

MAP^a: a Language for Modelling Conversations in Agent Environments

María Adela Grandó^{1*} and Christopher D. Walton^{2**}

¹ Research Group on Mathematical Linguistics,
Rovira i Virgili University, Tarragona, Spain.

² Centre for Intelligent Systems and their Applications,
School of Informatics, University of Edinburgh, UK.

Abstract. In this paper we present the MAP^a language for expressing dialogues in multiagent systems. This is accomplished by defining patterns of communication between groups of agents, expressed by protocols. Our language is directly implementable and allows to specify the connection between communication and knowledge management in a way that is independent of the specific reasoning techniques used. Here we introduce MAP^a formal syntax and we point out added features with respect to its predecessor, the MAP language.

1 Introduction

A Multi-Agent-System (MAS) may be defined as a collection of *agents*, which are autonomous and rational components, that interact within an environment and exhibit intelligent behaviour based on the dialogue with other agents and internal reasoning processes. To accomplish this, it is necessary for the agent to be able to *communicate* and *coordinate* with other agents in order to achieve certain tasks. This inter-operability between agents is surprisingly difficult to achieve.

In this paper we present our attempt to address these issues: the definition of a language that allows the communication between agents. We call it MAP^a because it is an extension of the MAP language which we previously presented in [10]. Like MAP it is based in the *societal approach* from [8]. In a society agents typically share a common interest, e.g. solving a particular problem. To become a member of this society, an agent must agree to observe certain rules and conventions expressed as *social norms*. For example, when engaging in a telephone conversation, the participants will take turns to speak. If the participants were to talk at the same time, then the conversation would become unintelligible. Therefore, it is these social norms that ensure the smooth running of the society.

* Sponsored by the “Programa Nacional para la Formación del Profesorado Universitario” from the Ministry of Education, Culture and Sports of Spain.

** Sponsored by the EPSRC Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (Grant GR/N15764/01).

A popular way to express the social norms between groups of agents is by means of an explicit *protocol*. A protocol is a statement of the social norms and acts as a guide for the agents to interact without affecting their autonomy. This approach has been successfully adopted in the Conversation Policy (CP) [4] and Electronic Institutions (EI) [2,9] formalisms, among others.

MAP^a retains from MAP its lightweight nature and the generation of *executable specifications*. Also the idea of providing decision points in which the agents can behave autonomously. These decision points are given by the agent reasoning processes, that we call decision procedures, whose output can determine next action to be performed in the protocol. New features are incorporated to MAP^a to allow the definition of more flexible and dynamic protocols. Now protocols are treated as first-class objects which can be passed between agents as opposed to static definitions from MAP. MAP^a also allows the definition of hierarchies of agent roles, with the purpose of re-usability and inheritance of role definitions. The concept of knowledge is also refined distinguishing between knowledge share by all the agents in a scene, knowledge associated to a role, and knowledge private to an agent.

2 MAP^a Language Definition

MAP^a is a lightweight language-based formalism for the expression of protocols, which is based on social interactions between groups of agents. It is a sugared process-calculus with an asynchronous semantics, which makes it suitable for describing interactions between agents. A BNF-style syntax of MAP^a is shown in Fig. 1. The remainder of this section is primarily an explanation of this syntax.

In MAP^a the concepts of *scenes* and *roles* are crucial. A *scene* can be thought of as a bounded space in which a group of agents interact on a single task. Formally a scene S comprises a *role hierarchy* $R^{(i)}$, a set of agent *protocols* $P^{(i)}$ which are parameterized on these roles, a set of axioms $K^{(s)}$ for the *common knowledge* and a set of performatives $M^{(t)}$ which defines all of the allowed performatives for the scene. An example of scene is the so called *negotiation protocol* where a buyer attempts to obtain an agreement with a group of sellers about the buying conditions for a set of items, and eventually buy them. For this scene we define $R^{(i)}$ as a hierarchy with tax-contributor as top role and buyer and seller as subroles, to denote that all the agents participating in the negotiation are required to be legal contributors. For $K^{(s)}$ that represents the set of facts which are believe to be true for all the agents in the scene, we can consider for the negotiation protocol: date, country, current coin, money exchange, etc. And about $M^{(t)}$, we can take as possible set of performatives that agents can utter: invitation to bid, acceptance or rejection of invitation, bid proposal, etc.

Agents participating in a scene adopt a *role* which they can dynamically change for other role from $R^{(i)}$. For example, an agent playing the role of

S	$::= \langle R^{(i)}, P^{(i)}, K^{(s)}, M^{(t)} \rangle$	(Scene)
R	$::= \langle id, Proc^{(l)}, K^{(m)}, r^{(n)} \rangle$	(Role)
P	$::= \mathbf{agent}(id, r, \phi^{(f)}) = op.$	(Protocol)
K	$::= axiom$	(Knowledge)
M	$::= id((\phi, type)^{(h)})$	(Performative)
$Proc$	$::= type :: id((\phi, type)^{(g)})$	(Procedure)
op	$::= v$	(Protocol Variable)
	$\mathbf{agent}(id, r, K^{(v)}, \phi^{(d)})$	(Agent Invocation)
	$op_1 \mathbf{then} op_2$	(Sequence)
	$op_1 \mathbf{or} op_2$	(Choice)
	$op_1 \mathbf{par} op_2$	(Parallel)
	(op)	(Precedence)
	α	(Action)
α	$::= \mathbf{null}$	(No Action)
	$v = id(\phi^{(g)})$	(Decision)
	$id(\phi^{(x)}) \Leftarrow \mathbf{agent}(id, r)$	(Receive)
	$id(\phi^{(y)}) \Rightarrow \mathbf{agent}(id, r)$	(Send)
ϕ	$::= c \mid - \mid v$	(Term)

Fig. 1. BNF for MAP^a Syntax.

seller can change its role to buyer if he has taken the compromise of selling more items than he has in stock. The role hierarchy is defined as a set of role definitions. Each of these definitions R has a unique identifier id , a set of role decision procedures definitions $Proc^{(l)}$, a set of role axioms $K^{(m)}$ and a set of upper roles $r^{(n)}$, which appear above the role in the hierarchy. A decision procedure $Proc$ is defined with an identifier, a set of typed input parameters, and an output type. The actual decision procedure is defined externally to the protocol. For example, a buyer agent may define the following procedures $Proc^{(l)}$: decide which bids to accept and which ones to reject, choose the best process of exchange and payment of items according to the conditions imposed by the winning bids, etc. All buyers are considered tax-contributors so the buyer upper roles set has tax-contributor as element. A possible definition for $K^{(m)}$ for the buyer can be a set of axioms that define market conditions, bank payment policies, item deliveries conditions, etc.

Associated with the concept of role, is the concept of *protocol* which is defined as a sequence of definitions parameterized on the role. Thus, the protocol steps which the agent follows in a scene are directly dependent on the role they assume. Formally, a protocol P is defined by an identifier id , a role r , a set of optional parameters $\phi^{(f)}$ and a body op . The core of the protocols are constructed from variables v replaced by operations during run time, agent invocations, control flow operations, and actions α which have side-effects and can fail. Also there is the possibility of changing operations natural precedence. The control flow operations can be either sequences *then*, non-deterministic choices *or*, or parallel compositions *par*.

MAP^a incorporates the use of variables as a new feature that provides additional flexibility and dynamism to the protocol. While in *MAP* language protocols were statically defined, here the use of variables allows the dynamic assignment of protocols during run time. Now we consider protocols as first order objects and it is possible to deal with situations in which agent behaviour can not be known before-hand but whose definition depends on agents interaction. Following with our example, it is better to introduce a variable for the buyer protocol fragment corresponding to the payment and delivery of items. And during run time, through the exchange of messages, the buyer can decide with the seller the best way to perform them.

Through agent invocation, an agent can change of role (same identifier, different role from the current one), introduce a new instance of agent (different identifier and role), or implement recursion (same identifier and role). A set of axioms K^v is used in this operation to denote the agent private knowledge. For our example we can mention the case of a seller that runs out of items to deliver to the buyer, so it changes its role to buyer applying an agent invocation. It incorporates as private knowledge the price, quantity and characteristic of the items he has to sell to the buyer, so he can try to get some profit buying the same items for less money and resell them.

Actions allow agents to send and receive messages, and to invoke external decision procedures. Also there are null actions. Interaction between agents is performed by the exchange of *messages*. Messages can be sent or received and this exchange happens in a blocking manner. Every message has an associated *performative* which is used to indicate the type of the message and parameters. The origin or destination of a message is specified by an identifier *id* together with a role *r*. We do not assign any fixed semantics to these performatives. In this way, we can represent FIPA-style agent communication [3], e.g. the contract-net protocol, or others. Messages can also contain protocols, and therefore an agent can inform another what steps to take in a given situation, where the protocol is not defined in advance.

About *decision procedures* it is important to note that they provide an interface between the communicative and the rational process of the agent. This makes it straightforward to understand the operation of the protocol without extraneous details, and makes it possible to verify the protocols using automated means, e.g. model checking. For an agent to be able to invoke a decision procedure, it has to be defined for the role the agent is playing, or for any upper role in the hierarchy from where it descends directly or indirectly. We allow decision procedures to access agent private knowledge, scene knowledge and role knowledge corresponding to the current role the agent is playing and the roles knowledges from which the role descend. We fix the restriction that decision procedures can only modify agents private knowledge. This first approach is quite simplistic and allows as to avoid inconsistencies in the sharing knowledge but we have future plans of giving a broader criteria.

3 Conclusions

In this paper we have presented the MAP^a language definition. For space reasons we could only introduce its formal syntax but we have also defined MAP^a formal semantics based on operational relational formalism. The provision of a clean and unambiguous semantics for the language was a primary consideration in the design process. We consider this to be a failing of the formal semantics of FIPA [3], which is expressed in a BDI-like logic. As it is shown in [5–7], the BDI modalities can be interpreted in a number of different ways, meaning that implementations of BDI agents have typically been *ad-hoc* in nature. The practical benefits of introducing MAP^a with a formal syntax and semantic are numerous, we can mention: application of formal deductive proof strategies and combination of MAP^a protocol definitions with other formal systems. As we have already done with genetic algorithms [1], we are currently working on the incorporation of MAP^a protocols into a grammatical framework called eco-grammar systems to study them from a formal language point of view. With regard to future work, we plan to use MAP^a in conjunction with Web Services, to perform service composition on the Semantic Web.

References

1. Dediu A. H., Grando M. A. (2005) Simulating evolutionary algorithms with ecogrammar systems. Proceedings of IWINAC05, Lecture Notes in Computer Science **3562**. Springer Verlag, 112-121
2. Esteva M., Rodryguez J. A., Sierra C., Garcia P., Arcos J. L (2001) On the Formal Specification of Electronic Institutions. Agent-mediated Electronic Commerce. Lecture Notes in Artificial Intelligence **1991**, 126-147
3. Foundation for Intelligent Physical Agents (1999) FIPA specification part 2 - agent communication language. Available at: www.fipa.org.
4. Greaves M., Holmback H., Bradshaw J. (1999) What is a Conversation Policy?. Proceedings of the Workshop on Specifying and Implementing Conversation Policies, Autonomous Agents 99
5. Jennings N. R. (1993) Specification and Implementation of a Belief-Desire-Joint-Intention Architecture for Collaborative Problem Solving. Journal of Intelligent and Cooperative Information Systems **2(3)**, 289-318
6. Rao A. S., Georgeff M. P. (1995) BDI-agents: from theory to practise. Proceedings of ICMAS-95. AAAI Press, 312-319
7. Sadek M. D. (1992) A Study in the Logic of Intention. Proceedings of KR92, 462-473.
8. Shoham Y., Tennenholtz M. (1992) On the Synthesis of Useful Social Laws for Artificial Agent Societies. Proceedings of AAAI-92
9. Vaconcelos W. (2004) Norm Verification and Analysis of Electronic Institutions. DALT-04, 141-155
10. Walton C. (2004) Multi-Agent Dialogue Protocols. Proceedings of the Eighth International Symposium on Artificial Intelligence and Mathematics