Implementing Norms in Electronic Institutions (Extended Abstract)

A. García-Camino[1] P. Noriega[1] J. A. Rodríguez-Aguilar[1]

[1] IIIA-CSIC, Campus UAB 08193 Bellaterra Spain

{andres,pablo,jar}@iiia.csic.es

Ideally, open multi-agent systems (MAS) involve heterogeneous and autonomous agents whose interactions ought to conform to some shared conventions. The challenge is how to express and enforce such conditions so that truly autonomous agents can adscribe to them. One way of addressing this issue is to look at MAS as environments regulated by some sort of normative framework. There have been significant contributions to the formal aspects of such normative frameworks, but there are few proposals that have made them operational. In this paper a possible step towards closing that gap is suggested. A normative language is introduced which is expressive enough to represent the familiar types of MAS-inspired normative frameworks; its implementation in JESS is also shown. This proposal is aimed at adding flexibility and generality to electronic institutions by extending their deontic components through richer types of norms that can still be enforced on-line.

The objective of this paper is to try to fill this gap by adapting a formal approach [5] to enrich an existing framework [1] . More precisely, we have extended the normative language of an EI to increase its expressiveness and flexibility.

It is worth remarking that we consider norm compliance from an institutional point of view. That is, we do not care how an agent decides which norms to comply with, but instead we define the norms and sanctions to be applied when the violation of norms occurs as part of the institution. With this approach we allow agents to reason about norm compliance while the choice and implementation of agents' architectures is left to agent developers.

We define a normative language to specify obligations, permissions, prohibitions, violations and sanctions to restrict agents' dialogical actions. This normative language can be used as an extension of the normative rules of the current version of electronic institutions obtaining a higher degree of expressiveness and flexibility. We also implement a norm engine which maintains the normative state of an institution, i.e. the permissions, prohibitions and pending obligations that hold in the current state of execution.

We extend the normative language recently proposed in [5]. That proposal is enriched with new types of norms, namely norms that we keep active during a time interval, and conditional norms over the institutional state, (e.g. the observable attributes of agents and objects of the environment). Moreover, our extension of that language includes the possibility to sanction agents by modifying their institutional state, i.e. their observable attributes[1]. Nonetheless, since in EIs alls actions are speech acts, actions expressed by the language are limited to the utterance of illocutions.

Once we define the normative language, we need to handle the normative state of an institution. A rule-based system is chosen to implement norms because the normative language is of the form *preconditions* $\rightsquigarrow$ *postconditions*, which is easily expressable with rules. In order to facilitate the integration with AMELI [1] we decided to implement this tool with Jess since both are written in Java.

We propose the use of Jess [4] for the implementation of the norm engine which maintains the normative state of an institution, i.e. the permissions, prohibitions and obligations that hold in the current state of execution. Our implementation has been carried out by translating the norms specified in our normative language into Jess rules. At run-time our norm engine can be updated with new utterances and queried about permissions, prohibitions or pending obligations.

To introduce utterances, permissions, prohibitions and obligations in the norm engine, a translation from our language into Jess rules is needed. Now we introduce part of the criteria presented

[1] For a entire formalization of the normative language we address the reader to the full paper [3].

in [3] to be followed to carry out this translation:

We define four types of Jess unordered facts: `O`, `P`, `F` and `V` that stand, respectively, for obligations, permissions, prohibitions and violations.

- *Conditional norms* are those norms that include an IF section. The translation of IF sections is directly realised by placing the conditions in the LHS of a Jess rule.
- *Action-dependent norms* are those norms that include a BEFORE, AFTER or BETWEEN section followed by an action. For example, to translate an obligation to be fulfilled before the utterance of an illocution i_1, we add a rule that asserts a violation fact if illocution i_1 has been uttered but the obliged illocution has not.
- *Time-dependent norms* are those norms that include a BEFORE, AFTER or BETWEEN section followed by a date. To translate rules with temporal constraints (i.e. the BEFORE, AFTER and BETWEEN constructs with time objects) into Jess rules we use the user-defined function `(set-deadline ?deadline ?rule)` where ?*deadline* is an absolute date object indicating when the rule fires and ?*rule* is a string-based representation of a rule. In this way the `set-deadline` function adds the given rule to the Jess engine only when the specified absolute date arrives.

There are some differences between our normative proposal and other recent ones, the more salient are that we do not make the strong assumption (as in, for example, [5]) that there is a prohibition before an action iff there is a permission after that action, and we do have a working proof of concept implementation of a computationally feasible framework. With respect to other implementation proposals for normative frameworks, ours is more expressive in the sense that other implementations do not include temporal aspects in the definition of norms and, in the test of conditional norms or in the application of sanctions, fail to consider observable agent attributes or attributes of objects in the environment.

As to future work, we intend to produce an upward compatible extension of the EIDE environment through the addition of an automatic translation module to map our normative language into Jess rules and integrates our norm engine with AMELI. Furthermore, we are extending the notions above in the formalization and development of a norm-based programming language [2].

Acknowledgments

The present paper was funded by the Spanish Science and Technology Ministry as part of the Web-i-2 project (TIC-2003-08763-C02-00). A. Garcia-Camino enjoys an I3P grant of the Spanish Council for Scientific Research (CSIC).

References

[1] M. Esteva, B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos. AMELI: An Agent-Based Middleware for Electronic Institutions. In *Procs. AAMAS 2004*, 2004.

[2] A. García-Camino, J.A.Rodriguez-Aguilar, C. Sierra, and W. Vasconcelos. A distributed architecture for norm-aware agent societies. In *Proceedings of the Declarative Agent Languages and Technologies (DALT) workshop*, Utrecht, July 2005.

[3] A. García-Camino, P. Noriega, and J. A. Rodríguez-Aguilar. Implementing Norms in Electronic Institutions. In *4th Int'l Joint Conf on Autonomous Agents and Multiagent Systems (AAMAS)*, 2005.

[4] Jess. The Rule Engine for Java. Sandia Nat'l Labs. `http://herzberg.ca.sandia.gov/jess`, Oct. 2005.

[5] J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Norms in Multiagent Systems: Some Implementation Guidelines. In *2nd European Workshop on Multi-Agent Systems*, Barcelona, 2004.