Engineering Applications of Artificial Intelligence 65 (2017) 388–399

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Practical reasoning with norms for autonomous software agents

Zohreh Shams [a,*,1], Marina De Vos [a], Julian Padget [a], Wamberto W. Vasconcelos [b]

[a] Department of Computer Science, University of Bath, BA2 6AH, UK
[b] Department of Computing Science, University of Aberdeen, AB24 3UE, UK

ARTICLE INFO

Keywords:
Intelligent agents
Practical reasoning
Norms
Goals

ABSTRACT

Autonomous software agents operating in dynamic environments need to constantly reason about actions in pursuit of their goals, while taking into consideration norms which might be imposed on those actions. Normative practical reasoning supports agents making decisions about what is best for them to (not) do in a given situation. What makes practical reasoning challenging is the interplay between goals that agents are pursuing and the norms that the agents are trying to uphold. We offer a formalisation to allow agents to plan for multiple goals and norms in the presence of *durative* actions that can be executed *concurrently*. We compare plans based on decision-theoretic notions (i.e. utility) such that the utility gain of goals and utility loss of norm violations are the basis for this comparison. The set of *optimal* plans consists of plans that maximise the overall utility, each of which can be chosen by the agent to execute. We provide an implementation of our proposal in Answer Set Programming, thus allowing us to state the original problem in terms of a logic program that can be queried for solutions with specific properties.

1. Introduction

Reasoning about what to do – known as practical reasoning – for an agent pursuing different goals is a complicated task. When conducting practical reasoning, the agents might exhibit undesirable behaviour that was not predicted. The necessity of controlling undesirable behaviour has given rise to the concept of norms that offer a way to define ideal behaviour for autonomous software agents in open environments. Such norms often define obligations and prohibitions that express what the agent is obliged to do and what the agent is prohibited from doing.

Depending on their computational interpretation, norms can be regarded as *soft* or *hard constraints*. When modelled as hard constraints, the agent subject to the norms is said to be *regimented*, in which case the agent has no choice but to blindly follow the norms (Esteva et al., 2001). Although regimentation guarantees norm compliance, it greatly restricts agent autonomy. Moreover, having individual goals to pursue, self-interested agents might not want to or might not be able to comply with the norms imposed on them. Conversely, *enforcement* approaches, in which norms are modelled as soft constraints, leave the choice of complying with or violating the norms to the agent. However, in order to encourage norm compliance, there are consequences associated, namely a punishment when agents violate a norm (López y López et al., 2005; Pitt et al., 2013) or a reward when agents comply with a norm (Aldewereld et al., 2006). In some approaches (*e.g.*, Aldewereld et al., 2006; Alrawagfeh and Meneguzzi, 2014; Oren et al., 2011) there is a utility gain/loss associated with respecting norm or not, whereas in the *pressured norm compliance* approaches (*e.g.*, López y López et al., 2005), the choice to violate a norm or not is determined by how the norm affects the satisfaction or hindrance of the agent's goals.

Existing work (e.g. Oren et al., 2011; Panagiotidi et al., 2012a; Criado et al., 2010; Meneguzzi et al., 2015) on normative practical reasoning using enforcement either consider plan generation or plan selection where there is a set of pre-generated plans available to the agent. In these works, the attitude agents have towards norms is often one of compliance, meaning that their plans are often selected or, in some approaches, customised, to ensure norm compliance (e.g., Kollingbaum, 2005; Alechina et al., 2012; Oren et al., 2011). We argue that in certain situations, an agent might be better off violating a norm which, if followed, would make it impossible for the agent to achieve an important goal or complying with a more important norm.

In this paper we set out an approach for practical reasoning that considers norms in both plan generation and plan selection. We extend current work on normative plan generation such that the agent attempts to satisfy a set of potentially conflicting goals in the presence of norms,

* Corresponding author.
E-mail addresses: z.shams@bath.ac.uk (Z. Shams), m.d.vos@bath.ac.uk (M. De Vos), j.a.padget@bath.ac.uk (J. Padget), wvasconcelos@acm.org (W.W. Vasconcelos).
[1] Present address: Computer Laboratory, University of Cambridge, CB3 0FD, UK.

http://dx.doi.org/10.1016/j.engappai.2017.07.021
Received 2 October 2016; Received in revised form 4 March 2017; Accepted 25 July 2017
0952-1976/

as opposed to conventional planning problems that generate plans for a single goal (Oren et al., 2011; Panagiotidi et al., 2012a). Such an extension is made on top of STRIPS (Fikes and Nilsson, 1971), the most established planning domain language that lays the foundation of many automated planning languages. Additionally, since in reality the actions are often non-atomic, our model allows for planning with durative actions that can be executed concurrently. Through our practical reasoning process agents consider *all* plans (i.e., sequences of actions), including those leading to norm compliance and violation; each plan gets an associated overall utility for its sequence of actions, goals satisfied, and norms followed/violated, and agents can decide which of them to pursue by comparing the relative importance of goals and norms via their utilities. The plan an agent chooses to follow is not necessarily norm-compliant; however, our mechanism guarantees that the decision will maximise the overall plan utility, and this justifies the occasional violation of norms in a plan. Both plan generation and plan selection mechanisms proposed in this paper are implemented using Answer Set Programming (ASP) (Gelfond and Lifschitz, 1988).

ASP is a declarative programming paradigm using logic programs under Answer Set semantics. In this paradigm the user provides a description of a problem and ASP works out how to solve the problem by returning answer sets corresponding to problem solutions. The existence of efficient solvers to generate the answers to the problems provided has increased the use of ASP in different domains of autonomous agents and multi-agent systems such as planning (Lifschitz, 2002) and normative reasoning (Cliffe et al., 2006; Panagiotidi et al., 2012b). Several action and planning languages such as event calculus (Kowalski and Sergot, 1986), $\mathcal{A}$ (and its descendants $\mathcal{B}$ and $\mathcal{C}$ (Gelfond and Lifschitz, 1998a), Temporal Action Logics (TAL) (Doherty et al., 1998), have been implemented in ASP (Lee and Palla, 2012, 2014), indicating that ASP is appropriate for reasoning about actions. This provides motive and justification for an implementation of STRIPS (Fikes and Nilsson, 1971) that serves as the foundation of our model in ASP.

This paper is organised as follows. First we present a scenario in Section 2 which we use to illustrate the applicability of our approach. This is followed by the formal model and its semantics in Section 3. The computational implementation of the model is provided in Section 4. After the discussion of related work in Section 5, we conclude in Section 6.

2. Illustrative scenario

To illustrate our approach and motivate the normative practical reasoning model in the next section, we consider a scenario in which a software agent acts as a supervisory system in a disaster recovery mission and supports human decision-making in response to an emergency. The software agent's responsibility is to provide humans with different courses of actions available and to help humans decide on which course of actions to follow. In our scenario, the agent is to plan for a group of human actors who are in charge of responding to an emergency caused by an earthquake. The agent monitors the current situation (e.g., contamination of water, detection of shocks, etc.) and devises potential plans to satisfy goals set by human actors. In our scenario we assume the following two goals:

1. Running a hospital to help wounded people: this goal is fulfilled when medics are present to offer help and they have access to water and medicines.
2. Organising a survivors' camp: this is fulfilled when the camp's area is secured and a shelter is built.

We also assume the two following norms that the agent has to consider while devising plans to satisfy the goals above:

1. It is forbidden to built a shelter within 3 time units of detecting shocks. The cost of violating this norm is 5 units.
2. It is obligatory to stop water distribution for 2 time units once poison is detected in the water. The cost of violating this norm is 10 units.

The formulation of this scenario is provided in Appendix A.

3. A model for normative practical reasoning

We use STRIPS (Fikes and Nilsson, 1971) as the basis of our normative practical reasoning model. In STRIPS, a planning problem is defined in terms of an initial state, a goal state and a set of operators (e.g. actions). Each operator has a set of preconditions representing the circumstances/context in which the operator can be executed, and a set of postconditions that result from applying the operator. Any sequence of actions that satisfies the goal is a solution to the planning problem. In order to capture the features of the normative practical reasoning problem that we are going to model, we extend the classical planning problem by:

(i) replacing atomic actions with *durative* actions: often the *nature* of the actions is non-atomic, which means that although executed atomically in a state, the system state in which they finish executing is not necessarily the same in which they started (Nunes et al., 1997). Refinement of atomic actions to durative actions reflects the real time that a machine takes to execute certain actions.
(ii) Allowing a set of potentially inconsistent goals instead of the conventional single goal: the issue of planning for multiple goals distributed among distinct agents is addressed in collaborative planning. We address this issue for a single agent.
(iii) Factoring in norms: having made a case for the importance of norms in Section 1, we combine normative and practical reasoning. Just like goals, a set of norms is not necessarily consistent, making it potentially impossible for an agent to comply with all norms imposed on it.

A solution for a normative practical reasoning problem that features (i), (ii) and (iii) is any sequence of actions that satisfies at least one goal. The agent has the choice of violating or complying with norms triggered by execution of a sequence of actions, while satisfying its goals. However, there may be consequences either way that the agent has to foresee.

We explain the syntax and semantics of the model in Section 3.2–3.6. First we present the architecture of our envisaged system in the next section.

3.1. Architecture

The architecture, depicted in Fig. 1, shows how re-planning can be considered if a plan in progress is interrupted due to a change in circumstances. This change can be the result of a change in the environment or unexpected actions of other agents in the system. As is customary in multi-agent systems, the agent will regularly check the viability of its plan. The frequency depends on the type of system the agent is operating in, the agent's commitment to its intentions, and the impact of re-computation on the agent's overall performance.

When an agent decides that re-planning is in order, it will take the state in which the plan is interrupted as the initial state for the new plan and its current goal set as the goals to plan towards. The current goal set does not have to be the same as the goal set the original plan was devised for. Goals can already be achieved in the interrupted plan, previous goals may no longer be relevant and others may have been added. Even if the goals remain the same, the resulting new optimal plan might change due to changes in the state of the system. Similarly, there might be new norms imposed on the agent that will have to be considered in replanning.

We cater for agents which need to create their own individual plans. However, in doing so, in multi-agent scenarios agents will inevitably