MIC*: Algebraic Agent Environment

Abdelkader Gouaich, Yves Guiraud

MIC*: Algebraic Agent Environment

Abdelkader Gouaich[1,2] and Yves Guiraud[1,3]

[1] LIRMM, Montpellier, France,
{gouaich,guiraud}@lirmm.fr
[2] Centre de Recherche Motorola, Paris, France,
gouaich@crm.mot.com
[3] Laboratoire Géométrie, Topologie, Algèbre – Université Montpellier 2 – France
guiraud@math.univ-montp2.fr

Abstract. This paper presents the MIC* algebraic structure modelling an environment where autonomous, interacting and mobile entities evolve.

1 Introduction

Understanding and representing explicitly the *deployment environment* where software processes or *agents* are deployed is a crucial issue especially for dynamical and open software environment such as ubiquitous software systems. In fact, as mentioned by [2] ignoring properties of the deployment environment may lead to dysfunctions that cannot be explained when the software system is isolated from its deployment environment. This paper presents an algebraic model, named MIC*, of such deployment environment where autonomous, interacting and mobile agents evolve. Multi-agents system (MAS) considers a computational system as a coherent aggregation of autonomous entities, named *agents*. The deployment environment has already been pointed out as a fundamental concept for the design and implementation of MAS-based applications [8, 4]. However, few works have actually addressed the problem of studying the general properties of this entity. By contrast to MAS, mobile computing [7] and coordination communities have represented explicitly the deployment environment that joins respectively mobile entities and coordinables in order to establish their interactions [5,1]. We suggest generalising their concepts in order to define a deployment environment that rules interactions among entities; their movement laws and how to react to their actions. On the other hand, formal models of mobile and distributed calculus such as π-calculus [6], Ambient [2] and Join calculus [3] present a formal programming language capturing modern computing concepts such as mobility and distribution. MIC* adopts a different view by clearly separating the calculus from its environment. Consequently, mobility, interaction and observations of the entities' computation are defined and studied at the environmental level.

2 Informal Example: Ubiquitous Electronic Chat Scenario

In order to introduce the MIC* formal structure, this section extracts main concepts starting from a simple ubiquitous application scenario. The ubiquitous electronic chat

N. Zhong et al. (Eds.): ISMIS 2003, LNAI 2871, pp. 216–220, 2003.

application emulates verbal conversations between several humans about some specific topics. Users are no longer connected permanently to a central network but own a small device equipped with some ad hoc networking capabilities. Thus, when several users are spatially joined they can converse together. The general description of the application can be summarised as following: *(i)* each user *participates* in one or several *discussions*; *(ii)* the interaction between the users are conducted by explicitly sending *messages*. The first reflection concerns the interactions among agents. These interactions are materialised by concrete objects that we identify as *interaction objects*. Interaction objects are structured objects. For instance, they can be composed in simultaneous interactions. Moreover, a special empty interaction object (the zero 0) can be abstractly identified to express 'no interaction'. In the presented scenario, messages represent the interaction objects and receiving simultaneous messages is viewed as receiving a *sum* ($\sum o$) of interaction objects. *Interaction spaces* are abstract locations where several entities interact by exchanging explicitly interaction objects. An interaction space rules interactions among agents and may alter the exchanged interaction objects. In the ubiquitous chat scenario, each topic is represented by an interaction space, where several *human agents* can exchange messages. Concerning the mobility over the interaction spaces, it is easy to encode the agents' desires to participate in certain topics as a logical movement inside these interaction spaces. *Agents* perceive and react to external interaction objects by a local computation and the emission of other interaction objects in interaction spaces. These reactions are considered as *attempts* to influence the universe (others) that are committed only when they are coherent with the deployment environment rules.

3 $\{$Movement, Interaction, Computation$\}^*$ (MIC*)

Due to space constraints this section presents semi-formally and briefly some aspect of the $\{$Movement, Interaction, Computation$\}^*$ structure. The algebraic theoretical definitions are omitted. Thus, a more intuitive view of the manipulated algebraic objects is designed using matrices. In fact, matrix representations are familiar to computer scientists and give spatial representation better than complex linear formulas. To present the matrix view, the reader should assume the following minimal definitions:

- $(\mathcal{O}, +)$ represents the commutative group of interaction objects. This means that interaction objects can be composed commutatively by the $+$ law, and that the empty interaction object exists ($0 \in \mathcal{O}$). Furthermore, each interaction object x has an opposite $(-x)$ and $x + (-x) = 0$;
- $\mathcal{A}$ and $\mathcal{S}$ represent respectively the sets of agents and interaction spaces. $\mathcal{S}$ contains a special element: $1 \in \mathcal{S}$ representing the universe as a global interaction space. Moreover, this special element has the following features: *(i)* no interaction between the entities is possible; *(ii)* all the interaction objects can move inside or outside this interaction spaces without restriction.

Each MIC* term is represented by the following matrices:

Outboxes Matrix: Rows of this matrix represent agents $A_i \in \mathcal{A}$ and the columns represent the interaction spaces $S_j \in \mathcal{S}$. Each element of the matrix $o_{(i,j)} \in \mathcal{O}$ is the representation of the agent A_i in the interaction space S_j.

Inboxes Matrix: Rows of this matrix represent agents $A_i \in \mathcal{A}$ and the columns represent the interaction spaces $S_j \in \mathcal{S}$. Each element of the matrix $o_{(i,j)} \in \mathcal{O}$ defines how the agent A_i perceives the universe in the interaction space S_j.

Memories Vector: Agents $A_i \in \mathcal{A}$ represent the rows of the vector. Each element m_i is an abstraction of the internal memory of the agent A_i. Except the existence of such element that can be proved using the Turing machine model, no further assumptions are made in MIC* about this element.

MIC* terms model naturally ubiquitous environments. In fact, the union or split of computational environments are simply represented as an addition $+$ and a subtraction $-$ defined between the sub matrices representing sub environments.

3.1 MIC* *Dynamics*

This section characterises three main transformations of environmental terms : the movement, the interaction and the computation.

- Movement μ: A movement is a transformation μ, of the environment where both inboxes and memories matrices are unchanged, and where outboxes matrix interaction objects are changed but globally invariant. This means that interaction objects of an agent can change their positions in outboxes matrix and no interaction object is created or lost.
- Interaction ϕ: Interaction is characterised by a transformation ϕ that leaves both outboxes and memories matrices unchanged and transform a row of the inboxes matrix. Thus, interaction is defined as modifying the perceptions of the entities in a particular interaction space.
- Computation γ: An observable computation of an entity transforms its representations in outboxes matrix and the memories vector. For practical reasons, the inboxes of the calculating entity are reset to 0 after the computation.

Finally, the structure of MIC* is fully defined for a particular system by giving the interaction objects group; the sets of processes and interaction spaces; the sets of transformations defining its dynamics.

4 Ubiquitous Chat

4.1 Situation A

Each topic is represented by an interaction space. For instance, "sport" and "computing" topics are represented by two interaction spaces (figure 1). When the user selects a chat topic x, this is expressed as sending an interaction object go^x. This interaction object is absorbed by the *correct* interaction space. In fact, the interaction space controls its local movement policy allowing certain interaction objects to enter and refusing access to others. Here the movement policy of an interaction space x is to absorb go^x and to move outside $-go^x$. Situation expressed in figure 1 is be described formally as follows:

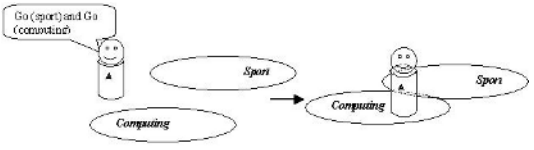

Fig. 1. Agent 'A' moving inside two interaction spaces

$$e_0^{outbox} = \begin{array}{|c|c|c|c|} \hline & 1 & sport & computing \\ \hline A & go^{sport} + go^{computing} & 0 & 0 \\ \hline \end{array} \text{ that evolves to :}$$

$$\mu(\mu((e_0^{outbox})) = e_1^{outbox} = \begin{array}{|c|c|c|c|} \hline & 1 & sport & computing \\ \hline A & 0 & go^{sport} & go^{computing} \\ \hline \end{array}$$

After these two movements, agent A exists in both interaction spaces: $sport$ and $computation$.

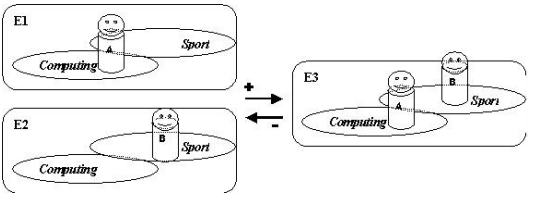

Fig. 2. Union and disjunction of environments

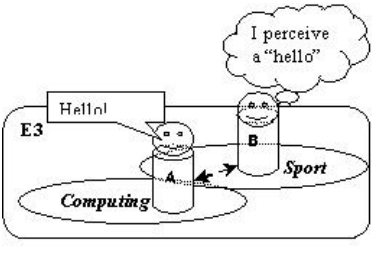

Fig. 3. Interaction among agents

4.2 Situation B

As illustrated in figure 2, when two environments E_1 and E_2 are joined a new environment E_3 is defined. This environment, the interaction schema among the entities is modified. On the other side, when the physical network link is disconnected, the environment E_3 is split into E_1 and E_2. This situation is formally described as follows:

$$\begin{array}{|c|c|c|c|} \hline & 1 & sport & computing \\ \hline A & 0 & go^{sport} & go^{computing} \\ \hline \end{array} + \begin{array}{|c|c|c|c|} \hline & 1 & sport & computing \\ \hline B & 0 & go^{sport} & 0 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|} \hline & 1 & sport & computing \\ \hline A & 0 & go^{sport} & go^{computing} \\ B & 0 & go^{sport} & 0 \\ \hline \end{array}$$

4.3 Situation C

Computation modifies the agent's memory. Consequently, an agent can modify the surrounding entities only by sending interaction objects. For instance, when a human agent computes internally what he should write as message, the observation of this process is the written message (interaction object). The surrounding entities receive this message through the interaction space (see figure 3). For instance, when agent A writes a $hello$ message, the outboxes matrix is changed as follows:

	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

Both agents A and B receive the $hello$ message that was emitted in the outboxes matrix.

5 Conclusion

This paper has presented, semi-formally, the MIC* algebraic structure modelling combinable environments where mobile, autonomous and interacting entities evolves. Evolutions of this environment are described as composition of three atomic evolutions: movement, interaction and computation. The next step of our works is to generate specific environments and interaction spaces starting form the engineering specification of a system. Following the MIC* approach, it would be possible to guarantee these specifications in an unpredictable and dynamic environments such as ubiquitous systems.

References

1. Giacomo Cabri, Letizia Leonardi, and Franco Zambonelli. Coordination in mobile agent applications. Technical Report DSI-97-24, Dipartimento di Scienze dell Ingegneria Universitá di Modena, 1997.
2. Luca Cardelli. Abstractions for mobile computation. *Secure Internet Programming*, pages 51–94, 1999.
3. Cedric Fournet. *Le Join-Calcul: Un Calcul Pour la Programmation Repartie et Mobile*. PhD thesis, Ecole Polytechnique, 1998.
4. James J. Odell, H. Van Dyke Parunak, Mitch Fleischer, and Sven Brueckner. Modeling agents and their environment. In *AOSE 2002, AAMAS 2002*, Bologna, 2002.
5. George A. Papadopoulos. Models and technologies for the coordination of Internet agents: A survey. In Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies, and Applications*, chapter 2, pages 25–56. Springer-Verlag, March 2001.
6. Milner Robin, Parrow Joachim, and Walker David. A calculus for mobile processes, parts 1 and 2. *Information and Computation*, 100(1), 1992.
7. G.-C. Roman, G. P. Picco, and A. L. Murphy. Software engineering for mobility: A roadmap. *The Future of Software Engineerin,*, pages 241–258, 2000.
8. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.