Knowledge-Based Systems 29 (2012) 31–43

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Graphical norms via conceptual graphs

Madalina Croitoru [a], Nir Oren [b,*], Simon Miles [c], Michael Luck [c]

[a] LIRMM, University Montpellier II, France
[b] Department of Computing Science, University of Aberdeen, UK
[c] Department of Informatics, King's College London, UK

ARTICLE INFO

Article history:
Available online 14 July 2011

Keywords:
Norms
Conceptual graphs
Reasoning
Graph-based reasoning
Normative violations

ABSTRACT

The specification of acceptable behaviour can be achieved via the use of obligations, permissions and prohibitions, collectively known as *norms*, which identify the states of affairs that should, may, or should not hold. Norms provide the ability to constrain behaviour while preserving individual agent autonomy. While much work has focused on the semantics of norms, the *design* of normative systems, and in particular understanding the impact of norms on a system, has received little attention. Since norms often interact with each other (for example, a permission may temporarily derogate an obligation, or a prohibition and obligation may conflict), understanding the effects of norms and their interactions becomes increasingly difficult as the number of norms increases. Yet this understanding can be critical in facilitating the design and development of effective or efficient systems. In response, this paper addresses the problem of norm explanation for Naïve users by providing of a graphical norm representation that can explicate why a norm is applicable, violated or complied with, and identify the interactions between permissions and other types of norms. We adopt a *conceptual graph* based semantics to provide this graphical representation while maintaining a formal semantics.

1. Introduction

Solutions to problems arising in the domain of multi-agent systems have often been inspired by approaches from human societies. Nowhere is this more evident than in addressing the problem of controlling the behaviour of agents within open systems. Here, interactions between agents can cause unexpected system behaviour, and traditional procedural approaches fail due to the unpredictability and complexity of these interactions, as well as the inherent autonomy of the agents involved. In human societies, behavioural control is achieved in a declarative manner, by specifying expectations regarding the behaviour of others, such as with laws or rules. These specifications, or *norms*, identify obligations, permissions and prohibitions that individuals are expected to comply with in particular situations. Drawing on this, there has been much work concerning the application of norms to artificial systems, in which agents are able to make use of concepts such as obligations, permissions, and prohibitions, to represent and reason about socially imposed goals and their execution. Such *norm aware* agents are able to decide whether to act in a manner consistent with norms, or whether to ignore them. In this context, norms are generally imposed on a set of agents in order to increase the overall utility of a system (often at the cost of individual utility) [18], or to reduce computational or communication overhead [4].

While the design and architecture of norm aware agents is critically important, this is not the only problem that must be addressed when utilising norms. Perhaps more interesting (and more challenging) is the problem of design time identification of which norms are needed in order to achieve some desired behaviour. Norms can interact with each other in unpredictable ways, and determining the effects of a norm on a system can thus be difficult. To identify these problematic norm interactions requires us to be able to *explain* the effects of a norm, and why, in some specific situation, it is applicable, violated, complied with, or in some other state, yet this has not been investigated to any real depth. Moreover, the ability to provide such explanations can enable designers to better understand the interactions between different norms, thereby allowing them to avoid introducing redundant norms [3], and to specify norms more precisely. Norm explanations can thus provide vital support for the design a normative system. In addition, from the perspective of users, norm explanation can facilitate a more intuitive appreciation of a system by providing a stronger understanding of the *reasons* why particular norms may have been brought to certain states in response to system events. Such a facility can increase and enhance the trust of a user in relation to operation of the system, providing confidence that it is in fact operating correctly.

Since much of the research into the formal properties of norms has taken place within the area of philosophy and deontic logic

* Corresponding author.
E-mail addresses: croitoru@lirmm.fr (M. Croitoru), n.oren@abdn.ac.uk (N. Oren), simon.miles@kcl.ac.uk (S. Miles), michael.luck@kcl.ac.uk (M. Luck).

0950-7051/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.06.025

[13,23], norms are typically specified within a knowledge-based system (KBS) using a logic which, for non-technical users, is often difficult to understand. However, in order for a KBS to be usable by such users, it is essential that they can understand and control not only the knowledge base construction process, but also how results are obtained from the running system. It should be easy for users not only to enter different pieces of knowledge and to understand their *meaning* but also to understand the *results* of the system, and *how* the system computed these results. This latter aspect, namely the ability to understand *why* the system gives a certain answer, is especially important since the expertise of different users may vary, and explaining each step of the logical inference process poses a difficult problem.

However, due to the core properties of norms, providing such explanations is not trivial. First, norms can be applicable only in specific circumstances, rather than over a system's entire lifetime. Thus, examining norms in isolation from a running system may not provide any useful explanation regarding an individual agent's behaviour. Second, multiple norms can interact with each other, collectively placing complex expectations on the various agents involved. Thus, while it may appear that an agent is violating some obligation, it may actually be the case that the agent is either currently exempt from this obligation due to it not being applicable in the current situation, or due to there being some *permission* that applies in the current circumstances, overriding the obligation. Given this, it should be clear that it is extremely difficult for non-technical users (indeed, also for technical experts) to interpret a large set of textually (logically) specified norms and identify their effects, and that an alternative solution to norm understanding is required.

In response, our aim in this paper is to provide a sound graphical representation of norms, by adopting a graph-based semantics and applying the semantics to normative systems. To do so, we adopt the normative framework of Oren et al. [17], a generic framework that enables updating and monitoring of the changing *status* of norms, and supports the normative reasoning process. Now, in order to provide such a graphical representation, we must be able to provide a sound and complete translation between the operations of the normative framework and the operations on the graph-based representation. Not only can this help in *understanding* the results of an update to the status of a norm, but it also allows for structural *optimisations* of norms that might not be obvious from the textual (logical) representation of the norm. Each of these is a significant challenge; in this paper, we focus on the former aspect of the graphical representation, leaving the latter for future work.

Oren et al.'s framework represents norms by means of sets of first order logic tuples, which are manipulated using a set of rules that can be reduced to first order logic subsumption on the individual tuple elements. The contribution of this paper is to map norms onto *conceptual graphs* [19,20], the only graph based formalism to have a sound and complete semantics corresponding to deduction (via subsumption) in first order logic. This formal semantics enables us to easily link Oren et al.'s norms, with their textual representation, to the conceptual graph's graphical representation, thereby providing a graphical explanation regarding the system's normative state to non-technical users. This aspect of our work was first discussed in [6], in which it was shown how individual obligations can be represented graphically. Representing permissions, and their interaction with obligations, introduces further complications, but we can extend the basic model to address this, as originally outlined in [16].

The remainder of this paper is structured as follows. In the next section, we provide the necessary formal background to the paper by briefly reviewing the normative framework and introducing the conceptual graph formalism. In Section 3, we show how the status of norms can be computed graphically. Section 4 then considers the graphical representation of interactions between permissions and other norm types. In Section 5, the paper provides a discussion in two parts: first it offers an evaluation of the effectiveness of our approach, together with an assessment of what is needed for more substantial user studies; second, it reviews some important related work. Finally, Section 6 concludes the paper by considering possible extensions to our work.

2. Background

In order to provide the requisite context for the contributions of the paper, and the basis on which we are able to develop norm explanations, we begin in this section by reviewing the formal model of norms. The model focuses on the problem of monitoring in that it facilitates identification of the *status* of norms as the environment changes over time. We then introduce the graphical formalism used in the remainder of this paper, conceptual graphs (CGs), which we map to the normative model in Section 3. This mapping allows us to address the problem of *explanation*, identifying why a norm has a particular status at some point in time.

2.1. The normative model

We introduce the normative model in a somewhat informal manner, motivating it in the context of a small example and examining how the model can be applied. Consider a situation in which an agent takes their car to a repair shop in order to be repaired. This repair shop provides a guarantee to its customers that their cars will be repaired within seven days, and thus has an *obligation* upon it, whenever a car arrives, to repair it within seven days. Clearly, once this obligation is fulfilled, it is lifted, and the repair shop no longer needs to repair the car. However, the obligation remains on the repair shop *as long as* the car is not repaired (even after seven days have passed). Finally, circumstances beyond the repair shop's control (for example, a power failure), will give the repair shop permission to repair the car seven days later than otherwise required.

The requirement on the repair shop to mend a car within seven days only obliges the repair shop to take action once a car actually arrives. Until then, the norm is an *abstract norm*. When a customer brings in a car, the norm is instantiated, thereby obtaining normative force over the repair shop and obliging it to repair the car within seven days. A single abstract norm can result in multiple *instantiated norms*; if two cars arrive at the repair shop, two instantiations of the abstract norm will occur.

Given this example, we observe that a norm may be defined in terms of five components. First, a norm has a *type*, such as an obligation, or a permission. Second, a norm has an *activation condition*, identifying the situations in which the norm affects some agents. Third, a norm imposes some *normative condition* on the affected agents; if this normative condition does not hold, then the norm is not being complied with (or made use of in the case of a permission). Fourth, norms have an *expiration condition*, identifying the situations after which the norm no longer affects the agent. Finally, the norm must identify the agents to which it is directed (i.e. those it affects), referred to as the *norm targets*.

More formally, we assume that the permissions and obligations represented by the norm refer to states and events in some environment, represented by some logical predicate language $\mathcal{L}$, such as first order logic. A norm is then a tuple of the form:

$\langle NormType,$
$NormActivation,$
$NormCondition,$
$NormExpiration,$
$NormTarget\rangle,$