Reactive Policy Checking for Action Languages*

Zeynep Gözen Saribatur
Technische Universität Wien
Vienna, Austria
zeynep@kr.tuwien.ac.at

Abstract

As autonomous systems become more common in our lives, the issue of verifying that they behave as intended and that their design policies are correct becomes more important. This thesis aims to build foundations for such a verification capability for policies with a reactive behavior, with a focus on combining the representation power of action languages with model checking techniques.

1 Introduction

Agents are autonomous systems that have a knowledge base that describes their capabilities, represents facts about the world and helps them in reasoning about their course of actions. Thus, they are able to decide for themselves what to do to satisfy their design objectives. Reactive agents, in particular, are able to interact with the environment according to a designed policy. They can perceive the current state of the world, figure out their next actions, execute them and observe the outcomes. One needs to be sure that by following the policy, the agent will achieve the desired results. It would be highly costly, time consuming and sometimes even fatal to realize on runtime that it does not provide the expected properties.

For example, in search scenarios, an agent needs to find a missing person in unknown environments. A policy for the agent can be "move to the farthest unvisited point that is visible, until a person is found". Following this reactive policy, the agent would traverse the environment by choosing its actions accordingly, and reiterating the decision process after reaching a new state. The agent may also remember the locations it has been in and gain information (e.g. obstacle locations) it senses on the way. Verifying beforehand whether or not the designed policy of the agent satisfies the desired goal (e.g. can the agent always find the person?), in all possible instances of the environment is nontrivial.

Action languages [Gelfond and Lifschitz, 1998] are a useful tool for defining actions and reasoning about them, by modeling dynamic systems as transition systems. Their declarative setup helps in describing dynamic systems in an understandable, concise language. They also address the problems encountered when reasoning about actions. As they are closely related to classical logic and answer set programming (ASP) [Brewka *et al.*, 2011], they can be translated into logic programs and queried for computation.

As action languages are convenient tools to describe dynamic systems, one can make use of them to represent reactive agents and define reactive policies. However, the shortage of representations that are capable of modeling reactive policies prevents one from verifying such policies using action languages before putting them into use. Furthermore, verifying reactive behaviors of agents in environments by considering all possible instances and sizes is a challenging task by itself. The infamous state explosion problem arises when dealing with large environments with different types in terms of observability and determinism.

In this thesis, our goal is to gain the capability of checking and verifying properties of reactive policies with a focus on action languages. Such a capability should give the power to determine the possible outcomes of an agent executing a policy. This way, the shortcomings of the policy can be detected and thus improved.

1.1 Related Work

Notice that our aim is different from *finding a policy* (i.e. global plan) that satisfies certain properties (i.e. goals) in an unknown environment. On the contrary, we assume that we are already given a representation of a system with a certain policy, and we want to check what it is capable (or incapable) of. Therefore, we do not address the problem of synthesizing plans by verifying properties.

The works on verifying properties of dynamic systems [Calvanese *et al.*, 2013] can be relevant. However, the problem is addressed in the presence of description logic, and real life environment settings are not considered. Meanwhile, works on verifying properties of agent programs based on the BDI approach [Dennis *et al.*, 2012] consider complex architectures which may be effective only for specific settings.

2 Research Goals

Our key objectives comprise of the following goals.

Semantics for Reactive Policies. In order to gain the capability of verifying policies with a reactive behavior, our first goal is to express these policies using the representation power of the transition systems described by action languages.

*This research has been supported by the Austrian Science Fund (FWF) project W1255-N23.

Competence in Verifying Properties of Policies. After obtaining a representation of reactive policies, we aim to address to issue of checking and verifying properties of such policies. In order to solve these problems practically, making use of model checking techniques is necessary. More specifically, we aim to investigate the following aspects and incorporate them into the syntax and semantics of action languages.

- *Abstraction and Refinement*
  Removing or simplifying details of the representation that are irrelevant to the policy or the desired properties would help in omitting some of the constraints and enriching the behavior. We want to describe such an abstraction to reduce the state space, and make verifying more efficient. In addition, we intend to employ ideas from the counterexample-guided abstraction refinement (CEGAR) method [Clarke *et al.*, 2003] to compute precise abstract representations.
- *Compositional Reasoning*
  In order to deduce the properties of large systems, we aim to find a method to decompose them into properties that describe the sub-behaviors of the components, inspired by the ideas in [Inclezan and Gelfond, 2016].
- *Parameterization*
  The desire to verify policies in all possible sizes of the environment brings the issue of verification in parameterized systems which is undecidable in general. We aim to address this issue and reduce it to verifying policies in a finite number of instances, up to some cutoff size.

Implementation. Last but not least, a prototypical implementation should be provided for the evaluation of the elaborated methods within the thesis.

3 Progress

We consider agents with a reactive behavior that determine targets to achieve during their interaction with the environment. Such agents also come with an (online) planning capability that computes plans to reach the targets.

We have described the semantics of such a reactive policy, by a high-level transition system that integrates components of target establishment and online planning [Saribatur and Eiter, 2016]. The flexibility in the components allow for a range of possibilities for designing behaviors. For example, one can use HEX [Eiter *et al.*, 2005] to describe a program that determines a target given the current state of an agent, finds the respective plan and the execution schedule. ACTHEX programs [Fink *et al.*, 2013], in particular, are a tool to define such reactive behaviors by allowing iterative evaluation of the logic programs. In addition, we have related these semantics to action languages (in particular action language $\mathcal{C}$) and investigated possibilities of policy formulation.

Currently, we are focused on fixed-size environments and exploring ways of abstracting away from the described transition system. We consider the possibility of an abstraction by omitting the details irrelevant to the policy, and investigate the employment of the CEGAR method. We use branching-time logics to express the properties of the policies. Additionally, we are looking into the possibilities of integrating such an abstraction with the syntax and semantics of action languages and similarly with answer set programming.

4 Future Work

It remains to consider a parameterization of the system where the parameter will be the size of the environment. To address the undecidability issue, instead of putting restrictions on the agent or on the topology of the environment, we plan to have this parameter bounded. Furthermore, the presence of a compositional structure should help in dealing with large systems.

In order to understand the scalability limits, a *complexity analysis* on the methods will be done. Finally, in the development of a prototype implementation, we may use available model checkers. However, issues may arise from reasoning about actions or outsourced tasks. In that case, we intend to make use of ASP reasoners.

In summary, this thesis will yield a theoretical foundation for gaining the capability of verifying (human-designed) reactive policies for AI agents with a focus on action languages.

References

[Brewka *et al.*, 2011] Gerhard Brewka, Thomas Eiter, and Mirosaw Truszczyski. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[Calvanese *et al.*, 2013] Diego Calvanese, Giuseppe De Giacomo, Marco Montali, and Fabio Patrizi. Verification and synthesis in description logic based dynamic systems. In *RR*, pages 50–64, 2013.

[Clarke *et al.*, 2003] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, 50(5):752–794, 2003.

[Dennis *et al.*, 2012] Louise A. Dennis, Michael Fisher, Matthew P. Webster, and Rafael H. Bordini. Model checking agent programming languages. *Automated Software Engineering*, 19(1):5–63, 2012.

[Eiter *et al.*, 2005] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *Proc. of IJCAI*, pages 90–96, 2005.

[Fink *et al.*, 2013] Michael Fink, Stefano Germano, Giovambattista Ianni, Christoph Redl, and Peter Schüller. Acthex: Implementing HEX programs with action atoms. In *LP-NMR*, pages 317–322, 2013.

[Gelfond and Lifschitz, 1998] Michael Gelfond and Vladimir Lifschitz. Action languages. *Electronic Transactions on AI*, 3(16), 1998.

[Inclezan and Gelfond, 2016] Daniela Inclezan and Michael Gelfond. Modular action language ALM. *Theory and Practice of Logic Programming*, 16(2):189–235, 2016.

[Saribatur and Eiter, 2016] Zeynep G. Saribatur and Thomas Eiter. Reactive policies with planning for action languages. In *Proc. of NMR Workshop*, pages 143–152, 2016.