Towards Improving the Comprehension of HTN Planning Domains by Means of Preconditions and Effects of Compound Tasks

Conny Olz,[1] **Eva Wierzba,**[1] **Pascal Bercher,**[2] **Felix Lindner**[1]
[1] Ulm University, Germany
[2] The Australian National University, Australia
{conny.olz, eva.wierzba, felix.lindner}@uni-ulm.de, pascal.bercher@anu.edu.au

Abstract

Hierarchical Task Network (HTN) planning is a paradigm that offers engineers a formalism for modeling planning domains in terms of possible decompositions of compound tasks. A complete decomposition of a compound task results in totally or partially ordered primitive tasks, i.e., plans in the classical sense. Existing specification languages for HTN domains, such as HDDL, do only allow the assertion of preconditions and effects for primitive tasks but not for compound ones. Recently, a method for inferring preconditions and effects for compound tasks was proposed. It was hypothesized that inferred preconditions and effects can aid knowledge engineers in understanding HTN domain specifications and in predicting the meaning of particular compound tasks. We describe preliminary results from a study that supports this hypothesis and discuss future research directions.

Introduction

One important prerequisite for AI planning being used more extensively in practice is to make it as accessible as possible. This can include support for experts and non-experts in writing their own planning domains. As pointed out by Lindsay et al. (2020) and McCluskey, Vaquero, and Vallati (2017), validating whether a domain models an application accurately is still considered to be a challenge. If one wants to use or adapt an existing domain, it is crucial to understand its mechanics as effortlessly and thoroughly as possible. We approach this issue in case of totally ordered (t.o.) Hierarchical Task Network (HTN) planning (Erol, Hendler, and Nau 1996; Ghallab, Nau, and Traverso 2004). In HTN planning, an abstract task is solved by refining it recursively into more fine-grained subtasks until a concrete action plan known from classical planning results. Besides this different view on solving a planning problem, it is strictly more expressive than STRIPS planning in the sense that a t.o. HTN planning formalism can express context-free languages whereas STRIPS planning corresponds to regular language classes (Höller et al. 2014). So, while it is certainly not the case for every HTN planning domain one can argue that the expressiveness can lead to even more complex domains, which are harder to examine compared to STRIPS domains. So-called *compound tasks* play an important role in HTN domains as they specify, together with their decomposition methods, the problem hierarchy. However, in contrast to the actions (now called primitive tasks) from STRIPS planning, these compound tasks do not have preconditions or effects, particularly in HDDL, a standard description language for HTN planning problem (Höller et al. 2020). Therefore, one cannot see their impact on states directly (Goldman 2009). This leads to two issues: First, it can be hard to verify whether a compound task functions as intended by the domain modeler. Second, for someone unfamiliar with a domain it can be tedious to oversee and understand all mechanisms of it. Recently, Olz, Biundo, and Bercher (2021) formalized different kinds of preconditions and effects of compound tasks based on the actions deeper down in the decomposition hierarchy and presented possibilities to calculate them. More precisely, this information specifies which state features will possibly or definitely hold after the execution of a refinement of a compound task and thus reveals some of their implications. In this work, we investigate whether these inferred preconditions and effects of compound tasks can improve comprehensibility of HTN planning domains. We report results from a user study which supports this hypothesis.

Related Work

As we consider a robotics domain in our study, we briefly mention some work from robot programming: Tools aiming at supporting both novices and experts in the programming of robots include *Code3* by Huang, Lau, and Cakmak (2016) and Huang and Cakmak (2017). In general, high level primitives with a close mapping to social interactions appear to be the best abstraction level for programming social robots (Diprose et al. 2017). With ROS-TiPlEx, La Viola et al. (2019) present a framework that aims at facilitating the communication between robot and planning experts to support the development of plan-based autonomous robots. Orlandini et al. (2020) provide a recent overview over more such results concerning engineering tools, which support non-planning experts when integrating planning tools in robotic systems. In our study, we suggest using HTN planning as a way to modeling abstract robot behavior.

Regarding tool support for planning domain modeling, Strobel and Kirsch (2020) show that when programming in PDDL, using a tool that highlights syntax and allows to generate type diagrams can improve the programmers'

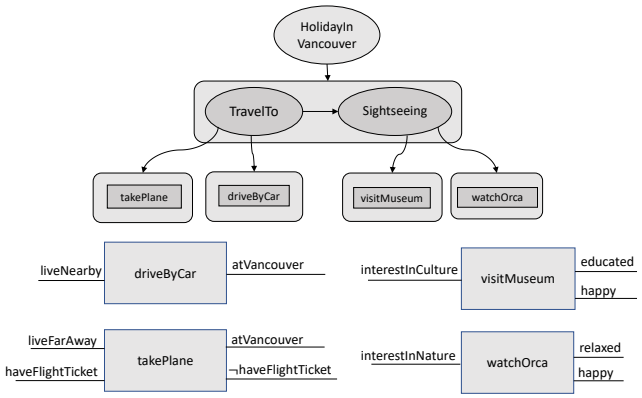

Figure 1: Visualization of a HTN planning domain to plan holidays in Vancouver. Ellipses depict compound tasks, rectangles are primitive tasks and rectangles with rounded corners are methods.

performance. Jannach, Jugovac, and Lerche (2015) found that users of the RapidMiner software environment need less time and less actions to fulfill a modeling task when an intelligent extension which recommends additional operators is used. More loosely related, Lindsay et al. (2020) proposed a method for the automated refinement of pre-engineered hybrid domains based on machine learning techniques. While there has been conducted quite a lot of research in the process of constructing domain models in AI planning (Vallati and Kitchin 2020), there is not much for HTN planning in particular so far. Bercher et al. (2016) brought up the usefullness of preconditions and effects of compound tasks in domain modelling, however, here they were not inferred but already given and checked according to some criteria. Recently, Lin and Bercher (2021) conducted a computational complexity analysis of making certain types of changes to the domain model motivated by modeling assistance.

Formal Framework

HTN Planning Formalism We will first introduce totally ordered (t.o.) HTN planning and a corresponding formalism, which is based on the ones by Geier and Bercher (2011) and Behnke, Höller, and Biundo (2018). To illustrate our explanations, we make use of an example, shown in Fig. 1, which we also used in a small tutorial on HTN planning at the beginning of our study. Assume you run a tourist agency and want to plan holidays in Vancouver. Such a holiday bundle consists of two tasks: the journey to Vancouver and some sightseeing on-site. These two tasks in turn can be implemented in different ways. One could take the car to Vancouver or the plane, depending on where the clients live. There are also multiple possibilities for sightseeing. As time is short, the clients can only see a selection, which should be chosen according to their interests. Your aim is to find such holiday plans satisfying for clients and model it as a planning problem. As we described the problem, it consists of multiple subtasks and some actions exclude others. In a classical planning problem we would need to make use of appropriate preconditions and effects to take these constraints into account. In HTN planning, however, this can be modeled quite naturally. Here, not only the actions known from classical planning (now called primitive tasks) exist, but also so-called *compound tasks*. They represent possibilities of series of further primitive or compound tasks, thereby embodying some sort of problem hierarchy. The possibilities are given by *decomposition methods*, which specify how compound tasks were refined, which means that they were replaced by other tasks. To solve an HTN planning problem one tries to refine an initially given set of tasks (a task network), which can be viewed as an unfinished plan. So, step by step always one method of a compound task must be chosen and all of its subtasks but no others must be inserted into the task network until only primitive tasks remain that are executable in the classical sense. In the example, the initial task would be *HolidayInVancouver*. As it has only one method, there is not much to choose, so the resulting task network consists of the tasks *TravelTo* and *Sightseeing*, which in turn both have two methods. Here one should take the ones such that the resulting primitive tasks are executable, which depends on the initial state. E.g., if the clients live far away, one should pick *takePlane*.

More formally: A *t.o. HTN planning domain* $\mathcal{D} = (F, A, C, M)$ consists of a finite set of facts F, *primitive tasks* A, *compound tasks* C, and *decomposition methods* $M \subseteq C \times T^*$, respectively, where $T = A \cup C$. A primitive task $a = (prec, add, del) \in A$ is described—like the actions in STRIPS planning—by its *preconditions* $prec(a) \subseteq F$ and its *effects* $add(a), del(a) \subseteq F$ (the *add, resp. delete effects*). Then, $a \in A$ is *applicable* in a state $s \in 2^F$ if $prec(a) \subseteq s$. If applicable to s and applied to it, it produces the successor state $s' = (s \setminus del(a)) \cup add(a)$. A sequence of primitive tasks or actions $\bar{a} = \langle a_0 \dots a_n \rangle$ with $a_i \in A$ for $0 \leq i \leq n$ is applicable in a state s_0 if and only if for all $0 \leq i < n$ a_i is applicable in s_i, where s_{i+1} results from applying a_i in s_i. A *t.o. task network* tn is a (possibly empty) finite sequence of tasks $\bar{t} = \langle t_0 \dots t_n \rangle \in T^*$. Compound tasks serve as abstractions for primitive and/or compound tasks and can be refined to a sequence of them specified by methods. A method $m = (c, \bar{t}) \in M$ *decomposes* a compound task $c \in C$ within a task network $tn_1 = \langle \bar{t}_1\, c\, \bar{t}_2 \rangle$ into a task network $tn_2 = \langle \bar{t}_1\, \bar{t}\, \bar{t}_2 \rangle$, written $tn_1 \rightarrow_{c,m} tn_2$. We write $tn \rightarrow tn'$ if there is a (possible empty) sequence of methods transforming tn into tn'. We then call tn' a refinement of tn. A *t.o. HTN planning problem* $\Pi = (\mathcal{D}, s_I, tn_I, g)$ contains the domain $\mathcal{D} = (F, A, C, M)$, an *initial state* $s_I \in 2^F$, an *initial task network* $tn_I \in T^*$, and a goal description $g \subseteq F$. Then, a sequence of actions $tn = \langle a_0 \dots a_n \rangle \in A^*$ is a solution to Π if and only if $tn_I \rightarrow tn$, tn is applicable in s_I, and results in a goal state $s \supseteq g$. Moreover, the set of *executability-enabling states* of a compound task $c \in C$ is $E(c) = \{s \in 2^F \mid \exists\, \bar{a} \in A^* : c \rightarrow \bar{a}$ and $\bar{a}$ is applicable in $s\}$ and $R_s(c) = \{s' \in 2^F \mid \exists\, \bar{a} \in A^* : c \rightarrow \bar{a}, \bar{a}$ is applicable in s and results in $s'\}$ is the set of all states into which the execution of c in a state $s \in 2^F$ can result (Olz, Biundo, and Bercher 2021).

Preconditions and Effects of Compound Tasks We have seen that compound tasks do not have preconditions and effects in standard HTN planning. However, recently, Olz, Biundo, and Bercher (2021) introduced various kinds of such preconditions and effects, which were not specified by a domain modeler but follow from the primitive tasks deeper down in the hierarchy. They can be inferred automatically by analyzing the domain structure. We will briefly recap the most important definitions since the aim of our study was to test their potential for the comprehension of HTN planning domains. *State-independent positive and negative effects* of a compound task c are facts that hold or do not hold, resp., after the successful execution of a refinement of c independent of the state in which the task is executed, i.e., $\mathit{eff}_*^+(c) := (\bigcap_{s\in E(c)} \bigcap_{s'\in R_s(c)} s') \setminus \bigcap_{s\in E(c)} s$ and $\mathit{eff}_*^-(c) := \bigcap_{s\in E(c)} (F \setminus \bigcup_{s'\in R_s(c)} s')$ if $E(c) \neq \emptyset$, otherwise $\mathit{eff}_*^{+/-}(c) := \mathit{undef}$. *Mandatory preconditions* of c $\mathit{prec}(c) := \bigcap_{s\in E(c)} s$ (if $E(c) \neq \emptyset$ and $\mathit{prec}(c) := \mathit{undef}$ otherwise) are facts that hold in every state for which there exists an executable refinement. So, they are required in every state in which a refinement of c shall be executed.

Applied to our example we can see that the state-independent positive effects of *HolidayInVancouver* are $\{atVancouver, happy\}$, the ones of *Sightseeing* are $\{atVancouver\}$, and for *TravelTo* we get $\{happy\}$. The sets of negative effects and mandatory preconditions are empty. Certainly, it has not been difficult to determine the information for the example. However, in general the structure of the domain can be more tangled such that it is not easy to overview all combinations and the inference gets harder. More precisely, determining whether a fact is a state-independent effect is as hard as deciding whether a t.o. HTN planning problem has a solution, which, depending on the underlying hierarchy structure, ranges from **PSPACE**-complete to **EXPTIME**-complete (Olz, Biundo, and Bercher 2021).

User Study

We have set up an online study to test our main hypothesis: *Presenting inferred preconditions and effects of abstract tasks increases the understandability of an HTN planning domain.* The online study is designed as between-subject experiment with two groups, viz., the treatment group who gets presented the inferred preconditions and effects and a control group who does not.

Methods

Materials The questionnaire starts out with asking for demographic data, viz., age and gender. Next, the participants are asked for their prior knowledge in computer science and artificial intelligence. A tutorial on HTN planning based on the Vancouver trip example is presented, where the distinction between compound and primitive tasks is introduced to the participants. After the tutorial, the participants are asked for a self assessment of their understanding of the planning method on a 5-point Likert scale. Subsequently, the first task is presented. The participants are introduced to the robot arm-movement domain as depicted in Fig. 2.

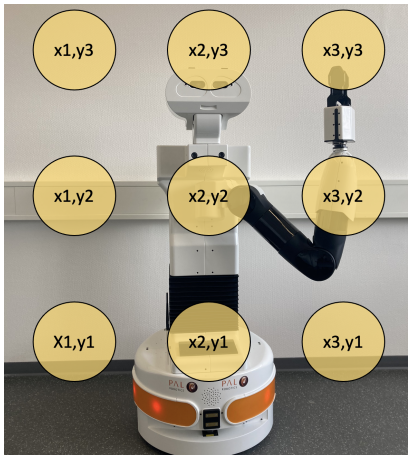

Figure 2: Depiction of the planning domain constituted by nine end-effector positions.

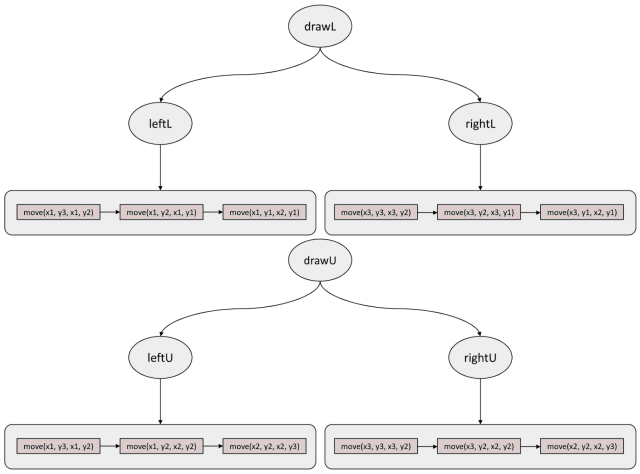

Figure 3: The two task hierarchies of drawing an L and drawing an U.

They are informed about the initial state of the robot's end-effector ($at(x3, y3)$), the preconditions and effects of the primitive task $move(x, y, xNew, yNew)$, and about the decomposition hierarchy for drawing the letter L, see Fig. 3. The treatment group gets shown the compound tasks including the inferred preconditions and effects, see Fig. 4. The control group sees the same picture but with the preconditions and effects removed, i.e., only the ellipses with the compound tasks' names. The first question that tests the participant's understanding reads: *Which of these facts hold in the final state?* All nine facts of the form $visited(x, y)$ are listed along with a checkbox to be checked if the participant considers the respective fact to be true after the execution of the abstract task $drawL$. For the treatment group, the answer to this question is immediately available due to the inferred effects of the $rightL$ compound task. The control group has to infer this information from the description of the primitive tasks and the decomposition hierarchy. The second question reads: *Which of these facts hold in the final*

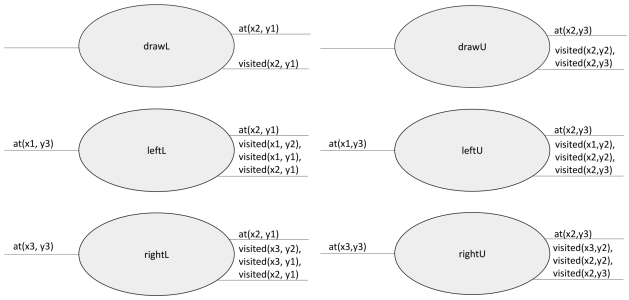

Figure 4: The abstract tasks for drawing an L and drawing an U, including inferred preconditions and effects.

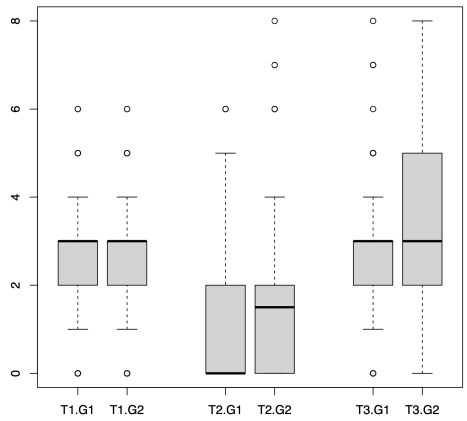

Figure 5: Number of mistakes for tasks T1-3 with comparison between the treatment group (G1) and the control (G2).

state independent of whether the arm is at the top right or left in the initial state? For the treatment group, the answer again is immediately available through the inferred effects of the $drawL$ compound task. Next, the questionnaire asks for a self assessment of the perceived difficulty of the task (5-point Likert scale), and gives participants the opportunity to give feedback on the difficulty via a text field. Afterwards, the questionnaire asks for the effects of another compound task $drawU$ in the very same manner as for $drawL$. The decomposition hierarchy of the $drawU$ task is presented as depicted in Fig. 3, the abstract tasks along with their inferred preconditions and effects is depicted in Fig. 4. Finally, participants get a de-briefing where the trajectory of the robot's movement is shown as an animated GIF.

Participants We have recruited $N = 200$ participants via prolific. We made a prescreening esnuring we only pick participants with computer programming skills. One hundred participants were directed to the questionnaire that contains the inferred preconditions and effects (the treatment group), and 100 participants were directed to the control questionnaire without this extra information. The mean age of participants was 25 ($SD = 8$), 47 were female, 149 were male, 2 defined themselves as other, and 2 did not specify gender. All participants were paid $£2.80$ for an estimated effort of 20 minutes to complete the questionnaire.

Results

We take the number of mistakes as a measure for how well each participant has understood the HTN domain. For each participant, we calculated how many wrong choices they made regarding whether a fact holds or does not hold in the final state. In support of our hypothesis, participants in the treatment group (with inferred preconditions and effects) in sum made less mistakes than participants in the control group (without inferred preconditions and effects). The median difference is 2 ($Mdn_{treatment} = 6$, $Mdn_{control} = 8$). A directed Wilcoxon rank-sum test indicates that this difference is statistically significant ($W = 5724, p = .03$). Comparisons of the number of mistakes for each of the three tasks—$drawL$ relative to initial state (T1), $drawL$ independent of initial state (T2), and $drawU$ (T3)—are shown in Fig. 5. We generally find that there is a tendency for less mistakes in the treatment group also within the sub-tasks, but the difference does not reach significance. We did not find any significant difference of self-assessed difficulty between the two groups. Qualitative analysis of the text-field answers reveals that many participants had problems understanding the HTN planning setting.

Discussion

We found that participants who were presented the inferred preconditions and effects made significantly less mistakes than those in the control group. This supports the hypothesis that presenting inferred preconditions and effects of abstract tasks increases the understandability of an HTN planning domain. However, the number of mistakes only differed significantly between the two groups when the overall number across all three tasks was looked at. Comparing the number of mistakes between groups for each of the three tasks separately showed the same pattern as the overall comparison, but the differences were not significant. There are several possible explanations for this finding. First of all, the results indicate that the task was perceived to be rather difficult by most of the participants. The mean self-reported difficulty was above average with values of 4.62 and 4.71 for the treatment group and the control group respectively. In addition, qualitatively analyzing the participants' comments as to why they found the task difficult or easy revealed that about two third of the the participants reported that they had problems with understanding or completing the task. The generally high difficulty of the task could have decreased the possibly beneficial effect of presenting inferred preconditions and effects, as in many cases participants were still unable to complete the task even with the additional support. One reason for the high perceived difficulty of the task could be an insufficient tutorial and explanations for the task. Therefore, future studies should try to further improve the tutorial in order to make the task clearer for all participants. Another possible reason could be the population sample. For participants with no prior experience in planning or even programming, the task was possibly too difficult. Also, they might have gotten frustrated in the course of the study when not being able

to solve the task and therefore lost motivation, which might be the reason for the many mistakes by the control group especially in the last task.

Conclusions

We have reported results from a user study which indicates that presenting inferred preconditions and effects for compound tasks can help humans to comprehend HTN planning domains. This result motivates future work on the development of a knowledge engineering tool that integrates the inference of such preconditions and effects. More research is needed to better understand the conditions under which additional information inferred from domain specifications can aid the knowledge engineer.

References

Behnke, G.; Höller, D.; and Biundo, S. 2018. totSAT – Totally-Ordered Hierarchical Planning through SAT. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*, 6110–6118. AAAI Press.

Bercher, P.; Höller, D.; Behnke, G.; and Biundo, S. 2016. More than a Name? On Implications of Preconditions and Effects of Compound HTN Planning Tasks. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016)*, 225–233. IOS Press.

Diprose, J.; MacDonald, B.; Hosking, J.; and Plimmer, B. 2017. Designing an API at an appropriate abstraction level for programming social robot applications. *Journal of Visual Languages & Computing* 39: 22–40.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1996. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence (AMAI)* 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. AAAI Press.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Goldman, R. P. 2009. A Semantics for HTN Methods. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 146–153. AAAI Press.

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 Press.

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 *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*, 9883–9891. AAAI Press.

Huang, J.; and Cakmak, M. 2017. Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts. In *Proceedings of the 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI 2017)*, 453–462. IEEE.

Huang, J.; Lau, T.; and Cakmak, M. 2016. Design and evaluation of a rapid programming system for service robots. In *Proceedings of the 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI 2016)*, 295–302. IEEE.

Jannach, D.; Jugovac, M.; and Lerche, L. 2015. Adaptive recommendation-based modeling support for data analysis workflows. In *Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI 2015)*, 252–262.

La Viola, C.; Orlandini, A.; Umbrico, A.; and Cesta, A. 2019. ROS-TiPlEx: How to make experts in A.I. Planning and Robotics talk together and be happy. In *Proceedings of the 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN 2019)*, 1–6. IEEE.

Lin, S.; and Bercher, P. 2021. 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)*. IJCAI.

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)*, volume 30, 469–477. AAAI Press.

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

Olz, C.; Biundo, S.; and Bercher, P. 2021. Revealing Hidden Preconditions and Effects of Compound HTN Planning Tasks – A Complexity Analysis. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, 11903–11912. AAAI Press.

Orlandini, A.; Mayer, M. C.; Umbrico, A.; and Cesta, A. 2020. Design of Timeline-Based Planning Systems for Safe Human-Robot. In *Knowledge Engineering Tools and Techniques for AI Planning*, 231–248. Springer Nature.

Strobel, V.; and Kirsch, A. 2020. MyPDDL: Tools for efficiently creating PDDL domains and problems. In *Knowledge Engineering Tools and Techniques for AI Planning*, 67–90. Springer Nature.

Vallati, M.; and Kitchin, D., eds. 2020. *Knowledge Engineering Tools and Techniques for AI Planning*. Springer Nature.