On Total-Order HTN Plan Verification with Method Preconditions – An Extension of the CYK Parsing Algorithm

Songtuan Lin[1] **Gregor Behnke**[2] **Simona Ondrčková**[3] **Roman Barták**[3] **Pascal Bercher**[1]

[1]School of Computing, The Australian National University, Canberra, Australia

[2]ILLC, University of Amsterdam, Amsterdam, The Netherlands

[3]Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

[1]{songtuan.lin, pascal.bercher}@anu.edu.au, [2]g.behnke@uva.nl, [3]{ondrckova, bartak}@ktiml.mff.cuni.cz

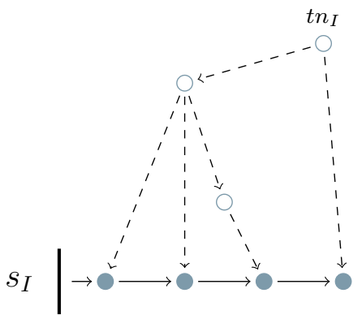
Introduction

Objective: Our objective is to develop a novel Total Order HTN (TOHTN) plan verification approach by extending the CYK-parsing Algorithm which can deal with method preconditions.

Motivations: 1) TOHTN plan verification can be used in many applications, e.g., in verifying plans in IPC on HTN Planning, and 2) the current SOTA parsing-based TOHTN plan verification approach relies on a brute force search.

TOHTN Planning

A TOHTN planning problem $\mathcal{P} = ((\mathcal{F}, \mathcal{A}, \mathcal{C}, \delta, \mathcal{M}), c_I, s_I)$:



- $\mathcal{F}$: A set of propositions
- $\mathcal{A}$: A set of primitive tasks
- $\mathcal{C}$: A set of compound tasks
- $\delta : \mathcal{A} \to 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$
- $\mathcal{M} \subseteq 2^{\mathcal{F}} \times \mathcal{C} \times (\mathcal{A} \cup \mathcal{C})^*$: A set of methods
- $c_I \in \mathcal{C}$: The initial task
- $s_I \in 2^{\mathcal{F}}$: The initial state

Solution: A solution to a TOHTN planning problem is an action sequence into which is decomposed from the initial task by methods, it is executable in the initial state, and the precondition of every used method is satisfied.

TOHTN Planning Problems and CFGs

The basis for using the CYK algorithm in TOHTN plan verification is the connection between TOHTN planning prblems and context-free grammars (CFGs):

- A primitive task is a terminal symbol
- A compound task is a non-terminal symbol
- A method **without** preconditions is a production rule

Idea: We need to check whether the precondition of each method is satisfied when constructing the CYK table.

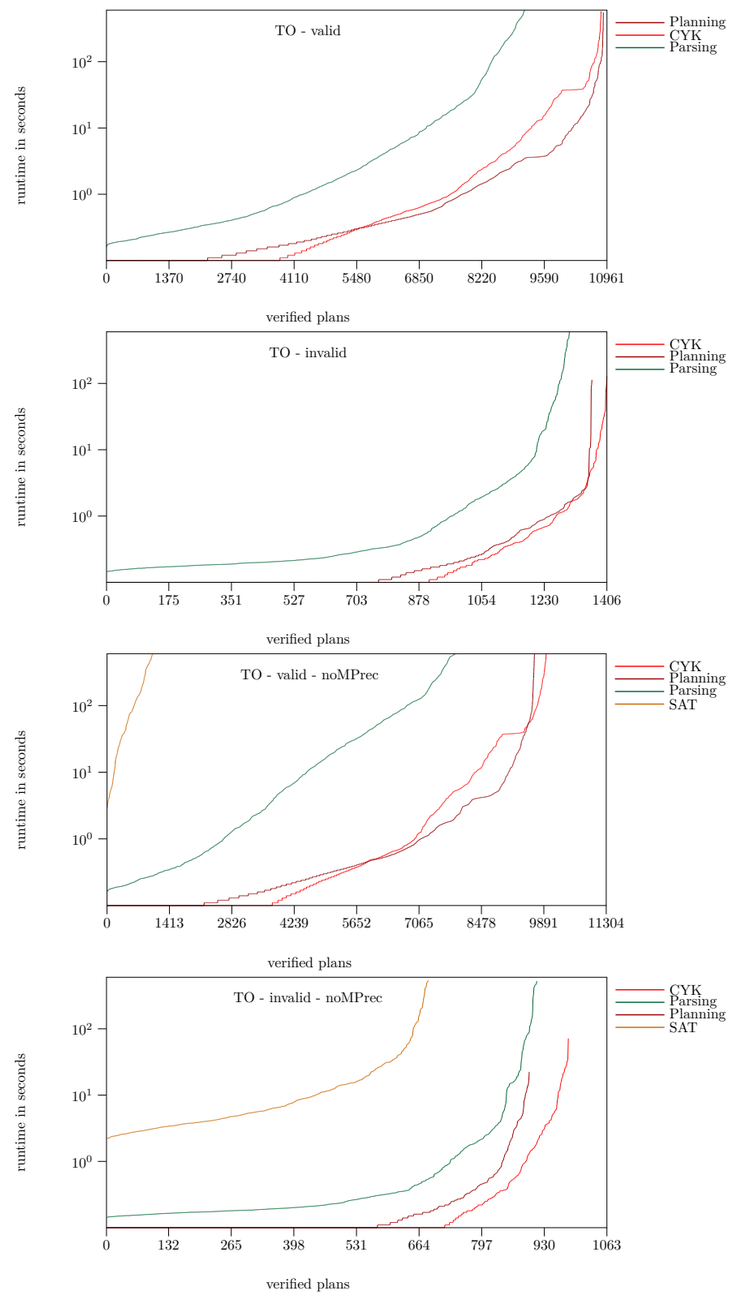
TOHTN Plan Verification Algorithm

Input: A plan $\pi = \langle p_1 \cdots p_n \rangle$
A planning problem $\mathcal{P}$ in 2RF
Output: True or false depending on whether π is a solution to $\mathcal{P}$

1: ▷ Let $\langle s_0 \cdots s_n \rangle$ be the state sequence *s.t.* $s_0 = s_I$, and $s_{i-1} \to_{p_i} s_i$ for each $i \in \{1 \cdots n\}$
2: **for** $i \leftarrow n$ to 1
3: $\quad A[i,i] = \{c \mid c \to \langle p_i \rangle\} \cup \{p_i\}$
4: $\quad$ **for** $j \leftarrow i$ to n
5: $\quad\quad$ **for** $k \leftarrow i$ to $j-1$
6: $\quad\quad\quad$ **for** $m \in \left\{ m \;\middle|\; \begin{array}{l} m = (prec(m), c, tn), \\ tn = \langle c'_1\, c'_2 \rangle, c'_1 \in A[i,k], \\ c'_2 \in A[k+1,j] \end{array} \right\}$
7: $\quad\quad\quad\quad$ ▷ Checking the method precondition
8: $\quad\quad\quad\quad$ **if** $prec(m) \subseteq s_{i-1}$
9: $\quad\quad\quad\quad\quad A[i,j] \leftarrow A[i,j] \cup \{c\}$
10: $\quad\quad$ ▷ Finding the unit productions
11: $\quad\quad$ **for** $\overline{m} \in \left\{ \overline{m} \;\middle|\; \begin{array}{l} c' \to^*_{\overline{m}} \langle c \rangle, c' \in N_c, \\ c \in A[i,j] \end{array} \right\}$
12: $\quad\quad\quad$ **if** $prec(m) \subseteq s_{i-1}$ for each m in $\overline{m}$
13: $\quad\quad\quad\quad A[i,j] \leftarrow A[i,j] \cup \{c'\}$
14: **if** $c_I \in A[1,n]$ **return** true
15: **else return** false

Input: An input TOHTN planning problem is in 2-regularation form (2RF) instead of Chomsky Normal Form (CNF) for the purpose of keeping the size of the planning problem small. In 2RF, every method contains **at most** two subtasks.
Line 8–9: Checking whether a method precondition is satisfied.
Line 11–13: Finding all unit productions leading to $c \in A[i,j]$.

Runtimes



Empirical Evaluation

Benchmark	Instances	Parsing-based	Planning-based	SAT-based	CYK-based (Ours)
to-val	10961	9158 (83.55%)	**10925** (**99.67%**)	Not support	10917 (99.60%)
to-inval	1406	1301 (92.53%)	1364 (97.01%)	Not support	**1406** (**100.00%**)
to-val-no-mprec	11304	7889 (69.79%)	9679 (85.62%)	1036 (9.16%)	**9946** (**87.99%**)
to-inval-no-mprec	1063	915 (86.08%)	898 (84.48%)	684 (62.01%)	**981** (**88.94%**)

The benchmark sets are from the IPC 2020 on HTN Planning. We compared our approach with the parsing-based plan verification approach, the planning-based approach, and the SAT-based approach. We ran the experiments on the benchmark sets which respectively consist of planning problems with and without method preconditions.