

The MAP^a Language of Agent Dialogues

María Adela Grando^{* †}
Research Group on Mathematical Linguistics,
Rovira i Virgili University,
Tarragona, Spain.

Christopher D. Walton[‡]
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 coordination and communication between agents in multi-agent systems. This is accomplished by defining patterns of dialogues between groups of agents, expressed as protocols. Our protocols are flexible and directly executable. Furthermore, our language allow us to specify the connection between communication and knowledge management in a way that is independent of the specific reasoning techniques used.

1. INTRODUCTION

Multi-Agent Systems (MAS) are a popular technique in Artificial Intelligence for effecting reasoning in a distributed manner. These systems are defined from collections of individual agents which are composed together into systems within an environment. Each individual agent is an autonomous and rational entity, and exhibits intelligent behaviour based on interactions with other agents in the environment. It is this intelligent behaviour that distinguishes a MAS from a conventional distributed software system.

An agent in a MAS is intended to perform in a similar way to a human agent. That is, the agent should be capable of acting on behalf of others, either humans or agents. To accomplish this, it is necessary for the agent to be able to *communicate* and *coordinate* with other agents to achieve certain tasks. This inter-operability between agents is surprisingly difficult to achieve. There are a wide range of issues that have to be addressed, including at least the following: The agent must divide the problem into tasks for which ex-

ternal cooperation is required, and find suitable agents to accomplish these tasks. The tasks must be communicated to the external agents, who must in-turn agree to participate. The agent should interact with these external agents to ensure the task is accomplished, and finally interpret the outcome to obtain the results. These issues may be further complicated as the external agents may use different formalisms or have a different world-view from the initiating agent. Furthermore, it is generally not possible to compel another agent to perform any particular task. Instead, it is typically necessary for agents to negotiate to reach a consensus on a given task. The provision of a solution to many of these issues is the focus of this paper.

To address the issues of communication and coordination that we have highlighted, it is useful to take our inspiration from human *social systems*, where these issues are solved in the real world. Following this approach, it is common to treat a group of agents as a *society* [3]. The agents in the society will 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. In return, the agent can itself benefit from the society, e.g. the expertise of other agents. In this societal view, it becomes possible to define conventions for the agents to follow, and the incentives for agents to operate together are made clear. This in-turn makes the issues of communication and coordination more manageable.

In any society, the rules and conventions are expressed as *social norms*. These norms may often be implicit, but are necessary nonetheless. For example, at a conference talk there is an implicit norm that the speaker does the talking while the audience listens. If everyone were to talk at the same time then the conference would break down. Thus, it is these social norms that ensure the smooth running of the society. Returning to MAS, we can readily define social norms for a group of agents to follow. These rules should be defined in such a way to facilitate the successful interaction between agents. However, the rules should not be overly restrictive in that they affect the autonomy of the agents.

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. The protocol can be examined in-advance by an agent to inform their decision on whether to join a particular society. The protocol also acts as a guide for the agents to follow once they are operating within the society. This approach has been successfully adopted in the Conversation Policy (CP) [2] and Electronic Institutions (EI) [1, 4] formalisms, among others.

^{*}The primary author is a Ph.D. student.

[†]Sponsored by research grant “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).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS’06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

In this paper, we define a language which can express protocols. Our protocols define patterns of *dialogue* between groups of agents. If a group of agents follow a particular dialogue then they should reach a particular well-defined outcome. Nonetheless, the protocols contain decision points in which the agents can behave autonomously. The protocols that we define are *executable specifications*. That is, the formalisms that we use to specify our protocols are exactly the same as those used for enactment. This is in contrast to the more traditional means of MAS construction, where an initial specification is refined into an executable model. For example, a MAS specified in a Belief-Desire-Intention (BDI) logic, and implemented in a FIPA-compliant architecture.

Our language defines Multi-Agent Protocols, and is called MAP^a. This language is an extension of the MAP language which we previously presented in [6]. The extensions in this paper make the protocols more flexible and dynamic, but retain the lightweight nature of the previous definitions. In MAP^a, we treat protocols as first-class objects which can be passed between agents as opposed to our previous static definitions. We also permit the organisation of roles in hierarchies, structuring role definitions in a modular and reusable way. Finally, we permit the management of knowledge to be expressed in our protocols, which enables a better connection between the protocol and the agent reasoning processes at the decision points.

2. MAP^a LANGUAGE DEFINITION

MAP^a is a lightweight language-based formalism for the expression of protocols, which express social interactions between groups of agents. The key concepts in MAP^a are *scenes*, and *roles* which we now explain. A scene can be thought of as a bounded space in which a group of agents interact on a single task. We assume that a scene places barrier conditions on the agents, such that a scene cannot begin until all the agents are present, and the agents cannot leave the scene until the dialogue is complete.

The concept of an agent *role* is also central to the definition of our protocols. Agents entering a scene assume a role which they can change during the scene. For example, a hospital scene may involve agents with the roles of *doctor* and *patient*. Roles are defined as a hierarchy, where more specialised roles appear further down in the graph of roles. For example, an agent may initially assume the role *doctor* but may change to the more specialised role *surgeon* during a scene.

A protocol is defined as a sequence of definitions parameterised on the role. Thus, the protocol steps which an agent follows in a scene are directly dependent on the role, or sub-role, which they assume. The role also identifies capabilities which the agent must provide. It is important to note that a protocol only contains operations which are specific to the mechanisms of communication and coordination between agents. 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 [5]. All of the other agent facilities, e.g. the reasoning processes, are encapsulated by *decision procedures* which are external to the protocol. In effect, the decision procedures provide an interface between the communicative and the rational process of the agent. We allow a decision procedures to be private to a particular agent, or shared between agents playing the same role.

Interaction between the agents in a scene 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 identifier which is used to indicate the type of the message and parameters. For convenience, we do not assign any fixed semantics to these messages. However, individual agents can agree on a semantics for a particular scene. In this way, we can readily represent FIPA-style agent communication, e.g. the contract-net protocol.

The final concept in MAP^a is the representation of knowledge management during agent interaction. The language allows us to define *knowledge*, represented as axioms, at a scene level, a role level, or the level of an individual agent. In this way, we can clearly express the exchange of knowledge between one agent and another, and the transition from private to public knowledge of certain facts during protocol execution.

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

Figure 1: MAP^a Language Syntax.

A BNF-style abstract syntax for MAP^a is shown in Figure 1. Superscripts are used to indicate a set, e.g. $P^{(i)}$ is a set with elements P of size i . To illustrate the MAP^a language syntax we now define in Figure 2 a general negotiation protocol. We have used the following abbreviations: IS is an Information Seeker role, IP is an Information Provider role, t and $newt$ are a negotiation topic and new negotiation topic, f and $newf$ are a fact and new fact, a is an answer, and $prot$ is a protocol variable.

There are four parts to the negotiation protocol, parameterised on the roles of Patient, GPDoctor, IS, and IP. The core of the dialogue is between an information seeker (IS) and an information provider (IP). The roles Patient and GPDoctor begin the dialogue. The Patient starts searching for a suitable information provider for a given topic and fact (t, f) and will then initiate a dialogue with this provider. $K^{(b)}$ and $Proc^{(a)}$ stand for Patient knowledge and Patient procedures. $K^{(b)}$ can be defined as the basic knowledge that patients have about illnesses, and $Proc^{(a)}$ can be the set of possible actions to perform in presence of some illness. GPDoctors adopt the role of providers with advanced knowledge of medicine, $K^{(d)}$, and treatments for the known diseases, $Proc^{(c)}$.

The information provider (IP) is defined in four cases separated by the choice operator. The evaluation model of the protocol is such that the protocol will backtrack and retry

agent(id₁, Patient, Proc^(a), K^(b), t, f) =
 id₂=search_doctor(t, f) then (agent(id₁, IS,
 Proc^(a), K^(b) ∪ {answer(t,f)=null}, t, f, id₂)).

agent(id₁, GPDoctor, Proc^(c), K^(d))=
 agent(id₁, IP, Proc^(c), K^(d)).

agent(id₁, IP, Proc^(t), K^(z)) =
 (request(t, f) ⇐ agent(id₂, IS)
 then agent(id₁, IP, ∅, ∅)) or
 (a=get_answer(t, f) then
 inform(t, f, a) ⇒ agent(id₂, IS)) or
 (prot = get_protocol(t, f) then
 what_to_do(t, f, prot) ⇒ agent(id₂, IS)) or
 ((newt, newf) = next_to_know(t, f) then
 agent(id₁, IS, ∅,
 {answer(t,f)=null, answer(newt,newf)=null},
 newt, newf, id₂) then agent(id₁, IP, ∅, ∅, id₂)).

agent(id₁, IS, Proc^(m), K^(s), t, f, id₂) =
 (request(t, f) ⇒ agent(id₂, IP) then
 agent(id₁, IS, ∅, ∅, t, f, id₂)) or
 (inform(t, f, a) ⇐ agent(id₂, IP) then
 agent(id₁, IS, ∅, {answer(t,f)=a}, t, f, id₂) or
 result=Newknowledge({answer(t,f)=a})) or
 (what_to_do(t, f, prot) ⇐ agent(id₂, IP) then
 prot
 then agent(id₁, IS, ∅, {answer(t,f)=prot}, t, f, id₂)
 or result=Newknowledge({answer(t,f)=prot}))
 or (request(newt, newf) ⇐ agent(id₂, IS)
 then agent(id₁, IP, ∅, ∅)
 then agent(id₁, IS, ∅, ∅, t, f, id₂)).

Figure 2: MAP^a Negotiation Protocol.

each of these cases until one of them can be satisfied. In the first case, the information provider receives a request for information (on some topic and fact) from an information seeker. The information provider will then update its state and attempt to answer the request. The appropriate answer is determined in the remaining cases. The second case is triggered when the information provider has a direct answer for the request, in which case this answer is returned to the seeker. In the third case, it is necessary for the seeker to perform additional actions before the request can be satisfied, so the provider sends a protocol to the seeker containing the appropriate actions. In the final case, more information is required, so the provider becomes a seeker for this new information. What he learned as IP is taken as private knowledge for the new role of IS (answer(t,f)=null, answer(newt,newf)=null). Upon obtaining the extra information, the agent reverts back from a seeker to a provider again.

The information seeker (IS) protocol is essentially the complement of the information provider protocol, and there are four corresponding cases. In the first case, the seeker sends a request for information to the provider. The remaining cases deal with the responses from the provider. In the second case, the requested information is received from the provider and the protocol either restarts with the new private knowledge (answer(t,f)=a) or terminates. In the third case, a protocol is received from the provider which is evaluated. Lastly, in the fourth case, a request for more information is received, and the agent assumes the role of a provider until this request is satisfied.

3. CONCLUSIONS

In this paper we have presented an overview of our MAP^a language. The purpose of this language is the specification of the communication and coordination processes between groups of agents in a multi-agent system. This is accomplished by the definition of protocols which express social behaviours between the agents. In particular, these protocols define social norms which agents must observe if they are to participate in, and benefit from, the society. The use of protocols ensures that the interactions between agents in the society happen in a controlled and predictable manner. Nonetheless, these protocols do not overly restrict the individual autonomy of the agents.

Our approach differs from previous work on multi-agent protocols, in a number of important ways. Our protocols are executable specifications, and we have constructed a meta-interpreter for our protocols in Java. This direct execution eliminates a potentially error-prone refinement task between the specification and implementation. Our protocols are defined separately from the reasoning processes of the agents. This enables our protocols to be independently examined and verified before an agent subscribes to the society. Furthermore, it permits a range of reasoning strategies to be adopted, e.g. reactive or planning systems, rather than restricting us to a single model.

With regard to future work, we plan to use MAP^a in conjunction with Web Services, to perform service composition on the Semantic Web. We also plan to use the language with eco-grammars to perform automatic agent-based grammar generation. Our future plans for extensions to the language include facilities for constraint satisfaction and the inclusion of deontic relationships between agents.

4. REFERENCES

- [1] M. Esteva, J. A. Rodríguez, C. Sierra, P. Garcia, and J. L. Arcos. On the Formal Specification of Electronic Institutions. In *Agent-mediated Electronic Commerce (The European AgentLink Perspective)*, number 1991 in Lecture Notes in Artificial Intelligence, pages 126–147, 2001.
- [2] M. Greaves, H. Holmback, and J. Bradshaw. What is a Conversation Policy? In *Proceedings of the Workshop on Specifying and Implementing Conversation Policies, Autonomous Agents '99*, Seattle, Washington, May 1999.
- [3] Y. Shoham and M. Tennenholtz. On the Synthesis of Useful Social Laws for Artificial Agent Societies. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, San Diego, USA, 1992.
- [4] W. Vaconcelos. Norm Verification and Analysis of Electronic Institutions. In *2004 Workshop on Declarative Agent Languages and Technologies (DALT-04)*, pages 141–155, New York, USA, July 2004.
- [5] C. Walton. Model Checking Multi-Agent Web Services. In *Proceedings of the 2004 AAAI Spring Symposium on Semantic Web Services*, Stanford, California, March 2004. AAAI.
- [6] C. Walton. Multi-Agent Dialogue Protocols. In *Proceedings of the Eighth International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida, January 2004.