# Learning Hierarchical Task Knowledge for Planning

**Pat Langley**
ARCAID, Information and Communications Laboratory
Georgia Tech Research Institute, Atlanta, GA 30318
Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, CA 94306
http://www.isle.org/~langley/

**Abstract**

In this paper, I review approaches for acquiring hierarchical knowledge to improve the effectiveness of planning systems. First I note some benefits of such hierarchical content and the advantages of learning over manual construction. After this, I consider alternative paradigms for encoding and acquiring plan expertise before turning to hierarchical task networks. I specify the inputs to HTN learners and three subproblems they must address: identifying hierarchical structure, unifying method heads, and finding method conditions. Finally, I pose seven challenges the community should pursue so that techniques for learning HTNs can reach their full potential.

## Background and Motivation

Early applications of AI revolved around expert systems, which produced many successes but also relied on manual elicitation and entry of domain knowledge (Waterman, 1986). This made them expensive to develop, difficult to maintain, and subject to errors. The field of machine learning, launched in 1980, aimed to automate creation of such knowledge bases. This movement also had many successes, initially with supervised learning from tabular data for classification (Langley and Simon, 1995) but recently with other types of content for vision, language, and game playing.

One area that has seen comparatively slow progress on learning has been *planning*. Although early examples appeared in the literature (e.g., Fikes et al., 1972; Minton, 1988), the automated acquisition of expertise has never become a central focus of planning research. One reason may be that mainstream planning methods rely on little domain knowledge and, despite the combinatorial character of many tasks, there has been steady progress toward techniques that scale to complex problems. This has led many researchers to conclude that learning is unnecessary for effective planning.

Nevertheless, there is clear evidence, both theoretical and empirical, that domain knowledge can offer substantial reduction in search and time required to find solutions. This holds especially for HTNs – hierarchical task networks (Nau et al., 2003) – and related formalisms, which specify ways to decompose complex planning problems into subproblems and thus eliminate many unproductive candidates from consideration. The challenge here is the same as that confronted by the expert system community: extracting and encoding this knowledge is a tedious, expensive, and error-prone process. This in turn suggests that acquiring HTNs automatically, rather than constructing them by hand, would offer the same advantages as learning in other AI areas.

In the following sections, I review the benefits of hierarchical organization and alternative ways to acquire plan knowledge. After this, I concentrate on HTN learning, examining the inputs to this task and identifying three primary subproblems that it must address, in each case citing relevant work. Finally, I present seven additional hurdles that the research community must overcome before techniques for the automated acquisition of HTNs can offer the same advantages that machine learning has provided in other settings.

## Benefits of Hierarchical Decomposition

We all have experience decomposing complex problems into subproblems, as people organize everyday activities in this manner. For example, a common morning routine involves getting out of bed, taking a shower, and getting dressed, but each of these is broken down further into substeps. Taking a shower in turn requires entering the bathroom, starting the water, getting into the shower, washing with soap, and so forth. This decomposition continues until it reaches primitives that handle low-level movements. Our repertoires of routine activities are organized into rich hierarchies of such skills, each of them aimed at achieving a specific goal but also having considerable generality. People store this knowledge in long-term memory and retrieve relevant structures as needed to achieve their current objectives.

The benefits of activity decomposition are even greater when one encounters novel problems that require search to solve. Recall that, in the worst case, a problem space with branching factor $b$ and depth $d$ includes $b^d$ alternative paths, which rapidly becomes intractable for complex tasks with long solutions. Decomposing such a task into subproblems reduces the exponent $b$ and dividing subtasks into still smaller ones can give further savings. Breaking down a problem recursively can lower its computational complexity substantially. A classic example involves logistics planning, which requires selecting materials to transport and allocating vehicles for moving them to target destinations. Such problems have regular structures that suggest efficient ways to partition them into sets of simpler subtasks.

However, such savings only materialize if one chooses useful problem organizations. Uninformed search through a space of decompositions can actually increase complexity over search in the original 'flat' space. This means that specifying appropriate problem decompositions, including the conditions under which to select them, is essential for their effectiveness. And although we can specify such hierarchical breakdowns manually, this approach can take considerable effort and introduce errors. Thus, the greatest potential lies in acquiring these structures from experience, as in other applications of machine learning.

We should also discuss the performance element that uses hierarchical plan knowledge. Most work on HTNs accesses this content for plan generation, invoking it in a top-down manner to find action sequences that achieve specified goals. But one can use the same structures for knowledge-guided reactive control and even for monitoring progress of executed plans. Finally, one can draw on them for plan recognition and understanding, which involves inferring another agent's plans or goals from observed behavior. The hierarchical organization of knowledge has benefits in each setting, as the resulting plans have interpretable steps and describe activity at multiple levels of abstraction. The plan knowledge itself is also understandable, being closely akin to computer programs' procedures and subroutines.

## Varieties of Learning Plan Expertise

Before we can discuss approaches to learning plan knowledge, we must first specify the generic problem, which we can express in terms of inputs and outputs:

- *Given:* Knowledge of relations that describe states / goals
- *Given:* Knowledge about conditional effects of actions
- *Given:* Sample traces of problem solutions / failures
- *Find:* Knowledge that constrains / guides search for plans

Like other branches of machine learning, we can divide research into paradigms that differ in how they encode expertise (Langley, 1996), with implications for how planners use such content to make search more effective. These include:

- *Acquiring search-control knowledge*. This paradigm represents learned expertise as a set of rules for selecting, rejecting, or preferring states, goals, or operators. Planners use these structures as heuristics to guide their search for solutions and reduce the effective branching factor. The literature has explored two primary approaches to learning rules for search control. One involves induction over solution paths, treating decisions that produced them as positive training cases and decisions that would have diverged as negative ones (e.g., Sleeman et al., 1982). Another relies on logical analysis of solution paths and failures, compiling their steps into conditions under which similar decisions will and will not lead to problem solutions (e.g., Mitchell et al., 1986; Minton, 1988).
- *Forming macro-operators*. This framework encodes plan knowledge as sequences of actions or operators, along with their effects under specified conditions. These structures let planners take larger steps through a given problem space, which in turn reduces the effective depth of their search. Learning involves composing the conditions and effects for action sequences to produce new operators that describe the conditional results of the sequences (e.g., Fikes et al., 1972; Iba, 1989). Advanced variants can learn disjunctive, iterative, and even recursive macro-operators that are similar in spirit to HTNs (e.g., Shell and Carbonell, 1989; Shavlik, 1990).
- *Learning hierarchical task networks*. This paradigm, our central concern here, represents planning expertise as a set of hierarchical methods, each specifying a task, a set of subtasks, and its conditions for application. A close relative, *hierarchical goal networks*, indexes methods by goals they achieve rather than arbitrary task names. Typical HTN planners accept an initial state and a task or set of goals, then apply methods recursively to decompose the problem until it grounds out in primitive actions. This process involves search, but usually far less than required in the absence of hierarchical knowledge. At first glance, HTN learning appears similar to forming macro-operators, but the need to find conditions on methods has produced both inductive responses (e.g., Ilghami et al., 2005) and analytical ones (e.g., Hogg et al., 2008).

A fourth paradigm, which encodes expertise in a very different format, instead relies on numeric evaluation functions. This typically carries out heuristic search without explicit goals, using the function to select among successor states. Induction can take different forms, but a common variety is reinforcement learning, which often propagates state values backward over sequences of actions that have produced more or less desirable outcomes. This comes closest to techniques for learning search-control rules, although the two paradigms encode content in quite different ways.

I will limit consideration here to techniques for learning plan knowledge from sample solutions, but I should note two other frameworks that operate over different types of input. One is *interactive task learning*, in which a human instructs an agent, using a mixture of natural language and examples, how to break down complex procedures into simpler ones (e.g., Kirk and Laird, 2014). Another is *learning from written instructions*, such as those found in manuals, which involves reading textual descriptions and extracting interpretable procedures (Langley et al., 2024). Both paradigms show considerable promise for acquiring hierarchical plan knowledge, but I lack space to cover them in this paper.

## Forms of Input for HTN Learning

Above I reviewed different paradigms for acquiring plan expertise, but mainly to provide context for our focus – learning hierarchical task networks – to which I now turn. As noted earlier, this task requires three types of input: domain predicates that are used to represent states and goals; domain operators that serve as elements of plans; and sample plans that serve as training problems to drive learning.

The first two items are the same as for classic knowledge-lean planners (e.g., as encoded in PDDL) and require no further elaboration. However, the solutions for training problems can vary along three dimensions that merit discussion:

- *Source of sample solutions*. These may be generated by the planner itself, typically using knowledge-lean heuristic search on simple training problems (e.g., Choi and Langley, 2005; Li et al., 2009; Langley, 2023). Alternatively, the solutions may be provided by a human trainer who understands the domain and its problems (e.g., Ilghami et al., 2005; Nejati et al., 2006; Hogg et al., 2008). This difference is not as important as it first appears because systems that rely on self-generated solutions can pass them on to the same types of mechanisms that learn from externally provided solutions.
- *Structure of sample solutions*. These may involve a simple sequence of operator instances that transform the initial state into one that satisfies the goal description (e.g., Nejati et al., 2006).[1] Another option is for the human trainer to organize each sample solution into a hierarchy that decomposes the problem into subproblems (e.g., Ilghami et al., 2005; Hogg et al., 2008). This dimension interacts with the first one, as it requires extra effort from the human who provides sample solutions, but it can be generated automatically by some knowledge-lean planning methods (e.g., Li et al., 2009).
- *Presentation of sample solutions*. The learner may encounter solutions in an online manner and process them incrementally (e.g., Ilghami et al., 2005; Nejati et al., 2006; Hogg et al., 2008). Alternatively, the system may receive a set of training solutions en masse, which it then can process in batch mode (e.g., Li et al., 2014; Grand et al., 2022). The first option has been more common, partially because research in the area has often been inspired by the incremental nature of human learning, but both are legitimate approaches to HTN acquisition.

Choices along these dimensions serve as design decisions that set the stage for learning. We will see that they lend themselves to different approaches to the task of acquiring hierarchical task networks from training solutions.

## Three Elements of HTN Learning

An important step in addressing any complex task is to identify its constituent processes. We can decompose the problem of HTN learning into three subtasks that are naturally addressed in a logical sequence:

- *Identifying hierarchical structures*. The first step in constructing HTN methods is extracting the hierarchical structure of solutions for given training problems. Although some approaches assume this is provided (e.g., Hogg et al., 2008), it must often be found automatically. For instance, systems that generate their own solutions using backward chaining from goals produce a hierarchical decomposition for each problem (e.g., Choi and Langley, 2005; Langley, 2023). However, even learners that are given sample solutions can analyze goal dependencies (e.g., Nejati et al., 2006; Li et al., 2009) or landmarks (Fine-Morris et al., 2020) to generate hierarchies. A very different approach applies grammar-induction techniques to solution sequences to find recurring hierarchical structures (e.g., Li et al., 2014; Grand et al., 2022). Analytical approaches typically operate incrementally, whereas ones that invoke grammar induction rely on batch processing.
- *Unifying heads of methods*. The second step involves determining when two or more HTN methods should have same task name or head, which is essential for domains that benefit from disjunctive or recursive knowledge. Some approaches provide task names as part of hierarchical problem solutions (e.g., Ilghami et al., 2005; Hogg et al., 2008), but in other cases the learner must make this decision itself. One response is to unify and merge the heads of methods that achieve the same goal, which can be extracted from solutions generated by backward chaining (e.g., Langley and Choi, 2006; Langley, 2023) or from analysis of trainer-provided solutions (e.g., Nejati et al., 2006). In contrast, batch learners that use grammar-induction techniques to identify hierarchical structure can unify the heads of methods that appear in similar contexts (e.g., Li et al., 2014; Grand et al., 2022).
- *Finding method conditions*. The final step is finding conditions on methods that determine when the planning system should apply their associated decompositions. Some researchers have used analytical or explanation-based techniques (e.g., Hogg et al., 2008) for this purpose; these compile elements from the solution sequence into conditions that ensure the same results. This approach learns from individual cases, but it can generate idiosyncratic conditions that are unnecessarily specific. Others have relied on inductive techniques, ranging from the candidate elimination algorithm (Ilghami et al., 2005) to inductive logic programming (Reddy and Tadepalli, 1997; Cropper and Muggleton, 2015), which compare positive and negative instances. These find more general conditions, but they are either limited to conjunctive conditions or require more training data. Identifying conditions on methods is the most challenging facet of HTN learning and the one that remains its weakest link.

Despite the differences among approaches, they share the idea that learning plan knowledge involves the creation of new hierarchical structures. We have seen that the three constituent tasks can be tackled analytically, by reasoning over relations among plan steps, or statistically, by finding regularities in sample solutions. The first approach typically processes training cases incrementally, while the second usually operates in batch mode. To date, there have been no experimental comparisons of the alternatives, but such studies would be a valuable addition to the literature.

## Challenges for HTN Learning

Despite two decades of research on the issues above, techniques for acquiring HTNs from sample solutions have not yet replaced their manual construction. This may be due partly to advances in knowledge-lean planning that do not rely on hierarchical content, but another reason is that HTN learning has not delivered on its promise. In this section, I consider seven challenges that the community should address to achieve its original vision. These include:

[1] A rarely used alternative is a sequence of state descriptions, from which one can infer responsible operators with action models.

- *Improving condition finding*. Existing mechanisms for identifying conditions on methods do well given enough training data, but their learning rate is slower than humans, who master a new skill from only a few examples. Both analytical and inductive schemes fall short on this front, so we should explore alternatives. Langley (2023) reports a third option – noting mutually exclusive relations that occur before and after successful decompositions – that let it propose conditions based on single cases. However, there may exist other techniques that produce rapid, human-like learning of HTN conditions.
- *Acquiring binding constraints*. Although learned HTNs can greatly reduce the search for solutions, they suffer from the same limitation as macro-operators: an inability to distinguish desirable bindings from undesirable ones. For instance, a recursive method for building a tower can specify that one should stack A on B and stack B on C, but not the order of these two actions. A promising response is to extend the representation of methods to specify goals they achieve and to analyze sample solutions for interactions among them. Detecting interactions would introduce ordering constraints on methods, an idea related to work on partial-order planning but only recently applied to learning plan knowledge (Langley, 2023).
- *Integration with action learning*. Most approaches to acquiring HTNs depend on having correct action models, but this poses issues for complex domains, as specifying such descriptions manually can be time consuming and error prone. To automate HTN acquisition more fully, we should integrate techniques reviewed earlier with ones that learn action models from observation and experimentation, as in early work by Gil (1994) and Benson (1995). A straightforward scheme would apply the two processes in sequence, but the research community should also explore approaches that interleave them, including refinement of incomplete and incorrect action models.
- *Learning from partial solutions*. Approaches to HTN acquisition generally posit that sample solutions are complete and correct but, if provided by humans, this assumption may not hold. Thus, we also need research on how to acquire useful methods even when training cases have missing or extra actions, as well as when states have omitted or incorrect relations. Zhuo et al. (2009) and Grand et al. (2022) describe responses to a number of these issues, which they treat as intertwined with the previous challenge of learning and refining action models.
- *Forming conceptual predicates*. Mechanisms for learning HTNs offer a powerful way to find useful temporal abstractions, but they still rely on manual handcrafting of knowledge for state abstractions. We should also develop approaches that automate this latter process by introducing new relational predicates that describe regularities in states. As with learning action models, this could occur prior to HTN acquisition or be interleaved with it, although the second option would be desirable for extended operation. There has been some preliminary work along these lines (e.g., Li et al., 2012; Cropper and Muggleton, 2015), but we need more efforts on this topic.
- *Cumulative acquisition of methods*. Most work on HTN learning has focused on domains of moderate complexity that can be mastered rapidly. However, acquiring hierarchical plan knowledge in complex domains poses new challenges. E.g., a reconnaissance agent that operates on extended missions must address many goals and store methods to achieve them. Such scenarios will require online learning not only over long periods, but also cumulative learning in which later knowledge elements build on those acquired earlier. HTNs offer a natural response, in that the agent can recode its experiences in terms of high-level tasks and apply learning to create new methods stated in these abstract terms.
- *Evaluating candidate methods*. Sometimes learning can reduce search but actually increase planning time, a situation Minton (1988) has called the *utility problem*. The literature on HTN learning has never noted instances of this issue, but it seems likely to arise in complex domains that require acquisition of many methods. Thus, we need research on detecting the utility problem, say by keeping statistics for each method, and responding appropriately, say by removing ones whose costs outweigh their benefits. Another response would be to associate expected values with methods and use reinforcement learning to update them, with planners using these estimates to rank decompositions during search. Both Hogg et al. (2010) and Patra et al. (2020) report progress along these lines.

Taken together, advances in these areas would augment the abilities of HTN-learning systems and decrease reliance on human assistance. This in turn should improve their learning rates and asymptotic performance, letting them reduce or eliminate search on problems of substantial complexity.

## Concluding Remarks

In this paper, I reviewed the problem of learning expertise to guide planning, a topic that has not received the attention it deserves. The analysis focused on acquiring knowledge for decomposing complex tasks into simpler ones, with HTNs as the classic target, and I contrasted this problem with other formulations of acquiring plan expertise – learning search-control rules and forming macro-operators – that encode knowledge in different ways. I also identified the inputs to HTN learning and three issues it must handle – extracting the network's hierarchical structure, unifying heads of different methods, and finding conditions on their application.

In addition, I raised seven key challenges that future work on HTN learning should address. These included improved finding of conditions, acquiring constraints on bindings, integrated learning of methods and action models, constructing methods from partial solutions, inventing new predicates that describe states in abstract terms, cumulative acquisition of methods that build on ones created earlier, and evaluating candidates based on track records. In summary, learning hierarchical task networks from sample solutions offers great potential for improving the efficiency and scalability of AI planning systems, but further progress in this understudied area is essential if we want it to provide the same benefits that machine learning has produced elsewhere.

## Acknowledgements

This research was supported by Grant FA9550-20-1-0130 from the US Air Force Office of Scientific Research and by Grant N00014-23-1-2525 from the Office of Naval Research, which are not responsible for its contents.

## References

Benson, S. 1995. Inductive learning of reactive action models. *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 47–54. Tahoe City, CA: Morgan Kaufmann.

Choi, D., and Langley, P. 2005. Learning teleoreactive logic programs from problem solving. *Proceedings of the Fifteenth International Conference on Inductive Logic Programming*, pp. 51–68. Bonn, Germany: Springer.

Cropper, A., and Muggleton, S. H. 2015. Learning efficient logical robot strategies involving composable objects. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pp. 3423–3429. Buenos Aires, Argentina: AAAI Press.

Fikes, R. E., Hart, P. E., and Nilsson, N. J. 1972. Learning and executing generalized robot plans. *Artificial Intelligence*, *3*, 251–288.

Fine-Morris, M., et al. 2020. Learning hierarchical task networks with landmarks and numeric fluents by combining symbolic and numeric regression. *Proceedings of the Eighth Annual Conference on Advances in Cognitive Systems*.

Gil, Y. 1994. Learning by experimentation: Incremental refinement of incomplete planning domains. *Proceedings of the Eleventh International Conference on Machine Learning*. New Brunswick, NJ: Morgan Kaufmann.

Grand, M., Fiorino, H., and Pellier, D. 2022. An accurate HDDL domain learning algorithm from partial and noisy observations. *Proceedings of the 2022 Workshop on Knowledge Engineering for Planning and Scheduling*. Singapore.

Hogg, C., Muñoz-Avila, H., and Aha, D. W. 2008. HTN-Maker: Learning HTNs with minimal additional knowledge engineering required. *Proceedings of the Twenty-Third National Conference on Artificial Intelligence*. San Francisco, CA: AAAI Press.

Hogg, C., Kuter, U., and Muñoz-Avila, H. 2010. Learning methods to generate good plans: Integrating HTN learning and reinforcement learning. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pp. 1530–1535. Atlanta, GA: AAAI Press.

Iba, G. A. 1989. A heuristic approach to the discovery of macro-operators. *Machine Learning* **3**, 285–317.

Ilghami, O., Nau, D. S., Muñoz-Avila, H., and Aha, D. W. 2002. CaMeL: Learning method preconditions for HTN planning. *Proceedings of the Sixth International Conference on AI Planning and Scheduling*, pp. 131–141. AAAI Press: Toulouse, France.

Kirk, J. R., and Laird, J. E. 2014. Interactive task learning for simple games. *Advances in Cognitive Systems*, *3*, 13–30.

Langley, P. 1996. *Elements of machine learning*. San Francisco, CA: Morgan Kaufmann.

Langley, P. 2023. Learning hierarchical problem networks for knowledge-based planning. *Proceedings of the Thirty-First International Conference on Inductive Logic Programming*, pp. 69–83. Windsor Great Park, UK: Springer.

Langley, P., Shrobe, H. E., and Katz, B. 2024. A cognitive task analysis of rapid procedure acquisition from instructional documents. *Advances in Cognitive Systems*, *10*.

Langley, P., and Simon, H. A. 1995. Applications of machine learning and rule induction. *Communications of the ACM*, *38*, November, 55–64.

Li, N., Cushing, W., Kambhampati, S., and Yoon, S. 2014. Learning probabilistic hierarchical task networks as probabilistic context-free grammars to capture user preferences. *ACM Transactions on Intelligent Systems and Technology*, *5*, 1–32.

Li, N., Stracuzzi, D. J., Langley, P., and Nejati, N. 2009. Learning hierarchical skills from problem solutions using means-ends analysis. *Proceedings of the Thirty-First Annual Meeting of the Cognitive Science Society*. Amsterdam.

Li, N., Stracuzzi, D. J., and Langley, P. 2012. Improving acquisition of teleoreactive logic programs through representation extension. *Advances in Cognitive Systems*, *1*, 109–126.

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. 1986. Explanation-based generalization: A unifying view. *Machine Learning*, *1*, 47–80.

Nau, D., Au, T-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. 2003. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, *20*, 379–404.

Patra, S., Mason, J., Kumar, A., Ghallab, M., Traverso, P., and Nau, D. 2020. Integrating acting, planning, and learning in hierarchical operational models. *Proceedings of the International Conference on Automated Planning and Scheduling*, pp. 478–487. Nancy, France: AAAI Press.

Reddy, C., and Tadepalli, P. 1997. Learning goal-decomposition rules using exercises. *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 278–286. Nashville, TN: Morgan Kaufmann.

Shavlik, J. W. 1990. Acquiring recursive and iterative concepts with explanation-based learning. *Machine Learning*, *5*, 39–70.

Shell, P., and Carbonell, J. G. 1989. Towards a general framework for composing disjunctive and iterative macro-operators. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 596–602. Detroit, MI: Morgan Kaufmann.

Sleeman, D., Langley, P., and Mitchell, T. 1982. Learning from solution paths: An approach to the credit assignment problem. *AI Magazine*, *3*, 48–52.

Waterman, D. A. 1986. *A guide to expert systems*. Reading, MA: Addison-Wesley.

Zhuo, H. H., Hu, D. H., Hogg, C., Yang, Q., and Muñoz-Avila, H. 2009. Learning HTN method preconditions and action models from partial observations. *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pp. 1804–1809. Pasadena: Morgan Kaufmann.