# Specifying and reasoning about normative systems in deontic logic programming

## (Extended Abstract)


Ricardo Gonçalves
rjrg@fct.unl.pt

José Júlio Alferes
jja@fct.unl.pt

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.


**Appears in:** *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.




Works using dyadic modal logics [19, 9] and input-output logic [11] are examples of approaches that model conditional obligations in order to have a behavior more reasonable than SDL in the face of the aforementioned paradoxes.

Another fundamental ingredient for modeling norms is the ability to express defeasible knowledge. This is important for representing exceptions, which are very common in normative rules. Several approaches using non-monotonic logics where applied to the problem of representing and reasoning about norms [16, 14, 1, 2].

For all the reasons aforementioned, the representation and reasoning about normative systems would greatly benefit from a framework combining deontic logic and rule based non-monotonic reasoning. We thus propose a language for representing and reasoning about normative systems that combines deontic logic with non-monotonic logic programming. Several features distinguish our approach from other formalisms, viz. [13, 12, 15, 6, 7, 10], that combine deontic operators with non-monotonic reasoning. First of all, we have a rich language, which allows complex deontic logic formulas to appear in the body and in the head of rules, combined with the use of default negation. Moreover, at the level of the semantics, we endow the normative systems with a purely declarative semantics, which stems from a well-known semantics: the stable model semantics of logic programs.

The fundamental notion in our framework is that of a deontic logic program. This is composed by rules that resemble usual logic program rules but where complex SDL formulas can appear in the place where only atoms were allowed.

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

As usual, the symbol $\leftarrow$ represents rule implication, the symbol "," represents conjunction and the symbol *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 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 $\mathcal{N}$* 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

[1] G. Boella, G. Governatori, A. Rotolo, and L. van der Torre. A logical understanding of legal interpretation. In F. Lin, U. Sattler, and M. Truszczynski, editors, *KR*. AAAI Press, 2010.

[2] G. Brewka. Well-founded semantics for extended logic programs with dynamic preferences. *J. Artif. Intell. Res. (JAIR)*, 4:19–36, 1996.

[3] B. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[4] R. Chisholm. Contrary-to-duty imperatives and deontic logic. *Analysis*, 24(2):33–36, 1963.

[5] R. Gonçalves and J. Alferes. Parametrized logic programming. In T. Janhunen and I. Niemelä, editors, *Logics in AI – JELIA*, volume 6341 of *LNCS*, pages 182–194. Springer, 2010.

[6] G. Governatori and A. Rotolo. Bio logical agents: Norms, beliefs, intentions in defeasible logic. *Autonomous Agents and Multi-Agent Systems*, 17(1):36–69, 2008.

[7] J. F. Horty. Deontic logic as founded on nonmonotonic logic. *Ann. Math. Artif. Intell.*, 9(1-2):69–91, 1993.

[8] J. Hubner, J. Sichman, and O. Boissier. Developing organised multiagent systems using the moise+ model: programming issues at the system and agent levels. *Int. J. Agent-Oriented Softw. Eng.*, 1:370–395, 2007.

[9] D. Lewis. *Semantic analyses for dyadic deontic logic*. Cambridge University Press, 1999.

[10] D. Makinson, Leendert, and V. D. Torre. Input-output logics. *Journal of Philosophical Logic*, 29:2000, 2000.

[11] D. Makinson and L. van der Torre. Constraints for input/output logics. *Journal of Philosophical Logic*, 30:155–185, 2001.

[12] L. T. McCarty. Defeasible deontic reasoning. *Fundam. Inform.*, 21(1/2):125–148, 1994.

[13] D. Nute. *Defeasible deontic logic*. Springer, 1997.

[14] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7(1), 1997.

[15] Y. U. Ryu and R. M. Lee. *Defeasible deontic reasoning: a logic programming model*, pages 225–241. John Wiley and Sons Ltd., Chichester, UK, 1993.

[16] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. The british nationality act as a logic program. *Com. ACM*, 29:370–386, 1986.

[17] N. A. M. Tinnemeier, M. Dastani, and J.-J. C. Meyer. Roles and norms for programming agent organizations. In C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman, editors, *AAMAS (1)*, pages 121–128. IFAAMAS, 2009.

[18] N. A. M. Tinnemeier, M. Dastani, J.-J. C. Meyer, and L. W. N. van der Torre. Programming normative artifacts with declarative obligations and prohibitions. In *IAT*, pages 145–152. IEEE, 2009.

[19] L. W. N. van der Torre. Contextual deontic logic: Normative agents, violations and independence. *Ann. Math. Artif. Intell.*, 37(1-2):33–63, 2003.

[20] G. H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

[21] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. pages 1070–1080. MIT Press, 1988.