Introducing Hierarchy to Non-Hierarchical Planning Models – A Case Study for Behavioral Adversary Models

Louisa Pragst, Felix Richter, Pascal Bercher, Bernd Schattenberg, and Susanne Biundo

Institute of Artificial Intelligence, Ulm University, Germany

Abstract. Hierarchical planning approaches are often pursued when it comes to a real-world application scenario, because they allow for incorporating additional expert knowledge into the domain. That knowledge can be used both for improving plan explanations and for reducing the explored search space. In case a non-hierarchical planning model is already available, for instance because a bottom-up modeling approach was used, one has to concern oneself with the question of how to introduce a hierarchy. This paper discusses the points to consider when adding a hierarchy to a non-hierarchical planning model using the example of the BAMS Cyber Security domain.

Keywords: Hybrid Planning, Hierarchical Planning, POCL Planning, Domain Modeling, Task Hierarchy, Abstractions

1 Introduction

This work is an extended abstract based on the Bachelor's Thesis "Hybrid Planning in Cyber Security Applications" [5]. The thesis is mainly concerned with converting the classical planning domain Behavioral Adversary Modeling System (BAMS) by Boddy et al. [2] into a hybrid planning domain. Hybrid planning is a planning approach that fuses hierarchical planning with concepts from Partial Order Causal Link (POCL) planning. When introducing a hierarchy to a non-hierarchical planning domain, several issues arise. This work gives an overview about some techniques that may be applied to come up with such a hierarchy.

Foremost the non-hierarchical planning model BAMS is presented. Subsequently hybrid planning, the planning formalism into which BAMS is transformed, will be presented. The main part explicates what to consider when adding a hierarchy to a non-hierarchical planning model.

2 Behavioral Adversary Modeling System

The non-hierarchical domain model that was used as a basis for this work is called BAMS and was introduced by Boddy et al. [2]. Its purpose is to help detecting

possible attacks against a computer network, especially from malicious insiders. Generated solution plans for a BAMS planning problem are supposed to portray possible attacks against the given network. Those plans can then be analyzed and weak points of the computer network can be deducted.

The BAMS domain enables the modeling of a broad range of attacks. It includes physical attacks such as keylogging, but also malware attacks, network sniffing and even social exploits. The virtual and physical interconnection of computers can be modeled as well as email communication and encryption.

3 Hybrid Planning

Hybrid Planning [1] is a planning formalism that fuses hierarchical planning with POCL Planning. Here, not only primitive tasks have pre- and postconditions, but also the abstract tasks. These conditions are conjunctions of literals. For primitive tasks, the conditions specify in which states they can be executed. The predicates used by the literals describe connections between objects (that are represented using *constants*). For instance, in the BAMS domain, a specific human could be the administrator of a specific computer. Each constant is of a specific *sort* that represents a class of constants, such as humans or computers. Sorts can be arranged in a hierarchy. Tasks describe the possible actions such as *logging in* or *writing an email*. Only primitive tasks may be executed directly. Abstract tasks are abstractions of one or more task sequences, so-called partial plans. For each such partial plan, the domain model contains a so-called decomposition method mapping the respective abstract task to that partial plan. An abstract task's pre- and postconditions describe its intended meaning. Any partial plan of a task's decomposition method needs to satisfy a legality criterion to ensure that it is an "implementation" of its abstract task. To allow more flexibility, the pre- and postconditions of abstract tasks may use *abstract literals*, which are abstractions of ordinary literals. Those are defined by means of other (possibly abstract) literals using so-called *decomposition axioms*.

The planning problem is given in terms of an initial partial plan P_{init} (possibly containing primitive and/or abstract tasks) that specifies an initial state as well as the goal properties that should hold after the execution of a solution. A solution is an executable plan P that satisfies the goal properties and that is a refinement of P_{init}. Refinement means that P is obtained from P_{init} via decomposing abstract tasks (replacing them by their implementations) and the insertion of ordering and variable constraints, causal links, and, if desired, the insertion of tasks.

4 Introducing a Hierarchy

This section describes the most important points one should consider when adding a hierarchy to a non-hierarchical domain model. It can be used either for bottom up approaches of building a hierarchical model or in cases where

there already exists a non-hierarchical model that is to be transformed into a hierarchical (or hybrid) model.

Before we explain the deployed techniques, we want to mention that there is only very little work in the literature that is concerned with the topic of automatically inferring a hierarchy of tasks. The constructive proof of Thm. 5 by Erol et al. [3] shows how a classical planning problem can be translated into an HTN planning problem with the same set of solutions than the original one. Such a hierarchical domain does, however, not constitute a "meaningful" hierarchical model as it does not calculate abstractions of tasks, but merely simulates task insertion via decomposition. The paper "Automatically Generating Abstractions" [4] is dedicated to the automated generation of abstractions. However, the produced domain model is tailored to a given problem instance, while we aim at introducing a hierarchy that is problem independent. Furthermore, the approach aims at reducing the search effort for planners, while we focus on developing a domain than can easily be read and understood by humans.

4.1 Task Hierarchy

When building a hybrid planning domain based on a non-hierarchical model, one mainly focuses on abstracting tasks. Given below are two types of abstraction: merging alternatives and abstracting a task sequence.

Abstracting Alternatives. One possible kind of abstraction is to merge alternatives. In the simplest case, there are two alternatives, each being a single task. As an example, consider a task for *logging in with a password* and another one for *logging in with an installed certificate*. Both tasks are used for logging in, but they have slightly different preconditions. They can hence be merged into a single abstract task *login*. The two primitive tasks still remain in the domain model, but the additional abstract task *login* is introduced together with two decomposition methods, each for one of the alternatives. The pre- and postconditions of abstract tasks must reflect the conditions of its primitive sub plans with respect to the legality criterion of decomposition methods [3]. For that purpose, abstract literals may be introduced. We will give an example later.

Abstracting Task Sequences. Another possibility is the abstraction of a sequence of tasks that is often done in a specific way, like checking emails. The pattern is always the same: logging in, reading all emails, opening attachments and logging out. One can introduce an abstract task *checkEmails* that is an representative for this task sequence.

4.2 Sort Hierarchy

A hierarchy of sorts is not only useful for imposing a logical structure on a domain model, properly used, it can also help to reduce the search space. We do not go into details how to find a plausible hierarchy of sorts, in particular, because the concept is not specific to hierarchical planning approaches. For instance, PDDL also allows to define a hierarchy on types, which is PDDL's equivalent of sorts.

4.3 Relation Hierarchy

Hybrid Planning uses abstract relations to model accurate pre- and postconditions of abstract tasks. Recall that any partial plan P that is used by a decomposition method for t must be an "implementation" of t [1]. The implementation criterion is defined by means of the pre- and postconditions of the task t.

Consider the two primitive tasks mentioned earlier: one performs a *login using a password* and another *using an installed certificate*. The two tasks only differ in their preconditions: the former requires a password and the latter requires an installed certificate. The previously introduced abstract task *login* cannot use any of these preconditions, since they do not hold for both alternatives. Instead, we can introduce an abstract relation *loginIsPossible* that is used as precondition of the abstract task. Then, we add a so-called decomposition axiom that defines *loginIsPossible* as satisfied if and only if either of the two primitive relations is satisfied.

5 Summary

To build a hierarchy for a non-hierarchical planning model one has to consider the hierarchy of sorts, relations, and, most importantly, tasks. The hierarchy of sorts can be easily used to reduce the search space. The hierarchy of relations is needed to keep as much information as possible about pre- and postconditions of tasks at higher levels of the hierarchy. These abstract tasks can mainly be introduced by abstracting from different alternatives or by abstracting from sequences of tasks that are often performed together.

Acknowledgment

This work is done within the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

1. Biundo, S., Schattenberg, B.: From Abstract Crisis to Concrete Relief (A Preliminary Report on Combining State Abstraction and HTN Planning). In: Proceedings of the 6th European Conference on Planning (ECP 2001). pp. 157–168 (2001)
2. Boddy, M., Gohde, J., Haigh, T., Harp, S.: Course of Action Generation for Cyber Security Using Classical Planning. In: Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005). pp. 12–21 (2005)
3. Erol, K., Hendler, J.A., Nau, D.S.: Complexity results for HTN planning. Annals of Mathematics and Artificial Intelligence 18(1), 69–93 (1996)
4. Knoblock, C.A.: Automatically generating abstractions for planning. Artificial Intelligence 68(2), 243–302 (1994)
5. Pragst, L.: Hybrid Planning in Cyber Security Applications. Bachelor thesis, Ulm, University (2013), http://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.090/Publikationen/2013/Pragst13CyberSecurityDomain.pdf