Extracting Planning Domains from Execution Traces: a Progress Report

Simona Gugliermo [1,2], Erik Schaffernicht [1], Christos Koniaris[2], Alessandro Saffiotti [1].

[1]Center for Applied Autonomous Sensor Systems, Örebro University, S-701 82 Örebro, Sweden
[2]Intelligent Transport Systems, Scania CV AB, S-151 87 Södertälje, Sweden
simona.gugliermo@oru.se, erik.schaffernicht@oru.se, christos.koniaris@scania.com, alessandro.saffiotti@oru.se

Abstract

One of the difficulties of using AI planners in industrial applications pertains to the complexity of writing planning domain models. These models are typically constructed by domain planning experts and can become increasingly difficult to codify for large applications. In this paper, we describe our ongoing research on a novel approach to automatically learn planning domains from previously executed traces using Behavior Trees as an intermediate human-readable structure. By involving human planning experts in the learning phase, our approach can benefit from their validation. This paper outlines the initial steps we have taken in this research, and presents the challenges we face in the future.

Introduction

In the current industrial practice within the domain of autonomous systems, plans are typically manually designed by "planning experts" who have extensive knowledge of the application domain. Even when automated planners (AI planners) are employed, they still rely on manually specified domain knowledge about the system and application context to derive plans. Such AI planners typically work by taking in information about the environment, encoded in the so called *planning domain*, in the form of action models with preconditions and effects (Ghallab, Nau, and Traverso 2016). Generating domains by hand is time-consuming or even infeasible for large applications (Jiménez et al. 2012). As a result, one of the reasons why AI planning techniques are not widely adopted in real applications is the burden of designing accurate and complete domain descriptions. To address this issue, our ongoing research aims to automatically learn planning domains by extracting and formalizing expert knowledge from historical executed traces.

We propose a 2-step learning approach, shown in Figure 1, that utilizes an intermediate structure to involve the human planning expert in the learning phase, as we want to integrate the manual planning experts' knowledge and their experience. However, engaging human planning experts requires a structure that is understandable even to those without an AI planning background. Domain description languages, such as PDDL and its extensions (Ghallab et al. 1998; Kovacs 2011) are difficult to understand for manual planning experts who are not used to logical specifications. Indeed, planning domains are able to represent a wide range of problems from different domains such as robotics, logistics, manufacturing and more. Nevertheless, the flexibility of these domains often leads to an increase in complexity and may pose challenges in terms of readability and comprehension, particularly for individuals who lack experience with the syntax and conventions of the planning domain languages. Therefore, we propose to use Behavior Trees (BTs) (Biggar, Zamani, and Shames 2021) as an intermediate structure to make the learning process transparent and intuitive. BTs provide a hierarchical structure that breaks down complex behaviors into smaller, simpler subtasks that are intuitive to understand and easy to modify. In essence, the advantages of using BTs align with those of HTN planning, an extension of classical planning that incorporates a hierarchical structure into domain representations (Ghallab, Nau, and Traverso 2016). Therefore, although the purposes of HTN planners and BTs differ, the adoption of BTs promises to be beneficial.

In the next section, we review some previous approaches for learning planning domains from execution traces. We then outline our approach and the main challenges that arise

Figure 1: The proposed approach generates planning models from a set of execution traces using Behavior Trees as an intermediate representation in order to engage human planning experts and account for their experiential knowledge.

in each step. Finally, we present our current progress in realizing these steps, discuss the open points, and conclude.

Related Work

Learning planning domains has been a focus area in the knowledge engineering literature. Methods such as LOCM2 (Cresswell and Gregory 2011), NLOCM (Gregory and Lindsay 2016) and AMAN (Zhuo and Kambhampati 2013) generate action models based on plan traces. The LOCM family of algorithms inputs entirely deterministic plan traces and operates under the assumption that each object can be represented by parameterized finite state machines. In contrast, the AMAN algorithm takes in noisy plan traces and employs a probabilistic graphical model-based approach. However, the aforementioned methodologies lack an interpretable intermediary structure that can facilitate interaction with humans, and are restricted to certain specific types of representations.

Konidaris, Kaelbling, and Lozano-Perez (2018) use reinforcement learning to learn a set of low-level action represented as a policy. In a second step, the learned actions are used to automatically generate a symbolic representation, which serve as the basis for the domain representation. In this case, the traces are not historical yet generated through exploration and trial-and-error within the environment. On the other hand, Ahmetoglu et al. (2022) exploit deep learning to generate symbols and rules that can be used for planning. This is an end-to-end learning approach that maps traces directly to planning domain without giving any guarantees about the resulting planning domain.

Arora et al. (2018) poses the problem of learning the user domain knowledge. The authors claim that even the most advanced state-of-the-art ML techniques may not eliminate the need and benefits of human intervention, especially for real industrial applications. There have been several studies on AI planning with humans in the loop. One well known framework is Human-in-the loop planning or HILP (Sengupta et al. 2017) where the idea is to add an AI planning layer to support the human expert in decision making and planning. Another research field where humans are involved in planning is mixed-initiative goal manipulation (Cox and Zhang 2005). In this case, users can establish and "steer" goals to provide better plan quality. Collaborative planning techniques were also explored by Kim, Banks, and Shah (2017), who proposed to involve humans before the planning phase by providing high-level strategies encoded as soft preferences to guide the low-level search of the planner. However, all these approaches aim at improving the generated plan rather than addressing our main objective of enhancing the learned domain.

Ilghami et al. (2005) explored learning algorithms that aim to acquire HTN domain knowledge from plan traces. This is based on the assumption that it is known in advance how to decompose tasks, but not under what conditions each of the several methods to decompose a task is applicable. An alternative approach proposed by Xu and Muñoz Avila (2005) is a lazy learning algorithm that leverages the structure of methods and an ontology of types to learn preconditions. However, both methods lack the ability to learn the method structure and infer causal relationships between multiple actions, limiting their applicability. Zhuo et al. (2009) developed the HTNlearner algorithm, which learns method preconditions and action models simultaneously, assuming annotated tasks with preconditions and effects. To overcome the deterministic assumption, (Hogg, Kuter, and Muñoz Avila 2009) investigated learning hierarchical task networks in non-deterministic planning domains, where actions may have multiple outcomes, to overcome the deterministic assumption. Despite the successes achieved by these previous learning systems, it is important to note that they exhibit a limitation that is relevant to our objectives, as they presume perfect observability, which may pose a limitation in real-world scenarios.

Proposed Approach

The proposed method, shown in Figure 1 above, aims to extract planning domains from execution traces of previously manually written plans, using a two-phase approach. In the first phase, the objective is to extract the human domain knowledge encoded in the historical execution traces into an intermediate representation. The second phase of the learning process involves the obtaining of a domain theory, which delineates the collection of actions that can be enacted, the conditions under which these actions may be executed, and the effects of such actions.

The reason for introducing an intermediary representation between these phases is two-fold. First, to simplify the learning problem by reducing the abstraction gap between execution traces and symbolic action representations. Second, to enable human inspection and validation, thereby enhancing transparency in the learning process. Accordingly, we suggest utilizing BTs, which, as per Colledanchise and Ögren (2018), are known to be comprehensible by humans.

We now present a succinct outline of BTs, as well as the stages and challenges associated with the proposed approach, along with the related state-of-the-art methods.

Behavior Trees

BTs have recently gained popularity in the robotics community due to their effectiveness in modelling the decision-making policy of an agent designed for reactivity, modularity, while also promoting transparency. Its hierarchical structure provides a clear and intuitive representation of the agent's decision-making process, thus facilitating comprehension and modification for individuals without programming or AI background. BT execution starts from its root node, generating signals called *Ticks* with a given frequency. The internal nodes are control flow nodes, and leaf nodes are execution nodes. During the execution, the possible states of each node are *Running*, *Success* or *Failure*. In the classical formulation, the standard control flow nodes are Sequence, Fallback or Parallel. A Sequence node executes its children in a depth-first order until one returns *Failure*, at which point it stops and fails itself. In contrast, a Fallback node executes its child nodes in a depth-first order until one returns *Success*, upon which it stops and succeeds itself. Lastly, a Paral-

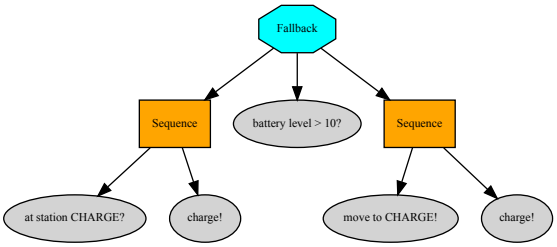

Figure 2: Example of a Behavior Tree.

lel node propagates the tick simultaneously to all its children and it succeeds if a set amount of them return *Success*. The standard execution nodes are action and condition nodes. The former represent the physical actions and typically stands for the execution of a command. The latter check a proposition when receiving signals.

To better understand how BTs work in practice, consider the example of the battery management of an autonomous agent, depicted in Figure 2. If the agent is located at the charging station, it engages in the battery charging behavior. In case the agent is not situated at the charging station but the battery level exceeds 10%, the condition node returns *Success*, and the system remains idle until the following *Tick*. Otherwise, if the battery level drops below 10%, the agent moves towards the charging station to recharge its battery. Note that this BT can be used as a subtree within a more intricate BT that integrates multiple behaviors (modularity).

From executed traces to BT

We consider execution traces consisting of high-level snapshots of the environment during the plan execution. Therefore, they are represented in the format of ($State_0$,$Action_0$)...($State_n$,$Action_n$), where n denotes the total number of snapshots, $State_i$ describes the environment variables at the i-th snapshot, including the agent's position, carried weight and the battery status, while $Action_i$ explicates the corresponding action undertaken by the agent at the i-th snapshot.

Learning BTs from previous executed plans is similar to Learning from Demonstation (LfD), where traces are typically recorded during a demonstration, and robots acquire new skills by learning to imitate an expert. One common approach to learning a BT from demonstration is to leverage Decision Trees (DT) as intermediaries (French et al. 2019). To this end, data mining algorithms, such as C5.0 (Kuhn and Johnson 2013), and CART (Breiman et al. 1984) can be employed to infer the knowledge that is implicitly encoded in the previous plans in the form of if-then rules. This simple representation shows what conditions must be met for an action to occur. Finally, the DT can be converted into a BT by using the naive Decision Tree to Behavior Tree algorithm proposed by French et al. (2019). Although the algorithm can convert the DT's if-then rules into a BT's hierarchical structure, it does not fully leverage the interpretable and expressive power of the BT architecture. Wathieu et al. (2022) partially addressed this issue by removing some of the logical redundancies in the generated BT.

We have developed a novel framework for automatically extracting and representing the implicit action knowledge encoded in execution traces within a BT. This framework is comprehensively detailed in (Gugliermo et al. 2023), where we provide the detailed algorithm, link to the source code and include examples to aid understanding of the methodology. In detail, the method is based on decision trees and logic factorization techniques, resulting in an interpretable BT that can be inspected by a human expert.

Interactive validation of BT

By involving a human planning expert, it is possible to ensure that the learned BT is correct (verification), satisfies the requirements (validation), and is free of errors and bugs (debugging). Tools are available for interactive validation of behavior trees such as, Behavior3 Editor [1], which is a simulation editor that includes a simulation mode for interactive validation, and PyTree [2] and Groot[3], which are Python and C++ libraries, respectively, that include tools for interactive validation. These methods are primarily based on testing and simulating BTs to assess them.

What is currently missing is a well-defined way to measure and evaluate a BT and its properties through specific, quantifiable metrics. The use of such metrics would aid the human expert in a more immediate and efficient evaluation.

From BT to planning domains

Once we have a validated BT, we need to extract an abstract planning domain from it. In the field of AI planning, causality holds a fundamental role in defining the planning domain, as it enables the definition of dependencies between different actions and their effects.

BTs do not inherently possess a causal structure, but it is possible to incorporate causal relationships between the different behaviors represented in the tree. Therefore, to extract a planning domain from a BT, it is necessary either to embed causal relationships into the BT's design or to enrich the BT with a causal structure. One possible approach to enhance the causal structure of a BT is by using causal inference techniques (Nogueira et al. 2022) to automatically extract causal relationships between actions and variables. This can be achieved by analyzing the statistical dependencies between actions and changes in state variables, thus inferring the underlying causal mechanisms that give rise to these dependencies. Once the causal relationships have been identified, they can be leveraged to ensure that the plan is consistent with the causal structure of the domain, resulting in more effective and robust planning.

Current Results and Next Steps

Our research efforts so far have been focused on the automatic extraction of BTs from execution traces, while we have only recently begun exploring methods to identify and extract causal dependencies between actions and variables in the domain.

[1] https://www.behaviortrees.com/#/dash/home

[2] https://py-trees.readthedocs.io/en/devel/

[3] https://www.behaviortree.dev/groot/

Table 1: Results of correctness experiments for BT-Factor and RE:BT-Espresso. The table shows the percentage of simulations in which each system achieved correct behavior. BT-Factor achieved 100% correct behavior in all 200 simulations, while RE:BT-Espresso achieved correct behavior in 62.5% of simulations when selecting the BT based on the amount of material delivered, and in 72% of simulations when the learned tree was not pruned.

Method	**% correct behavior**
BT-Factor	100%
RE:BT-Espresso	62.5%
RE:BT-Espresso (no pruning)	72.0%

Table 2: Results of efficiency experiments for BT-Factor and RE:BT-Espresso. The table shows the mean and variance values for all simulations, as well as for the subset of simulations where RE:BT-Espresso learned correct behaviors (in parentheses). Higher mean values indicate better performance, while lower variance values indicate more stability.

	Mean	**Variance**
BT-Factor	38.505 (37.928)	22.642 (24.151)
RE:BT-Espresso	32.433 (36.704)	108.348 (24.084)

BT extraction

To evaluate our approach (Gugliermo et al. 2023), we conducted experiments in a simulated environment that closely mirrors realistic industrial logistics applications. The environment features an autonomous agent equipped with a battery, two loading areas, a depot, and a charging station. Empirical evidence demonstrates that our method captures the logical elements implicitly encoded in the historical data and generates correct behavior in the simulated scenario, where correctness denotes the agent's ability to execute the policy during the entire duration of the simulation. Furthermore, we examine the generalization ability of our framework by introducing noise into executed traces to disrupt determinism. The framework generalizes and discards outliers when the signal-to-noise ratio (SNR) is higher than 5, allowing for generalization across possible task executions. Finally, we conducted a comparison between our method and a recent state-of-the-art algorithm, RE:BT-Espresso (Wathieu et al. 2022). It is worth noting that the latter is incapable of identifying the appropriate pruning level for the extracted BT, resulting in the extraction of multiple BTs and requiring the user to select the desired one. Our method outperforms RE:BT-Espresso across various key aspects. Firstly, in terms of domain-dependent measures, such as correctness (Table 1) and efficiency (Table 2), which is quantified by the amount of material delivered. Additionally, our method improves BT synthesis and enhances its readability, resulting in a reduction in the number of nodes.

Overall, the key innovations of the developed method are (i) the use of C5.0 as an algorithm for learning the Decision Tree and (ii) the introduction of a logical factorization step based on a state of the art method developed for designing logic circuit (Wang 1989).

Causal inference

To write a planning domain, it is essential to infer causal relationships between actions and state variables, which are encoded in the form of preconditions and effects. In order to deduce the preconditions, the DT obtained with the previous method can be used, as it expresses the conditions necessary for an action to be performed. In contrast, inferring the effects of an action on a state is a more complex task.

We are currently investigating an approach that makes use of observable historical data, as illustrated in Figure 3. The underlying idea is that some state variables change even without performing any action, such as the agent's battery level. Therefore, we initially analyze the changes of the state variables when no action, denoted by Σ, is performed. This study is then repeated for each action $a \in A$. The effects of an action a incorporate the effects of the null action Σ, which do not depend on the action a itself. This approach is inspired by the Difference-in-Differences (DID) technique (Hausman 1978). Note that this is applicable when the statistical assumption of parallel trends holds, which requires that the trend of the state variables would have been parallel in the absence of action.

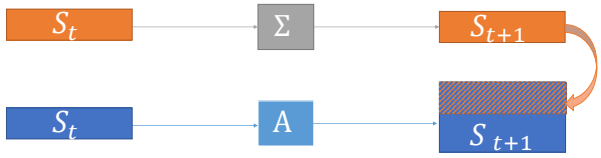

Figure 3: Schematic representation of the proposed approach for inferring action effects, drawing inspiration from the Difference-in-Differences technique.

Conclusions

By automating the generation of planning domains and formalizing expert knowledge, our approach can facilitate the transfer of knowledge among multiple individuals responsible for planning in industrial contexts. The initial steps presented here suggest that BTs are a suitable intermediate structure to automate this generation. We envision that BTs will also help to bridge the gap between human knowledge and machine representation, making the planning process more efficient and transparent for real-world applications. Finally, we hope that our learning approach will help industrial automated planning and scheduling systems to scale up, and produce high-quality plans across diverse domains such as mining, construction, and material transport.

Acknowledgment

This work is financially supported by the Swedish Foundation for Strategic Research (SSF).

References

Ahmetoglu, A.; Seker, M. Y.; Piater, J.; Oztop, E.; and Ugur, E. 2022. DeepSym: Deep Symbol Generation and Rule Learning for Planning from Unsupervised Robot Interaction. *J. of Artificial Intelligence Research*, 75: 709–745.

Arora, A.; Fiorino, H.; Pellier, D.; Métivier, M.; and Pesty, S. 2018. A review of learning planning action models. *The Knowledge Engineering Review*, 33.

Biggar, O.; Zamani, M.; and Shames, I. 2021. An expressiveness hierarchy of behavior trees and related architectures. *IEEE Robotics and Autom. Letters*, 6(3): 5397–5404.

Breiman, L.; Friedman, J.; Stone, C. J.; and Olshen, R. 1984. *Classification and Regression Trees*. Chapman and Hall.

Colledanchise, M.; and Ögren, P. 2018. *Behavior Trees in Robotics and AI*. CRC Press.

Cox, M. T.; and Zhang, C. 2005. Planning as mixed-initiative goal manipulation. In *Int Conf on Automated Planning and Scheduling*, 282–291.

Cresswell, S.; and Gregory, P. 2011. Generalised Domain Model Acquisition from Action Traces. In *Int Conf on Automated Planning and Scheduling*, ICAPS'11, 42–49.

French, K.; Wu, S.; Pan, T.; Zhou, Z.; and Jenkins, O. C. 2019. Learning Behavior Trees From Demonstration. In *Int Conf on Robotics and Automation (ICRA)*, 7791–7797.

Ghallab, M.; Howe, A.; Knoblock, C.; McDermott, D.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. The planning domain definition language.

Ghallab, M.; Nau, D.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

Gregory, P.; and Lindsay, A. 2016. Domain Model Acquisition in Domains with Action Costs. In *Int Conf on Automated Planning and Scheduling*, ICAPS'16, 149–157.

Gugliermo, S.; Schaffernicht, E.; Koniaris, C.; and Pecora, F. 2023. Learning Behavior Trees From Planning Experts Using Decision Tree and Logic Factorization. *IEEE Robotics and Automation Letters*, 8(6): 3534–3541.

Hausman, J. A. 1978. Specification Tests in Econometrics. *Econometrica*, 46(6): 1251–1271.

Hogg, C.; Kuter, U.; and Muñoz Avila, H. 2009. Learning Hierarchical Task Networks for Nondeterministic Planning Domains. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, 1708–1714. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ilghami, O.; Muñoz-Avila, H.; Nau, D. S.; and Aha, D. W. 2005. Learning approximate preconditions for methods in hierarchical plans. *Proceedings of the 22nd international conference on Machine learning*.

Jiménez, S.; De La Rosa, T.; Fernández, S.; Fernández, F.; and Borrajo, D. 2012. A review of machine learning for automated planning. *The Knowledge Engineering Review*, 27(4): 433–467.

Kim, J.; Banks, C. J.; and Shah, J. A. 2017. Collaborative Planning with Encoding of Users' High-Level Strategies. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, 955–961.

Konidaris, G.; Kaelbling, L. P.; and Lozano-Perez, T. 2018. From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning. *J. Artif. Int. Res.*, 61(1): 215–289.

Kovacs, D. L. 2011. BNF definition of PDDL 3.1. *Unpublished manuscript from the IPC-2011 website*, 15.

Kuhn, M.; and Johnson, K. 2013. *Applied Predictive Modeling*, volume 26. Springer.

Nogueira, A. R.; Pugnana, A.; Ruggieri, S.; Pedreschi, D.; and Gama, J. 2022. Methods and tools for causal discovery and causal inference. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(2): e1449.

Sengupta, S.; Chakraborti, T.; Sreedharan, S.; Vadlamudi, S. G.; and Kambhampati, S. 2017. RADAR - A Proactive Decision Support System for Human-in-the-Loop Planning. In *AAAI Fall Symposia*.

Wang, A. R. R. 1989. *Algorithms for multilevel logic optimization*. Ph.D. thesis, University of California, Berkeley.

Wathieu, A.; Groechel, T. R.; Lee, H. J.; Kuo, C.; and Mataric, M. J. 2022. RE:BT-Espresso: Improving Interpretability and Expressivity of Behavior Trees Learned from Robot Demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Xu, K.; and Muñoz Avila, H. 2005. A Domain-Independent System for Case-Based Task Decomposition without Domain Theories. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1*, AAAI'05, 234–239. AAAI Press. ISBN 157735236x.

Zhuo, H. H.; Hu, D. H.; Hogg, C.; Yang, Q.; and Munoz-Avila, H. 2009. Learning HTN Method Preconditions and Action Models from Partial Observations. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, 1804–1809. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Zhuo, H. H.; and Kambhampati, S. 2013. Action-Model Acquisition from Noisy Plan Traces. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, 2444–2450.