ABSTRACT
We propose a notation to specify and reason about an expressive class of protocols. Our notation is logic-based, employing a restricted form of first-order constructs added with sets. We formally associate the syntax of our constructs with a precise semantics. The semantics enables us to automatically build explicit models which can be used to restrict the possible executions of a given Multi-Agent System. Our notation has been incorporated to a variant of electronic institutions – an encompassing and flexible approach to specify open agent organisations – and we have implemented a proof-of-concept platform to enact arbitrary protocols using our notation.

Categories and Subject Descriptors
D.2.2 [Software Engineering]: State Diagrams—Multi-Agent Systems.
General Terms
Languages, Design.
Keywords

1. INTRODUCTION
A defining property of a multi-agent system (MAS) is the communication among its components: a MAS can be depicted by the kinds and order of messages its agents exchange [3]. We adopt the view that the design of MASs should thus start with the study of the exchange of messages, that is, the protocols among the agents, as explained in [6]. Such protocols are called global because they depict every possible interaction among all components of a MAS.

We propose a logic-based formalism that allows the representation of a useful class of protocols involving many agents. This formalism combines first-order logic and set theory to allow the specification of interactions among agents, whether an auction, a more sophisticated negotiation or an argumentation framework. We introduce and exploit the logic-based formalism within the context of electronic institutions: these are means to modularly describe open agent organisations. As well as providing a flexible syntax for dialogues, we also formalise their semantics via the construction of models.

Our effort aims at providing MASs engineers with a notation for specifying interactions among the components of a MAS, but which allows relationships to be forged among the interactions. A typical interaction we are able to express in our formalism is “all seller agents advertise their goods; after this, all buyer agents send their offers for the goods to the respective seller agent”. In this interaction, it is essential that the buyer agents send offers to the appropriate seller agents, that is, each seller agent should receive an appropriate offer to the good(s) it advertised.

2. A SET-BASED LOGIC FOR PROTOCOLS
We aim at a broad class of protocols in which many-to-many interactions (and, in particular, one-to-one, one-to-many and many-to-one) can be formally expressed. The protocols are global in the sense that they describe any and all interactions that may take place in the MAS. We define our set-based logic $\mathcal{L}$ as:

DEF. 1. $\mathcal{L}$ consists on formulae $Qf$ ($Atfs \Rightarrow SetCtrs$) where $Qf$ is the quantification, $Atfs$ is a conjunction of atomic formulae and $SetCtrs$ is a conjunction of set constraints.

$Qf$ provides us constructs with universal and existential quantification over (finite) sets; $Atfs$ expresses atomic formulae that must hold true and $SetCtrs$ represents set constraints that are made to hold true. The set constraints are restrictions on set operations such as union, intersection, and so on:

DEF. 2. Set constraints are conjunctions of set operations, defined by the following grammar:

$$SetCtrs \rightarrow SetCtrs \land SetCtrs \mid \{SetCtrs\} \mid MTest \mid SetProp$$

$$MTest \rightarrow \text{Term} \in \text{SetOp} \mid \text{Term} \notin \text{SetOp}$$

$$SetProp \rightarrow \text{card}(\text{SetOp}) \text{ Op N} \mid \text{card}(\text{SetOp}) \text{ Op card}(\text{SetOp})$$

$$\text{SetOp} = \text{SetOp}$$

$$\text{Op} = \mid > \mid > \mid > \mid < \mid < \mid < \mid <$$

$$\text{SetOp} \rightarrow \text{SetOp} \cup \text{SetOp} \mid \text{SetOp} \cap \text{SetOp} \mid \text{SetOp} \setminus \text{SetOp} \mid \text{SetOp} \times \text{SetOp} \mid \text{card}(\text{SetOp}) \mid \text{Set} \mid \emptyset$$

$MTest$ is a membership test, that is, a test whether an element belongs or not to the result of a set operation $\text{SetOp}$ (in particular, to a specific set). $SetProp$ represents the set properties, that is, restrictions on set operations as regards to their size (card) or their contents. $N$ is the set of natural numbers. $\text{Op}$ stands for the allowed operators of the set properties. $\text{SetOp}$ stands for the set operations, that is, expressions whose final result is a set. An example of a formula of $\mathcal{L}$ is $\exists B \in \text{Ags}(m(B, \text{adm}, \text{enter}(\text{buyer}))) \Rightarrow (B \in \text{Bs} \land 1 \leq |\text{Bs}| \leq 10))$.

We define the semantics of our formulae as below:
**DEF. 3.** An interpretation $\mathfrak{I}$ for Fml is the pair $\mathfrak{I} = (\sigma, \Omega)$ where $\sigma$ is a possibly empty set of ground atomic formulae (i.e. af$s$ without variables) and $\Omega$ is a set of sets.

Intuitively our interpretations provide in $\sigma$ what is required to determine the truth value of $Qtf(Atfs)$ in $\Omega$ what is needed in order to assign a truth value to $Qtf(\text{SetCtrs})$.

The semantic mapping $k : \text{Fml} \times \mathfrak{I} \mapsto \{ \top, \bot \}$ is defined as, for instance, $k(Atfs_1 \wedge Atfs_2, \mathfrak{I}) = \top$ iff $k(Atfs_1, \mathfrak{I}) = k(Atfs_2, \mathfrak{I}) = \top$, for conjunctions; the remaining operators are defined similarly [5]. The auxiliary mapping $k' : \text{SetOp} \times \mathfrak{I} \mapsto \text{Set}$ in $\Omega, \mathfrak{I} = (\sigma, \Omega)$, referred to above and which gives meaning to the set operations is defined as, for instance, $k'((\text{SetOp}_1 \cup \text{SetOp}_2), \mathfrak{I}) = \{ e \mid e \in k'(\text{SetOp}_1, \mathfrak{I}) \text{ or } e \in k'(\text{SetOp}_2, \mathfrak{I}) \}$. The remaining set operations are defined similarly [5].

We are interested in models for our formulae, that is, interpretations that map Fml to the truth value $\top$ (true). We are only interested in those interpretations in which both sides of the “⇒” in the Fml’s hold true. Formally:

**DEF. 4.** An interpretation $\mathfrak{I} = (\sigma, \Omega)$ is a model for $\text{Fml} = Qtf(Atfs \Rightarrow \text{SetCtrs})$, denoted by $\mathfrak{m}(\text{Fml}, \mathfrak{I})$ iff $\sigma$ and $\Omega$ are the smallest possible sets such that $k(Qtf Atfs, \mathfrak{I}) = k(Qtf \text{SetCtrs}, \mathfrak{I}) = \top$.

3. **LOGIC-BASED E-INSTITUTIONS**

In the same way that social institutions are somehow forged, the laws that should govern the interactions among heterogeneous agents can be defined by means of electronic institutions (e-institutions, for short) [2]. E-institutions are non-deterministic finite-state machines describing possible interactions among agents.

We have incorporated our proposed logic $\mathcal{L}$ to the definition of e-institutions. In this combination, constructs of $\mathcal{L}$ label edges of finite-state machines. This allows for precisely defined and expressive edges thus extending the class of e-institutions one can represent. Furthermore, by embedding $\mathcal{L}$ within e-institutions, we can exploit the model-theoretic issues in an operational framework.

4. **CONCLUSIONS AND FUTURE WORK**

We have proposed a formalism to represent global protocols, that is, all possible interactions among components of a multi-agent system, from a global perspective. The formalism is $\mathcal{L}$, a set-based restricted kind of first-order logic that allows engineers to describe a protocol and to forge relationships among messages of one-to-one, one-to-many and many-to-many interactions.

We have put this formalism to work by embedding it within the definition of electronic institutions [1], giving rise to $\mathcal{L}$-based electronic institutions. Existing formulations of electronic institutions resort to informal explanations when defining the meaning of their constructs. Our rendition, on the other hand, has its syntax and semantics formally defined using $\mathcal{L}$.

We have incorporated the concepts above into a distributed enactment platform. This platform, implemented in SICStus Prolog [4], uses the semantics of our constructs to perform a simulation of an e-institution. The platform relies on a number of administrative agents, implemented as independent processes, to overlook the enactment, building models and interacting with the agents partaking the enactment via a blackboard architecture, using SICStus Linda tuple space.

Our proposal has been exploited for rapid prototyping of large Multi-Agent Systems [8].

Our implementation does not take into account message loss or delays. We also assume that there are no malign agents intercepting messages and impersonating other agents. Our platform can be seen as an idealised correct version of a multi-agent system to be built, whereby the performing agents stands for “proxies” of foreign heterogeneous agents, guaranteed to follow an e-institution. The practical security issues that actual heterogeneous agents are prone to are not transferred on to the e-institution platform. We are working on how agents synthesised from the e-institution specification [7] could be presented to foreign agents and customised as their proxy agents.

**Acknowledgements:** This work was partially sponsored by the European Union, contract IST-1999-10208, research grant Sustainable Lifecycles in Information Ecosystems (SLIE).

**REFERENCES**


