From PCP to HTN Planning Through CFGs

Daniel Höller[1] and **Songtuan Lin**[3] and **Kutluhan Erol**[2] and **Pascal Bercher**[3]
[1] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[2] İzmir University of Economics, Turkey
[3] The Australian National University, Australia
hoeller@cs.uni-saarland.de, kutluhan.erol@ieu.edu.tr, {songtuan.lin, pascal.bercher}@anu.edu.au

Abstract

The International Planning Competition in 2020 was the first one for a long time to host tracks on HTN planning. The used benchmark set included a domain describing the undecidable *Post Correspondence Problem (PCP)*. In this paper we describe the two-step process applied to generate HTN problems based on PCP instances. It translates the PCP into a grammar intersection problem of two context-free languages, which is then encoded into an HTN problem.

Introduction

Hierarchical Task Network (HTN) planning is a widely-used planning approach with many practical applications (Bercher, Alford, and Höller 2019). It provides two means of modeling: a grammar-like decomposition structure as well as actions with preconditions and effects. The hierarchy makes it more expressive than e.g. classical planning, where only the latter is available. Erol, Hendler, and Nau (1996) showed that it enables the description of recursively enumerable, undecidable problems like the intersection problem of context-free languages. The International Planning Competition (IPC) in 2020 was the first for a long time to host tracks on HTN planning. To include provable hard domains, we included one describing the *Post Correspondence Problem (PCP)*, which is one of the standard decisions problems known to be undecidable (Hopcroft and Ullman 1979).

In this paper we give a short introduction of the applied encoding. We first translate PCP into the language intersection problem of the context-free languages. For the language intersection model, we can use the original encoding used to prove hardness of the plan existence problem in HTN planning (Erol, Hendler, and Nau 1996).

Background

We first shortly introduce PCP (for further details see e.g. (Hopcroft and Ullman 1979)). An instance of a PCP consists of two finite lists of finite strings (over an alphabet A with at least two symbols) with the same length: $P^1 = s^1_1, \ldots, s^1_n$ and $P^2 = s^2_1, \ldots, s^2_n$ ($n \in \mathbb{N}$). There is no restriction on the length of the individual s^1_i and s^2_i. A solution is a finite sequence of indices $j_1 \ldots j_k$ ($k \in \mathbb{N}$) with $1 \leq j_r \leq n$ for each $1 \leq r \leq k$ such that the compound string $s^1_{j_1} \ldots s^1_{j_k}$ is identical to $s^2_{j_1} \ldots s^2_{j_k}$.

A context-free grammar is a tuple $G = (\Gamma, \Sigma, P, S)$, where Γ is a finite set of non-terminal symbols, Σ is a finite set of terminal symbols, P is a finite set of production rules mapping a single non-terminal symbol to a finite sequence of terminal and non-terminal symbols. S is the start symbol. With the language $L(G)$ of a grammar we refer to the set of words of that grammar, i.e., all terminal symbol sequences that can be obtained by refining S via adhering the rules in P. Erol, Hendler, and Nau (1994; 1996) were the first to recognize the close relationship to HTN models, which they exploited for HTN's undecidability proof. Höller et al. (2014; 2016) have taken this further and studied the close relationship between various hierarchical (and non-hierarchical) planning problems and the Chomsky Hierarchy.

From PCP to HTN Planning Problems

We first translate a given PCP instance into a grammar intersection problem. For each list of strings, we construct a grammar such that words derived from that grammar begin with newly introduced letters representing the selected string indices from the respective P^i, followed by the actual concatenation of these strings. This is done for both P^is, when these languages have an intersection, this means that there is a list of indices leading to the same overall string, which solves our PCP problem.

For each P^i from the PCP, we construct a CFG $G^i = (\{S^i\}, A \cup L, P^i, S^i)$, where $L = \{l_1, \ldots, l_n\}$, n is the length of the string list and the production rules P^i are given by two rules $S^i \rightarrow l_j S^i s^i_j$ and $S^i \rightarrow l_j\, s^i_j$ for each s^i_j in the list, where j is the string's index, and l_j a terminal symbol.

Now that we have constructed the grammars G^1 and G^2 for P^1 and P^2, we check whether they may both produce the same string, relying on the encoding introduced by Erol, Hendler, and Nau (1996). The resulting problems include two tasks in the initial task network, which are not ordered with respect to each other. Each of them can be decomposed in sequences of actions representing the words of the language of one of the grammars. Preconditions and effects of the actions ensure that there is an applicable linearization if and only if the actions derived from the two grammars are applied in turns, and some letter from the second grammar follows the same letter from the first grammar. That way, the HTN problem has a solution if and only if the languages have a non-empty intersection.

We use the formalism by Geier and Bercher (2011). An HTN problem is a tuple $P = (F, N_p, N_c, M, \delta, tn_I, s_I, g)$. F is a set of propositional state features, N_p the set of primitive tasks, N_c the set of abstract (also *compound*) tasks, M the set of decomposition methods, δ a function mapping primitive tasks to their precondition and effects, tn_I the initial task network, s_I the initial state, and g the state-based goal condition.

Let $G^1 = (\{S^1\}, \Sigma, P^1, S^1)$ and $G^2 = (\{S^2\}, \Sigma, P^2, S^2)$ be the two grammars constructed in the previous step. The set of state features is defined as $F = \Sigma \cup \{turn_1, turn_2\}$, the *primitive tasks* as $N_p = \{p^i_a \,|\, a \in \Sigma, i \in \{1,2\}\}$. δ is defined as follows: if $i = 1$ then $\delta(p^i_a) = (\{turn_1\}, \{a, turn_2\}, \{turn_1\})$, otherwise, $\delta(p^i_a) = (\{turn_2, a\}, \{turn_1\}, \{turn_2, a\})$. The domain contains two compound tasks $N_c = \{S^1, S^2\}$. The set of methods M which decompose those two compound tasks is constructed according to the set of production rules $P^1 \cup P^2$. Let $p \in P^1 \cup P^2$ be an arbitrary production rule. If p is in the form $S^i \to lS^is$ where $i \in \{1,2\}$, $l \in L$, and $s = c_1 \dots c_k$ is a string with $k \in \mathbb{N}$ and $c_j \in A$ for $1 \leq j \leq k$, we construct a method $m = (S^i, (T, \prec, \alpha))$ where

$$\begin{aligned} T &= \{t_S, t_l, t_1, \dots, t_k\} \\ \prec &= \{(t_l, t_S), (t_S, t_1), \dots, (t_{k-1}, t_k)\} \\ \alpha &= \{(t_S, S^i), (t_l, p^i_l), (t_1, p^i_{c_1}), \dots, (t_k, p^i_{c_k})\} \end{aligned}$$

For production rule not containing S^i, a similar method is constructed not including S^i. Initial task network, initial state, and state-based goal condition are defined as follows:

$$\begin{aligned} tn_I &= (\{s^1, s^2\}, \emptyset, \{(s^1, S^1), (s^2, S^2)\}) \\ s_I &= \{turn_1\} \quad g = \{turn_1\} \end{aligned}$$

Example

Fig. 1 shows the definitions of the tasks `t1G1` and `t1G2`, which are two actions that can be executed after each other, since one corresponds to the "creation" of the symbol *l1* by the first grammar, whereas the other, by the second grammar, deletes that symbol. The example code is provided in HDDL (Höller et al. 2020). The used predicate names in our figure differ only slightly from the respective names in the actual problem files (we adapted it slightly to match our formal definitions from before).

Benchmark Collection

For the IPC we selected problem instances where a solution exists, and we varied degree of difficulty as measured by the solution length. Meanwhile we also support the automatic generation of problem instances based on an external PCP problem generator. That generator creates random PCP instances given a minimal solution length, which is obtained by solving the respective problem. The random generator is available in the IPC benchmark repository next to the instances used in the IPC.

```
(:action t1G1              (:action t1G2
  :parameters ()             :parameters ()
  :precondition              :precondition
    (and                       (and
      (turn1)                    (turn2)
    )                            (l1))
  :effect                    :effect
    (and                       (and
      (not (turn1))              (not (turn2))
      (turn2)                    (turn1)
      (l1)))                     (not (l1))))
```

Figure 1: Examples of primitive tasks.

References

Bercher, P.; Alford, R.; and Höller, D. 2019. A survey on hierarchical planning – One abstract idea, many concrete realizations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 6267–6275. IJCAI Organization.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. Semantics for hierarchical task-network planning. Technical Report CS-TR-3239, UMIACS-TR-94-31, ISR-TR-95-9, Inst. for Advanced Computer Studies, Inst. for Systems Research, Computer Science Department, University of Maryland.

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

Geier, T., and Bercher, P. 2011. On the decidability of HTN planning with task insertion. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 1955–1961. AAAI Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2014. Language classification of hierarchical planning problems. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*, 447–452. IOS Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2016. Assessing the expressivity of planning formalisms through the comparison to formal languages. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*, 158–165. AAAI Press.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An extension to PDDL for expressing hierarchical planning problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 9883–9891. AAAI Press.

Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.