Designing Protocols for Agent Institutions*

Huib Aldewereld†
Institute of Information and Computing Sciences
Utrecht University
The Netherlands
huib@cs.uu.nl

Frank Dignum
Institute of Information and Computing Sciences
Utrecht University
The Netherlands
dignum@cs.uu.nl

John-Jules Ch. Meyer
Institute of Information and Computing Sciences
Utrecht University
The Netherlands
jj@cs.uu.nl

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*

General Terms

Theory, Legal Aspects, Design

Keywords

Norms, Electronic Institutions, Normative Systems

1. INTRODUCTION

Agent-mediated institutions (or e-institutions), introduced in [9], are *open* agent systems that allow heterogeneous agents to enter and perform tasks. The e-institutions specify the admissible behaviour of the agents by means of norms, which are declarative and abstract by nature. On the one hand this allows for a stable specification suitable for almost any conceivable situation that arises in the institution, but in the other hand the norms hardly give any indication which interaction patterns would guarantee satisfaction of the norms.

One could aim for the use of normative agents in e-institutions as propagated in [8]. In this perspective the agents are capable of reasoning about the norms and planning their actions accordingly. However, it seems not very realistic that all agents will have this capacity. Most agents will be standard agents that are only capable to reason about standard protocols as part of their interactions with other agents. Therefore, in this paper, we aim to provide ways to generate protocols (on the basis of the normative descriptions) that are guaranteed to fulfil all norms of the e-institution. Given these protocols agents entering the e-institution can just follow these protocols and be sure they will always stay within the "law". Note that we do not necessarily require the agents to follow these protocols. They can be seen as available templates for use by the agents. Agents can still perform normative reasoning if they are capable, but, with the help of these protocols, they do not need to do that in order to participate in the e-institution.

Our approach is inspired by how the gap between the normative and procedural dimensions is bridged in human institutions. Human laws express in a very abstract way wanted (legal) and unwanted (illegal) states of affairs. Although laws are very expressive, they do not express how to achieve a given state of affairs, and therefore they are very hard to use in practice to, e.g., guide each decision in a process. In practice more efficient representations are needed, such as protocols or guidelines. In rule-based legal systems (those based in Roman-Germanic law), *regulations* add an intermediate level between laws and practice, by giving some high-level specifications on some constraints about how things can or cannot be done. These high-level descriptions are therefore interpretations of the law that add some operational constraints to be met by practice. Using this idea, we introduce an intermediate level between institutional norm specifications and institutional protocols based on *landmarks* in a similar way as done in [7, 5, 3].

From the norms we will automatically generate finite state automata, using a technique introduced by Wolper in [10]. This is a general technique to convert temporal logic formulas in LTL into so-called Büchi automata. The landmark patterns can be obtained from these automata by looking at their corresponding recognised languages.

2. FROM NORMS TO PROTOCOLS

The relevance of landmarks in protocol specification is dictated by the simple observation that several different actions can bring about the same outcome. Once the outcomes of actions are organised in a structured description (a landmark pattern), it becomes possible to represent families of protocols abstracting from the actual transitions by which each protocol is constituted. This makes landmarks an ideal solution for bridging the gap between the abstract normative level and the procedural level of protocols. The landmark pattern fully captures the order in which states should occur, representing the important steps that any protocol should contain, while still abstracting from the actual procedural information on how the transition from one state to another should be achieved. In essence, a landmark pattern represents "those steps that should be taken and in which order".

Similar to the relation between norms, regulations and

*This is a student paper.

†The first author of this paper was supported by the Netherlands Organisation for Scientific Research (NWO) under project number 634.000.017

AAMAS'07, May 14–18, 2007, Honolulu, Hawai'i, USA.

978-81-904262-7-5 (RPS) ©2007 IFAAMAS

practice, where regulations add operational information to the restrictions given by the norms, a landmark pattern should add additional information to the normative goals in order to bridge the gap between the norms and the practice. The norms only give a (temporal) ordering of the states that should be reached. The normative landmark pattern extracted from these norms will thus leave many blanks, situations where the order of events/actions is undetermined by the norms or only minimally described. The choices in ordering that are not specified in the pattern, however, can influence the efficiency or even the feasibility of the pattern.

Even though all kinds of information concerning efficiency and feasibility can be added to the landmark pattern, some limitations still exist that should not be overlooked. The pattern will have to satisfy certain principles to be useful. Firstly, the pattern needs to be norm-compliant. No landmarks should be added that are in conflict with the requirements specified by the norms. Secondly, the pattern needs to contain only reachable landmarks; the pattern should not contain landmarks that are unachievable by definition (e.g. a state satisfying $a \wedge \neg a$), or express an ordering that is impossible to fulfil. Lastly, a landmark pattern should only express goals that are within the capabilities of the agents.

Creating a protocol for a normative domain is done by using the intermediate level of landmarks presented above. The process of generating a (proto-typical) protocol for a normative domain is the following. First a set of landmarks is extracted from the norms governing the domain. To extract this (normative) landmark pattern from the norms we use a technique presented in [10], which was originally developed with model-checking in mind (some model-checkers, like SPIN, [6], are built around principles very similar to those of this technique). The idea is that we create a generic, canonical-like model representing *all* LTL models that satisfy the norms of the system. This canonical model is, in fact, a finite state machine. From this finite state machine we generate a *regular expression* expressing the characteristic features of all models satisfying the norms. This expression is, in fact, a basic landmark pattern, exactly containing all the important states (landmarks) expressed in the norms, as well as the order in which these states must occur (thus making it a landmark pattern). This normative pattern will then be expanded with extra landmarks to strengthen it to a full landmark pattern (as described above). The landmark pattern, now including all important states, both normative-wise and efficiency-wise, will then be translated into a protocol for the normative domain.

2.1 Example

Let us discuss this procedure in a bit more detail by the use of an example. Consider, for illustrative purposes, a domain governed by a single deadline that ρ needs to happen before δ happens. This norm is expressed in linear-time temporal logic (similar to [5]). Details about the representation of norms in LTL can be found in [1]. The deadline of our example will be represented as follows:

$$O(\rho < \delta) \Leftrightarrow \Diamond\delta \wedge \Big[(\neg\delta \wedge \neg\rho \wedge \neg v(\rho,\delta)) \textbf{ until } ((\rho \wedge \neg\delta \wedge \Box\neg v(\rho,\delta)) \vee (\neg\rho \wedge \delta \wedge \bigcirc v(\rho,\delta)))\Big]$$

Using the relation between LTL and Büchi automata (Büchi automata are automata over infinite words, see [10, 1] for more details) presented in [10, 1] we can create an automaton that models all LTL sequences that satisfy the norms of a given domain.

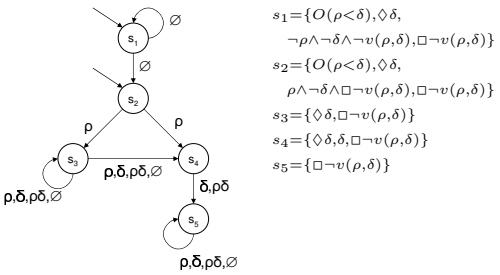

Figure 1: A Büchi automaton for $O(\rho < \delta)$.

The landmarks are then extracted from the norms by creating an automaton (which, in fact, is a general, canonical-like model of the norms, since it represents all LTL sequences satisfying that set of norms) by the procedure described in [10, 1]. However, building an automaton on basis of the logical representation of the norms gives a model of the norms that also includes the LTL sequences where (one of the) norms have been violated, since the violation of a norm is just as much a part of the logical representation because of its prescriptive nature (norms express what should be, not what is). Instead, we are more interested in only those LTL sequences where the norms hold but are not violated, since these sequences characterise the patterns that we want to capture in the protocol. In case of our example, this means we need to build an automaton for the formula $O(\rho < \delta) \wedge \Box\neg v(\rho,\delta)$. The resulting automaton A_φ is shown in figure 1. The alphabet of the automaton is taken as $\Sigma_\varphi = 2^{\{\rho,\delta,v(\rho,\delta)\}}$, the states $s_1, \ldots, s_5 \subseteq 2^{cl(\varphi)}$ (parts of the state labels are given in figure 1), the starting states of the automaton are s_1 and s_2, and the acceptance set is defined as $\mathcal{F}_\varphi = \{\{s_4, s_5\}, \{s_2, s_4, s_5\}\}$.

As can be seen, this automaton exactly represents our intuition of a deadline (as expressed in LTL logics). All LTL sequences satisfying a deadline, should have a number of states (possibly zero) in which nothing interesting happens (this is represented in state s_1). Then a state occurs where ρ holds (state s_2), after which, one or more states later, δ holds (the intermediate states are represented in state s_3, the state where δ occurs is represented in state s_4). After the obligation has been fulfilled (δ has happened, while ρ happened one or more states before δ), an infinite sequence of states occurs where anything can happen as long as $v(\rho,\delta)$ does not happen; this is represented in state s_5 (since no more restrictions are posed on ρ and δ, they can hold in any order at any state after the deadline has been fulfilled). Note that, as required, none of the states satisfy $v(\rho,\delta)$. To fulfil the Büchi acceptance condition the sequences before ρ has happened (i.e. the transition from s_1 to s_1), and the sequence before δ happens (i.e. the transition from s_3 to s_3), can only be of finite length, the only infinite recursion in this automaton is the transition from s_5 to itself.

After translating the norms to a Büchi automaton we can extract a regular expression characterising the language expressed by the automaton. The idea is that the main characteristics, the basic landmark structure, obtained from the norms that are represented in the Büchi automaton, can be

easily extracted through this regular expression.

For the automaton presented in figure 1, we can express the accepted language by the following regular expression (we use *all* to denote $(\varnothing + \rho + \delta + \rho\delta)$):

$$L_\omega(A_\varphi) = \varnothing^*.\rho.all^*.(\delta + \rho\delta).all^\omega.$$

As can be seen from this regular expression, the points of interest of every LTL sequence satisfying φ are the state where ρ holds and the state where either δ or $\rho\delta$ holds (the former being the state where $\delta \wedge \neg\rho \wedge \neg v(\rho, \delta)$ holds, the latter where $\delta \wedge \rho \wedge \neg v(\rho, \delta)$ holds). As seen in the expression, confirming the intuition about deadlines, all LTL sequences satisfying $O(\rho < \delta)$ have a state where ρ holds (possibly preceded by a number of states where neither ρ nor δ hold), which always happens before a state occurs where δ holds (the regular expression allows a number of states, possibly zero, between the occurrence of ρ and δ). The fact that the state where δ should hold is denoted in the expression as $\delta + \rho\delta$ is mainly because, since the restriction of ρ at least happening before δ has already been met, at this point it does not matter whether ρ holds (basically, the transition label $\delta + \rho\delta$ expresses no information concerning ρ or $\neg\rho$, but expresses that *at least* δ holds).

Given that we can deduce from $\delta + \rho\delta$ that at least δ holds, we can simplify the regular expression to the landmark pattern $\mathfrak{L}_\varphi = \langle\{\rho, \delta\}, \{\rho \leq \delta\}\rangle$. This is the normative landmark pattern extracted from the norms of the domain (just $O(\rho < \delta)$ in this case). This landmark pattern is the basis of the landmark pattern that we construct to create protocols. The landmark pattern can now be extended with additional landmarks to denote domain-specific information to increase the efficiency and feasibility of the pattern.

Given that the landmark pattern expresses states that should be achieved, it can be viewed as a collection of goals and the order in which these goals must be achieved. A translation from a landmark pattern to a basic, prototypical protocol is then achieved by means of use of *seeing to it that* operators (*stit*) [4]. While the *stit* operator ignores the means by which an agent will bring about a state of affairs, it does provide the link to make states (the landmarks) into procedural goals. It is then possible to create a protocol given this specification of goals (while retaining the order in which they should be achieved) by linking these goals to the capabilities of agents via a planning algorithm.

It might be possible that a landmark expresses a state that is not reachable by a single agent alone; a cooperation of agents might be needed to achieve (parts of) the landmark pattern. In this case, methods for designing interaction patterns between agents, such as the ones in OPERA, [5], can be applied to create the necessary interaction protocols for reaching those complex goals.

3. CONCLUSIONS

In this paper we have shown a procedure to derive a basic protocol from norms described as obligations with deadlines. This procedure can be used to create protocols for e-institutions governed by norms such that agents that follow these protocols will always fulfil all the norms of the e-institution.

We have not shown in this paper how the process can be conducted using multiple norms. Basically, this process is a very simple extension of the above described procedure. The different norms are all described in LTL. We can combine the norms in LTL by just taking the conjunction of them. If more knowledge is available on the relation between the norms that is not explicit in the norms themselves this can also be added in the LTL description. Often this amounts to temporal orderings between deadlines of the norms that derive from common-sense knowledge. E.g. asking consent for organ donation from family of a donor should be done after the donor has died (not before). The resulting formula can then be processed as before again.

Another point that we could not expose in depth due to space limitations is the construction of a protocol involving multiple parties. Mainly what comes out of the procedure above is a set of states that should be reached by the agents. If a state can only be reached by a coordinated action of several agents we can use techniques from MAS planning to create an interaction pattern to reach that state. Although not at all trivial we assume that existing techniques suffice to bridge this gap.

Finally, we have not formally shown that the resulting protocols indeed satisfy the norms. This can be done by verifying that the protocol will never lead to a violation state. In [2] it has been shown that it is indeed possible to perform this exercise, therewith ensuring the correctness of the complete procedure.

4. REFERENCES

[1] H. Aldewereld. *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*. PhD thesis, Universiteit Utrecht, 2007.

[2] H. Aldewereld, F. Dignum, J.-J. Meyer, and J. Vázquez-Salceda. Proving norm compliancy of protocols in electronic institutions. Technical Report UU-CS-2005-010, Utrecht University, 2005.

[3] H. Aldewereld, D. Grossi, J. Vázquez-Salceda, and F. Dignum. Designing normative behaviour via landmarks. In O. Bossier. et.al., editor, *Coordination, Organisation, Institutions and Norms in Agent Systems I*, pages 150–162. Springer-Verlag, 2006.

[4] N. Belnap and M. Perloff. Seeing to it that: a canonical form for agentives. *Theoria*, 54:175–199, 1988.

[5] V. Dignum. *A Model for Organizational Interaction: Based on Agents, Founded in Logic*. PhD thesis, Universiteit Utrecht, The Netherlands, 2004.

[6] G. Holzmann. The model checker spin. *IEEE Transactions On Software Engineering*, 23(5):279, 1997.

[7] S. Kumar, M. J. Huber, P. R. Cohen, and D. McGee. Toward a formalism for conversation protocols using joint intention theory. *Computational Intelligence*, 18(2):174–228, 2002.

[8] F. López y Lopez, M. Luck, and M. d'Inverno. A framework for norm-based inter-agent dependence. In *Proceedings of the 3rd Mexican International Conference on Computer Science*, pages 31–40. SMCC-INEGI, 2001.

[9] J. Rodriguez. *On the Design and Construction of Agent-mediated Electronic Institutions*. PhD thesis, Inst. d'Investigació en Intel.ligència Artificial, 2001.

[10] P. Wolper. Constructing automata from temporal logic formulas: A tutorial. In *Lectures on Formal Methods and Performance Analysis*, number 2090 in LNCS, pages 261–277. Springer-Verlag, New York, 2002.