Tight Bounds for Hybrid Planning

Pascal Bercher[1], Songtuan Lin[1], and Ron Alford[2]

[1] The Australian National University
[2] The MITRE Corporation

Australian National University

MITRE

Introduction

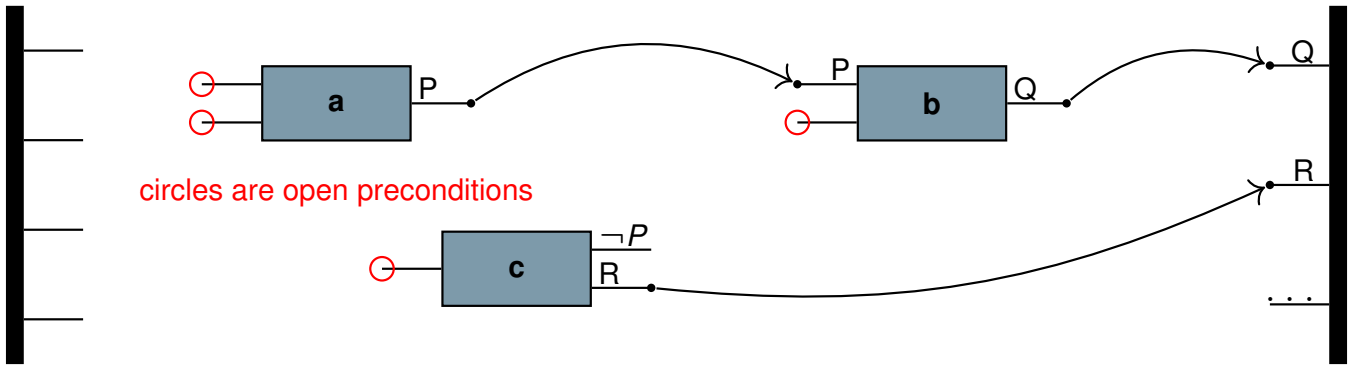
Overview

In this paper we study:

- A hierarchical approach to planning, more specifically:
  - An extension of Hierarchical Task Network (HTN) planning
  - Hybrid planning: HTN planning with causal links
- Specifically, we investigate the computational complexity of
  - The *plan existence problem* = Does there exist a solution?
  - The *plan verification problem* = Does plan P solve the problem?

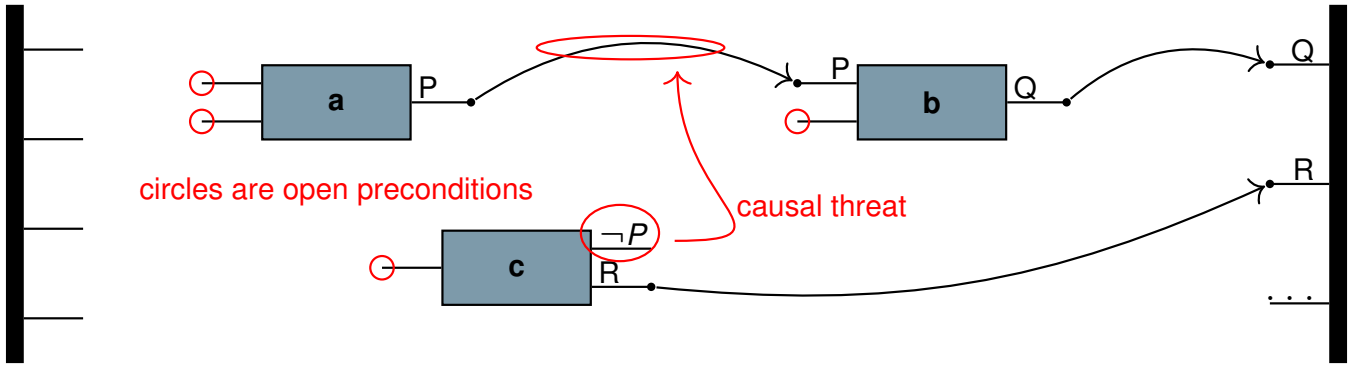
Hybrid Planning

Introduction to Actions and Causal Links



- Partial Order Causal Link (POCL) plans are partially ordered
- Causal dependencies are represented using causal links
- For solution POCL plans,
  - all preconditions must be supported by a causal link, and

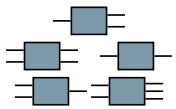
Introduction to Actions and Causal Links



- Partial Order Causal Link (POCL) plans are partially ordered
- Causal dependencies are represented using causal links
- For solution POCL plans,
  - all preconditions must be supported by a causal link, and
  - all causal threats must be resolved

Introduction to HTN Planning

primitive tasks



compound tasks

$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- F a set of facts
- P a set of primitive task names
- $\delta : P \rightarrow (2^F)^3$ the task name mapping
- C a set of compound task names

Introduction to HTN Planning

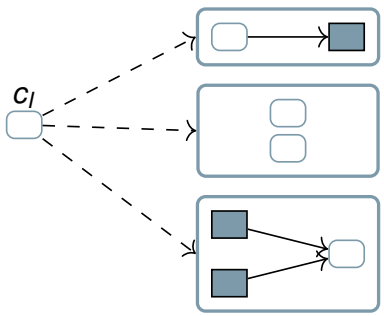
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- F a set of facts
- P a set of primitive task names
- $\delta : P \rightarrow (2^F)^3$ the task name mapping
- C a set of compound task names
- $c_I \in C$ the initial task

A solution task network *tn* must:

- be a refinement of c_I,

Introduction to HTN Planning



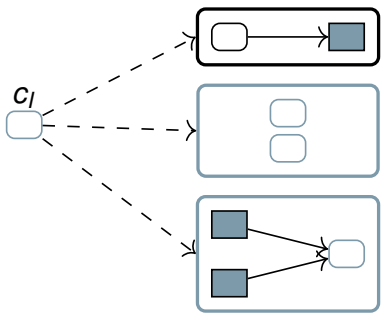
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- F a set of facts
- P a set of primitive task names
- $\delta : P \rightarrow (2^F)^3$ the task name mapping
- C a set of compound task names
- $c_I \in C$ the initial task
- $M \subseteq C \times 2^{TN}$ the methods

A solution task network *tn* must:

- be a refinement of c_I,
- only contain primitive tasks, and

Introduction to HTN Planning



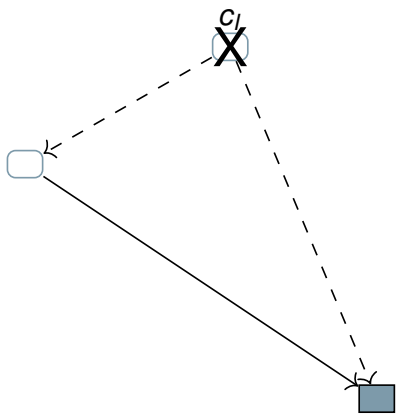
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- F a set of facts
- P a set of primitive task names
- $\delta : P \rightarrow (2^F)^3$ the task name mapping
- C a set of compound task names
- $c_I \in C$ the initial task
- $M \subseteq C \times 2^{TN}$ the methods

A solution task network *tn* must:

- be a refinement of c_I,
- only contain primitive tasks, and

Introduction to HTN Planning



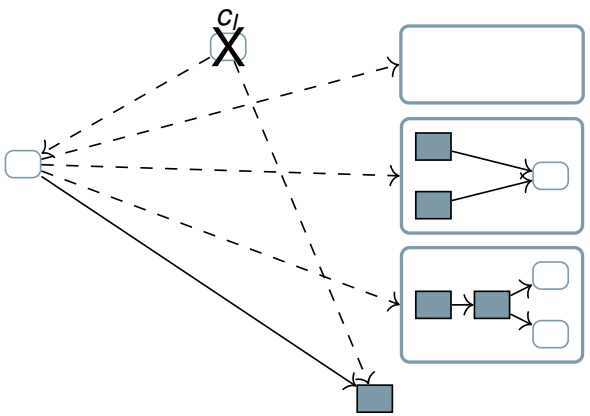
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- F a set of facts
- P a set of primitive task names
- $\delta : P \rightarrow (2^F)^3$ the task name mapping
- C a set of compound task names
- $c_I \in C$ the initial task
- $M \subseteq C \times 2^{TN}$ the methods

A solution task network *tn* must:

- be a refinement of c_I,
- only contain primitive tasks, and

Introduction to HTN Planning



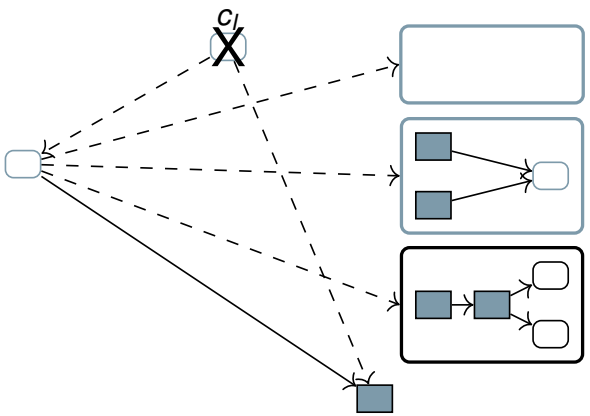
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- F a set of facts
- P a set of primitive task names
- $\delta : P \rightarrow (2^F)^3$ the task name mapping
- C a set of compound task names
- $c_I \in C$ the initial task
- $M \subseteq C \times 2^{TN}$ the methods

A solution task network *tn* must:

- be a refinement of c_I,
- only contain primitive tasks, and

Introduction to HTN Planning



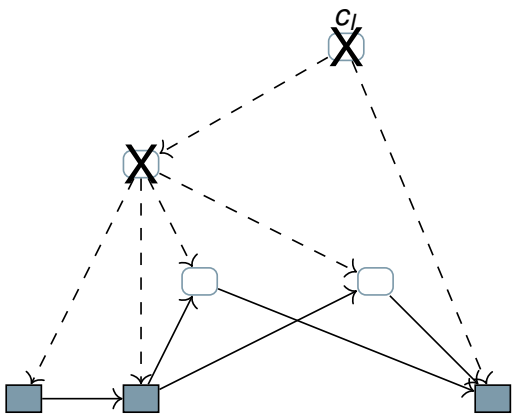
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- F a set of facts
- P a set of primitive task names
- $\delta : P \rightarrow (2^F)^3$ the task name mapping
- C a set of compound task names
- $c_I \in C$ the initial task
- $M \subseteq C \times 2^{TN}$ the methods

A solution task network *tn* must:

- be a refinement of c_I,
- only contain primitive tasks, and

Introduction to HTN Planning



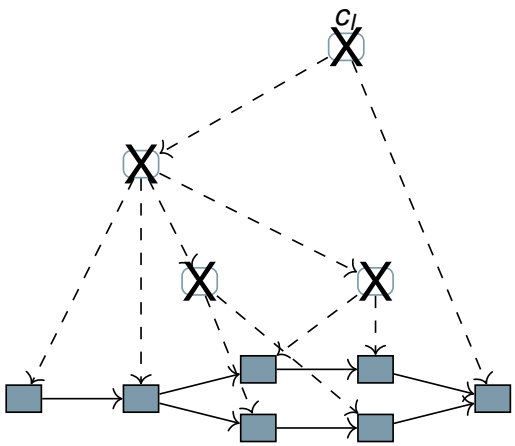
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- F a set of facts
- P a set of primitive task names
- $\delta : P \rightarrow (2^F)^3$ the task name mapping
- C a set of compound task names
- $c_I \in C$ the initial task
- $M \subseteq C \times 2^{TN}$ the methods

A solution task network *tn* must:

- be a refinement of c_I,
- only contain primitive tasks, and

Introduction to HTN Planning



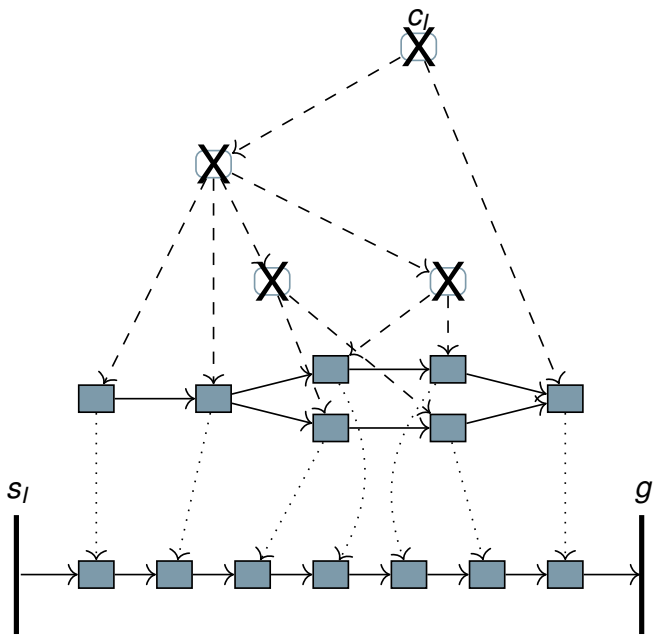
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- F a set of facts
- P a set of primitive task names
- $\delta : P \rightarrow (2^F)^3$ the task name mapping
- C a set of compound task names
- $c_I \in C$ the initial task
- $M \subseteq C \times 2^{TN}$ the methods

A solution task network *tn* must:

- be a refinement of c_I,
- only contain primitive tasks, and

Introduction to HTN Planning



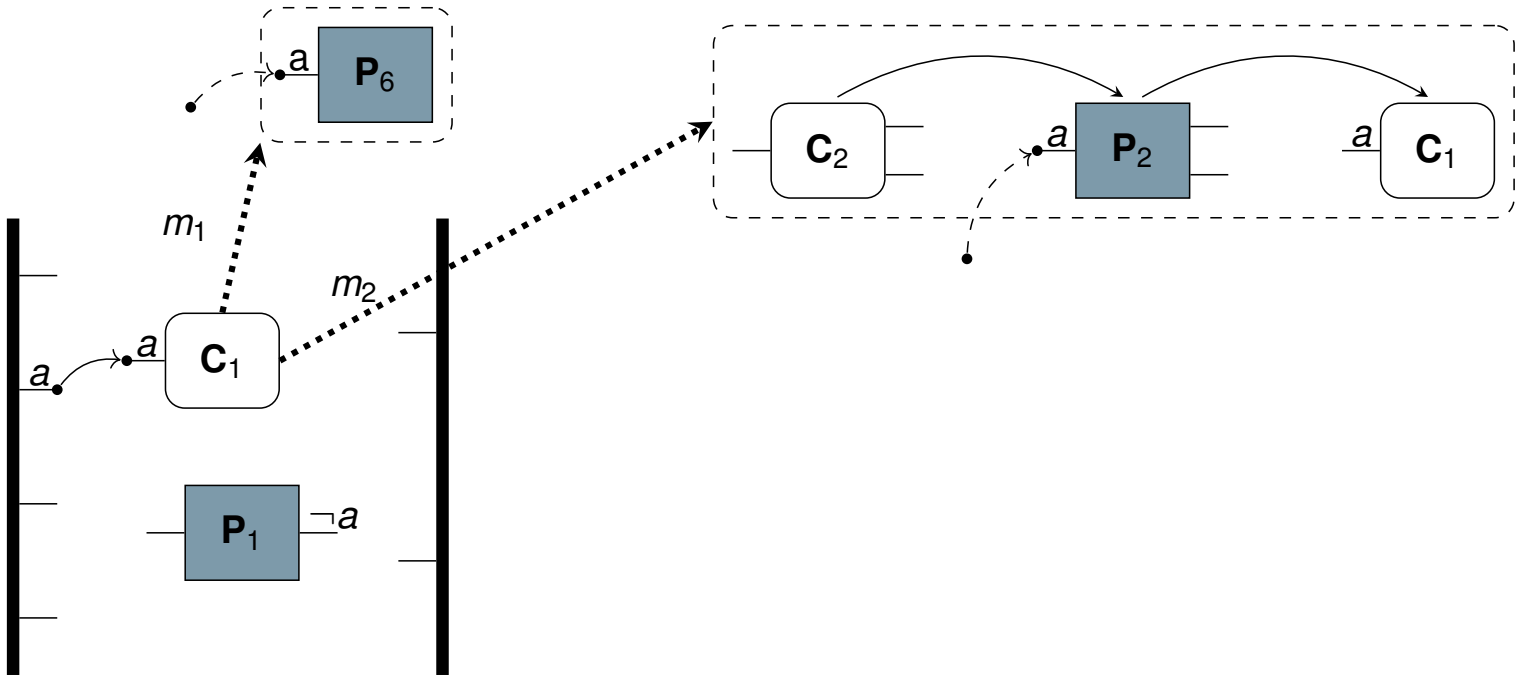
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- F a set of facts
- P a set of primitive task names
- $\delta : P \to (2^F)^3$ the task name mapping
- C a set of compound task names
- $c_I \in C$ the initial task
- $M \subseteq C \times 2^{TN}$ the methods
- $s_I \in 2^F$ the initial state
- $g \subseteq F$ the (optional) goal description

A solution task network *tn* must:

- be a refinement of c_I,
- only contain primitive tasks, and
- have an executable linearization.

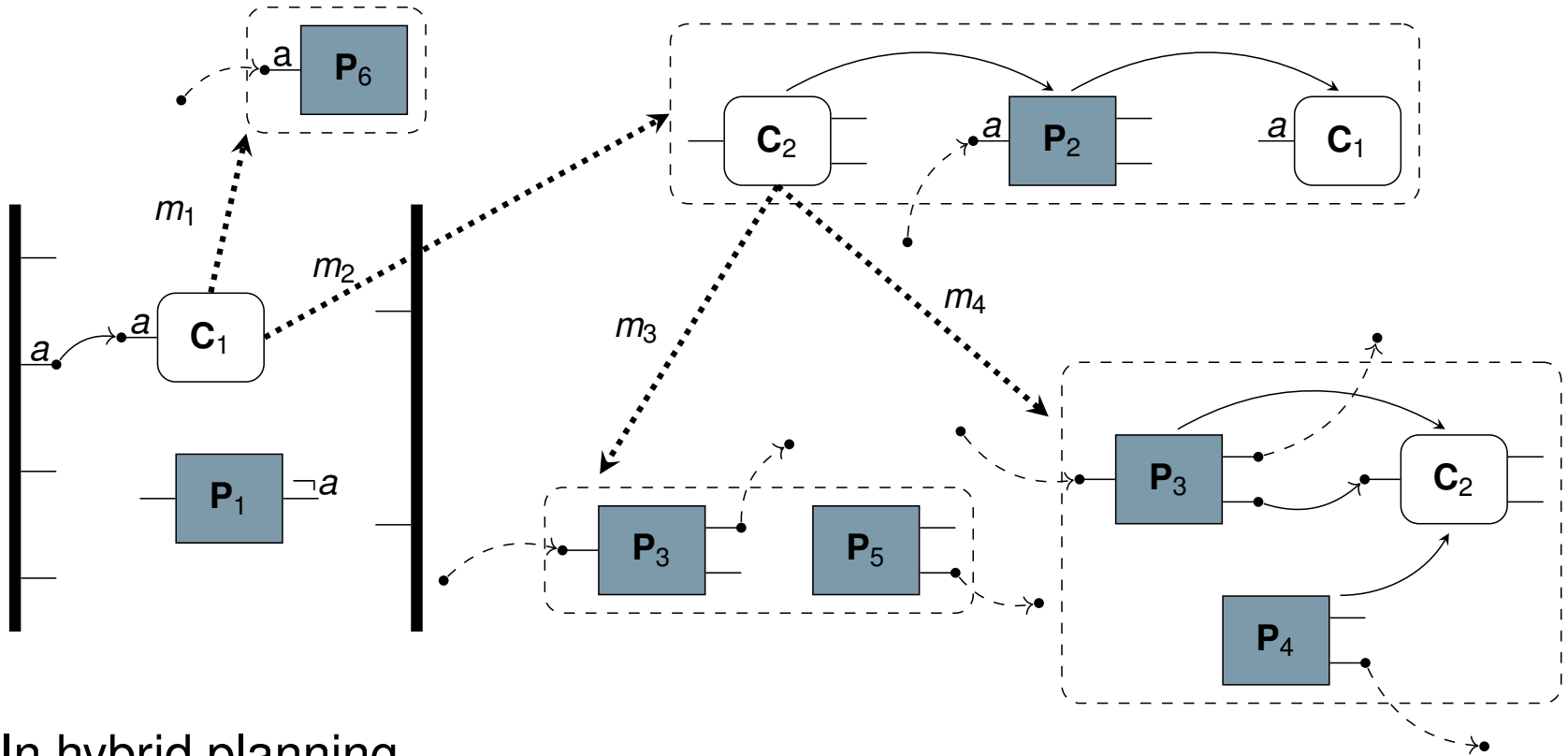
Hybrid Planning Problem Example



In hybrid planning,

- compound tasks also have preconditions and effects,
- decomposition methods specify to which subtasks causal links get inherited down

Hybrid Planning Problem Example



In hybrid planning,

- compound tasks also have preconditions and effects,
- decomposition methods specify to which subtasks causal links get inherited down

Complexity Results

Main messages:

1. Causal links don't increase computational complexity
   - We took all membership proofs from HTN planning and showed how to adapt them to still work despite the inheritance of causal links (no entirely new proofs were required)
   - Various complexity classes are covered: (N)P, (N)EXPTIME, PSPACE, EXPSPACE – all the same as in HTN planning
2. Plan verification is already NP-complete in the absence of a task hierarchy (i.e., if the initial plan is already primitive)

For the actual proofs check out our paper or see you at the poster

Thank you for listening!