Planning in a Normative System

Guilherme Krzisch

Abstract

In a normative system, agents need to consider existing norms while planning, in order to reason if it is better to follow a norm-compliant plan or a norm-violation plan, with its associated violation cost. Our proposal is the implementation of a planner that takes into account a set of norms, returning the optimal plan. In order to do this, we intend to use as a base the Graphplan algorithm.

Introduction

In a multi-agent system, agents are autonomous and can decide which actions to perform to reach some goal. As agents are self-interested and can perform actions that do not consider the impact on other agents or on the system, sometimes the expected general behavior of a system must be enforced with norms. Norms specify the desired behavior of agents, and an associated sanction cost that will affect violation agents. Therefore, agents in a normative system are still autonomous, but must reason whether to follow a norm-compliant or a norm-violation plan.

Our proposal is to implement a planner that takes into account a set of norms during planning. There are existing works that have considered this problem. In (Panagiotidi and Vázquez-Salceda 2011) they formalized norms with activation, deactivation, maintenance and repair conditions, and in order to find a plan they introduced intermediate states responsible to check norm compliance. In (Panagiotidi, Alvarez-Napagao, and Vázquez-Salceda 2013) norms are specified in Linear Temporal Logic (LTL), and they use TLPlan as their base planner.

Technical Approach

There are two main decisions we need to make in this work, namely how the system will be described (and consequently how the search of solutions will work) and how the norms will be formalized. The next two subsections will describe both of them in more details, and then we will talk about planning in this context, finishing this section with how we can evaluate the proposed solution.

System Formalization

To model the system we are going to use classical planning, because it allows a domain-independent formalization, in a deterministic and fully-observable environment. More specifically, we are going to use PDDL (Planning Domain Definition Language) to describe the domain, with available predicates and actions, and the problem, with the initial and goal states. The planning process, i.e. the decision of which sequence of actions to follow, traditionally only considers the initial and the goal state, but additions in subsequent versions of PDDL allows the specification of plan metrics and preferences.

There are three main ways to perform planning in classical planning (Russell et al. 2003, Ch. 10):

- Forward state-space search
- Backward relevant-space search
- Planning-graph search

We intend to implement our planner based on the planning-graph search, using the Graphplan algorithm (Blum and Furst 1997).

Norm Formalization

There are many different ways to formalize norms, but most of them use deontic logic to express the norm's modality, i.e. if a norm is an obligation, a permission or a prohibition. We are considering the following types of norms for the current work:

1. Norm has a context and a trigger condition (Chang and Meneguzzi ; Oren and Meneguzzi 2013)
2. Using Linear Temporal Logic (LTL) (Cranefield et al. 2015)
3. Norm has an activation, deactivation, maintenance and repair condition (Panagiotidi and Vázquez-Salceda 2011)

These norm formalization have different complexities associated. While the first one can be checked in a single state, the other ones need to be checked along a path (i.e. a sequence of states and actions). We intend to start with the more straightforward one (the first one), and then move to support one of the other two.

Planning with Norms

Having the system and the norm formalized, we can discuss how we intend to implement a planner that takes into account norms. Our proposal is to have three planner modes:

- Return a norm-compliant plan
- Return a norm-violation plan
- Return a plan with minimum cost

The last mode has the assumption that each action has cost one, and the notion that each norm has a violation cost. Thus, the planner goal in this case is to return a plan which minimizes its cost.

A first intuition in how we could implement a planner with the first type of norm is to prune solutions during the extract solution phase of the Graphplan algorithm. The other types would need to have a mechanism to store the current status of a given norm in a given path.

Evaluation

In order to evaluate our planner we need to consider its quality, i.e. whether it can return correct solutions, and its time efficiency, i.e. if it can return solutions quicker than a naive solution. This naive solution would be to use a classical planner, and then filter those returned plan results which are norm-compliant or norm-violation.

Project Management

In order to perform this work the major tasks are:

- Implement the Graphplan algorithm (or use an already existing version, e.g. JavaGP (Meneguzzi and Luck 2008)) (1 week)
- Modify Graphplan algorithm to consider the first type of norm (1 week)
- Modify Graphplan algorithm to consider another type of norm (2 weeks)
- Perform experiments (1 week)
- Write the final report and prepare the presentation (1 week)

Considering we have approximately six weeks for this project, we believe we will be able to follow the above schedule.

Conclusion

We think our proposal is interesting and challenging because it connects classical planning with norms. We will be able to put into practice concepts seen in class, as well as explore new concepts like norms. While there are already similar works in literature, we could not find one that uses planning graph, a widely used data structure, in order to perform this task.

References

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial intelligence* 90(1):281–300.

Chang, S., and Meneguzzi, F. Simulating normative behaviour in multi-agent environments using monitoring artefacts. *COIN@ AAMAS2015* 17.

Cranefield, S.; Savarimuthu, T.; Meneguzzi, F.; and Oren, N. 2015. A bayesian approach to norm identification. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 1743–1744. International Foundation for Autonomous Agents and Multiagent Systems.

Meneguzzi, F., and Luck, M. 2008. Leveraging new plans in AgentSpeak(PL). In Baldoni, M.; Son, T. C.; van Riemsdijk, M. B.; and Winikoff, M., eds., *Proceedings of the Sixth Workshop on Declarative Agent Languages*, 63–78.

Oren, N., and Meneguzzi, F. 2013. Norm identification through plan recognition. In *Proceedings of the workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN 2013@ AAMAS)*.

Panagiotidi, S.; Alvarez-Napagao, S.; and Vázquez-Salceda, J. 2013. Towards the norm-aware agent: Bridging the gap between deontic specifications and practical mechanisms for norm monitoring and norm-aware planning. In *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*, 346–363. Springer.

Panagiotidi, S., and Vázquez-Salceda, J. 2011. Norm-aware planning: Semantics and implementation. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03*, 33–36. IEEE Computer Society.

Russell, S. J.; Norvig, P.; Canny, J. F.; Malik, J. M.; and Edwards, D. D. 2003. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River.