

Norm Aware Agent Systems

N. Oren, S. Miles, S. Modgil, J. Keppens and M. Luck

Department of Computer Science, King's College London, Strand, London, WC2R 2LS

Abstract

In this paper, we outline an approach for implementing agents that function in a norm-aware manner; that is, agents that are capable of reasoning about, and acting, due to social obligations imposed upon them. We contrast our approach with more traditional techniques for creating agents, namely the BDI approach, and detail existing work in the area. Finally, we describe the domain in which we ground our work.

Keywords : Agents, Norms, Architecture, Practical Reasoning

Agents and Practical Reasoning

Introduction

Multi-agent systems (MAS) are a popular engineering approach to crafting distributed systems. Within a multi-agent system, autonomous entities, or *agents*, may communicate with each other, but reason and undertake action independent of other agents. Much work has been done on multi-agent systems in general, but work on the issues that arise when considering the constraints, rules, or norms that govern the behaviour of agents within organisational structures (such as in dynamic virtual organisations) is still immature. In response, this paper describes our proposed approach to norm-aware agent systems.

Practical Reasoning Agents

Agents within a MAS are typically treated as "black boxes"; that is, no outside entity may observe their inner organisation or operation. Instead, external entities may only observe the agent's interactions with the environment. One popular approach to engineering agents within a MAS [9] takes advantage of this fact, and ascribes to agents *beliefs*, *desires*, and *intentions*. Within this BDI model, beliefs represent the agent's view of the world. Desires consist of the agent's goals, and intentions are those goals that the agent has decided to

attempt to satisfy. Implementations of the BDI model typically represent desires using a library of plans [8], with the desire itself being the plan's effect. Here, a plan is selected and transformed into an intention as events are processed by the agent and courses of action committed to. Agents then execute actions according to their intentions. A BDI agent is thus capable of *practical reasoning*, or reasoning about which actions it should execute at any point in time.

Example

In a MAS representing a simple air traffic control system, agents may include the air traffic controller (ATC), and various aircraft. They may communicate with each other but, ultimately, the aircraft themselves decide whether to obey or ignore the ATC's instructions (such as in the case where the aircraft have access to better information than the ATC, and no time to communicate with it).

Within this domain, the ATC agent may have the position, velocity, destination and fuel levels of the various aircraft as beliefs. Since its desires involve moving aircraft to different locations, it may then have various plans regarding aircraft routing as its desires. If it receives a request from an aircraft to perform some manoeuvre, such

as an emergency landing, the ATC can select the appropriate plan from its library and instantiate it. This plan may have additional requirements (for example, needing to clear the taxiway of all aircraft), causing it to generate internal events to satisfy these (which in turn would instantiate additional intentions). In order to satisfy the generated intentions, the ATC might communicate with the various aircraft. Each communication may affect the position, heading and destination of some set of aircraft, which would spawn further (external) events, to which the ATC might then have to respond. A BDI system is thus both *proactive* and *reactive*, generating and acting in response to both internal and external events.

Overview

This BDI approach allows a developer to model and implement a single agent functioning as part of a MAS. However, it ignores a critical aspect of such systems, namely the *social* aspect. Concepts such as *roles* and *norms*, which we consider below, capture just this aspect. In this paper, therefore, we introduce a BDI-like architecture which takes into account these social concepts, allowing an agent to decide what action to perform in some situation, given not only its beliefs, desires and intentions, but also its norms. We begin by examining the concept of norms in more detail, and examine a number of influential architectures in the domain. We then describe our proposed architecture, and conclude by examining two domains in which our approach can be tested.

Social Practical Reasoning

A norm represents an obligation, permission or prohibition on one agent, issued by some agent or group of agents, stating that the agent *should*, *may*, or *should not* ensure that some action is taken, or that some state of affairs occurs. Thus, in the ATC case, an obligation on an aircraft could be to maintain a certain altitude

unless instructed otherwise by the ATC. Unlike standard rules, norms impose soft constraints on an agent. An aircraft might change altitude without permission, but may face sanction for doing so. Clearly, in some situations (such as an emergency), the ability for an agent to reason about, and decide to ignore, a norm may be useful. A system containing norm-aware agents (that is, agents that are capable of reasoning about norms) has a number of benefits.

- As mentioned above, norms may increase the robustness of a system. Given an unanticipated situation, agents will typically attempt to act without violating their norms. If this is not possible, an agent can be expected to meet a subset of its norms. The system will thus degrade gracefully in such circumstances.
- Norms allow agents to assume, by default, that other agents will act in a certain way, reducing the complexity of reasoning.
- Since it is natural for people to think in terms of norms, they also offer a strong and intuitive metaphor for programming, understanding and explaining why agents acted as they did in some situation.

However, making use of norms as part of a practical reasoning architecture poses a number of challenges, including determining how normative conflict should be dealt with, how norms evolve, and how an agent should actually perform reasoning given a set of norms. Norms are only one among a number of social concepts, and ensuring that they function effectively in concert with these other concepts is also important. In this paper, we outline our intended approach for constructing a normative reasoning architecture. Before doing so, we describe existing work in more detail.

Normative Systems

Norms

The study of normative systems originated in philosophy, and this led to most approaches to the formalisation of normative systems being built on the foundations of deontic logic [11]. In general, deontic logics specify *which* norms can be derived from some other set of norms, but provide no suggestion as to which norms an agent should go about fulfilling. Often, deontic logics are intractable, and make many impractical simplifying assumptions (for example, that action is always successful). They are thus not directly usable in any implementation of a practical reasoning agent. Nevertheless, by providing a formal semantics on which a system can be built, as well as suggesting solutions to many of the paradoxes that plague deontic reasoning, they are important to consider when designing a system.

More specifically, deontic logics typically contain a single modality, labelled O. The expression

$$O_x a$$

in standard deontic logic [11] is interpreted as the agent x being obliged to see to it that a holds. In other words, in x 's ideal world, a would hold. More complex deontic logics [4,10] introduce conditionals, with the formula $O_x(a|b)$ stating that x is obliged to see to it that a if b occurs.

Within such a deontic logic, the set of norms

$$O_a(l|r), P_p(r|f), O_p(\neg r|T)$$

could be interpreted as follows: an obligation on the ATC (a), to allow an aircraft to land (l) if an emergency radio message is received (r); a permission for an aircraft (p) to send the emergency radio message if it is low on fuel (f); and, finally,

a prohibition (a negated obligation) to send an emergency radio message by default (the T indicates *truth*, giving this default). As can be seen from this example, deontic logics are typically propositional, and without additional modalities and refinements, even these more expressive logics are insufficient to express typical situations in which normative practical reasoning should take place. Important norm related concepts that a logic should cater for include the concept of a violation (i.e. when $O_x a$ and a does not hold, which causes paradoxes to appear in standard deontic logic), as well as the notion of norm persistence. To illustrate these, consider the situation when an aircraft is coming in to land. The ATC has the obligation to guide the aircraft to the runway. If it does not do so, it has violated this norm. However, this violation does not discharge the norm; the ATC is still responsible for guiding the aircraft. Typically, logics are useful in proving various system properties, such as showing that a certain obligation will always hold, rather than driving reasoning.

The preceding example raises another interesting issue, namely the concept of an *organisational action*. There is no way to directly evaluate whether an ATC has guided an aircraft to the runway or not, as the act of guiding does not exist in the domain. Instead, one must map from agents' actions and domain states (which are referred to as *brute facts*), to the organisational action of guiding (i.e. to an organisational fact), in order to evaluate the norm. Thus, a set of brute facts *count as* some set of organisational facts within the context of some organisation. A number of researchers have investigated how these notions may be formally represented [3,5], but we do not concern ourselves with this issue further in this paper, and simply note that agents must be able to reason using both brute and organisational facts.

Normative Agent Architectures

The BOID architecture [1] introduces obligations into the BDI model. It represents obligations, intentions and desires as rules of the form $a \rightarrow b$; that is, if a , then b . There is also a priority function p , which is used to rank the different classes of rules. Whenever some belief is modified, the highest priority rule is activated, and may alter the belief base. If this happens, the cycle starts again. Otherwise, the rules in the next highest class are evaluated.

For example, consider a BOID agent with the belief rules:

$radioMessage \rightarrow fuelLow$

$landLast \rightarrow \neg landFirst$

the desire rule:

$true \rightarrow landLast$

and the obligation rule:

$fuelLow \rightarrow landFirst$

Assume that the *radioMessage* event occurs (i.e. *radioMessage* is true). This means that *fuelLow* is also true. An agent that prioritises desires over obligations would also decide that *landLast* is true, and this would overrule *landFirst*. However, an agent that allows obligations to overrule desires would infer *landFirst*.

Different types of agents have different priority relationships. For example, a selfish simple-minded agent ignores its obligations in favour of desires, and ignores these in favour of intentions it has already committed to fulfilling. While there are simple implementations of the BOID architecture, in general it is not computationally tractable. Agents using the BOID architecture also do not attempt to reason about the effects of norm violation.

The work of Dignum et al. [2] is related to BOID; they also use a BDI-like architecture, which is then filtered by the norms affecting an agent. Since an agent's actions may cause additional norms to come into force (so called *contrary-to-duty obligations*, such as an obligation on an aircraft at the wrong altitude to return to the correct altitude immediately), the reasoning cycle described by Dignum et al. involves determining the effects of norms based on any action that the agent may take. These effects may cause more normative effects and norms to come into force (such norms are called *potential norms*), and the agent must recursively determine the set of potential norms for any action it wants to undertake. When all potential norms have been inferred, the agent may decide how to act based on the potential norm's effects. Again, this approach faces tractability issues when calculating the normative closure of an action.

Kollingbaum's NoA [6] architecture has also attempted to integrate normative reasoning into a BDI-like system. Unlike the approaches described so far, NoA was designed from the beginning to be easily implemented, and makes use of ideas taken from the AgentSpeak(L) BDI architecture [8]. As in AgentSpeak(L), agents have a set of beliefs, desires and intentions, with the last two being operationalised by plans, which are associated with an explicit set of effects. Agents are also assigned a set of obligations and permissions, with permissions being viewed as exceptions to obligations. These norms are richer than the norms described in models up to now, and incorporate an activation and expiration condition. The activation condition determines when the norm should affect the agent, while the expiration condition states when a norm ceases to affect the agent. The agent may then reason about whether a plan violates any of its (active) norms by comparing the plan's effects to its prohibitions and obligations. It is then possible to identify different *levels of*

consistency between norms and plans. For example, a set of norms is defined to be *strongly inconsistent* with a set of plans if there is no plan that can be executed given the set of norms. A set of norms, together with a specific consistency level can thus be seen as a filter over the set of possible plans. NoA clearly provides only limited support for an agent to ignore a norm but, nevertheless, provides a computationally attractive framework for utilising norms to drive practical reasoning.

Towards a Framework for Norm Aware Practical Reasoning

We propose to develop an architecture for practical reasoning in the context of norms, that overcomes the weaknesses of existing approaches, and takes into account social concepts not often considered by existing work. In the remainder of this paper, we briefly outline our intended approach, and identify its advantages over current techniques.

More specifically, our architecture will address four main issues as follows.

- As a basic requirement, we will require an appropriate representation of norms, as well as a representation for the agent's internal structure.
- We will specify a reasoning process to enable an agent to deal with normative conflict. This process must form a core part of the agent's more general reasoning procedure and, building on top of this, we will describe tractable practical reasoning algorithms.
- We will identify and define techniques to enable an agent to adopt, modify and delete norms.
- These techniques will form part of an ontology of norm related actions over which the agent can perform reasoning.

Representation Issues

Agents in our framework will, as in most existing approaches, have a set of beliefs, desires and intentions. The agent's beliefs can be split into two groups. The first identifies facts about the world, while the second contains sets of norms. As in [7], we differentiate between abstract and instantiated norms. The former have no normative force on the agent, but may become instantiated (and thus have normative force on the agent) if some circumstances hold. Note that this division of an agent's beliefs is somewhat arbitrary; norms do not ultimately differ from other beliefs the agent may hold. In other words, an agent may have a belief about the world (for example, aircraft *x* is flying over the Atlantic), and also have a belief over norms (for example, that it is obliged to communicate with aircraft *x* in 2 hours, or that aircraft *x* is obliged to maintain a certain altitude).

As in NoA and Agentspeak(L), the agent's desires and intentions will be operationalised as plans. Also, like NoA, we associate a set of effects with a plan.

Dealing with Conflict and Reasoning

Our agent will have a set of defeasible rules that specify how norms interact with each other. These rules will initially be based on the norm's type and its issuer. Thus, for example, one rule could state that a norm issued by a superior will defeat a norm issued by a friend. This rule set can then be used to resolve normative conflict.

Reasoning in our framework is expected to be similar to that used by NoA, with norms providing a filter on plans. However, unlike NoA, we aim to annotate intentions with the norms that they satisfy, so that if norms are modified, the consequences of such modifications can be addressed.

Norm Related Action

We will also create a library of standard, norm-related actions, such as the creation, deletion and modification of a norm. These actions might form part of a plan, which could, for example, have as one of its effects, the creation of a new norm. Other important norm-related actions might alter the defeasible rules used in the norm conflict resolution mechanism. Then, whenever such a change takes place, the norm conflicts must be re-computed. If this results in instantiation of a plan that is in conflict with an existing plan, where the existing plan is constrained by a lower priority norm, then the existing plan would be suspended or removed from execution, and the higher priority plan would instead be adopted.

To pull all this together, we expect to provide a set of core, or *kernel*, plans to allow a norm-aware agent to be *created*. Through interaction with its environment, such a norm-aware agent would then be able to adopt new norms, and to adopt new plans related to both its domain, and its norm related actions. This is similar to the way in which we, as humans, deal with new environments. When integrating into a new society, a person slowly learns about its norms, and adopts new plans allowing it to interact with these norms.

In the long term, we aim to extend our repertoire of normative actions to deal with organisational concepts, enabling agents to issue commands to one another and reason about delegation and co-operation in this norm-aware context.

A Testbed Domain

In order to evaluate our approach, we intend to simulate a complex domain, and evaluate our agents' performance within it.

We are currently investigating two domains for the evaluation of our framework, namely the ATC domain (described

earlier), and a distributed power management domain.

Within the ATC domain, the agents include a number of air traffic controllers, aircraft and airports. ATC and airport actions are mainly communication oriented, while aircraft are able to manoeuvre and communicate. In this scenario, norms provide a means to enable agents to respond to emergencies efficiently, and to reduce an agent's computational load, by making sensible assumptions about agent behaviour.

We are also considering a domain built on simulating agent reasoning within a (distributed) power management system. The goal of such a system is to ensure a stable supply of power over the entire power grid. Broadly, such a system consists of three low level elements, namely power sources, power sinks and transmission capacity. Agents cooperate with each other (when acting as part of a single organisation), and compete with other organisations to maximise profit, all the while attempting to meet norms that guarantee system stability. In some situations, this will not be possible, and norms should allow the system to degrade gracefully in such cases. Different agents will have different capabilities in such a scenario, and only a few classes of actions need be considered.

Conclusions

In this paper, we have outlined a new approach to norm-aware agents, capable of operating as part of a multi-agent system. By enabling agents to determine how to act based not only on their physical environment and goals, but also on the norms that exist between them and other entities within a system, additional types of emergent behaviour may appear. For example, agents may begin to co-operate with each other (when it is beneficial for them to do so), by creation and imposition of the appropriate norms. In principle, this

allows a system to solve problems that would be difficult for an individual agent to tackle alone. A system of norm-aware agents is also more robust against failure, with the possibility of degraded performance rather than outright failure in the face of impossible situations.

The specific kind of system we propose is based on an extension of the standard BDI architecture, enabling norms to be handled in a computationally tractable manner.

Such a norm-aware system is applicable to a variety of domains, and we intend to investigate either the ATC or distributed power management domain to investigate and quantify the benefits of norm-aware agents.

References

- [1] Broersen, J.; Dastani, M.; Hulstijn, J.; Huang, Z. and van der Torre, L., *The BOID architecture: conflicts between beliefs, obligations, intentions and desires*, 2001, Proceedings of the Fifth International Conference on Autonomous Agents
- [2] Dignum, F.; Morley, D.; Sonenberg, E. A. and Cavedon, L., *Towards socially sophisticated BDI agents*, 2000, Proceedings of the Fourth International Conference on Multi-agent Systems, 111-118
- [3] Grossi, D., *Designing Invisible Handcuffs*, 2007, PhD thesis, Dutch Research School for Information and Knowledge Systems
- [4] Horty, J., *Nonmonotonic Foundations for Deontic Logic*, 1997, Defeasible Deontic Logic, Kluwer Academic Publishers, 17-44
- [5] Jones, A. J. I. & Sergot, M., *A Formal Characterisation of Institutionalised Power*, 1996, Journal of the IGPL, 3, 427-443
- [6] Kollingbaum, M., *Norm-governed Practical Reasoning Agents*, 2005, PhD thesis, University of Aberdeen
- [7] Oren, N.; Panagiotidi, S.; Vazquez-Salceda, J.; Modgil, S.; Luck, M. and Miles, S., *Towards a Formalisation of Electronic Contracting Environments*, 2008, Proceedings of the International Workshop at AAAI on Coordination, Organization, Institutions and Norms in Agent Systems, 61-68
- [8] Rao, A. S., *AgentSpeak(L): BDI agents speak out in a logical computable language*, 1996, Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World
- [9] Rao, A. S. and Georgeff, M. P., *Modeling Rational Agents within a BDI-Architecture*, 1991, Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning
- [10] van der Torre, L., *Contextual Deontic Logic: Normative Agents, Violations and Independence*, 2003, Annals of Mathematics and Artificial Intelligence, 37, 33-63
- [11] von Wright, G. H., *Deontic Logic*, 1951, Mind, 60, 1-15

Acknowledgements

The work reported in this paper was funded by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence.