Compiling HTN Plan Verification Problems into HTN Planning Problems

Daniel Höller[1], Julia Wichlacz[1], Pascal Bercher[2], and Gregor Behnke[3]

[1]Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[2]The Australian National University, Canberra, Australia
[3]University of Freiburg, Freiburg, Germany

hoeller@cs.uni-saarland.de, wichlacz@cs.uni-saarland.de, pascal.bercher@anu.edu.au, behnkeg@cs.uni-freiburg.de

Motivation (1)

- In HTN planning, the decomposition steps applied to find a solution are usually not part of the solution

→ The solution is a sequence (partially ordered set) of actions

- This makes verifying whether a solution returned by a planner correct a hard task (up to NP-hard)
- When the steps (and certain mappings) are given, it becomes polynomial

→ E.g. for the IPC, systems needed to provide them

Definition

Given an HTN planning problem and a sequence of actions π, decide whether there is a solution for the HTN planning problem such that π is a valid linearization of it

Motivation (2)

- However, the decomposition steps are sometimes needed (e.g. for generating explanations)
- Models are transformed during preprocessing and grounding

→ Tracking decomposition steps is sometimes (technically) complicated

- Sometimes, decomposition steps are even not available, e.g.
  - ▶ when using post-processing
  - ▶ when using compilation-based planning systems (see TOAD system on the main conference)

Related Work

Two approaches in the literature:

- Translation to propositional logic (Behnke, Höller, and Biundo 2017)
- Based on parsing (see e.g. Barták, Maillard, and Cardoso 2018)

Contribution

Here: Based on compilation to HTN planning

- We compile an HTN plan verification problem to an HTN planning problem that has a solution if and only if the verification problem has one
- We use HTN planning systems to solve the new problems
- The used systems return the decomposition steps

→ Witness for correctness (what about plans that could not be verified?)

Translation (1)

- We start planning with original initial task network
- We transform the model such that only the sequence to verify is applicable

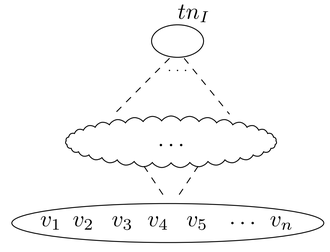

Figure 1: *Overview of the overall approach*

Translation (2)

- Copy actions contained in plan: $a \to a', a''$
- a' gets new preconditions and effects, it can only be placed at the position where it is in the plan
- Actions also hold their original preconditions and effects
- All actions have to be in the plan (enforced by a new state-based goal)
- Other actions are made inapplicable

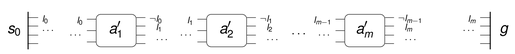

Figure 2: *Schema of action transformation*

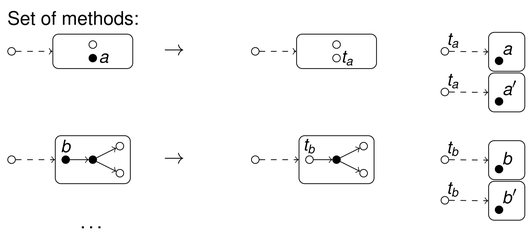

Figure 3: *Example for method transformation (plan to verify: ab)*

(Preliminary) Evaluation

- We combined our transformation with planning systems from the PANDA planning framework:
  - ▶ One based on heuristic progression search
  - ▶ One based on a translation to propositional logic
- We created a new benchmark set based on the instances, planners, and solutions from the 2020 IPC

→ Contains only very few "unsol" instances

- We compared our system with two systems from the literature

→ Some systems missing yet

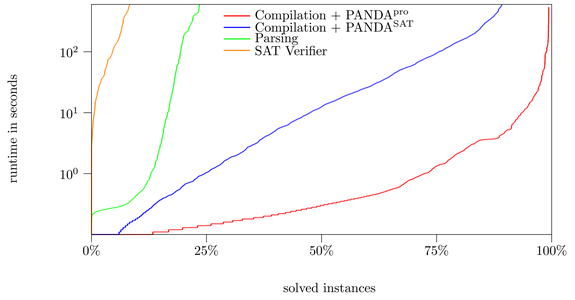

Figure 4: *Percentage of solved instances over time (TO setting)*

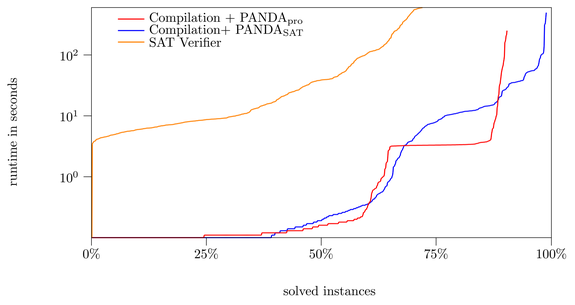

Figure 5: *Percentage of solved instances over time (PO setting)*

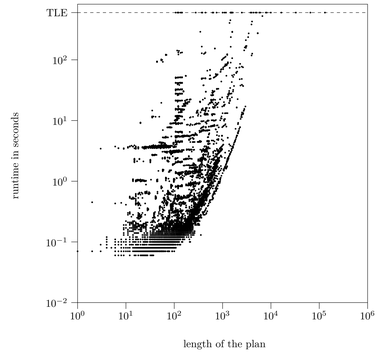

Figure 6: *Verification time vs. plan length*

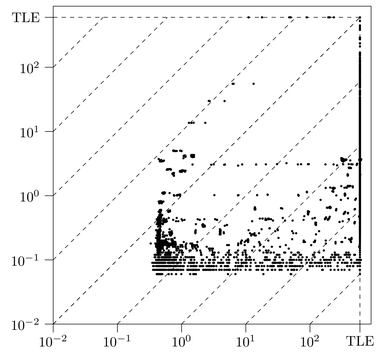

Figure 7: *Runtime of our system (y axis) on the TO set compared to the parsing-based approach on the x axis (log scale)*

Discussion

- We introduced a compilation-based approach to plan verification
- It empirically outperforms the approaches from the literature
- Technically, the transformation is very similar to one introduced for plan and goal recognition as planning
- Used planners output the decomposition steps

→ Witness for (positive) result

- However, our evaluation is yet preliminary
  - ▶ Not all systems from the literature have been included
  - ▶ Only very few instances of non-plans

UNIVERSITÄT DES SAARLANDES