proc. of MFCS*, volume 711 of *Lecture Notes in Computer Science*, pages 566–576. Springer, 1993.

[95] U. Nestmann. What is a "Good" Encoding of Guarded Choice ? *Inf. Comput.*, 156(1-2) :287–319, 2000.

[96] U. Nestmann and B. C. Pierce. Decoding Choice Encodings. *Inf. Comput.*, 163(1) :1–59, 2000.

[97] M. Nielsen, G. D. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part I. *Theor. Comput. Sci.*, 13 :85–108, 1981.

[98] P. W. O'Hearn. Resources, concurrency, and local reasoning. *Theor. Comput. Sci.*, 375(1-3) :271–307, 2007.

[99] P. W. O'Hearn and D. J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2) :215–244, 1999.

[100] C. Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous pi-calculus. In *Proc. of POPL*, pages 256–265, 1997.

[101] A. T. Phillips, N. Yoshida, and S. Eisenbach. A distributed abstract machine for boxed ambient calculi. In *Proc. of ESOP'04*, volume 2986 of *Lecture Notes in Computer Science*, pages 155–170. Springer, 2004.

[102] B. C. Pierce and D. N. Turner. Pict : a programming language based on the Pi-Calculus. In G. D. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 455–494. The MIT Press, 2000.

[103] D. Pous. *Techniques modulo pour les bisimulations*. PhD thesis, ENS Lyon, 2008.

[104] J. Rathke, V. Sassone, and P. Sobocínski. Semantic Barbs and Biorthogonality. In *Proc. of FoSSaCS*, volume 4423 of *Lecture Notes in Computer Science*, pages 302–316. Springer, 2007.

[105] J. Rathke and P. Sobocínski. Deconstructing behavioural theories of mobility. In *Proc. of IFIP TCS'08*. Springer, 2008. to appear.

[106] J. Rathke and P. Sobocinski. Deriving Structural Labelled Transitions for Mobile Ambients. In *Proc. of CONCUR*, volume 5201 of *Lecture Notes in Computer Science*, pages 462–476. Springer, 2008.

[107] J. C. Reynolds. Separation Logic : A Logic for Shared Mutable Data Structures. In *Proc. of LICS*, pages 55–74. IEEE Computer Society, 2002.

[108] J. Riely and M. Hennessy. Distributed Processes and Location Failures (Extended Abstract). In *Proc. of ICALP*, volume 1256 of *Lecture Notes in Computer Science*, pages 471–481. Springer, 1997.

[109] D. Sangiorgi. *Expressing Mobility in Process Algebras : First-Order and Higher-Order Paradigms*. PhD thesis, University of Edimburgh, 1992.

[110] D. Sangiorgi. Extensionality and Intensionality of the Ambient Logics. In *Proc. of POPL*, pages 4–13, 2001.

[111] D. Sangiorgi. On the Origins of Bisimulation and Coinduction. *ACM TOPLAS*, 2008. to appear – a preliminary version has appeared as Technical Report 2007-24, Department of Computer Science, University of Bologna.

[112] D. Sangiorgi and D. Walker. On Barbed Equivalences in pi-Calculus. In *Proc. of CONCUR'01*, volume 2154 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2001.

[113] D. Sangiorgi and D. Walker. *The π-calculus : a Theory of Mobile Processes*. Cambridge University Press, 2001.

[114] V. Sassone and P. Sobocínski. Locating reaction with 2-categories. *Theor. Comp. Sci.*, 333(1-2) :297–327, 2005.

[115] P. Sewell. From rewrite rules to bisimulation congruences. In *Proc. of CONCUR'98*, volume 1466 of *LNCS*, pages 269–284. Springer, 1998.

[116] C. Stirling. Modal Logics for Communicating Systems. *Theor. Comput. Sci.*, 49 :311–347, 1987.

[117] D. Teller, P. Zimmer, and D. Hirschkoff. Using Ambients to Control Resources. In *Proc. of CONCUR'02*, volume 2421 of *LNCS*. Springer Verlag, 2002.

[118] D. Teller, P. Zimmer, and D. Hirschkoff. Using Ambients to control resources. *Int. J. Inf. Sec.*, 2(3-4) :126–144, 2004.

[119] A. Unyapoth and P. Sewell. Nomadic Pict : correct communication infrastructure for mobile computation. In *Proc. of POPL'01*, pages 116–127. ACM, 2001.

[120] R. J. van Glabbeek. The Linear Time-Branching Time Spectrum (Extended Abstract). In *Proc. of CONCUR*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer, 1990.

[121] R. J. van Glabbeek. The Linear Time - Branching Time Spectrum II. In *Proc. of CONCUR*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1993.

[122] H. Vieira and L. Caires. A Spatial Logic Model Checker. `http ://ctp.di.fct.unl.pt/SLMC/`.

[123] J. Villard, É. Lozes, and C. Calcagno. Proving Copyless Message Passing. In *Proc. of APLAS'09*, LNCS. Springer Verlag, 2009. to appear.

[124] G. Winskel. Events, Causality and Symmetry. In *Visions of Computer Science - BCS International Academic Conference*, pages 111–127. British Computer Society, 2008.

A Articles dont il est question dans ce mémoire

Nous mentionnons ici les articles qui ont été mentionnés dans le texte de ce mémoire. Ils sont sont à disposition en cliquant sur la version électronique de ce document.

– parties 2.1 et 3.2.2

□[61] D. Hirschkoff. An extensional spatial logic for mobile processes. In *CONCUR*, pages 325–339, 2004.

– partie 2.2

□[68] D. Hirschkoff and D. Pous. A Distribution Law for CCS and a New Congruence Result for the *pi*-Calculus. *Logical Methods in Computer Science*, 4(2), 2008.

□[69] D. Hirschkoff and D. Pous. On characterising strong bisimilarity in a fragment of CCS with replication. *CoRR*, abs/0810.2061, 2008.

– partie 3.2.1

□[65] D. Hirschkoff, É. Lozes, and D. Sangiorgi. Minimality results for the spatial logics. In *Proc. of FSTTCS'03*, Lecture Notes in Computer Science, pages 252–264. Springer, 2003.

– partie 4.2

□[66] D. Hirschkoff, É. Lozes, and D. Sangiorgi. On the expressiveness of the ambient logic. *Logical Methods in Computer Science*, 2(2), 2006.

□[83] É. Lozes, D. Hirschkoff, and D. Sangiorgi. Separability in the ambient logic. *Logical Methods in Computer Science*, 4(3 :4), 2008.

– partie 4.3

□[70] D. Hirschkoff, D. Pous, and D. Sangiorgi. An Efficient Abstract Machine for Safe Ambients. *Journal of Algebraic and Logic Programming*, 71(2) :114–149, 2007.

□[63] D. Hirschkoff, T. Hirschowitz, D. Pous, A. Schmitt, and J.-B. Stefani. Component-oriented programming with sharing : Containment is not ownership. In *Proc. of GPCE*, volume 3676 of *LNCS*, pages 389–404, 2005.

□[62]. D. Hirschkoff, A. Pardon, T. Hirschowitz, S. Hym, and D. Pous. Encapsulation and dynamic modularity in the pi-calculus. *Electr. Notes Theor. Comput. Sci.*, 241 :85–100, 2009.

B Autres travaux

Les travaux suivants n'ont pas été traités dans ce document.

– *Typage et bornes sur l'utilisation des ressources en Mobile Ambients*, avec D. Teller et P. Zimmer.

□[117] D. Teller, P. Zimmer, and D. Hirschkoff. Using Ambients to Control Resources. In *Proc. of CONCUR'02*, volume 2421 of *LNCS*. Springer Verlag, 2002.

□[118]. D. Teller, P. Zimmer, and D. Hirschkoff. Using ambients to control resources. *Int. J. Inf. Sec.*, 2(3-4) :126–144, 2004.

– *Étude de l'encodage de l'opérateur amb de McCarthy dans le π-calcul*, avec A. Carayol et D. Sangiorgi.

□[28] A. Carayol, D. Hirschkoff, and D. Sangiorgi. On the representation of mccarthy's amb in the pi-calculus. *Theor. Comput. Sci.*, 330(3) :439–473, 2005.

– *Systèmes de types pour la terminaison en π-calcul*, avec R. Demangeon et D. Sangiorgi.

□[48] R. Demangeon, D. Hirschkoff, N. Kobayashi, and D. Sangiorgi. On the complexity of termination inference for processes. In *Proc. of TGC'07*, volume 4912 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 2007.

□[49] R. Demangeon, D. Hirschkoff, and D. Sangiorgi. Static and dynamic typing for the termination of mobile processes. In *Proc. of IFIP TCS*, volume 273 of *IFIP*, pages 413–427. Springer Verlag, 2008.

□[50] R. Demangeon, D. Hirschkoff, and D. Sangiorgi. Termination in higher order concurrent calculi. To appear in Journal of Algebraic and Logic Porgramming, 2009.