Structure algébrique pour des environnements à entités interactionnelles et mobiles

Abdelkader Gouaich, Yves Guiraud, Fabien Michel

Structure algébrique pour des environnements à entités interactionnelles et mobiles

Gouaich, Abdelkader*† Guiraud, Yves‡§ Michel, Fabien¶

Résumé

Les systèmes ubiquistes sont de *'bons'* défis lancés aux communautés de l'ingénierie des systèmes informatiques pour le développement fiable de systèmes ouverts, mobiles et coopératifs. L'interaction se trouve au centre de ces systèmes. En effet, le schéma d'interaction entre les composants est très variable et dépend des propriétés environnementales. Pour pouvoir comprendre ces environnements et leurs propriétés, il est donc indispensable de les décrire d'une façon globale et rigoureuse. Ce papier présente un modèle algébrique de tels environnements où des entités mobiles, interactionnelles et autonomes évoluent. Les évolutions de cette structure sont caractérisées pour définir les concepts de mouvement, d'interaction et de computation. Nous illustrerons également, comment ce modèle est utilisé concrètement pour développer une application ubiquiste.

1 Introduction

Les réseaux de communication ad hoc connectent spontanément différents systèmes informatiques hétérogènes. Ces systèmes, *ubiquistes*, sont définis partout et continuellement dans l'espace et dans le temps. Dans ce contexte particulier, l'environnement de déploiement, entourant le système informatique, influence le schéma d'interaction des entités et affecte ainsi les caractéristiques et les fonctionnalités du système. Les influences de l'environnement de déploiement ont souvent été ignorées ou rendues transparentes[GG02] dans les systèmes informatiques classiques. Cependant, avec la venue d'environnements complexes ces influences ne peuvent plus être ignorées et doivent être considérées durant la phase de conception et de réalisation du système. Cette corrélation entre le système informatique et son environnement de déploiement a déjà été soulevée par L. Cardelli dans [Car99]. Il démontre que selon l'environnement de déploiement cible, la conception du système différera car elle reposera sur des *hypothèses* environnementales différentes. Par exemple, la conception d'une application dans un environnement de réseau local (LAN) supposera une certaine fiabilité des liens d'interactions entre les composants distribués. Cependant, la même application devra prendre en compte des problèmes liés à l'imprédictibilité des liens et à lŠimpossibilité de contrôler globalement lŠenvironnement d'agglomération de réseau (WAN). Ceci illustre bien que les influences de l'environnement de déploiement sur le système deviennent incontournables dans les environnements complexes[GG02]. F. Zambonelli et H. Parunak dans [FZ02] soulignent également que l'intégration des influences de l'environnement de déploiement dans les phases d'ingénierie est indispensable pour les systèmes d'informations modernes. De plus, ils décrivent quelques propriétés générales de ces environnements de déploiement :

– localisation : Les composants logiciels sont exécutés dans un environnement délimité ;

*gouaich@lirmm.fr - Laboratoire Informatique, Robotique et Micro Electronic- UMR 5506 - Montpellier - France
†gouaich@crm.mot.com - MOTOROLA Labs - Networking and Application Lab - Centre de recherche de Motorola-Paris - France
‡guiraud@lirmm.fr - Laboratoire Informatique, Robotique et Micro Electronic- UMR 5506 - Montpellier - France
§guiraud@math.univ-montp2.fr - Laboratoire Géométrie, Topologie, Algèbre - Université Montpellier 2 - France
¶fmichel@lirmm.fr, Laboratoire Informatique, Robotique et Micro Electronic- UMR 5506 - Montpellier - France

– ouverture : Les systèmes informatiques sont fortement décentralisés et changent dynamiquement de structure ;
– localité du contrôle : Les composants logiciels sont autonomes et proactifs ;
– localité des interactions : Les composants logiciels privilégient des schémas d'interaction locale.

Les environnements ubiquistes exhibent aussi une propriété de *composabilité*. Ce qui permet de les unir ou de les séparer. Par exemple, quand des conditions environnementales tel que la distance ou la puissance d'un signal éléctro-magnétique sont réunies, un nouvel environnement est disponible par composition des environnements existants. Les entités de cet environnement peuvent exhiber des comportements différents car elles interagissent avec de nouvelles entités. D'une façon similaire, quand ces conditions environnementales sont insatisfaites, le système global se subdivise en plusieurs sous systèmes indépendants.

1.1 Motivations

Comprendre et représenter explicitement l'environnement de déploiement où des entités autonomes, mobiles et interactionnelles évoluent devient donc un point crucial spécialement pour les environnements dynamiques et imprédictibles. Pour répondre à ce besoin, nous avons défini une structure algébrique nommée {Mouvement, Interaction, Computation}* (MIC*). MIC* est un modèle formel d'environnement où des entités autonomes, interactionnelles et mobiles sont déployées. A la différence des modèles formels de calcul mobile tel que Ambient [Car99], π-calculus [Mil00] et le Join calculus [Fou98], MIC* étudie les propriétés externes de l'environnement entourant des processus de calcul. Les modèles de calcul étudient les propriétés internes d'un processus de calcul. Les motivations de notre approche peuvent être résumées comme suit :

Description rigoureuse : L'étude des systèmes informatiques complexes s'étale sur plusieurs domaines de recherches comme les systèmes distribués, code mobile et l'intelligence artificiel distribuée. Par conséquent, des concepts génériques comme : 'interaction', 'mobilité' et 'agent' sont utilisés avec ambiguïté. Sans imposer des définitions consensuelles, MIC* caractérise rigoureusement ces concepts dans le cadre de son étude.

Un modèle implémentable : En adoptant une démarche purement algébrique, la structure du modèle est implémentée directement avec des structures informatiques équivalentes aux objets algébriques. Par conséquent, il est possible de faire un lien direct entre les structures informatiques du système et son modèle algébrique. Ceci contraste avec les démarches classiques formelles qui définissent une syntaxe et des fonctions d'interprétation de cette syntaxe. Le lien entre l'implémentation du modèle et le modèle est alors plus difficile à établir.

1.2 Organisation du papier

Ce papier est organisé comme suit : la section 2 présente des domaines proches qui traitent des influences de l'environnement de déploiement sur les interactions d'un système et ainsi son comportement global. La section 3 introduit informellement une application ubiquiste d'où seront extraits les concepts fondamentaux de la structure formelle décrite dans la section 4. La section 5 reprend l'exemple informel et présente sa réalisation avec le modèle MIC*. La section 6 conclue en présentant des perspectives sur nos futurs travaux.

2 Domaines voisins

Cette section présente des domaines de recherche qui sont concernés par le problème du contrôle des influences de l'environnement sur les interactions internes entre les composants.

2.1 Les systèmes multi-agents

Les systèmes multi-agents (SMA) considèrent un système computationnel comme un agrégat cohérent d'entités ou *'agents'*. Un agent est une entité autonome qui accomplie ses buts propres en interagissant avec d'autres agents et son environnement[JW95]. Pour les agents dits cognitifs les interactions sont conduites à travers des messages sémantiques d'interaction comme FIPA ACL [FIP96a] ou KQML [DAR93]. Le contenu de ces messages peut être défini avec des ontologies qui fixent la sémantique des concepts échangés durant l'interaction. De plus, les SMA ont défini une continuité cohérente des interactions sous forme d'un dialogue entre agents. Ce dialogue peut être spécifié avec des protocoles d'interaction tel que les protocoles FIPA [FIP96b]. Malgré ses contributions non négligeables pour définir les interactions entre composants informatiques, lŠapproche SMA a du mal à convaincre à une grande échelle. En effet, pour la plupart des systèmes informatiques où l'environnement reste simple, les paradigmes classiques, comme l'orienté objet, suffisent largement et sont bien admis. Cependant, les environnements mobiles et dynamiques tel que les environnements ubiquistes peuvent constituer une frontière réelle entre les paradigmes classiques et le paradigme SMA pour l'ingénierie des systèmes informatiques. C'est pourquoi nous pensons que les systèmes ubiquistes devraient être considérés comme de vrais cas d'étude et d'expérimentation pour les SMA.

2.2 Code mobile

Le code mobile étudie les systèmes informatiques, où des composants logiciels changent d'environnement d'exécution durant leur cycle de vie. Comme dans les SMA, les composants mobiles interagissent[1] pour accomplir leurs buts. La communauté du code mobile s'est heurtée rapidement à la gestion des interactions entre les composants. Une des solutions apportée à ce problème est l'identification d'une entité nommée 'média de coordination' qui contrôle et concrétise les interactions entre les entités[CLZa, CLZb]. Parmi les médias de coordination on peut citer Lime [PMR99], Tuscon [OZ98] et MARS [CLZb]. Le média de coordination est donc une entité externe au système applicatif : il fait partie de ce que nous avons défini comme l'environnement de déploiement. Nous proposons de généraliser cette entité en définissant un modèle d'environnement qui définit non seulement les interactions entre les entités, mais également leur mobilité et les observations valides, et donc sures, du résultat de leur calcul interne.

2.3 Calcul mobile formel

Les modèles formels de calcul décrivent un langage de programmation 'formel'. Le λ-calcul [Mor68] est probablement le plus connu et étudié de ces modèles. Malheureusement, il se limite seulement aux algorithmes séquentiels et statiques. Ceci amena l'apparition d'autres modèles de calcul qui incluent des concepts de l'informatique moderne comme la mobilité, la distribution et la sécurité. Parmi ces modèles de calcul on peut citer le π-calculus[MPW92] et ses dérivés pour la gestion de la distribution Dπ[AG98] et la sécurité Sπ[MH98] ; Ambient [Car99] et le Join calculus [Fou98]. MIC* adopte une approche différente en définissant les concepts de mobilité, distribution et interaction comme des propriétés environnementales et non calculatoires.

3 Exemple informel

Cette section introduit les concepts fondamentaux du modèle MIC* à travers un scénario simple d'application ubiquiste.

[1] Dans le cadre de notre étude nous considérons les composants qui se meuvent pour interagir avec d'autres composants. Ce qui exclue l'étude des systèmes où la mobilité est motivée uniquement par la gestion de ressources ou la répartition de charge

3.1 Discussion électronique ubiquiste

Les applications de discussion électronique émulent les discussions verbales humaines par des moyens technologiques. Ce type d'application a déjà connu un certain succès dans le contexte d'Internet. Pour l'environnement ubiquiste, l'utilisateur n'est plus connecté à un réseau central, mais dispose d'un petit appareil capable de communiquer de façon ad hoc. Lorsque les utilisateurs se trouvent à proximité, un réseau local se définit spontanément et ils peuvent alors échanger leurs messages et discuter de différents sujets.

3.2 Objets d'interaction

L'interaction entre les différentes entités est conduite par l'échange explicite *d'objets d'interaction*. Ces objets possèdent naturellement une structure de monoïde. En effet, il est possible de définir une loi commutative de composition $+$. Ainsi, l'objet $a + b$ sera considéré comme la juxtaposition des objets d'interaction a et b. Le premier pas d'abstraction que nous menons consiste à définir l'objet d'interaction nul. Ceci se traduit par la définition d'un élément neutre 0 comme objet d'interaction. Dans le scénario présenté, les objets d'interaction représentent les messages échangés. La réception de plusieurs messages simultanés est vue comme la réception d'un objet d'interaction somme : $\sum o$.

3.3 Espace d'interaction

Les espaces d'interaction sont des abstractions qui représentent des localisations où différentes entités interagissent. L'espace d'interaction est une entité active qui régie les règles d'interaction entre les entités. A ce titre, elle peut par exemple altérer les objets d'interaction émis. Dans le scénario présenté chaque sujet de discussion est un espace d'interaction où les agents humains échangent leurs messages. Pour illustrer la nature active de l'espace d'interaction on peut imaginer une certaine politique de gestion des messages échangés. Ainsi, les messages jugés intolérables seront simplement réduits à l'objet d'interaction neutre. Les messages contenants des fautes d'orthographe seront corrigés. Concernant la mobilité des entités, elle est modélisée comme la mobilité des objets d'interaction dans les espaces d'interaction. Par exemple, le désir d'un utilisateur de participer aux discussions d'un sujet particulier est défini comme le mouvement d'un objet d'interaction dans l'espace d'interaction du sujet. De plus, un agent peut être présent dans plusieurs espaces d'interaction et réagir comme un tout cohérent à toutes ses interactions : ceci définit son ubiquité logique.

3.4 Agents

Les agents perçoivent et réagissent aux interactions externes par un calcul local et l'émission de nouveaux objets d'interaction. Ces réactions sont considérées comme des tentatives pour influencer l'univers (les autres). Ces influences sont effectivement validées par l'espace d'interaction uniquement si elles sont en adéquation avec ses règles d'interaction locales.

4 Formalisme de MIC^*

4.1 Agents de MIC^*

Dans ce paragraphe, nous allons définir la structure algébrique des agents de MIC^*. Soient $\mathcal{O}$ un groupe abélien, I et J deux ensembles et $(L_i)_{i\in I}$ une famille d'ensembles pointés (*i.e.* avec un élément distingué noté 0). Nous donnons l'interprétation suivante à ces objets : les éléments de $\mathcal{O}$ sont les *objets d'interaction*, ceux de I les *processus*, ceux de J les *espaces d'interaction* et ceux de L_i les termes du

langage interne du processus i.

On note $\mathcal{O}^{(I\times J)}$ l'ensemble des familles $(a_{i,j})_{(i,j)\in I\times J}$ d'éléments de $\mathcal{O}$ telles que tous les $a_{i,j}$ sont nuls sauf un nombre fini. De même, on note $\mathcal{L}$ l'ensemble des familles $(m_i)_{i\in I}$, avec $m_i \in L_i$, telles que $m_i = 0$ sauf pour un nombre fini.

On pose enfin : $\mathcal{A} = \mathcal{O}^{(I\times J)} \times \mathcal{O}^{(I\times J)} \times \mathcal{L}$ dont les éléments sont appelés *agents (de MIC^*)*. On note $out, in : \mathcal{A} \longrightarrow \mathcal{O}^{(I\times J)}$ et $mem : \mathcal{A} \longrightarrow \mathcal{L}$ les trois projections et, pour $A \in \mathcal{A}$, on appelle $outA$ (resp. inA) la *boîte d'émission* (resp. *de réception*) de A et $memA$ la *mémoire* de A. On peut encore projeter : $out = (out_{i,j})_{(i,j)\in I\times J}$, $in = (in_{i,j})_{(i,j)\in I\times J}$ et $mem = (mem_i)_{i\in I}$. Si $A \in \mathcal{A}$, on appelle $out_{i,j}A$ (resp. $in_{i,j}A$) la *boîte d'émission* (resp. *de réception*) *du processus i de A dans l'espace j*, et mem_iA la *mémoire du processus i de A.*

4.2 Règles d'évolution de MIC^*

Ici, nous allons caractériser les évolutions des agents.

4.2.1 Mouvement

Une *règle de mouvement* est une application $\mu : \mathcal{A} \longrightarrow \mathcal{A}$ vérifiant :

1) μ ne modifie pas les boîtes de réception ni les mémoires :

$$in \circ \mu = in \quad \text{et} \quad mem \circ \mu = mem\,;$$

2) μ préserve globalement les boîtes d'émission d'un processus donné :

$$\forall i \in I, \quad \sum_{j\in J} out_{i,j} \circ \mu \;=\; \sum_{j\in J} out_{i,j}\,;$$

3) les boîtes de réception *après mouvement* d'un processus donné ne dépendent que des boîtes de réception *avant mouvement* du même processus :

$$\forall i \in I, \quad (out_{i,j})_{j\in J} \circ \mu \;=\; (out_{i,j})_{j\in J} \circ \mu \circ (out_{i,j})_{j\in J}\,;$$

4) la règle de mouvement est la même pour tous les processus :

$$\forall A \in \mathcal{A}, \forall i, i' \in I, (out_{i,j}A)_{j\in J} = (out_{i',j}A)_{j\in J} \;\Rightarrow\; (out_{i,j}\mu(A))_{j\in J} = (out_{i',j}\mu(A))_{j\in J}.$$

La propriété suivante est alors immédiate :

Proposition. Il existe une correspondance bijective entre les règles de mouvement et les applications $\mathcal{O}^{(J)} \longrightarrow \mathcal{O}^{(J)}$. Plus précisément, une règle de mouvement μ est entièrement caractérisée par une application $\tilde{\mu}$ et :

$$\mu\Big((a_{i,j})_{i,j}, (b_{i,j})_{i,j}, (m_i)_i\Big) \;=\; \Big(\big(\tilde{\mu}((a_i, j)_j)\big)_i, (b_{i,j})_{i,j}, (m_i)_i\Big).$$

Parmi les mouvements, il en existe des particuliers, les mouvements *élémentaires* ; ils sont caractérisés par la donnée de $i, i' \in I$ et de $a \in \mathcal{O}$ et vérifient : pour tout agent A, $out_{k,j}(A)$ n'est pas modifié si $k \notin \{i, i'\}$ et, sinon, le mouvement lui ajoute (resp. retranche) a si $k = i$ (resp. $k = i'$). Cette définition, la proposition précédente, la propriété 2) des mouvements (conservation) et le fait que, dans un agent donné, il n'y a qu'un nombre fini de boîtes d'émission non nulles conduisent à la proposition :

Proposition. Pour tout mouvement μ et tout agent A, il existe une suite *finie* $(\mu_1, \ldots, \mu_n)$ de mouvements élémentaires vérifiant : $\mu(A) = \mu_n \circ \cdots \circ \mu_1(A)$.

4.2.2 Interaction

Une *règle d'interaction* est une application $\varphi : \mathcal{A} \longrightarrow \mathcal{A}$ qui vérifie :

1) φ ne modifie pas les boîtes d'émission ni les mémoires :

$$out \circ \varphi = out \quad \text{et} \quad mem \circ \varphi = mem\,;$$

2) un processus qui n'est pas présent dans un espace d'interaction donné ne reçoit rien dans cet espace :

$$\forall A \in \mathcal{A}, \forall i \in I, \forall j \in J, \quad out_{i,j} A = 0 \;\Rightarrow\; in_{i,j}\varphi(A) = in_{i,j} A\,;$$

3) la boîte de réception *après interaction* d'un processus donné dans un espace d'interaction fixé ne dépend que de sa valeur *avant interaction* et des boîtes d'émission des processus dans le même espace d'interaction :

$$\forall i \in I, \forall j \in J, \quad in_{i,j} \circ \varphi \;=\; in_{i,j} \circ \varphi \circ \Big((out_{k,j})_{j\in J}, in_{i,j}\Big).$$

On a alors la description suivante :

Proposition. Une interaction φ est entièrement déterminée par une famille d'applications $(\varphi_{i,j} : \mathcal{O}^{(I)} \times \mathcal{O} \longrightarrow \mathcal{O})_{i,j}$ de la manière suivante :

$$\varphi\Big((a_{i,j})_{i,j}, (b_{i,j})_{i,j}, (m_i)_i\Big) \;=\; \Big((a_{i,j})_{i,j}, \big(\varphi_{i,j}((a_{k,j})_k, b_{i,j})\big)_{i,j}, m_i\Big).$$

De plus, pour tout agent $A \in \mathcal{A}$, il existe des sous-ensembles *finis* $I_0 \subseteq I$ et $J_0 \subseteq J$ tels que $\varphi(A)$ se calcule par application successive des $\varphi_{i,j}$ pour $i \in I_0$ et $j \in J_0$, et ce dans n'importe quel ordre.

4.2.3 Calcul

Une *règle de calcul* est une application $\gamma : \mathcal{A} \longrightarrow \mathcal{A}$ qui vérifie :

1) si un processus est modifié par le calcul (une de ses boîtes d'émission ou sa mémoire), ses boîtes de réception sont remises à 0 :

$$\forall A \in \mathcal{A}, \forall i \in I, \quad \Big(\exists j \in J, out_{i,j}\gamma(A) \neq out_{i,j} A \text{ ou } mem_i\gamma(A) \neq mem_i A\Big) \;\Rightarrow\; (in_{i,j}\gamma(A))_j = 0\,;$$

2) les valeurs des boîtes d'émission et de la mémoire *après calcul* d'un processus donné ne dépendent que des valeurs des boîtes d'émission, de réception et de la mémoire *avant calcul* de ce processus :

$$\forall i \in I, \quad \begin{cases} (out_{i,j})_j \circ \gamma & = (out_{i,j})_j \circ \gamma \circ \Big((out_{i,j})_j \times (out_{i,j})_j \times mem_i\Big) \\ mem_i \circ \gamma & = mem_i \circ \gamma \circ \Big((out_{i,j})_j \times (out_{i,j})_j \times mem_i\Big) \end{cases}$$

On a la caractérisation suivante :

Proposition. Un calcul γ est entièrement déterminé par la famille d'applications $\big(\gamma_i : \mathcal{O}^{(J)} \times \mathcal{O}^{(J)} \times L_i \longrightarrow \mathcal{O}^{(J)} \times \mathcal{O}^{(J)} \times L_i\big)_i$ vérifiant :

$$\gamma\Big((a_{i,j})_{i,j}, (b_{i,j})_{i,j}, (m_i)_i\Big) \;=\; \Big(\gamma_i\big((a_{i,j})_j, (b_{i,j})_j, m_i\big)\Big),$$

avec, comme condition supplémentaire, que γ_i est l'identité ou alors sa deuxième composante (les boîtes de réception) est nulle. De plus, pour un agent $A \in \mathcal{A}$ fixé, il existe un sous-ensemble *fini* $I_0 \subseteq I$ tel que $\gamma(A)$ se calcule par application successive des γ_i à A, $i \in I_0$, et ce dans n'importe quel ordre.

4.3 Structure de MIC^*

Une *structure de* MIC^* est la donnée d'un groupe abélien $\mathcal{O}$ (objets d'interaction), d'ensembles I et J (les processus et les espaces d'interaction), d'ensembles de mouvements, d'interactions et de calculs et d'un sous-ensemble E de l'ensemble des évolutions possibles, c'est-à-dire de l'ensemble des combinaisons possibles des mouvements, interactions et calculs donnés.

Nous allons voir deux exemples formels de MIC^* : le premier décrit des ordinateurs calculant indépendemment les uns des autres (des agents effectuant des β-réductions de λ-termes) et le second des processus échangeant des noms (des agents incarnant des processus du π-calcul).

Exemple 1. Ici, on fixe $\mathcal{O} = 0$ (le groupe trivial), $I = \{1, \ldots, n\}$, $J = \{*\}$ et, pour tout $i \in \{1, \ldots, n\}$, $L_i = \Lambda \coprod \{0\}$ avec Λ l'ensemble des λ-termes modulo α-conversion. Un agent est donc constitué de n processus (au plus) ayant des 0 dans toutes leurs boîtes de réception et d'émission et un λ-terme chacun dans leur mémoire. Les seuls mouvements et interaction considérés sont l'identité sur $\mathcal{A}$. On se donne un calcul pour chaque processus, qui effectue une β-reduction du terme en mémoire, en suivant la stratégie *leftmost-outermost*. Enfin, l'ensemble E des évolutions est constitué de toutes les combinaisons possibles de ces calculs. Si on fixe un agent A, toutes les évolutions possibles vont converger vers $\hat{A}$ ayant en mémoire les formes β-normales des λ-termes de départ.

Exemple 2. Soit $\mathcal{N}$ un ensemble dénombrable (l'ensemble des *noms* du π-calcul). On note $\mathcal{O}$ le $\mathbb{Z}/2\mathbb{Z}$-groupe abélien libre engendré par $\mathcal{N}$. Soient $I = \mathbb{N}$ et $J = \mathcal{N}$. Enfin, pour tout i, on définit L_i comme l'ensemble des termes décrits par la grammaire suivante :

$$P \in L_i ::= 0 \big| \bar{x}y.P \big| x(y).P \quad \text{avec } x, y \in \mathcal{N}.$$

On considère comme mouvement l'identité, comme interactions l'ensemble des $\varphi^x_{i,i'}$ pour $x \in \mathcal{N}$ et $i \neq i' \in I$, définis par : $\varphi^x_{i,i'}(a_{k,z})$ met x en $in_{i,z}$ et $-x$ en $in_{i',z}$ si $a_{i,z} = -y$ et $a_{i',z} = x$, ne change rien sinon, et ne change rien aux autres boîtes de réception. Pour les calculs, on considère les γ_i qui envoient $((x_y), (-x_y), P)$ sur $(0, 0, P)$, $((-x_y), (x'_y), P)$ sur $(0, 0, P[x := x'])$, $(0, 0, \bar{y}x.P)$ sur $((x_y), 0, P)$ et $(0, 0, y(x).P)$ sur $((-x_y), 0, P)$ (et ne changent rien sinon). On a utilisé les conventions (x_y) est le vecteur ne contenant que des 0 sauf un x en y et $P[x := x']$ est la substitution sans capture de variable de x par x' dans P. Enfin, les seules évolutions considérées sont les composées d'applications de la forme $\gamma_i \circ \gamma_i \circ \gamma_{i'} \circ \gamma_{i'} \circ \varphi^x_{i,i'}$.

On considère à présent un sous-ensemble Π_0 des π-termes défini de la manière suivante :

$$P \in \Pi_0 ::= 0 \big| \bar{x}y.P \big| x(y).P \big| P|Q \quad \text{avec } x, y \in \mathcal{N}.$$

La première observation est : on peut "linéariser" tout $P \in \Pi_0$, c'est-à-dire l'écrire comme $P_1| \ldots |P_k$ où chaque P_l ne contient que des termes de L_i. On n'a pas d'égalité $P = P_1| \ldots |P_k$ mais une équivalence de

comportement classique en π-calcul. Par exemple :

$$\bar{x}y.(P|Q) \equiv \bar{x}y.z(t).z(t).0|\bar{z}t.P|\bar{z}t.Q$$

où les variables z et t sont choisies judicieusement selon le contexte (il ne faut pas qu'elles soient utilisées ailleurs). On va alors traduire chaque P_l comme $((x_y), 0, Q)$ si $P_l = \bar{y}x.Q$ et comme $((-x_y), 0, Q)$ si $P_l = y(x).Q$. On note $[P]$ l'agent de MIC^* ainsi obtenu à partir de $P \in \Pi_0$. Alors, toutes les évolutions possibles de $[P]$ mènent à des formes normales, qui sont les $[\hat{P}_p]$ où $\hat{P}_p$ parcourt l'ensemble des formes normales de P pour la réduction du π-calcul.

4.4 MIC* Matrices

Afin de présenter la structure formelle, une représentation matricielle plus intuitive a été développée. Cette représentation est familière aux informaticiens et donne une écriture spatiale plus accessible que des formules algébriques linéaires.
Chaque terme MIC* est représenté par les matrices suivantes :

Matrice des émissions : Les lignes de cette matrice représentent les agents (processes) $A_i \in I$ et les colonnes représentent les espaces d'interaction $S_j \in J$. Les éléments de la matrice $o_{(i,j)} \in \mathcal{O}$ représentent la boite d'émission de l'agent A_i dans l'espace d'interaction S_j, , cŠest-à-dire sa représentation dans cet espace.

Matrice des perceptions : Les lignes de cette matrice représentent les agents $A_i \in I$ et les colonnes représentent les espaces d'interaction $S_j \in J$. Les éléments de la matrice $o_{(i,j)} \in \mathcal{O}$ représentent la boite de réception dŠun agent A_i, cŠest-à-dire ses perceptions dans l'espace d'interaction S_j.

Vecteur des mémoires : Les agents $A_i \in I$ représentent les lignes de ce vecteur. Un élément m_i du vecteur représente la mémoire interne de l'agent au sens de Turing [Tur36]. Hormis l'existence, aucune autre hypothèse n'est formulée sur cet élément.

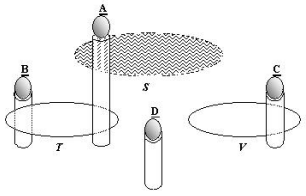

FIG. 1 – *Des agents dans un environnement.*

	1	S	T	V
A	a	b	c	0
B	d	0	e	0
C	f	0	0	g
D	h	0	0	0

TAB. 1 – Matrice des représentations de la figure 1.

Par exemple, la matrice des représentations de la table 1 modélise la situation décrite dans la figure 1. Quand un agent est présent dans un espace d'interaction, sa représentation diffère du 0. Ainsi, la valeur 0 traduit lŠabsence dŠun agent dans un espace dŠinteraction.

4.5 Composition environnementale

Une des justifications de l'approche algébrique pour construire le modèle est la composition 'naturelle' des termes pour représenter la composition des environnements computationnels. Considérons l'exemple de deux environnements où les matrices des émissions sont définies comme suit : $e_1^{outbox} =$

	S_j
A_i	$o_{i,j}$

et $e_2^{outbox} =$

	S_j
$A_{i'}$	$o_{i',j}$

Les agents A_i et $A_{i'}$ appartiennent au même espace d'interaction S_j mais se trouvent dans deux environnements indépendants e_1 et e_2. En conséquence, aucune interaction n'est possible entre ces deux processus. L'union de ces environnements définira un nouvel environnement $e_3 = e_1 + e_2$: $e_3^{outbox} = e_1^{outbox} + e_2^{outbox} =$

	S_j
A_i	$o_{i,j}$
$A_{i'}$	$o_{i',j}$

Dans l'environnement e_3, les agents A_i et $A_{i'}$ interagissent par l'échange d'objet d'interaction. Similairement, un environnement peut être disjoint en deux environnements indépendants.

4.6 Les évolutions de MIC* sur les matrices

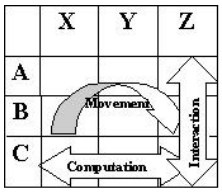

FIG. 2 – *Movement, Interaction et Computation*

Dans cette section nous illustrons les principales transformations des représentations matricielles illustrées dans la figure 2 :

Mouvement : Le mouvement est une transformation d'un terme où la matrice des perceptions et le vecteur des mémoires sont inchangés. La matrice des représentations est modifiée, mais reste globalement invariante pour un agent donné. Cela signifie que les objets d'interaction changent d'espace d'interaction sans perte ni création.

Interaction : L'interaction est caractérisée par une transformation matricielle où les représentations et le vecteur des mémoires sont inchangés. La matrice des perceptions est modifiée pour chaque espace d'interaction pour prendre en compte les nouvelles perceptions de chaque agent.

Computation : L'observation d'une computation est caractérisée par la modification des représentations pour un agent donné. La matrice des perceptions est inchangée par cette transformation.

4.7 Limitations

MIC* est une approche formelle *descriptive*. Des éléments abstraits y sont définis pour modéliser des concepts concrets comme le *movement*, *l'interaction* et *l'observation d'un calcul*. Pour l'instant aucun outil de preuve n'est défini pour analyser les propriétés d'un système modélisé.

5 Discussion électronique ubiquiste

5.1 Description de l'application

La section 3 a introduit l'application de discussion électronique ubiquiste émulant des discussions 'orales' d'un groupe d'humains. Cette démonstration concrète des concepts a été implémentée en utilisant un prototype du modèle MIC* implémenté en PYTHON [Pyt02]. Elle est disponible pour les réseaux locaux LAN entre des machines ; et des réseaux ad hoc avec la technologie WI-FI en mode Peer-to-Peer, en utilisant des assistants personnels.

5.2 Situation A :

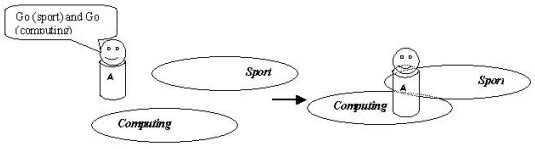

FIG. 3 – *L'agent 'A' entrant dans deux espaces d'interaction*

Comme présenté dans la section 3, chaque sujet est représenté par un espace d'interaction autour duquel les agents humains échangent des messages. Par exemple, les sujets de "sport" et "computing" sont représentés par deux espaces d'interaction (figure 3). Quand un utilisateur sélectionne un sujet de discussion x particulier, son agent informatique exprime cela par l'émission d'un objet d'interaction go^x. Cet objet d'interaction est automatiquement absorbé par le 'bon' espace d'interaction. En effet, l'espace d'interaction contrôle complètement ses lois locales de mouvement en permettant à certains objets d'entrer et en refusant l'accès à d'autres. Ainsi, la politique de mouvement d'un espace d'interaction x est de permettre l'accès à tous les objets d'interaction du type go^x ; les objets d'interaction du type $-go^x$ seront sortis de l'espace d'interaction s'ils y sont déjà. Ceci représente le désabonnement d'un agent humain du sujet x. La situation de la figure 3 est formellement représentée par les matrices des représentations suivantes :

$e_0^{outbox} =$

	1	*sport*	*computing*
A	$go^{sport} + go^{computing}$	0	0

qui évolue en :

$M(M((e_0^{outbox})) = e_1^{outbox} =$

	1	*sport*	*computing*
A	0	go^{sport}	$go^{computing}$

Après ces deux mouvements, l'agent A est présent dans les deux espaces d'interaction : *sport* et *computing*.

5.3 Situation B :

Comme illustré dans la figure 4, quand deux environnements E_1 et E_2 sont joints un nouvel environnement E_3 est défini. Dans cet environnement, le schéma d'interaction entre les entités est modifié : les agents A et B sont maintenant capables d'interagir dans l'espace d'interaction 'sport'. Cette situation est formellement décrite comme suit :

	1	*sport*	*computing*
A	0	go^{sport}	$go^{computing}$

+

	1	*sport*	*computing*
B	0	go^{sport}	0

→

	1	*sport*	*computing*
A	0	go^{sport}	$go^{computing}$
B	0	go^{sport}	0

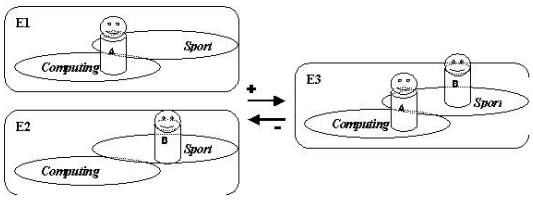

FIG. 4 – *union et disjonction des environnements*

5.4 Situation C :

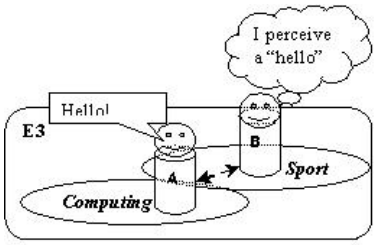

FIG. 5 – *Interaction entre les agents*

Le calcul est un processus interne à un agent qui modifie sa mémoire selon le modèle de Turing[Tur36]. Un agent ne modifie en aucun cas directement l'état de l'univers mais y parvient par un processus d'interaction. Par exemple, quand un agent humain calcule son prochain message en réponse à l'historique de la discussion, le seul résultat observable de ce calcul est le message rédigé. L'agent informatique prend en charge ce message et l'émet dans l'espace d'interaction (voir figure 5). Plus concrètement, quand un agent A envoie un message *hello*, la matrice des représentations est transformée comme suit :

	1	*sport*	*computing*
A	0	go^{sport}	$go^{computing}$
B	0	go^{sport}	0

$\rightarrow$

	1	*sport*	*computing*
A	0	*hello*	*hello*
B	0	go^{sport}	0

L'interaction entre les agents est ensuite définie avec les matrices de perceptions suivantes :

	1	*sport*	*computing*
A	0	0	0
B	0	0	0

$\rightarrow$

	1	*sport*	*computing*
A	0	*hello*	*hello*
B	0	*hello*	0

Les deux agents A et B reçoivent le message *hello* émis dans la matrice des représentations. Cette interaction sera ensuite considérée pour les prochains calculs de ces entités.

6 Conclusion

Ce papier a présenté la structure algébrique MIC* modélisant des environnements composables où des entités autonomes, interactionnelles et mobiles évoluent. La dynamique de ces environnements est décrite comme la composition de trois transformations atomiques : le mouvement, l'interaction et le calcul. Cette démarche identifie explicitement l'environnement de déploiement d'un système informatique. Ceci peut aider à la construction fiable de systèmes informatiques dans des environnements ouverts et dynamiques comme les environnements ubiquistes. En effet, un environnement de déploiement abstrait peut être généré

directement des spécifications du système. Ces spécifications seront garanties par les lois de mouvement et d'interaction de l'environnement indépendamment des entités computationnelles qui le peuplent.

Références

[AG98] M. Abadi and A.D. Gordan. A calculus for cryptographic protocols : The spi calculus. Information and Computation, 1998.

[Car99] Luca Cardelli. Abstraction for mobile computation. *Secure internet programming : security issues for mobile and distributed objects, LNCS 1603*, 1999.

[CLZa] Giacomo Cabri, Letizia Leonardi, and Franco Zambonelli. Coordination in mobile agent applications.

[CLZb] Giacomo Cabri, Letizia Leonardi, and Franco Zambonelli. Reactive tuple spaces for mobile agent coordination.

[DAR93] DARPA. Specifiaction of the kqml agent-communication language. Technical report, DARPA, 1993.

[FIP96a] FIPA. Fipa agent communication language, www.fipa.org/, 1996.

[FIP96b] FIPA. Foundation of intelligent and physical agents, 1996. http ://www.fipa.org.

[Fou98] Cedric Fournet. *Le join-calcul : un calcul pour la programmation repartie et mobile.* PhD thesis, Ecole Polytechnique, 1998.

[FZ02] H. Van Dyke Parunak Franco Zambonelli. Signs of a revolution in computer science and software engineering. In *AOSE 02*, 2002.

[GG02] A. Gouaich and Y. Guiraud. (movement, interaction, calculus)* : an algebraic environment for distributed and mobile calculus. In *ICAIS*. Deakin University, February 2002.

[JW95] Nicholas R. Jennings and Michael Wooldridge. Intelligent agents : theory and practice. *The Knowledge Engineering Review*, 10(2) :115–152, 1995.

[MH98] James Riely Mathew Hennessy. A typed language for distributed mobile processes. In Proc. Of POPL, ACM Press, 1998.

[Mil00] Robin Milner. Communication and mobile systems : the pi-calculus. Cambridge University Press, 2000.

[Mor68] James Hiram Morris. λ-calculus model of programming language, 1968. MIT.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus for mobile processes, parts 1 and 2. *Information and computation*, 1992.

[OZ98] A. Omicini and F. Zambonelli. The tucson coordination model for mobile information agents, 1998.

[PMR99] Gian Pietro Picco, Amy L. Murphy, and Gruia-Catalin Roman. Lime : Linda meets mobility. In *International Conference on Software Engineering*, pages 368–377, 1999.

[Pyt02] Python. Python web resources. web, March 2002. http ://www.python.org.

[Tur36] Alan Turing. On computable numbers, with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Soceity*, number 42 in 2, 1936.