Semantics and transformations of HTN models

David Kroupa

1. Introduction

- Classical planning is an approach of assembling *actions* into a valid *plan* that changes the current state of the world into the desired one.
- Hierarchical task network (HTN) is an extension to the classical planning based on task decompositions.
- HTN uses *decomposition methods* with specific decomposition constraints. Such constraints can define task-ordering (partial, total) and state-constraints.
- Example of *a plan* in classical planning:

$$\pi = (move\text{-}loc1, take\text{-}box, move\text{-}loc2, put\text{-}box).$$

- Example of *a decomposition method* in HTN planning:

$$T \to T_1, T_2\ [C],$$

with ordering-constraint $(T_1 \prec T_2) \in C$ and state-constraint $before(p, T_2) \in C$.

- Example of *an empty method* (task is decomposed into nothing):

$$T \to \varepsilon\ [C].$$

2. Goals

- Survey and compare various semantics describing hierarchical task networks (concerning issues with *empty methods*).
- Propose techniques for transformation between models. We want to find ways of modifying the hierarchical models without losing any properties of the models. For example, we might try to compile away some constraints and convert them into different ones.

3. Motivations

The theory of classical and hierarchical planning is closely related to the theory of Automata and Grammars. For this reason, we can utilize concepts, knowledge, and well-known structures from the theory of Automata and Grammars and apply them to the field of planning.

4. Semantics

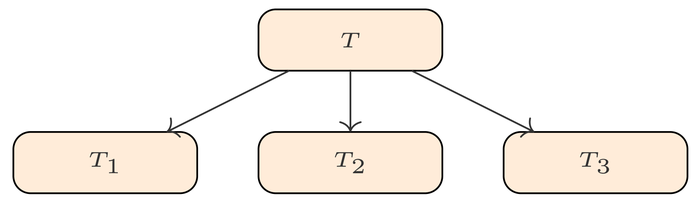
This thesis explores different types of HTN semantics, each with unique specifics and behavior.

- No Empty Methods Model,
- No-op Based Model,
- Constraint Graph Model,
- Index-Based Model,
- Increment-Base Model.

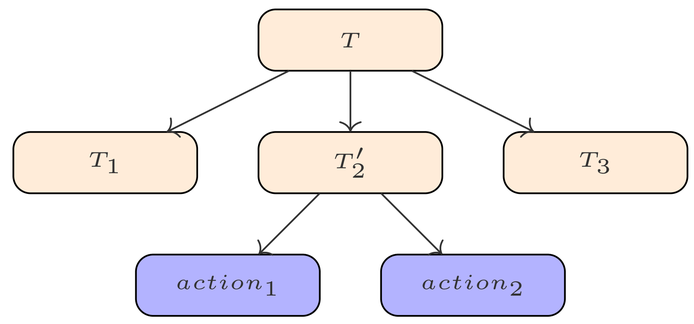
5. HTN Model Transformations

This thesis introduces many different transformations. The most interesting results are regarding *totally-ordered planning domains* which can utilize existing Context-free grammar algorithms. The main goal of each transformation is to preserve the set of *solutions* of the input model. There are two types of transformations:

- Normal forms.
- Adjusting/Deleting *state-constraints*.



$$T \to T_1, T_2, T_3\ [T_1 \prec T_2 \prec T_3, between(T_1, p, T_3)]$$



$$T \to T_1, T_2', T_3\ [T_1 \prec T_2' \prec T_3];$$
$$T_2' \to action_1, action_2\ [action_1 \prec action_2, before(p, action_1), before(p, action, 2)]$$

6. Implementation

The practical part of this thesis aims to deliver some of the *totally-ordered* transformations mentioned in the thesis. For this task, a user defines *a hierarchical planning domain* using a simple text format and selects the desired transformation.

The program is written in C# using the .NET8 framework and provides *totally-ordered* transformations e.g. compile away all *between-constraints* or *empty methods*.

7. Conclusions

To conclude, this project consists of three main parts: **HTN Semantics**, **Transformations**, and an attached command-line **program**. Some transformations might be useful to achieve prerequisites for particular HTN planners, such as compiling away unsupported features. Other transformations (HTN-ChNF, -GNF) might speed up the *planning* process and the reverse process of *plan verification*.

8. Additional Information

Supervisor: Prof. RNDr. Roman Barták, Ph.D.
Department: Department of Theoretical Computer Science and Mathematical Logic
Thesis: https://github.com/damodarak/bachelor-thesis