An Architecture of a Normative System

Counts-as Conditionals, Obligations and Permissions

Guido Boella
Università di Torino
Italy
guido@di.unito.it

Leendert van der Torre
University of Luxembourg
Luxembourg
leendert@vandertorre.com

ABSTRACT

Normative systems are traditionally described and analyzed using deontic logic, describing the logical relations among obligations and permissions. However, there is still a gap between deontic logic and normative multi-agent systems such as electronic institutions, which may be seen as an instance of the gap between on the one hand logical agent specification languages and on the other hand agent architectures and programming languages. To bridge the gap, in this paper we propose an architecture containing separate subsystems or components for counts-as conditionals, conditional obligations and conditional permissions. We add a norm database component in which the three kinds of rules are stored, and we use a channel based coordination model to describe the relations among the four normative components.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent systems

General Terms

MAS theory

Keywords

Normative systems, normative multi-agent systems

1. NORMATIVE ONTOLOGY

Searle [13, 14] distinguishes two types of norms: "Some rules regulate antecedently existing forms of behavior. For example, the rules of polite table behavior regulate eating, but eating exists independently of these rules. Some rules, on the other hand, do not merely regulate an antecedently existing activity called playing chess; they, as it were, create the possibility of or define that activity. The activity of playing chess is constituted by action in accordance with these rules. Chess has no existence apart from these rules. The institutions of marriage, money, and promising are like the institutions of baseball and chess in that they are systems of such constitutive rules or conventions" [13, p. 131].

Thus far the two kinds of norms have mainly been studied in isolation, which raises the question how regulative and constitutive norms are related. In the deontic logic literature, the interaction among obligations and permissions has been studied in some depth, see, e.g., [16, 7, 12]. In most formalizations, obligations, prohibitions and permissions have a conditional nature. Their conditions could directly refer to entities and facts of the commonsense world, but they often refer to a legal and more abstract classification of the world, making them more independent from the commonsense view. E.g., they refer to money instead of paper sheets, to properties instead of houses and fields. This more natural and economical way to model the relation between commonsense reality and legal reality uses constitutive norms, and allows regulative norms to refer to the legal classification of reality. In this way, it is not necessary that each regulative norm refers to all the conditions involved in the classification of paper as money or of houses and fields as properties. A constitutive norm has been defined by Searle as "X counts as Y in context C", and has been formalized as a counts-as conditional [10, 3, 9, 4].

Since all kinds of norms are represented by conditionals, we can represent them as subsystems within the normative system, or as input/output components in an architecture. This raises the question what kind of input and output these components have. According to Searle, institutional facts like marriage, money and private property emerge from an independent ontology of "brute" natural facts through constitutive norms of the form "such and such an X counts as Y in context C" where X is any object satisfying certain conditions and Y is a label that qualifies X as being something of an entirely new sort. Thus, the propositions describing the world are distinguished in two categories. First, "brute facts" are natural facts and events produced by actions of agents. Second, "institutional facts" are a legal classification of brute facts; they belong only to the beliefs of the normative system and have no direct counterpart in the world. Moreover, counts-as conditionals can be iterated, in the sense that X counts-as Y and Y counts-as Z (to construct the complexity of social reality), and therefore we also allow institutional literals in X. We call a set of brute literals a brute description, a set of institutional literals an institutional description, and a set of both brute and institutional literals a mixed description.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

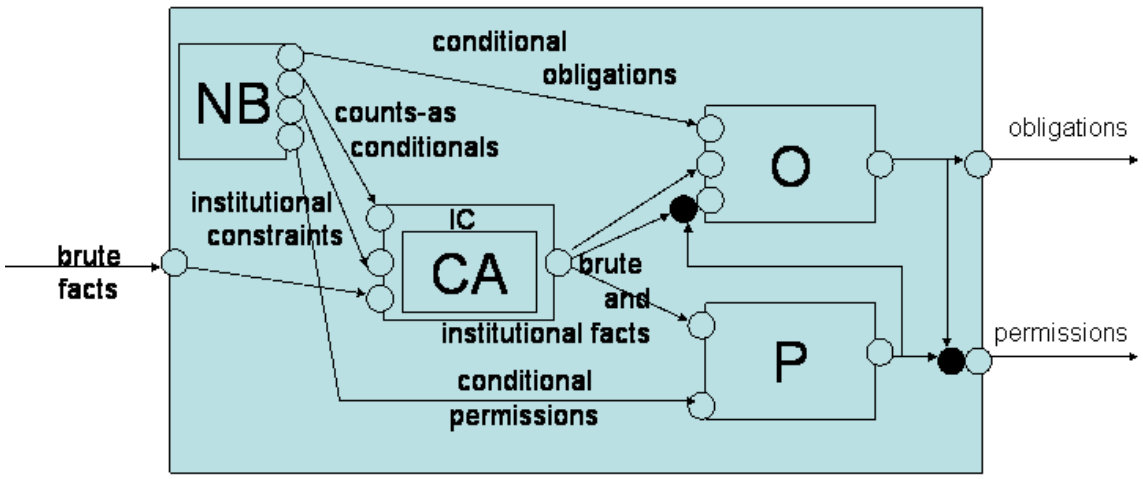

Figure 1: The Architecture of the Normative System.

2. NORMATIVE COMPONENTS

We follow the definition of normative systems proposed by Alchourrón and Bulygin [1], inspired by Tarski's definition of deductive systems. They consider as input factual descriptions and as output a classification of obligatory and permitted situations. Permissions are not closed under logical consequence, in the sense that a permission for p and a permission for $\neg p$ do not imply a permission for a contradiction. For technical reasons, we describe the permissions by sets of mixed descriptions. In our architecture, visualized in Figure 1, the input of the normative system is a context, a brute description, and the output is a set of obligations, a mixed description, and a set of permissions, a set of mixed descriptions.

There are four components in the architecture. Besides the ones for counts-as conditionals (CA), obligations (O) and permissions (P), there is a component that contains the norms (NB). The counts-as component has a wrapper (IC) to enable the connection with the other components. The components are connected by channels. The data flowing over the channels is not indexed by modal operators. So if a proposition p is flowing over a permission channel it is interpreted as a permission, usually represented in deontic logic by $P(p)$, and when it is flowing over an obligation channel it is interpreted as an obligation, usually represented by $O(p)$.

The norm database does not have any input, we therefore assume that the norms of the normative system are fixed. E.g., it can be extended with an input to update the norm database. We also assume that there can be background knowledge in the form of institutional constraints. The output contains sets of conditional obligations, conditional permissions, counts-as conditionals, and institutional constraints, four sets of pairs of mixed descriptions.

The counts-as component directly encodes Searle's conditional "X counts as Y in context C". Examples of constitutive norms are "X counts as a presiding official in a wedding ceremony", "this bit of paper counts as a five euro bill" and "this piece of land counts as somebody's private property". The input contains counts-as conditionals, X, a brute description, and C, a mixed description. The output is Y, an institutional literal.

The counts-as component is part of a component for institutional constraints, acting like a wrapper for the counts-as conditionals. Its input are the brute facts, the counts-as conditionals, and the institutional constraints. The output are the brute and institutional facts, a mixed description.

The notorious contrary-to-duty paradoxes like Chisholm's and Forrester's paradox have led to the use of constraints in deontic logic [11]. The strategy is to adapt a technique that is well known in the logic of belief change - cut back the set of norms to just below the threshold of making the current situation inconsistent. Therefore, the input of the obligation component contains conditional obligations, a set of pairs of mixed descriptions, a context, a mixed description, and constraints, a set of mixed descriptions. The output contains obligations, a mixed description.

The permission component is like an obligation component without constraints, since the issue of contrary-to-duty does not play a role for permissions. The input of the permission component is a set of conditional permissions, and a context of brute and institutional facts. As mentioned above in the description of the whole normative system, another distinction is that the output is a set of mixed descriptions. The input contains conditional permissions, a set of pairs of mixed descriptions, and a context, a mixed description. The output contains permissions, a set of mixed descriptions.

Finally, consider again the whole architecture. When designing a normative system, there is a tradeoff between obligations, permissions and counts-as conditionals, in the sense that there are many norm bases which have the same global output. For example, the same output can be obtained for example using many counts-as conditionals, or no counts-as conditionals at all. Likewise there is a tradeoff between obligations and permissions. Consequently, there is a need for a methodology to represent normative systems in our architecture. It is relatively straightforward to represent a piece of legal code in the architecture, as in legal code the three kinds of norms are usually clearly distinguished, but in general it is more problematic.

3. COORDINATION

A coordination model defines the interaction among components. In software engineering coordination models and languages based on tuplespaces are very popular, but such a coordination model does not reflect the topology of the architecture. Therefore, in this paper the coordination of the normative components is based on a channel-based coordination model. Our discussion is influenced by the coordination language Reo [2], which has also been used to coordinate agents and agent organizations [8].

Most of the channels are straightforward. The output of the norm database is connected to the input of the three other components. The output of the wrapper of the counts-as component is the context of the obligation and permission component, *et cetera*. All channels are synchronous, and we assume that they are all synchronized. This can be made explicit in the architecture. For example, in the Reo coordination language this can be achieved by connecting so-called SyncDrain channels between all channels. However, we did not represent this in the architecture to keep it as simple as possible. This synchronicity may seem like an unrealistic assumption, but it simply means that the calculation is atomic. This is a common way to model coordination in coordination languages. In actual implementations they do not have to be synchronous, but there is a mechanism in the coordination language to ensure the atomicity.

When there are multiple output channels on the same port, then the output is replicated to all channels. For example, the output of the wrapper of the counts-as component is replicated to the obligation and permission component, and the output of the permission component is replicated to the output of the normative system as well as to an input of the obligation component. When there are multiple input channels which are joined, then there are special merge nodes (a simple kind of component). In Reo there is one predefined merge as a non-deterministic choice among the inputs, if inputs are available at the same time. In the normative system architecture there are two mergers, represented by black circles. One merges the output of the counts-as component with the output of the permission component to result in the constraints for the obligation component, and the other merges the output of the obligation and permission component such that obligations are permitted. They cannot be modeled as a non-deterministic choice, but they are defined by $d \times \{d_1, \ldots, d_n\} = \{d \cup d_1, \ldots, d \cup d_n\}$. Since there are no cycles, it is straightforward to calculate the coordination.

The relation between the counts-as component and its wrapper is as follows. When the counts-as component outputs at least Y for input X and C, then the wrapper also outputs Y (and possibly more) when the input contains both X and C.

4. LOGICAL SPECIFICATIONS

The architecture is a bridge between logical specifications, for example based on deontic logic, and normative multi-agent systems. In a normative multi-agent system the components can be implemented in a variety of ways. The abstract behavior of the components and channels can be given in terms of real-timed timed data streams [2], based on general results in the Reo coordination language. The principles of deontic logic can be interpreted as functionality descriptions of the components [15, 6].

Though initially our motivation to develop the architecture of the normative system was to build such computer systems, we noticed that the architecture can be used too in deontic logic research. In particular, it can be used as a logical architecture to study interaction among logical systems by varying the logics to describe the components. For example, we can use either the logic for counts-as conditionals of Jones and Sergot [10], Artosi *et al.* [3], Grossi *et al.* [9], or our own proposal [4], and study its interaction with input/output logic [11], or some other deontic logic. To do so, we just have to find the flat conditional fragment of the logics, and ignore the modal operator. This is subject of further study [5].

5. REFERENCES

[1] C. Alchourrón and E. Bulygin. *Normative systems*. Springer, 1971.

[2] F. Arbab. Reo: A channel-based coordination model for component composition. *Mathematical Structures in Computer Science*, 14(3):329–366, 2004.

[3] A. Artosi, A. Rotolo, and S. Vida. On the logical nature of count-as conditionals. In *Procs. of LEA 2004 Workshop*, 2004.

[4] G. Boella and L. van der Torre. Regulative and constitutive norms in normative multiagent systems. In *Procs. of KR'04*, pages 255–265. AAAI Press, 2004.

[5] G. Boella and L. van der Torre. A logical architecture of a normative system. In *Procs. of ΔEON'06*, LNAI. Springer, 2006.

[6] J. Broersen, M. Dastani, and L. van der Torre. Beliefs, obligations, intension and desires as components in agent architectures. *International Journal of Intelligent Systems*, 20:9:893–919, 2005.

[7] E. Bulygin. Permissive norms and normative systems. In A. Martino and F. S. Natali, editors, *Automated Analysis of Legal Texts*, pages 211–218. Publishing Company, Amsterdam, 1986.

[8] M. Dastani, F. Arbab, and F. de Boer. Coordination and composition in multi-agent systems. In *Procs. of AAMAS05*, pages 439–446. 2005.

[9] D. Grossi, F. Dignum, and J.-J. Meyer. Contextual taxonomies. In *Procs. of CLIMA-V*, LNAI 3487, pages 33–51. Springer, 2005.

[10] A. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of IGPL*, 3:427–443, 1996.

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

[12] D. Makinson and L. van der Torre. Permissions from an input/output perspective. *Journal of Philosophical Logic*, 32 (4):391–416, 2003.

[13] J. Searle. *Speech acts: An essay in the philosophy of language*. Cambridge University, Cambridge, England, 1969.

[14] J. Searle. *The Construction of Social Reality*. The Free Press, New York, 1995.

[15] J. Treur. Semantic formalisation of interactive reasoning functionality. *International Journal of Intelligent Systems*, 17:645–686, 2002.

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