Modeling Assistance for AI Planning From the Perspective of Model Reconciliation – Dissertation Abstract

[1]Songtuan Lin
Supervisors: [1]Pascal Bercher, [2]Gregor Behnke, [1]Alban Grastien
[1]School of Computing, The Australian National University, Canberra, Australia
[2]Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, Netherlands
[1]firstname.secondname@anu.edu.au
[2]behnkeg@informatik.uni-freiburg.de

Abstract

Providing modeling assistance to domain modelers is a prominent challenge in incorporating humans into planning processes. Many efforts have been devoted to this direction in classical planning, however, only few works have been done in hierarchical planning. In this thesis, we will study a methodology for providing modeling assistance in HTN planning, which is the most commonly used hierarchical planning framework. Particularly, we will address two bottleneck problems for this purpose, namely domain model validation and domain model refinements. For the former one, we propose an approach based on plan verification, and for the latter, we view it as a model reconciliation problem and will study a novel approach for solving it.

Introduction

Human-AI interaction has evolved as the frontier of the research on automated planning in the last decades for its capability of solving complex problems. One prominent challenge in this direction faced by the community is providing *modeling assistance* from which a domain engineer can benefit. Designing a planning domain, also known as knowledge engineering in planning and scheduling (KEPS) (McCluskey, Vaquero, and Vallati 2017) has been shown to be a difficult task, as evidenced by the establishment of the International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS), whereas it is also a mandatory process in automated planning. Thus, providing modeling assistance to domain engineers is also in deep need.

Two bottleneck problems in engineering planning domains are domain validations and domain refinements, that is, deciding whether a domain is functioning correctly and how to refine the domain if that is not the case. A significant number of efforts have been made attempting to address these two problems in order to provide modeling assistance in classical planning, for example by Lindsay et al. (2020). However, only few have been done in hierarchical planning, e.g., by Olz et al. (2021). Consequently, in this thesis, we intend to deal with those two problems in the context of hierarchical planning for the purpose of providing modeling assistance for hierarchical domain modeling. The hierarchical planning framework we are concerned with is Hierarchical Task Network (HTN) planning (Erol, Hendler, and Nau 1996; Geier and Bercher 2011; Bercher, Alford, and Höller 2019), as it is the most widely used one in recent years. Additionally, although our focus is modeling assistance for hierarchical planning, the approach we study here is also compatible with classical planning, which thus generalizes the scope of our work.

Our approach for domain model validations is via plan verification (Behnke, Höller, and Biundo 2015). More concretely, we provide a plan to a planning problem in the domain we want to validate which is supposed to be a solution, and then we verify whether this is the case. This stems from how a software is validated in practice, i.e., via providing test cases to see whether the software's outputs are correct. In our context, a provided plan serves as a test case, and the failed verification indicates that there are some flaws within the domain model.

For domain refinements, we are essentially investigating the scenario where a given plan, which serves as a test case, turns out not to be a solution to a planning problem in a domain, and we want to refine the domain so that it can be. This can be viewed as a model reconciliation problem (Chakraborti et al. 2017; Chakraborti, Sreedharan, and Kambhampati 2020; Sreedharan, Chakraborti, and Kambhampati 2021) where we intend to reconcile a flawed domain model (engineered by a domain modeler) to the ground truth one (which is unfortunately unknown). However, the underlying assumption of the existing model reconciliation approaches (Chakraborti et al. 2017; Sreedharan et al. 2019; Sreedharan, Chakraborti, and Kambhampati 2021) is that there are two domain models given (one is a human's mental model, and the other is the real one used by a planner/robot). This is however not the case in our scenario where we only have one flawed domain model. Consequently, we will study a new model reconciliation approach in the thesis for serving our scenario, i.e., changing the domain model so that the given plan will be a solution.

Beyond the scope of modeling assistance, the idea of changing a domain model to make a plan be a solution can also serve as an approach for providing a contrastive explanation (Miller 2019) about why a given plan is not a solution and can henceforth be employed in building Explainable AI Planning (XAIP) systems.

The objective of this paper is to outline the contents of

the thesis and give an introduction to our methodology for providing modeling assistance. For this purpose, we will first present the HTN formalism which we based upon and, on top of this, give an example to illustrate how our approach can be used to provide modeling assistance. Afterwards, we will present theoretical foundations for our methodology and abstractly describe how our approach will be implemented. We start by presenting the HTN formalism employed.

HTN Formalism

Thus far there exist various HTN formalisms. Those of major importance are the ones by Erol, Hendler, and Nau (1996), Geier and Bercher (2011), and Bercher, Alford, and Höller (2019), where the last two only differ in syntax, whereas the first one features more constraints over task networks (which we will introduce later on). In the thesis, we adhere to the one by Bercher, Alford, and Höller (2019), as it is the one which most complexity investigations are based on in the last decade.

One foundation for the HTN formalism is the concept of task networks, which is a set of *labeled* task names with a partial order defined over them. The task names in a task network are further categorized as being primitive or compound. A primitive task name, also called an action, is associated with its preconditions, add, and delete list, each of which consists of a set of propositions. The add list together with the delete list of an action is called the effects of the action. On the other hand, a compound task name can be refined (decomposed) into a task network by some method.

Another fundamental concept in the HTN formalism is the decomposition of a task network. Informally, a task network tn is said to be decomposed into another one tn' if tn' is obtained from tn by substituting a compound task in tn with a task network into which this compound task is decomposed by a method.

An HTN planning problem is constituted of three components: a domain, an initial task network, and an initial state. The domain consists of a finite set of propositions, a finite set of actions, a finite set of compound tasks, a finite set of methods, and a function mapping each action in the domain to its preconditions and effects. A state in an HTN planning problem is a set of propositions which describes the world.

A plan is a solution to an HTN planning problem if it is a refinement of the initial task network and consists of solely actions, meaning that it is a primitive task network obtained from the initial task network by a sequence of decompositions. Further, this refined task network must possess a linearisation (i.e., an action sequence) which is executable in the initial state, i.e., the plan is executable in the initial state. In this paper, unless otherwise specified, a plan is referred to a *partially ordered primitive task network*. For formal definitions, we refer to the work by Bercher, Alford, and Höller (2019).

Domain Validation via Verification

Having presented the HTN planning formalism used in the thesis, now we would like to introduce more technical details about the thesis in the following sections, including the theoretical foundations and implementation techniques. We begin with domain model validation.

As mentioned earlier, our way to validate a domain model is by verifying whether a plan, which serves as a test case, is a solution to a planning problem in the domain. The core challenge in this step is clearly solving a plan verification problem (Behnke, Höller, and Biundo 2015). Hence, we will also exploit a plan verifier in our implementation.

There are currently two main approaches for plan verification in HTN planning. One is by transforming a plan verification problem into a SAT problem and exploiting a SAT solver to solve the problem, i.e., the SAT-based approach (Behnke, Höller, and Biundo 2017). The other one exploits the similarities between an HTN planning problem and an attribute grammar and regards a plan verification problem as a parsing problem, i.e., the parsing-based approach (Barták, Maillard, and Cardoso 2018; Barták et al. 2020). Thus far, the empirical evaluations show that the parsing-based approach outperforms the SAT-based approach. However, the SAT-based approach is implemented in JAVA in a deprecated version of the planning system called PANDA_3, and recently, a new version called PANDA_{pi} was developed in C++. Hence, despite the underperformance of the SAT-based approach compared with the parsing-based one, we are still planning to reimplement it in C++, integrate it into the new version of PANDA, and again compare its performance with the parsing-based approach. Further, we also plan to develop a new SAT approach for plan verification based upon solution order graphs (SOGs) proposed by Behnke, Höller, and Biundo (2019), which are used in solving an HTN planning problem via encoding it as a SAT formula and is proved that such a graph can significantly reduce the number of SAT clauses and variables required. We will also compare the performance of this new SAT approach with the reimplemented one and the parsing-based one.

Further, we will also pay extra attentions to plan verification for totally ordered (TO) HTN planning problems. In contrast to plan verifications for partially ordered (PO) HTN planning problems, which have been shown to be NP-complete (Behnke, Höller, and Biundo 2015), a TOHTN plan verification problem is computationally easy. This is because any TOHTN planning problem can be captured by a context-free grammar (Höller et al. 2014), and hence, a TOHTN plan verification problem can essentially be viewed as a membership decision problem for context-free grammars. Nevertheless, the CYK algorithm for the membership decision problem for context-free grammars cannot be directly employed in TOHTN, due to additional constraints a TOHTN planning problem might have, e.g., method preconditions. Barták et al. (2021) extended the parsing-based approach to adapt to totally ordered planning problems, whereas the extended approach still relies on blind search. Consequently, we are going to improve the approach by taking advantage of the CYK algorithm and conduct empirical evaluations to compare the performance of those two.

Domain Validations via Model Checking

Apart from providing test cases, another possible way to validate a domain model is by model checking (Baier and Ka-

toen 2008) which has already been successfully used in verifying whether a system, e.g., an Information and Communication Technology (ICT) system, preserves certain properties. Hence, we can also employ this approach to check, e.g., whether the plans produced in a domain model satisfy certain constraints.

In model checking, properties in need of verification are normally captured by an LTL formula. It is thus natural to think of incorporating LTL directly into the HTN formalism, e.g., using an LTL formula to serve as the goal description of an HTN planning problem and using such a fusion to catch those domains which fail to produce plans that satisfy the LTL formula. As a starting point, we have investigated the expressiveness power of LTL in conjunction with the HTN formalism as well as the STRIPS (classical) formalism (Lin and Bercher 2022). We believe those results can serve as the theoretical foundations for applying model checking to domain model validation.

Domain Refinements via Model Reconciliation

The major concern of the thesis is domain refinements, provided that a test case, i.e., a plan, given to an HTN planning problem built upon a domain model fails. Our goal is to refine the domain model so that the plan can become a solution. As mentioned in the introduction, although this is essentially a model reconciliation problem, existing model reconciliation approaches (Chakraborti et al. 2017; Sreedharan et al. 2019; Sreedharan, Chakraborti, and Kambhampati 2021) are not applicable here, due to the inconsistency between the underlying assumption of those approaches and our scenario where we only have a flawed domain model in hand, but the existing approaches demand two models, i.e., a human's mental model and the actual model employed by a planner/robot. Consequently, in the thesis, we will propose a novel method for solving our model reconciliation problem.

Framework

As our ultimate goal is to change a domain model so that a given plan will be a solution in the updated model, we shall first specify what changes are allowed to be imposed to the domain. We first observe that, according to the solution criteria of HTN planning problems, there are mainly two reasons for why a plan is not a solution to a planning problem:

1) The plan cannot be obtained from the initial task network via decompositions, and
2) the plan is not executable in the initial state.

We can fix the first problem via refining methods in a domain model, and for the second one, we can change actions' preconditions and effects. One might further notice that those two classes of changes are independent of each other, because changing methods will not affect the executability of a plan and changing actions will not affect the decomposition hierarchy as well from which a plan is obtained.

More concretely, we will define four change operations targeted at refining methods in a domain model, namely,

1) adding an action to a method,
2) removing an action from a method,
3) adding an ordering constraint (between two actions) to a method, and
4) removing an ordering constraint (between two actions) from a method.

The reason for restricting ourselves to those four operations is that we can implement other high-level changes targeted at turning a non-solution plan to a solution, e.g., adding a compound task to a method, in terms of those defined.

For changing actions' preconditions and effects, we are concerned with three operations:

1) adding a proposition to an action's add list,
2) removing a preposition from an action's preconditions,
3) and removing a preposition from an action's delete list.

We do not consider any other change here, e.g., adding a proposition to an action's preconditions, because it can only increase the possibility that a plan is not executable.

Theoretical Foundations – Complexity

Before developing a methodology for accomplishing domain refinements via the operations described above, we are interested in finding out how hard that might be, that is, we are going to study the computational complexity of turning a non-solution plan into a solution via domain refinements. For the purpose of complexity investigation, we will use *decision languages* to describe the scenario, that is, we want to decide, given an HTN planning problem and a plan, whether there exist a way to refine the domain model of the planning problem via the operations described above such that the given plan can be a solution to the planning problem.

Note that the decision problem we are studying here is *not* exactly the same as the actual domain refinement (model reconciliation) problem. The former one only demands a *yes* or *no* answer, whereas the actual domain refinement problem we want to solve demands an exact sequence of refinement operations that can turn the plan into a solution as an output. In spite of that, the complexity results for the decision problem quantify the computational efficiency of the actual domain refinement problem. Further, the complexity investigation (on the decision problem) is also a way to identify the sources that make the actual refinement problem hard, which can thus point out the direction of developing an actual methodology for accomplishing domain refinements.

Since changing methods and changing actions' preconditions and effects are orthogonal, it is safe for us to study these two classes of changes independently. Concretely, we will first study the complexity of deciding the existence of *method* refinement operations that turn the plan into a solution by assuming that the plan is already executable in the initial state. Under this assumption, the decision problem can be rephrased as: Given an HTN planning problem and a plan, is there a way to change the methods in the domain of the planning problem such that the given plan can be obtained from the initial task network via decompositions.

Our earlier work has shown that deciding whether such changes exist is already $\mathbf{NP}$-complete in TOHTN (Lin and Bercher 2021a), independent of what method refinement operations are allowed, and later on we extended the results from a TO setting to a PO setting (Lin and Bercher 2021b). Hence, unless $\mathbf{P} = \mathbf{NP}$, there exist no polynomial algorithms that can solve the refinement problem in poly-time.

Thus far we restrict an input test case to one single plan. In practice, a domain modeler may supply a plan together with the decomposition hierarchy that is supposed to lead to the plan as a test case. The decision problem asking for the existence of method refinement operations with regard to this scenario is similar to the previous one except that now we have an additional decomposition hierarchy as the input. Though the problem is still NP-complete in general, we identified several special cases in $\mathbf{P}$ where an input decomposition hierarchy satisfies certain constraints, e.g., every method in the hierarchy decomposes a unique compound task (Lin and Bercher 2021a), and by exploiting those special cases, we were able to find all hardness sources of the decision problem where a decomposition hierarchy is not given (Lin and Bercher 2021b).

We have also studied the complexity of the problem of deciding whether there exists a way to turn a plan into a refinement of the initial task of a planning problem by applying at most k method refinement operations, where $k \in \mathbb{N}$ is a given number. This decision problem corresponds to the domain refinement problem demanding the minimum number of change operations that turn a plan into a solution. Given the previous results, this k-existence decision problem is unsurprisingly $\mathbf{NP}$-complete as well, independent of whether a decomposition hierarchy is given as an additional input.

Next we introduce the complexity of deciding the existence of action refinement operations which turn a plan into a solution, by assuming that a given plan is an action sequence and is a refinement of the initial task network of a planning problem. Clearly, the answer to this decision problem is always *yes*, because we can at least empty the preconditions of each action in the given action sequence. Therefore, what we are really interested in here is the complexity of the k-existence decision problem asking whether the plan can become executable by applying at most k refinement operations. The complexity varies in what operations are allowed. The problem is $\mathbf{NP}$-complete if adding propositions to actions' add lists is allowed, otherwise it is in $\mathbf{P}$.

Implementation – A SAT-Baseed Approach

Lastly, we will develop a methodology for actually solving the domain refinement problem. Our first attempt toward this will be to encode the refinement problem as the SAT problem. In our previous works (Lin and Bercher 2021b), we have shown that two main sources that make the refinement problem $\mathbf{NP}$-hard are: 1) the non-deterministic choices of the decomposition hierarchy that leads to the given plan, i.e., the plan verification problem, and 2) the non-deterministic choices of the methods that are in need of changing. The natural properties of these hardness sources make SAT approaches the best tool to encode the non-determinism. As an example, we can use one SAT variable $x_{a \to m}$ to indicate whether the action a is added to the method m. Additionally, we can also exploit the procedure by Behnke, Höller, and Biundo (2017) that transforms the plan verification problem into the SAT problem. Further, since both the refinement problem and the SAT problem are both $\mathbf{NP}$-complete, it is reasonable to assume that transforming one into the other will not add too many overheads.

Our another concern is how to do the empirical evaluation. We are current considering two approaches. One is to use the existing *invalid* plans. We will use our implementation to alter the respective domains to make those plans become valid. The metric for this evaluation approach will thus be the number of successful instances in a certain time frame, i.e., how many instances can be successfully solved (refined) in a given time.

Alternatively, we are also considering using the existing *valid* plans and randomly changing the respective HTN domains to make those plans become invalid. Afterward, we will employ our implementation to change the domains and make those plan valid again. The metric for this evaluation approach is to estimate, for each domain, the similarity between the original unmodified version (the ground truth) and the version refined by our implementation.

Implementation – A Planning-Based Approach

Another implementation approach we are concerned with is by encoding the domain refinement problem as an HTN planning problem. The idea stems from the similarity between the plan verification problem and the domain refinement problem. Höller et al. (2022) have proposed the approach of transforming a plan verification problem into an HTN planning problem. The empirical evaluation shown that this approach significantly outperforms others, i.e., the SAT-based and parsing based approach. Hence, we are also planning to extend this approach to support model refinements and compare its performance with the SAT approach.

Future Work

In the thesis, we only concern grounded HTN planning models, i.e., models without variables. One straightforward extension of the thesis is generalizing our approach to lifted models, i.e., models with variables. In fact, such an extension is of great importance because most domain description languages for HTN planning, e.g., HDDL (Höller et al. 2020), are designed for lifted models.

Another direction of extending our work is to take into account *negative* test cases by assuming that a domain modeler provides a plan which is *not* supposed to be a solution, and we want to refine the domain to ensure this.

Conclusion

The thesis aims at addressing two major problems in providing modeling assistance in AI planning, namely, domain model validation and domain model refinements. The approach for domain validation is by plan verification. We assume that a plan which serves as a test case is provided to a planning problem built on a domain model that need to be validated, and a failed verification indicates that domain model is flawed. For domain refinements, we view it as a model reconciliation problem where given a plan and an HTN planning problem, we want to refine the domain model such that the plan can be a solution to the planning problem, and we will propose a novel approach for solving this reconciliation problem,

References

Baier, C.; and Katoen, J. 2008. *Principles of model checking*. MIT.

Barták, R.; Maillard, A.; and Cardoso, R. C. 2018. Validation of Hierarchical Plans via Parsing of Attribute Grammars. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling, ICAPS 2018*, 11–19. AAAI.

Barták, R.; Ondrcková, S.; Behnke, G.; and Bercher, P. 2021. On the Verification of Totally-Ordered HTN Plans. In *Proceedings of the 33rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2021*, 263–267. IEEE.

Barták, R.; Ondrcková, S.; Maillard, A.; Behnke, G.; and Bercher, P. 2020. A Novel Parsing-based Approach for Verification of Hierarchical Plans. In *Proceedings of the 32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020*, 118–125. IEEE.

Behnke, G.; Höller, D.; and Biundo, S. 2015. On the Complexity of HTN Plan Verification and Its Implications for Plan Recognition. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling, ICAPS 2015*, 25–33. AAAI.

Behnke, G.; Höller, D.; and Biundo, S. 2017. This Is a Solution! (... But Is It Though?) - Verifying Solutions of Hierarchical Planning Problems. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling, ICAPS 2017*, 20–28. AAAI.

Behnke, G.; Höller, D.; and Biundo, S. 2019. Bringing Order to Chaos - A Compact Representation of Partial Order in SAT-Based HTN Planning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019*, 7520–7529. AAAI.

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning – One Abstract Idea, Many Concrete Realizations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, 6267–6275. IJCAI.

Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2020. The Emerging Landscape of Explainable Automated Planning & Decision Making. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI 2020*, 4803–4811. IJCAI.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017*, 156–163. IJCAI.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1996. Complexity Results for HTN Planning. *Annals of Mathematics and Artificial Intelligence*, 18(1): 69–93.

Geier, T.; and Bercher, P. 2011. On the Decidability of HTN Planning with Task Insertion. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, 1955–1961. IJCAI.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2014. Language Classification of Hierarchical Planning Problems. In *Proceedings of the 21st European Conference on Artificial Intelligence, ECAI 2014*, 447–452. IOS.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 9883–9891. AAAI.

Höller, D.; Wichlacz, J.; Bercher, P.; and Behnke, G. 2022. Compiling HTN Plan Verification Problems into HTN Planning Problems. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling, ICAPS 2022*. AAAI.

Lin, S.; and Bercher, P. 2021a. Change the World - How Hard Can that Be? On the Computational Complexity of Fixing Planning Models. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI 2021*, 4152–4159. IJCAI.

Lin, S.; and Bercher, P. 2021b. On the Computational Complexity of Correcting HTN Domain Models. In *Proceedings of the 4th ICAPS Workshop on Hierarchical Planning, HPlan 2021*, 35–43.

Lin, S.; and Bercher, P. 2022. On the Expressive Power of Planning Formalisms in Conjunction with LTL. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling, ICAPS 2022*. AAAI.

Lindsay, A.; Franco, S.; Reba, R.; and McCluskey, T. L. 2020. Refining Process Descriptions from Execution Data in Hybrid Planning Domain Models. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling, ICAPS 2020*, 469–477. AAAI.

McCluskey, T. L.; Vaquero, T. S.; and Vallati, M. 2017. Engineering Knowledge for Automated Planning: Towards a Notion of Quality. In *Proceedings of the 9th Knowledge Capture Conference, K-CAP 2017*, 14:1–14:8. ACM.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267: 1–38.

Olz, C.; Wierzba, E.; Bercher, P.; and Lindner, F. 2021. Towards Improving the Comprehension of HTN Planning Domains by Means of Preconditions and Effects of Compound Tasks. In *Proceedings of the 10th Workshop on Knowledge Engineering for Planning and Scheduling, KEPS 2021*.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2021. Foundations of explanations as model reconciliation. *Artificial Intelligence*, 301: 103558.

Sreedharan, S.; Hernandez, A. O.; Mishra, A. P.; and Kambhampati, S. 2019. Model-Free Model Reconciliation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, 587–594. IJCAI.