Specifying and reasoning about normative systems in deontic logic programming

(Ricardo Gonçalves, José Júlio Alferes)

CENTRIA & Dep. Informática, Faculdade Ciências e Tecnologia, Universidade Nova de Lisboa, Portugal

ABSTRACT

In this paper we propose the usage of a framework combining standard deontic logic (SDL) and non-monotonic logic programming – deontic logic programs (DLP) – to represent and reason about normative systems.

Categories and Subject Descriptors

I.2 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods

General Terms

Languages, Theory

Keywords

Norms, Knowledge representation, Organisations and institutions, Logic-based approaches and methods, Design languages for agent systems

1. INTRODUCTION

Normative systems have been advocated as an effective tool to regulate interaction in multi-agent systems. Essentially, norms encode desirable behaviours for a population of a natural or artificial society. In general, they are commonly understood as rules specifying what is expected to follow (obligations, permissions, ...) from a specific set of facts. Moreover, in order to encourage agents to act according to the norms, normative systems should also be able to specify the application of rewards/sanctions.

Deontic logic [20] deals precisely with the notions of obligation and permission, and it is, therefore, a fundamental tool for modeling normative reasoning. The modal logic KD has emerged as the Standard Deontic Logic (SDL) [3].

Although necessary, SDL has shown not to be sufficient for the task of representing norms [4]. For instance, it is well known its inability to deal with some paradoxes, namely those involving the so-called contrary-to-duty obligations. The main difficulty of SDL is the fact that classical implication does not provide a faithful representation for the conditional obligations that usually populate a normative system.

DEFINITION 1. A deontic logic program is a set of rules $\varphi \leftarrow \psi_1, ..., \psi_n, \text{not } \delta_1, ..., \text{not } \delta_m$ (1) where each of $\varphi, \psi_1, ..., \psi_n, \delta_1, ..., \delta_m$ is an SDL formula.

As usual, the symbol $\leftarrow$ represents rule implication, the symbol $\&$ represents conjunction and the symbol \text{not} represents default negation. A rule as (1) has the usual reading that $\varphi$ should hold whenever $\psi_1, ..., \psi_n$ hold and $\delta_1, ..., \delta_m$ are not known to hold.

Note that, contrarily to some works in the literature [6, 10, 7], deontic formulas can appear both in the head and in the body of a rule, and they can be complex formulas rather than just atomic formulas. This extra flexibility is fundamental, for example, to deal with non-compliance and application of sanctions.

A normative system is usually understood as a set of rules that specify what obligations and permissions follow from the task of representing norms.
a given set of facts, and, moreover, that specify sanction and/or rewards. In our approach, we use the deontic logic programs to represent normative systems.

**Definition 2.** A normative system $N$ is a deontic logic program.

In order to allow agents and institutions to reason about a normative system, it is very important that it has a rigorous formal semantics which, at the same time, should be clean and as simple as possible. We endow our rich normative language with a declarative semantics, by defining a stable model based semantics [21] for deontic logic programs. The definition of such a semantics for deontic logic programs is not straightforward due to their complex language where, instead of atoms, we can have complex SDL formulas in the head and body of rules. The problem is that, contrarily to the case of atoms, these formulas are not independent. To overcome this difficulty we need to define a notion of interpretation that accounts for such interdependence between these “complex atoms”. The key idea of taking theories of SDL as interpretations, contrasted with the usual definition of an interpretation as any set of atoms, allows the semantics to cope with the interdependence between the SDL formulas appearing in the rules. This construction of a stable model semantics for deontic logic programs can be seen as a special case of the general construction of [5] for parametrized logic programs in which SDL is taken as the parameter logic.

The thus obtained normative language is quite expressive, and can be shown to embed extant approaches such as an important fragment of input-output logic [10]. The fact that our language has a purely declarative semantics also allows us to have several interesting properties. First of all, the agents (the ones that are subject to the normative system), the modeler (the one that writes down the norms) and the electronic institution (the one responsible for monitoring the agents and applying the sanctions/rewards) can all reason about the normative system in a simple and clear way. Moreover, in this semantics we can define the fundamental notion of equivalence between normative systems, and, what is the more, we are able to define a logic in which we can verify equivalence of normative systems using logical equivalence.

The results achieved open very interesting paths for future research. An example is the use of abductive reasoning over our stable model semantics to allow agents to plan their interaction with the normative system, in order, for example, to avoid sanctions. Being declarative, our normative framework could easily be integrated in normative multi-agent system that use declarative languages for modeling norms [17, 8], allowing an important increasing of expressivity of these norm languages. Although this is not the main focus of such systems, it was realized, viz. [18], the need for more expressive declarative norm languages.

Other interesting topics for future work include the study of how tools for updating logic programs could be used for the fundamental problem of updating normative systems, and how to define a well-founded based semantics for DLP, that is a sound skeptical approximation of the stable model semantics with more favorable computational complexity.

### 2. ACKNOWLEDGMENTS

This work was partially supported by FCT Project ERRO PTDC/EIA-CCO/121823/2010. R. Gonçalves was partially supported by FCT Grant SFRH/ BPD/ 47245/ 2008.

### 3. REFERENCES


