On Trustworthy, Explainable, and Verifiable High-Level Autonomy via Hierarchical Planning

Roman Barták

Charles University, Faculty of Mathematics and Physics
Prague, Czech Republic
bartak@ktiml.mff.cuni.cz

Abstract

Current mainstream AI, at least as presented in media and measured by the number of people involved and papers published, is mainly about big data, deep learning, and recently trendy large language models. All these are techniques that are data-driven, model-free, and number-crunching. Their immense success in some areas, such as computer vision and natural language processing, started the next hype in the era of AI, which brings a question whether neural approaches, after being dismissed at the beginning of the AI era, finally conquered the world of AI and proved applicable to every problem. A deeper look at these new techniques shows they have similar issues as the old-fashioned AI techniques in the past: brittleness, making strange mistakes, and being highly dependent on data used for training. Moreover, there are problems with the explainability of results and no guarantees provided, which is a crucial issue in some application areas.

In this paper, we look at core principles of the neural ML techniques, that is, being data-driven rather than knowledge-based and being model-free rather than model-based, and we argue that symbolic knowledge models can still contribute to the design of trustworthy and explainable AI systems. Specifically, we focus on hierarchical reasoning, namely hierarchical planning, which is useful for highly complex problems but is not addressed by current neural models. We propose a research plan consisting of solving specific problems in hierarchical planning as an example of a knowledge-intensive approach to problem-solving. We show close connections between these problems that allow a smooth transition between solving techniques used to solve these problems. We also propose an ultimate goal of this endeavor, that is, autonomous construction of hierarchical planning models, that addresses the crucial problem of knowledge-based approaches – how to obtain a formal model (extract knowledge from data).

Introduction

Artificial Intelligence (AI) started as a research area attempting to design computer systems to solve problems so hard that they seemed solvable only by humans or, in general, by intelligent entities. The area differentiated from the already existing area of cybernetics by exploiting computers, that is, machines for processing *symbolic information*.

For a long time, AI was dominated by *symbolic-based systems* mimicking, to some level, human reasoning processes. These systems were based on exploring candidates for a solution using search techniques, and search was the dominant technology. To solve the problem, one needs to formulate carefully the problem in terms of states, transitions between states, and properties of the goal state we are looking for. Such a problem formulation, called a *model*, can be separated from the reasoning algorithm. Researchers soon realized that search algorithms require some guidance to direct them to areas of the search space where the solution can be expected. The search algorithm can exploit these so-called *heuristics* to improve its efficiency. Otherwise, the now-known combinatorial explosion would make the problem practically unsolvable if it becomes bigger. The AI systems, called expert systems, based on the above core ideas, proved themselves applicable to real problems. The models were used not only to guide the search algorithm but also to explain the solutions produced. We had systems solving problems that only experts in the domain could solve. However, these systems also showed the core issues of the technology, such as brittleness – when being slightly outside their initially intended domain, they returned non-sense answers. It took much work to design them (acquire proper knowledge and formulate it in the model) and maintain them (update the knowledge). This is called a *knowledge acquisition gap*, and it led to an acceleration of research in machine learning, with the idea that the models can be generated and updated automatically.

Neural networks representing a bottom-up approach to human reasoning are not a new invention. However, their early research has been frozen for a long time, partly due to the limited representational power of two-layer networks. For their current success, they required two ingredients missing in the past: much training data and much computational power needed for much larger neural architectures. When these became available, the techniques based on deep neural networks started to crunch one problem by another, especially in areas challenging for traditional symbolic AI techniques such as computer vision (Krizhevsky, Sutskever, and Hinton 2012). More and more success stories brought another hype to AI. Again, there are huge expectations that these techniques can finally solve all AI issues and that we have reached the holy grail of AI. These techniques do not

require extra knowledge from humans; they learn by themselves, using a vast amount of data and computing power. Most users also do not deal much with the architectures of neural networks and use those suggested by others.

In summary, there are no explicit models and widely available data substitute knowledge. The new systems seem to understand the data, build their own models, and use them for reasoning. However, there are already clues that things are not that straightforward. The neural systems may provide wrong solutions with high confidence, as the expert systems did in the past. One may even prepare fake inputs that confuse the system (Nguyen, Yosinski, and Clune 2015). The reason is that when the input is outside the scope of data used for training, the system has no mechanism to deduce the correct answer. The hidden issue is that the internal model constructed during the learning stage may not correspond to the solved task. For example, when recognizing fish in the picture, the system may build a model recognizing blue color, so anytime the picture contains much blue, the system may classify it as a fish. Despite many efforts and human resources devoted to deep learning and reinforcement learning research, these techniques still need to improve in some areas, such as automated planning (Valmeekam et al. 2023). The Flatland challenge may be a particular example where classical (search) approaches beat the Reinforcement Learning techniques (Li et al. 2021). The neural systems also require careful architecture design, and a big problem is their lack of trustworthiness and explainability of results.

After the above anecdotic, short, and personal view of AI history, we will focus on model-based symbolic techniques for encoding knowledge. We will argue that they reflect part of human reasoning and could be more appropriate for solving some problems, specifically when trustworthiness and explainability are required.

Abstract Models for Behaviour Description

Thanks to the popular AI textbook (Russell and Norvig 2020), AI is nowadays seen as a research area for designing rational agents. Though chatbots are also rational agents, entities with "body" and more complex behavior are more accessible to recognize as agents. There are two popular areas where these agents are naturally visible: computer games with their virtual characters and robotics. These areas use easy-to-recognize agents, non-player characters and robots, that naturally interact with the environment and other agents, including humans. From the human perspective, these agents should behave predictably regarding their role in the environment. The easiest way to achieve such behavior is via predefined scripts describing the expected behavior. Frequently, these scrips are encoded as finite state automata (FSA). However, when the behavior is more complex, such automata become huge, making them hard to understand and maintain. Therefore *behavior trees* were suggested as a mathematical model of plan execution (Colledanchise and Ögren 2017). Their primary advantage over FSA is the capability to encode complex tasks composed of simple tasks, which makes the concept easy to understand. Finite state automata and behavior trees describe predefined behaviors, making them less flexible and adaptive to new tasks and environments. Automated planning has a similar but richer concept of *hierarchical task networks* (Erol, Hendler, and Nau 1996). In the rest, we will discuss how this formal model contributes to the design of rational agents.

Hierarchical Task Networks

Hierarchical task networks (HTNs) have been suggested as a knowledge-based approach to automated planning (Erol, Hendler, and Nau 1996). The flat model of agent capabilities described using actions with preconditions and effects (Fikes and Nilsson 1971) is augmented with a hierarchical structure describing how complex tasks are solved by decomposition to simpler tasks until the primitive tasks – actions – are obtained (Figure 1). A decomposition of a compound task T is modelled using a decomposition method:

$$T \rightarrow T_1, ..., T_n \; [C]$$

specifying the sub-tasks $T_1, ..., T_n$, to which T decomposes, and additional constraints C such as the ordering of subtasks or state constraints specifying extra preconditions or prevailing conditions over the world states. More decomposition methods might describe alternative ways of achieving the task; the method might even be empty for a task already achieved (Ondrcková, Pantucková, and Barták 2024). Note also that actions of different tasks may interleave (Figure 1) so solving one task may help achieve another task. A task may also recursively decompose to the task of the same type (for example, the *move* task uses *go* action for a single step and another *move* task to continue the movement – Figure 1). Formal semantics of task decompositions need to assume these complex interactions (Ondrcková and Barták 2023) and modern planning domain modeling languages such as *Hierarchical Domain Definition Language* (HDDL) (Holler et al. 2019) support some of the above constraints.

The rich structure of task decompositions serves as additional knowledge that speeds up the planning process (Nau et al. 2003). However, the knowledge acquisition gap we discussed is the issue again: somebody must provide the hierarchical task decomposition model. We shall now present a research plan to bridge this gap and even to go beyond it.

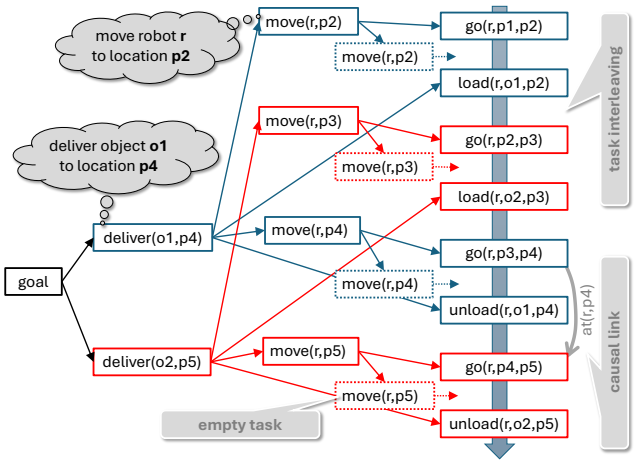

Figure 1: Example of task decomposition tree

Research Problems for Hierarchical Task Networks

This section will describe various problems related to hierarchical task networks. It goes beyond existing surveys (Georgievski and Aiello 2015; Bercher, Alford, and Höller 2019) focusing more on planning aspects of HTN as we propose a research plan that identifies problems to be solved and explains why these problems are essential to designing trustworthy, explainable, and verifiable autonomous agents.

Hierarchical Planning

Hierarchical planning deals with the problem of how to generate a plan for a given task. Planning is realized by decomposing the task into simpler tasks via decomposition methods. Multiple decomposition methods per task describe alternative ways to achieve the task. Moreover, the final set of primitive tasks must form a valid plan; preconditions of all actions must be satisfied, which is a somehow orthogonal requirement to task decompositions as the causal links between actions might go through different tasks (see Figure 1). These requirements on plans illustrate that HTNs are richer than behavior trees and that the planning process still requires a search to select a plan that best fits the current situation. HTN planning is an undecidable problem (Erol, Hendler, and Nau 1996), which is contrary to the intuition that the introduced task hierarchy makes planning easier.

SHOP2 is the most iconic HTN planner (Nau et al. 2003), but many other hierarchical planners exist. For example, the PANDA planning system (Bercher, Keen, and Biundo 2014) includes several solving approaches such as hierarchical and POCL (Partial-Order Causal-Link) planning. We should highlight, however, that no existing hierarchical planner covers all the theoretical capabilities of HTN models, for example, there is no planner supporting the prevailing constraint that requires some property to be true across a sequence of states. The PANDA planner is a part of a complex development environment covering some of the tasks discussed below. One of its capabilities is *plan repair*, allowing to repair plans that failed during execution. We highlight this specific but practically significant problem (Zaidins, Roberts, and Nau 2023; Goldman, Kuter, and Freedman 2020; Höller et al. 2020) to distinguish it from later-discussed *plan correction* problem that solves a different task.

On the one hand, the task decomposition structure provides guidelines for solving particular tasks, dramatically speeding up planning and making this concept popular in applications (Kaelbling and Lozano-Pérez 2011; Kelly, Botea, and Koenig 2021). On the other hand, the decomposition structure restricts what actions may appear in the plan. This reduces the flexibility of an agent to respond to situations that the modeler of the task decomposition methods did not anticipate. There exists a theoretical concept to overcome this limitation (which is quite significant for the design of autonomous agents) called TIHTN (HTN Planning with Task Insertion) (Geier and Bercher 2011). TIHTN allows inserting tasks/actions not obtained by decomposition of the initial task network and hence can respond to unanticipated situations similarly to classical planning. However, hierarchical planners still need to support the TIHTN concept. Exploiting the flexibility of hierarchical planning domain models and maintaining the efficiency of planning is the biggest challenge of hierarchical planning.

Plan Recognition

Somehow, a reverse problem to planning is *plan recognition*. This well-established research area (Mirsky, Keren, and Geib 2021; Van-Horenbeke and Peer 2021) has great practical importance in multiagent environments as it deals with the problem of recognizing the plans of another agent based on observations of actions performed by that agent. The agent observes the actions of another agent and uses a model of that agent's behavior (or model of own behavior, if the agent believes that the other agent behaves identically) to guess what the other agent is doing. This approach is practical in cooperative environments (agents do not need to communicate the following actions explicitly) and in competitive environments (game theory approaches are based on knowing what the other agent can do). For example, plan recognition is related to behavior recognition, such as recognizing suspicious behavior in public space (Niu et al. 2004). In computer security, plan recognition can predict cyber attacks (Li et al. 2020). Other applications include multiagent systems (Kaminka, Pynadath, and Tambe 2002), where each agent needs to recognize the goals of other agents, or artificial intelligence in computer games (Ha et al. 2011), which needs to predict the future actions of human players.

In literature (Mirsky, Keren, and Geib 2021; Van-Horenbeke and Peer 2021), three different problems are formulated. *Activity recognition* refers to a problem of labeling low-level data from sensors as activities. It recognizes short spatiotemporally localized activities or events, such as the agent visiting a shop. *Goal recognition* analyzes a sequence of observed activities to infer what goal the agent is going to achieve or what the final objective of the agent is; for example, by observing the activity of visiting a shop, the goal would probably be to buy something. This task is also referred to as intent recognition, and opposite to activity recognition, it contains a predictive component as it deals with possible future actions of the agent. Activity recognition is a component of goal recognition that discretizes sensory input to a sequence of observed activities and hence neural approaches seem well suited for this problem. *Plan recognition* is an extension of goal recognition, where we are also looking for future actions that will lead to the goal. Interaction between observed and predicted actions is significant, so formal planning models play a crucial role. We formulate the problem of plan recognition by giving a sequence of observed actions and completing this sequence to a valid plan for achieving some goal. This observed action sequence usually forms a prefix of the plan (and we are looking for the missing suffix), but there might be more complex settings. For example, some actions may be missed in the observation (hence, they need to be added), or some observed actions are unrelated to the plan (hence, they need to be removed).

A popular approach to classical plan recognition is based on compilation to planning (Ramírez and Geffner 2009). Two plans are found for each possible goal: the overall op-

timal plan and the optimal plan, which uses the observed actions. Based on the agent's rationality assumption, the goal with the minimal difference in costs of these two plans should be the agent's true goal. Pre-selecting a set of candidate goals is part of the problem formulation. Another approach is based on landmarks (Pereira, Oren, and Meneguzzi 2017). A landmark of a goal is a fact that must be true at a point in each plan for this goal. Instead of compilation to planning, they rank candidate goals based on the achieved landmarks. Other approaches to classical plan recognition deal, for example, with online plan recognition (Vered et al. 2018), epistemic plan recognition (Shvo et al. 2020), or using reinforcement learning (Amado, Mirsky, and Meneguzzi 2022).

In hierarchical plan recognition, the aim is to find a root (goal) task, which decomposes into a plan starting with the observed plan prefix. Unlike classical plan recognition, where the possible goals are formulated as properties of world states (for example, possible destinations of the agent), hierarchical plan recognition exploits the task decomposition structure, which naturally defines possible goal tasks. The decomposition tree rooted in the goal task is constructed to explain the belief in other agents' intentions. The agent may also predict the other agent's following action from this decomposition tree.

In the past, hierarchical plan recognition approaches working with models weaker than HTN have been proposed, some of which were based on parsing of grammars (e.g., (Geib and Goldman 2009), (Geib, Maraist, and Goldman 2008), (Mirsky, Gal, and Shieber 2017)). Currently, two approaches to plan recognition appear in HTNs. The first of these approaches is based on compilation to HTN planning (Höller et al. 2018). The second approach was inspired by parsing of grammars (Barták, Maillard, and Cardoso 2020). The approach of Höller et al. (2018) performs better on instances with a high number of missing (unobserved) actions, while the technique of Barták, Maillard, and Cardoso (2020) is faster on instances with few missing actions. This is somehow expected as the missing actions need to be planned – added to the observed plan. Motivation in formal grammars is reflected in re-using the approaches developed for context-free grammars, for example, the Earley parser (Earley 1970), in hierarchical plan recognition, which brings further speed-up (Pantucková and Barták 2023). Moreover, parsing-based approaches do not need pre-selection of possible goal (root) tasks that compilation to planning requires (a dummy root task that decomposes to any other task can be used).

As hierarchical plan recognition requires adding actions to the observed part of the plan, planning capabilities seem important there. On the other hand, rediscovering the observed actions might be overkill, mainly when a large portion of the plan is observed. A challenge in hierarchical plan recognition is efficiently combining information about observed actions (where parsing excels) with knowledge about the task decomposition structure (where planning has the lead). Also, practical applications must solve more challenging plan recognition problems with partial and noisy observability and interleaving plans. Some ideas in this direction will be presented later in the section on plan correction.

Plan Verification

Now, assume the complete plan – a sequence of actions – be given, and the problem is to determine whether it is a valid hierarchical plan. This problem is called *plan verification* and can be seen as a particular case of goal recognition – the complete sequence of actions is given, and the question is what goal task is solved by it. Plan verification involves checking that the plan is executable (action preconditions are satisfied) and can be obtained by decomposing some task. The first component – checking that the plan is executable – is easy to solve by simulating plan execution (Howey and Long 2003). Starting with the initial state, we apply the actions from the plan to calculate intermediate states (according to the planning domain model) used to verify that each action is executable and that the goal condition is satisfied in the final state. The second component of the problem usually includes the construction of the task decomposition tree that explains the plan, and this problem is NP-hard (Behnke, Höller, and Biundo 2015; Bercher et al. 2016).

Plan verification was introduced to verify plans generated by planners in planning competitions, but it also has direct practical applicability. Many organizations, such as hospitals, insurance companies, or manufacturers, have models of internal processes and workflows. When a process is executed, for example, surgery in a hospital, an insurance case with a customer, or an aircraft production, and the result is not as expected, one may ask whether the process complies with the workflow model. The workflow models frequently have a hierarchical structure, called a nested structure (Barták 2016), which resembles hierarchical task networks. Hence, checking whether the process complies with the model is the hierarchical plan verification problem.

As of now, three approaches have been proposed to hierarchical plan verification. One uses a translation of the verification problem into a Boolean satisfiability problem (Behnke, Höller, and Biundo 2017) (SAT approach). The second one translates the problem into a planning problem (Höller et al. 2022) (planning approach). The third one uses parsing (Barták et al. 2020; Barták, Maillard, and Cardoso 2018; Barták et al. 2021) (parsing approach). Parsing and planning approaches are currently the leading techniques, but only the parsing approach covers all features of HTNs. Like in plan recognition, parsing-based approaches may exploit decades of research on formal grammars. Two famous parsing techniques, CYK and Earley parsers, have been applied to hierarchical plan verification (Lin et al. 2023; Pantucková, Ondrcková, and Barták 2024) and are currently the fastest approaches. Their downside is that they work only for totally ordered plans. Hence, the current challenge is applying the power of parsing techniques to general HTN models that include partial order among tasks (and hence go beyond context-free languages due to task interleaving) and empty decomposition methods.

Plan Correction

When verification finds the plan valid, the verification system can return the task decomposition tree to justify this conclusion. This decomposition tree is a formal explanation

that the plan is valid, and because it is based on the task model, anyone can quickly check the conclusion. The situation is considerably more complicated if the plan is invalid. There could be various reasons for invalidity: some actions may have wrong attributes, some actions might be swapped in the plan, an action may not fit into the decomposition structure, or some decomposition constraints might be violated. The pioneering paper (Barták et al. 2021) proposing the concept of *plan correction* suggests a simple form of describing and correcting bugs in the plan. In particular, the paper suggests that the plan can be corrected/modified using two methods: deleting or inserting actions into the plan. The plan, which has minimal modifications, is assumed to be the correct version of the input plan. One can easily see that action insertion and deletion cover the abovementioned bugs. For example, when the attributes of an action are wrong, we can delete that action and then insert the action with the correct attributes. If an action is not located correctly in the plan, we can delete the action and insert it in the position where the action should be located. The plan correction problem is formulated as follows (Barták et al. 2021): Given a sequence of (observed) actions (and an initial state), delete or insert a minimal number of actions to obtain a valid plan according to a given HTN model.

Before continuing, we should emphasize that the plan correction problem is semantically different from the plan repair problem. In plan repair, we are given a valid plan being executed. During its execution, something unexpected happens, so the rest of the plan cannot be executed as planned. Hence, the plan's remainder (unexecuted part) needs to be modified (repaired) to achieve the original goal task. Notice that the final sequence of actions consisting of the executed prefix and the repaired suffix may not be a valid plan according to the domain model. In plan correction, the problem solved is different – we are given an action sequence that may not be valid, and the task is to correct the sequence to obtain a valid plan. Still, plan correction can solve the plan repair problem (Pantucková and Barták 2025a).

Plan correction is a natural extension of plan verification. If the input action sequence is a valid plan, the plan correction returns the same plan (zero modifications is the minimum number of changes to obtain a valid plan). Suppose the action sequence is not a valid hierarchical plan. In that case, the plan correction reports that and returns the valid plan closest to the action sequence, where the distance between plans is measured by the number of deleted and inserted actions. Although plan correction has been suggested as an extension of plan verification for invalid plans to return some information about the bugs in the plan (and correct the bugs), plan correction and its modifications can solve other HTN-related problems. For example, if we are given an empty plan (and an initial state) and a goal task, then correcting that empty plan means solving the planning problem. If an observed plan prefix is given, correcting this plan when action insertion is allowed only in the (missing) plan suffix solves the classical plan recognition problem. By restricting if and where actions can be deleted and inserted, one can solve variants of the plan recognition problem. For example, we may allow action deletion in the observed plan prefix, which removes wrong observations, or we may allow action insertion in that prefix, which adds missing observations so noisy and incomplete input can be naturally covered by plan correction. A minimum number of changes is a natural objective to ensure the corrected plan is as close as possible to the original partial plan. Some variants of the objective, for example, using different weights for different correction steps (e.g., based on correction location), may control the type of corrections preferred. Hence, plan correction is a universal concept for solving various problems related to hierarchical task networks (planning, verification, recognition).

The pioneering paper (Barták et al. 2021) introducing the problem of plan correction suggested the first technique to do the corrections. This technique uses bottom-up parsing (starts with the given actions and proceeds to a goal task) and supports action deletion only. In the recent work (Pantučková and Barták 2025), a new technique using the Earley parser can correct the plan by action insertion and action deletion. Supporting action insertion is a considerable step in plan correction as it allows plan correction to solve many other problems, as discussed above. However, this recent approach supports for totally-ordered domains only.

The plan correction problem still needs to be solved for all domains, specifically partial-order domains and empty decomposition methods. Action insertion requires some planning capabilities, so techniques for hierarchical planning, such as heuristics developed for HTN planning, are valuable. Solving the plan correction problem is the ultimate problem of HTN planning because, as we sketched above, plan correction covers many other problems in HTN planning.

Model Correction

So far, we have discussed only problems related to hierarchical planning. In particular, we assumed that the formal model consisting of task decomposition methods and action descriptions is given; the agent knows which tasks are there, how to decompose them into subtasks, and how to execute the actions. The open questions are how we get such a model and how we know that this model is correct. Currently, hierarchical planning models are defined by experienced human designers, and the design and maintenance of models are the major bottlenecks of symbolic (model-based) AI techniques.

Numerous approaches for learning action models already exist with various assumptions (Aineto, Jiménez, and Onaindia 2022). These ML approaches transform observed plans into action models by learning under which conditions the action is applicable (action preconditions) and how the action changes the environment (action effects). The existence of the symbolic notion of action is the core assumption, so using action recognition techniques should precede learning action models. Learning task description structure sits on top of action model learning, and it can be seen as a particular problem of learning typical structures of plans (Vomlelová et al. 2020) describing how actions are usually organized in plans (and going beyond the physics-only model of action models). There are approaches for learning HTN models; for example, the system *HTN Maker* has been proposed (Hogg, Muñoz-Avila, and Kuter 2016), but it requires semantically-annotated tasks as input. A recent system HPNL (Langley

Figure 2: Overview of HTN tasks discussed: black part of the decomposition tree is given, the red part is looked for.

2022) shows that techniques from other areas, such as inductive logic programming, can be exploited in learning task hierarchies.

Let us now put the problem of learning hierarchical models in the context of problems discussed earlier in this paper. In the plan verification problem, we asked whether the action sequence was valid concerning the task model. We assumed that if this was not the case, there was a bug in the plan, and we tried to remove it in the plan correction. However, what if the plan is reported to be invalid, but it is, in fact, correct? Then, the bug must be in the model (or the verification algorithm, but we assume this algorithm is correct). This observation leads to another problem that we call *model correction*. The problem is how to modify the formal model to cover the valid plans and not give invalid plans.

Similarly, as plan correction extends planning, we see model correction as an extension of domain model learning. The vision of the complete system is as follows. The agent is given some vanilla model (even an empty one) and examples of valid (good) plans. The valid plans can be provided by human users (learning by observation) or generated by the planner based on the current model. How can the hierarchical planner generate a plan if the model is empty? This is where the TIHTN concept (Estlin, Chien, and Wang 1997; Schattenberg and Biundo 2006; Gerevini et al. 2008) may help as it can generate plans containing actions not included in task decompositions (so in principle, even if there are no tasks yet). This solves the problem of HTN planning, which generates plans only for tasks already described in the model. Theoretical properties of this concept have been studied (Geier and Bercher 2011) and a TIHTN planner exists (Pantucková and Barták 2025b). An interesting question is whether invalid plans are also required when building the HTN model. Intuition says the answer is yes, but no existing HTN learning approaches use such invalid plans.

Let us focus on one more problem related to formal task decomposition models. How do we know that such a formal model is correct, and can we verify that a model is correct? We are unaware of any formal definition of the correctness of the HTN model. We already discussed that the task decomposition rules in HTN models resemble the rewriting rules of formal grammar, namely context-free grammar, so we can exploit notions from formal grammar, particularly the notion of reduced grammar. In a reduced context-free grammar, every non-terminal symbol is reachable from the initial non-terminal symbol, and from every non-terminal symbol, we can generate a terminal word. These concepts can be naturally applied to HTN models. We will require that it is possible to generate a sequence of actions from any task (a plan can solve the task), and we may assume that any task can be generated from one of the root (goal) tasks (the task is helpful). This second condition is less critical because any task could be the goal task in HTN models, but the requirement may be used in model learning to reduce the number of learned tasks. We introduced the notion of soundness for attribute grammars with arbitrary constraints over finite domains, and we also suggested a technique to verify the soundness of the model based on reducing context-free grammars and maintaining consistency among constraints (Barták and Dvorák 2016). An open question is how to verify the soundness of HTN models.

Solving the model correction problem naturally covers the knowledge acquisition gap.

Conclusions

The hierarchical task structure is the form of abstraction that allows agents to understand how the world works and to improve the agents' reasoning efficiency. Hierarchical planning proved to be an efficient way of automated planning (Nau et al. 2003), but it still needs to bridge the knowledge acquisition gap. The importance of hierarchical planning is identified not only in symbolic AI; it also appeared recently as the possible next important step in bringing neural models closer to the capabilities of humans (LeCun 2022).

In this paper, we argue that symbolic hierarchical planning may contribute to designing agents that are easier to understand and trust and can explain their decisions. The explicit formal model behind it is the core concept there. We described several research topics and showed that they are closely related (see Figure 2). We identified two essential areas of future research. One area is correcting hierarchical plans that can, in principle, solve planning, plan verification, and plan recognition problems. Hence, it is a universal approach covering various plan-related problems. The second area is correcting hierarchical models, an incremental version of model learning techniques. The proposed concept integrates hybrid planning materialized in TIHTN and automated updating of the task hierarchy (when the plan does not correspond to the model). Automated correction of hierarchical models brings the idea of continuous (never-ending) learning that may lead to the design of genuinely autonomous agents whose reasoning processes are still accessible (and explainable) to humans. If successful, this approach would solve one of the crucial problems of knowledge-based approaches – how to automatically learn the knowledge represented by hierarchical planning models.

Acknowledgments

Roman Barták is supported by the project 25-18003S of the Czech Science Foundation.

References

Aineto, D.; Jiménez, S.; and Onaindia, E. 2022. A Comprehensive Framework for Learning Declarative Action Models. *J. Artif. Intell. Res.*, 74: 1091–1123.

Amado, L.; Mirsky, R.; and Meneguzzi, F. 2022. Goal Recognition as Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9): 9644–9651.

Barták, R. 2016. Using Attribute Grammars to Model Nested Workflows with Extra Constraints. In Freivalds, R. M.; Engels, G.; and Catania, B., eds., *SOFSEM 2016: Theory and Practice of Computer Science*, 171–182. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-662-49192-8.

Barták, R.; and Dvorák, T. 2016. On Verification of Workflow and Planning Domain Models Using Attribute Grammars. In Sidorov, G.; and Herrera-Alcántara, O., eds., *Advances in Computational Intelligence - 15th Mexican International Conference on Artificial Intelligence, MICAI 2016, Cancún, Mexico, October 23-28, 2016, Proceedings, Part I*, volume 10061 of *Lecture Notes in Computer Science*, 332–345. Springer.

Barták, R.; Maillard, A.; and Cardoso, R. C. 2018. Validation of Hierarchical Plans via Parsing of Attribute Grammars. In *Proc. of the 28th Int. Conf. on Automated Planning and Scheduling (ICAPS 2018)*, 11–19. AAAI Press.

Barták, R.; Maillard, A.; and Cardoso, R. C. 2020. Parsing-based Approaches for Verification and Recognition of Hierarchical Plans. In *The AAAI 2020 Workshop on Plan, Activity, and Intent Recognition*.

Barták, R.; Ondrcková, S.; Behnke, G.; and Bercher, P. 2021. Correcting Hierarchical Plans by Action Deletion. In Bienvenu, M.; Lakemeyer, G.; and Erdem, E., eds., *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, 99–109.

Barták, R.; Ondrčková, S.; Behnke, G.; and Bercher, P. 2021. On the verification of totally-ordered HTN plans. In *2021 IEEE 33rd Int. Conf. on Tools with Artificial Intelligence (ICTAI)*, 263–267. IEEE.

Barták, R.; Ondrčková, S.; Maillard, A.; Behnke, G.; and Bercher, P. 2020. A Novel Parsing-based Approach for Verification of Hierarchical Plans. In *Proc. of the 32nd Int. Conf. on Tools with AI (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 *Proc. of the 25th Int. Conf. on Automated Planning and Scheduling (ICAPS 2015)*, 25–33. AAAI Press.

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

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning - One Abstract Idea, Many Concrete Realizations. In Kraus, S., ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 6267–6275. ijcai.org.

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 *Proc. of the 22nd European Conf. on AI (ECAI 2016)*, 225–233. IOS Press.

Bercher, P.; Keen, S.; and Biundo, S. 2014. Hybrid Planning Heuristics Based on Task Decomposition Graphs. In Edelkamp, S.; and Barták, R., eds., *Proceedings of the Seventh Annual Symposium on Combinatorial Search, SOCS 2014, Prague, Czech Republic, 15-17 August 2014*, 35–43. AAAI Press.

Colledanchise, M.; and Ögren, P. 2017. Behavior Trees in Robotics and AI: An Introduction. *CoRR*, abs/1709.00084.

Earley, J. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2): 94–102.

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

Estlin, T. A.; Chien, S. A.; and Wang, X. 1997. An Argument for a Hybrid HTN/Operator-Based Approach to Planning. In Steel, S.; and Alami, R., eds., *Recent Advances in AI Planning, 4th European Conference on Planning, ECP'97, Toulouse, France, September 24-26, 1997, Proceedings*, volume 1348 of *Lecture Notes in Computer Science*, 182–194. Springer.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. In *Proc. of the 2nd Int. Joint Conf. on AI (IJCAI 1971)*, 608–620.

Geib, C.; and Goldman, R. 2009. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11): 1101–1132.

Geib, C.; Maraist, J.; and Goldman, R. 2008. A New Probabilistic Plan Recognition Algorithm Based on String Rewriting. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling*, 91–98.

Geier, T.; and Bercher, P. 2011. On the decidability of HTN planning with task insertion. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, 1955.

Georgievski, I.; and Aiello, M. 2015. HTN planning: Overview, comparison, and beyond. *Artif. Intell.*, 222: 124–156.

Gerevini, A.; Kuter, U.; Nau, D. S.; Saetti, A.; and Waisbrot, N. 2008. Combining Domain-Independent Planning and HTN Planning: The Duet Planner. In Ghallab, M.; Spyropoulos, C. D.; Fakotakis, N.; and Avouris, N. M., eds.,

ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings, volume 178 of *Frontiers in Artificial Intelligence and Applications*, 573–577. IOS Press.

Goldman, R. P.; Kuter, U.; and Freedman, R. G. 2020. Stable plan repair for state-space HTN planning. In *Proceedings of the 3rd ICAPS Workshop on Hierarchical Planning (HPlan 2020)*, 27–35.

Ha, E.; Rowe, J.; Mott, B.; and Lester, J. 2011. Goal recognition with Markov logic networks for player-adaptive games. In *Proceedings of the seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 6.

Hogg, C.; Muñoz-Avila, H.; and Kuter, U. 2016. Learning Hierarchical Task Models from Input Traces. *Comput. Intell.*, 32(1): 3–48.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2018. Plan and goal recognition as HTN planning. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence*, 466–473.

Holler, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2019. HDDL – A Language to Describe Hierarchical Planning Problems. arXiv:1911.05499.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2020. HTN plan repair via model transformation. In *KI 2020: Advances in Artificial Intelligence: 43rd German Conference on AI, Bamberg, Germany, September 21–25, 2020, Proceedings 43*, 88–101. Springer.

Höller, D.; Wichlacz, J.; Bercher, P.; and Behnke, G. 2022. Compiling HTN Plan Verification Problems into HTN Planning Problems. In *Proc. of the 32nd Int. Conf. on Automated Planning and Scheduling (ICAPS 2022)*. AAAI Press.

Howey, R.; and Long, D. 2003. VAL's Progress: The Automatic Validation Tool for PDDL2.1 used in the Int. Planning Competition. In *Proc. of the ICAPS'03 Workshop on the Competition: Impact, Organization, Evaluation, Benchmarks*.

Kaelbling, L. P.; and Lozano-Pérez, T. 2011. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, 1470–1477.

Kaminka, G. A.; Pynadath, D. V.; and Tambe, M. 2002. Monitoring teams by overhearing: A multi-agent plan-recognition approach. *Journal of artificial intelligence research*, 17: 83–135.

Kelly, J.-P.; Botea, A.; and Koenig, S. 2021. Offline Planning with Hierarchical Task Networks in Video Games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 60–65.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Bartlett, P. L.; Pereira, F. C. N.; Burges, C. J. C.; Bottou, L.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, 1106–1114.

Langley, P. 2022. Learning Hierarchical Problem Networks for Knowledge-Based Planning. In Muggleton, S. H.; and Tamaddoni-Nezhad, A., eds., *Inductive Logic Programming - 31st International Conference, ILP 2022, Windsor Great Park, UK, September 28-30, 2022, Proceedings*, volume 13779 of *Lecture Notes in Computer Science*, 69–83. Springer.

LeCun, Y. 2022. A Path Towards Autonomous Machine Intelligence. OpenReview.net:Position Paper.

Li, J.; Chen, Z.; Zheng, Y.; Chan, S.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2021. Scalable Rail Planning and Replanning: Winning the 2020 Flatland Challenge. In Biundo, S.; Do, M.; Goldman, R.; Katz, M.; Yang, Q.; and Zhuo, H. H., eds., *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling, ICAPS 2021, Guangzhou, China (virtual), August 2-13, 2021*, 477–485. AAAI Press.

Li, T.; Liu, Y.; Liu, Y.; Xiao, Y.; and Nguyen, N. A. 2020. Attack plan recognition using hidden Markov and probabilistic inference. *Computers & Security*, 97: 101974.

Lin, S.; Behnke, G.; Ondrcková, S.; Barták, R.; and Bercher, P. 2023. On Total-Order HTN Plan Verification with Method Preconditions - An Extension of the CYK Parsing Algorithm. In Williams, B.; Chen, Y.; and Neville, J., eds., *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, 12041–12048. AAAI Press.

Mirsky, R.; Gal, Y.; and Shieber, S. M. 2017. CRADLE: an online plan recognition algorithm for exploratory domains. *ACM Transactions on Intelligent Systems and Technology*, 8(3): 1–22.

Mirsky, R.; Keren, S.; and Geib, C. W. 2021. *Introduction to Symbolic Plan and Goal Recognition*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. ISBN 978-3-031-00461-2.

Nau, D. S.; Au, T.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *J. Artif. Intell. Res.*, 20: 379–404.

Nguyen, A. M.; Yosinski, J.; and Clune, J. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 427–436. IEEE Computer Society.

Niu, W.; Long, J.; Han, D.; and Wang, Y.-F. 2004. Human activity detection and recognition for video surveillance. In *Proceedings of the 2004 IEEE International Conference on Multimedia and Expo (ICME)*, volume 1, 719–722.

Ondrcková, S.; and Barták, R. 2023. On Semantics of Hierarchical Planning Domain Models with Decomposition Constraints and Empty Methods. In *35th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2023, Atlanta, GA, USA, November 6-8, 2023*, 349–353. IEEE.

Ondrcková, S.; Pantucková, K.; and Barták, R. 2024. Handling Empty Decomposition Methods in Hierarchical Planning. In Chun, S. A.; and Talbert, D. A., eds., *Proceedings of the Thirty-Seventh International Florida Artificial Intelligence Research Society Conference, FLAIRS 2024, Sandestin Beach, FL, USA, May 19-21, 2024*. AAAI Press.

Pantucková, K.; and Barták, R. 2023. Using Earley Parser for Recognizing Totally Ordered Hierarchical Plans. In Gal, K.; Nowé, A.; Nalepa, G. J.; Fairstein, R.; and Radulescu, R., eds., *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, 1819–1826. IOS Press.

Pantucková, K.; and Barták, R. 2025a. Hierarchical Plan Repair via Plan Recognition and Plan Correction. In Talbert, D. A.; and Biskri, I., eds., *Proceedings of the 38th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2025, Daytona Beach, FL, USA, May 20-23, 2025*. Florida Online Journals.

Pantucková, K.; and Barták, R. 2025b. Parsing-based Planner for Totally Ordered HTN Planning with Task Insertion. In *Proceedings of the 37th International Conference on Tools with Artificial Intelligence (ICTAI)*, 483–490. IEEE.

Pantucková, K.; Ondrcková, S.; and Barták, R. 2024. Using Earley Parser for Verification of Totally Ordered Hierarchical Plans. In Chun, S. A.; and Talbert, D. A., eds., *Proceedings of the Thirty-Seventh International Florida Artificial Intelligence Research Society Conference, FLAIRS 2024, Sandestin Beach, FL, USA, May 19-21, 2024*. AAAI Press.

Pantučková, K.; and Barták, R. 2025. What to Do if a Plan Does Not Comply with an HTN Model. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)*, 4953–4960.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2017. Landmark-based heuristics for goal recognition. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 3622–3628.

Ramírez, M.; and Geffner, H. 2009. Plan recognition as planning. In *Proceedings of the twenty-first International Joint Conference on Artifical Intelligence*, 1778–1783.

Russell, S.; and Norvig, P. 2020. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson. ISBN 9780134610993.

Schattenberg, B.; and Biundo, S. 2006. A Unifying Framework for Hybrid Planning and Scheduling. In Freksa, C.; Kohlhase, M.; and Schill, K., eds., *KI 2006: Advances in Artificial Intelligence, 29th Annual German Conference on AI, KI 2006, Bremen, Germany, June 14-17, 2006, Proceedings*, volume 4314 of *Lecture Notes in Computer Science*, 361–373. Springer.

Shvo, M.; Klassen, T. Q.; Sohrabi, S.; and McIlraith, S. A. 2020. Epistemic Plan Recognition. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 1251–1259.

Valmeekam, K.; Sreedharan, S.; Marquez, M.; Hernandez, A. O.; and Kambhampati, S. 2023. On the Planning Abilities of Large Language Models (A Critical Investigation with a Proposed Benchmark). *CoRR*, abs/2302.06706.

Van-Horenbeke, F. A.; and Peer, A. 2021. Activity, Plan, and Goal Recognition: A Review. *Front. Robot. AI*, 8.

Vered, M.; Pereira, R. F.; Kaminka, G.; and Meneguzzi, F. R. 2018. Towards online goal recognition combining goal mirroring and landmarks. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, 2112–2114.

Vomlelová, M.; Vodrázka, J.; Barták, R.; and Chrpa, L. 2020. Automated Acquisition of Control Knowledge for Classical Planners. In Rocha, A. P.; Steels, L.; and van den Herik, H. J., eds., *Proceedings of the 12th International Conference on Agents and Artificial Intelligence, ICAART 2020, Volume 2, Valletta, Malta, February 22-24, 2020*, 959–966. SCITEPRESS.

Zaidins, P.; Roberts, M.; and Nau, D. 2023. Implicit Dependency Detection for HTN Plan Repair. *Proceedings of the 6th ICAPS Workshop on Hierarchical Planning (HPlan 2023)*, 10–18.