Model Update for Automated Planning

Maria Viviane de Menezes and Leliane Nunes de Barros
Department of Computer Science
Institute of Mathematics and Statistics
University of São Paulo, Brazil
mariavm, leliane@ime.usp.br

Introduction

Model checking is a formal approach that consists of automatically solving the problem of verifying if $M$ satisfies $\varphi$, where $M$ is a formal model of a system and $\varphi$ is a formal specification of a temporal property (Figure 1). A model checker will return success if $M$ satisfies $\varphi$, otherwise, it will return a counter-example, e.g., a set of states where $\varphi$ is not satisfied.

Automated planning (Ghallab, Nau, and Traverso 2004) is the artificial intelligence field that studies the process of reasoning about actions and goals, seeking for implementation of planners. Essentially, a planner is an algorithm that synthesizes a plan of actions by analyzing a formal specification of the environment’s dynamic and the agent’s goal. Since the trajectory of a plan have temporal properties that can be formally specified, formal methods based on model checking techniques have been used to guarantee the reliability of plans (Cimatti et al. 1997; Cimatti, Roveri, and Traverso 1998; Daniele, Traverso, and Vardi 1999).

A planning algorithm based on model checking (Figure 2) receives a planning domain $M^*$, a planning goal $\varphi$ and either returns a plan, i.e., a substructure $M \subseteq M^*$ that satisfies the goal $\varphi$; or failure if there is not such a plan. Traditionally, the planning domain is represented by a Kripke structure (Kripke 1963) and the goal is specified in the Computational Tree Logic - CTL (Clarke and Emerson 1982).

A limitation of this approach is that CTL temporal logic, which is the main formalism used in planning as model checking, does not take into account the actions behind the state transitions and therefore this logic is not appropriate to deal with most of the planning problems. In order to overcome these limitations, Pereira and Barros (2008) proposed an extension of CTL, called $\alpha$-CTL, whose semantics considers the actions behind the transitions. They have also defined a planning as model checking approach based on $\alpha$-CTL that is able to cope with complex goals.

Given that $\alpha$-CTL semantics is an action labeled transition system that can be induced by a planning domain specification, can this formal approach be used to develop useful Knowledge Engineering tools for planning?

Given the planning goal $\varphi$ and $\varphi'$, this PhD proposal aims to develop a KE tool based on $\alpha$-CTL logic that is able to:

• modify an incorrect plan in order to satisfy $\varphi$ (plan repair).
• modify a plan that satisfies $\varphi$, to also satisfy $\varphi'$ (planning for complex goals).
• improve the quality of a suboptimal plan for $\varphi$. (plan optimization).
• modify a planning domain when no plan can be found to satisfy $\varphi$ and return such plan (knowledge engineering for planning).

Background and Progress to Date

Model update is a technique that extends model checking functions in order to support the repair of a faulty system. Zhang and Ding (2008) proposed a model update algorithm (Figure 3) that takes a Kripke model $M$, which does not satisfy an arbitrary CTL formula $\varphi$, and generates an updated model $M'$ that: (i) satisfies $\varphi$ and (ii) has a minimal change with respect to the original model. To generate $M'$, this approach uses primitive update operations such as: add a relation element, remove a relation element, change labelling function on one state, add a state and remove a state.
Based on Zhang and Ding ideas, we propose an $\alpha$-ctl model update approach to perform knowledge engineering for planning. The proposed system (Figure 4) receives: (i) a set of actions $\mathbb{A}$ (used to induce the planning domain $M'_A$); (ii) a partial model $M$, which is a substructure of the planning domain $(M \subseteq M'_A)$ and represents the plan that we want to update; and (iii) an $\alpha$-ctl formula $\varphi$ representing a planning goal. The system will perform modifications in the input model $M$ to (Figure 4):

1. update $M$ generating a plan $M'$ that satisfies $\varphi$ or
2. update the planning domain $M'_A$ in order to get such a plan, if there is no plan that satisfies $\varphi$, generating a new planning domain $M'_{A'}$ and a new set of actions $\mathbb{A}'$.

![Diagram](image.png)

Figure 4: $\alpha$-ctl Model Update.

In order to extend the ctl model update proposed by Zhang and Ding (2008) for planning, we have also defined a set of primitive operations, named Primitive Update in Action operations (PUAs for short). In (Menezes, Pereira, and Barros 2011), we defined a set of four PAUs to update a plan without modifying the planning domain, that are: PUA1 - adding transitions induced by an action; PUA2 - removing transitions induced by an action; PUA3 - adding a new state; and PUA4 - removing an isolated state. In (Menezes, Pereira, and Barros 2010), we defined three extra primitive update operations to update a planning domain (i.e., to modify action specifications), that are: PUA5 - adding transitions by relaxing an action precondition; PUA6 - adding transitions by adding an action effect; and PUA7 - removing a transition by deleting only one of non-deterministic effects).

Formally, given a complete model $M'_{A'}$ induced by a set of actions $\mathbb{A}$; a partial model $M \subseteq M'_{A'}$; and an $\alpha$-ctl formula $\varphi$ defining a planning goal, the set of primitive update operations $\text{PUA1-PUA4}$ can be used to refine and validate the partial model $M$. If this is not possible, i.e., there is no way to modify $M$ to satisfy $\varphi$ throughout $\text{PUA1-PUA4}$, then the planning domain has to be updated by using operations $\text{PUA5-PUA7}$.

Considering all the primitive update operations, $\text{PUA1-PUA7}$, we have defined a minimal change criterion that is able to suggest a set of minimal changes to the planning domain designer.

**Discussion and Future Work**

The main motivation for this work is to build a tool that can support a knowledge engineer to specify a planning domain for real applications where there is no guarantee to find an optimal or correct plan solution.

Therefore, one of the challenges for future work is to define an $\alpha$-ctl model update algorithm that is scalable with the size of the induced planning domain ($M'$). One possible way to deal with this is to work directly with the action description (i.e., applying symbolic techniques).

We also intend to perform empirical experiments using domains from the International Planning Competition (IPC-6 2008). By analyzing (i) different plans generated by out-of-shelf planners and (ii) planning problems for which no planner is able to solve, we expect to be able to propose updates either in the plans or in the planning domains.

**Acknowledgements.** This work has been supported by FAPESP project 2010/10845-0.

**References**


Menezes, M. V.; Pereira, S. L.; and Barros, L. N. 2010. Model Updating in Action. In Workshop on Knowledge Engineering for Planning and Scheduling – ICAPS.

