Hierarchical Planning and Reasoning about Partially Ordered Plans – From Theory to Practice

Pascal Bercher

College of Engineering and Computer Science
School of Computing
the Australian National University (ANU)

21 July 2022

Australian
National
University

Introduction

About the Speaker

Who am I? Why am I here?

- Did my PhD in Ulm, at the Institute of Artificial Intelligence
  - Under the supervision of Prof. Biundo
  - In the context of Hierarchical Planning, applied to Companion Technology (cognitive systems), specifically to provide planning-based user assistance
- Worked closely with Prof. Glimm (13 publications! – and counting)
- Bio:
  - 2002–2009: Studied Computer Science in Freiburg im Breisgau
  - 2009–2017: Doctoral Studies in Ulm
  - 2017–2019: Post-Doc – still in Ulm
  - 2019–2021: Lecturer (Assistant/Junior Professor) at the ANU
  - 2022–. . . : Senior Lecturer (Associate/W2 Professor) at the ANU

Talk/Expertise Overview: Main Research Interests

Problem classes:

- (Non-hierarchical) Partial Order Causal Link (POCL) Planning
- Hierarchical Task Network (HTN) Planning

Talk/Expertise Overview: Main Research Interests

Problem classes:

- (Non-hierarchical) Partial Order Causal Link (POCL) Planning
- Hierarchical Task Network (HTN) Planning

Research areas:

- Complexity investigations
  (of various interesting problems)
- Heuristic search
  (algorithms, heuristics, etc.)
- Practical applications
  (or questions related to it)

Talk/Expertise Overview: Main Research Interests

Problem classes:

- (Non-hierarchical) Partial Order Causal Link (POCL) Planning
- Hierarchical Task Network (HTN) Planning

Research areas:

- Complexity investigations
  (of various interesting problems)
- Heuristic search
  (algorithms, heuristics, etc.)
- Practical applications
  (or questions related to it)

} from theory to practice!

Planning in a Nutshell: Planning in a Nutshell

We consider *classical planning problems*, which consist of:

- An initial state s_I – all "world properties" true in the beginning.
- A set of available actions – how world states can be changed.
- A goal description g – all properties we'd like to hold.

What do we want?

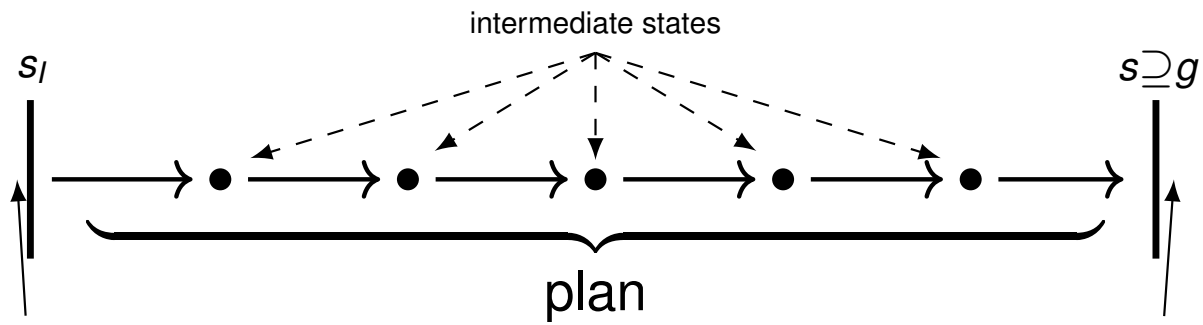
Planning in a Nutshell: Planning in a Nutshell

We consider *classical planning problems*, which consist of:

- An initial state s_I – all "world properties" true in the beginning.
- A set of available actions – how world states can be changed.
- A goal description g – all properties we'd like to hold.

What do we want?

→ Find a *plan* that transforms s_I into g.



Planning in a Nutshell: Example: Home Theater Assembly Assistant

Sink devices:

- Television (requires video)
- Amplifier (requires audio)

Source devices:

- Blu-ray player
- Satellite receiver
  (both produce audio & video)

Planning in a Nutshell: Definitions, Examples

- Planning problems are usually defined in terms of a description language based on a first-order predicate logic.
  - Predicates, like $HasPort(?device, ?port)$, express relationships between variables representing objects.
  - Constants, like *AMPLIFIER* and *CABLE_HDMI*, represent objects.
- States are sets of (ground) propositions, e.g.,
  $s \supseteq \{HasPort(AMPLIFIER, HDMI),$
  $HasPort(AMPLIFIER, CINCH),$
  $HasPort(CABLE\_HDMI, HDMI),$
  $IsConnected(AMPLIFIER, CABLE\_HDMI, HDMI)\}$

(connected to each other)

Planning in a Nutshell: Definitions, Examples

- Actions are defined by preconditions and effects, e.g.,

  plugIn(?*cable*, ?*device*, ?*port***)**

  precondition: $HasPort(?device, ?port) \wedge$
  $HasPort(?cable, ?port) \wedge$
  $\nexists ?cable' : IsConnected(?device, ?cable', ?port)$
  $\nexists ?device' : IsConnected(?device', ?cable, ?port)$

  effect: $IsConnected(?device, ?cable, ?port)$

  (Signal flow not shown for the sake of simplicity)

Planning in a Nutshell: Planning Problem Definition in the Home Theater Domain

Initial state:

- HasPort(. . . , . . .) // which device has which ports?
- IsConnected(. . . , . . . , . . .) // how are the connections initially?
- HasSignal(. . . , . . . , . . .) // which device has which signals?

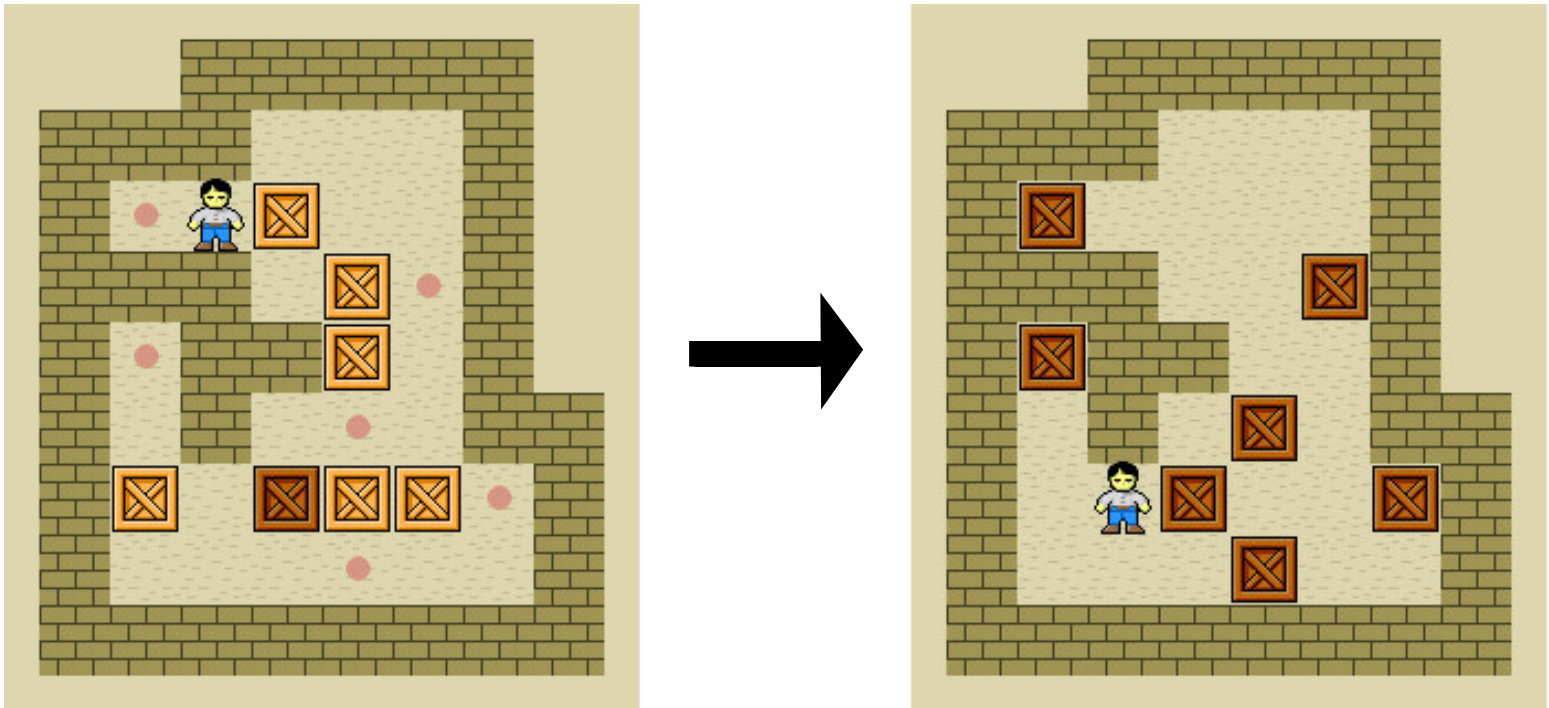
Action portfolio:

- ***plugIn(****?cable*, ?*device*, ?*port****)*** // plugging in a cable
- ***plugOut(****?cable*, ?*device*, ?*port****)*** // in case plugging out is allowed

Goal description:

- HasSignal(. . . , . . . , . . .) // e.g., HasSignal(TV, VIDEO, BR) denoting
- . . . that the TV has the video signal of the blu-ray player

Examples of Planning Problems: Computer Games Sokoban



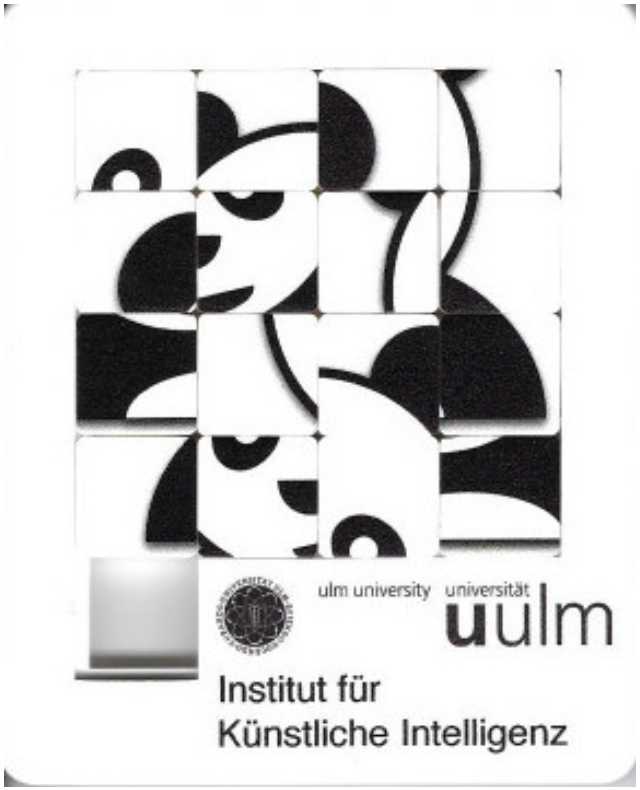
Title: A Sokoban puzzle and its solution

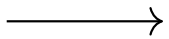
Source: `https://en.wikipedia.org/wiki/Sokoban`

Puzzle Author: Carlos Montiers Aguilera

Graphics Author: Borgar Þorsteinsson and Pascal Bercher.
The graphic has been modified multiple times (e.g., conversion from animated gif into this one.)

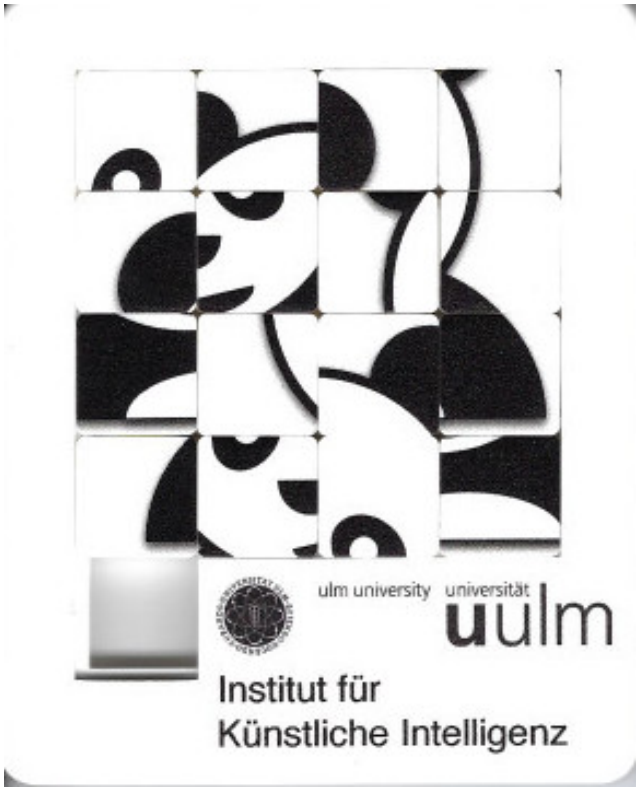
Examples of Planning Problems: Puzzle Games (here: Sliding Tile Puzzle)





Initial State

Examples of Planning Problems: Puzzle Games (here: Sliding Tile Puzzle)

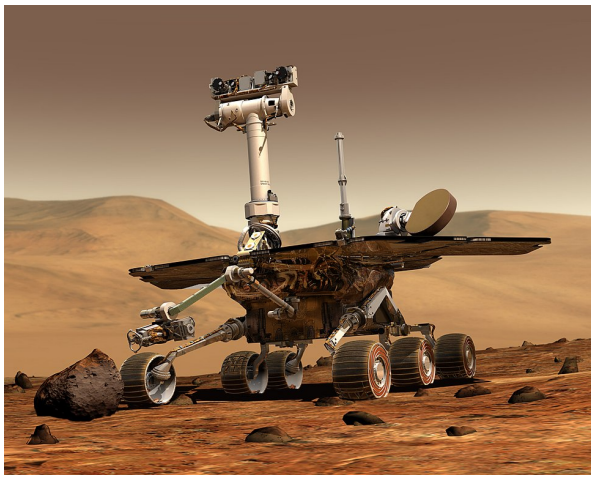


Initial State

Goal State

Examples of Planning Problems: Mars Rovers

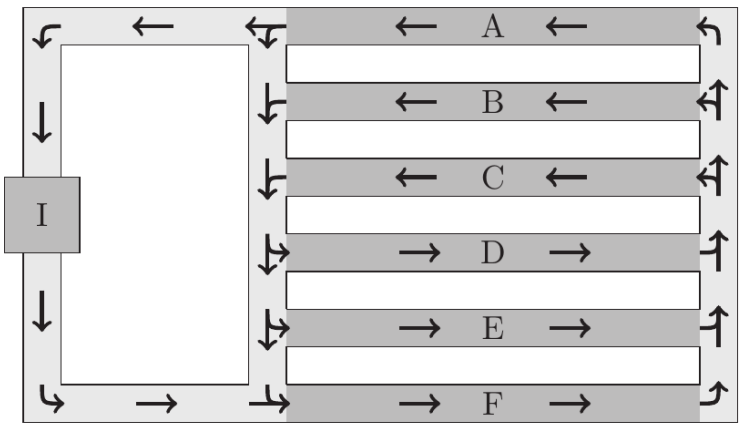




- MAPGEN (Mixed Initiative Activity Planning Generator) is a ground-based decision support system for Mars Exploration Rover mission operations and science teams that begins to give content to the notion of autonomous planetary exploration.
- The paradigm is to enable the person using the software to critique a plan that the system automatically produces and ensure that resulting plans are viable within mission and flight rules.

from *https://www.nasa.gov/*

Examples of Planning Problems: Automated Factories (here: a Greenhouse)



Source: `https://www.lemnatec.com/`

Copyright: With kind permission from *LemnaTec GmbH*

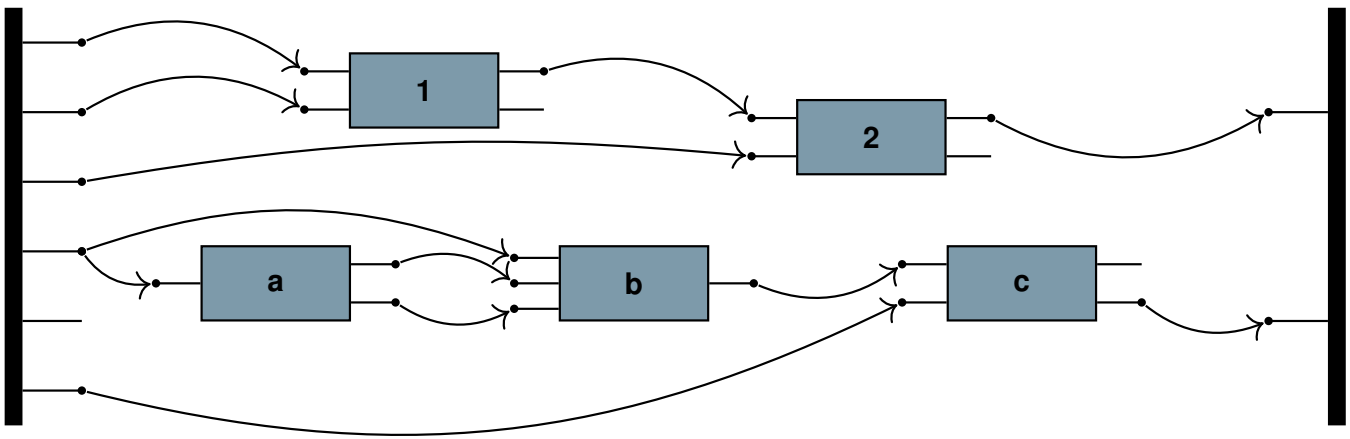
Further reading:

- M. Helmert and H. Lasinger. "The Scanalyzer Domain: Greenhouse Logistics as a Planning Problem". In: *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS 2010)*
- The IPC Scanalyzer Domain in PDDL (see paper above).

POP and POCL Planning

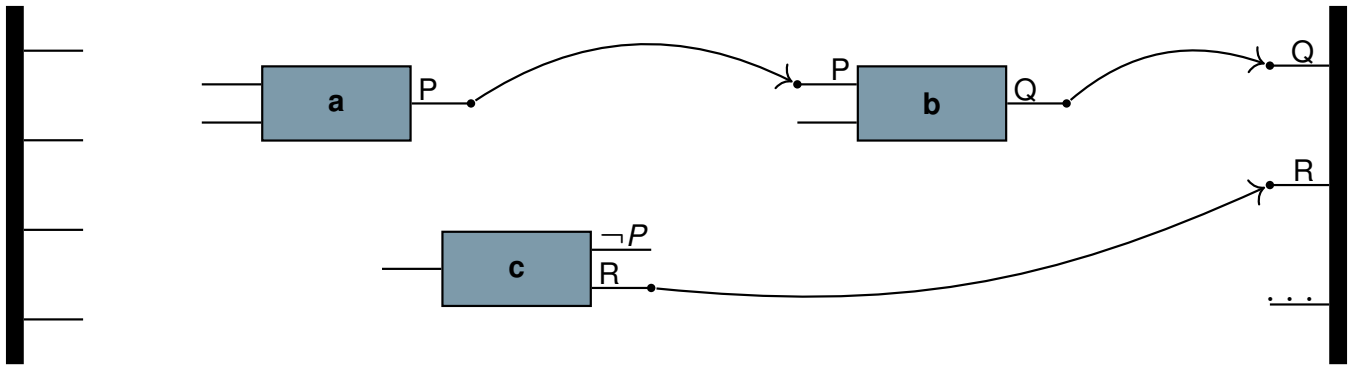
Introduction

- Most planning systems generate *action sequences* as solutions!
- But plans may be just *partially ordered* in general.



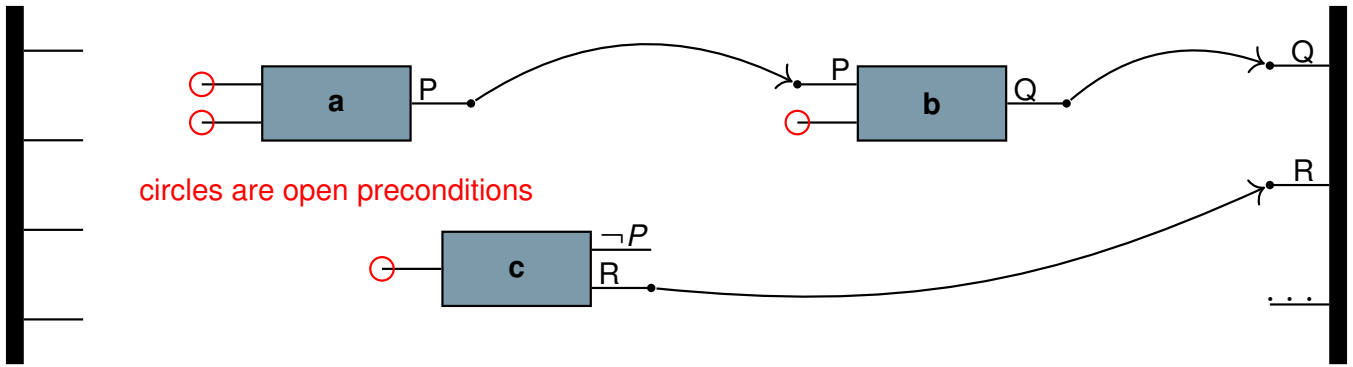
- Partial Order Causal Link (POCL) plans are partially ordered
- POCL plans are interesting for many tasks like plan optimization

(A Glance at the) Planning Procedure



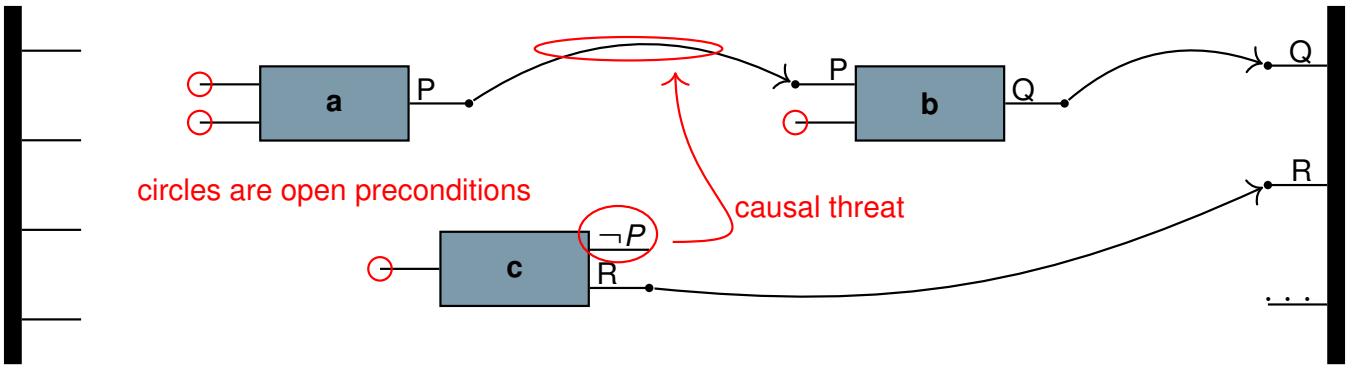
- Partial Order Causal Link (POCL) plans are partially ordered
- Causal dependencies are represented using causal links

(A Glance at the) Planning Procedure



- 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, and

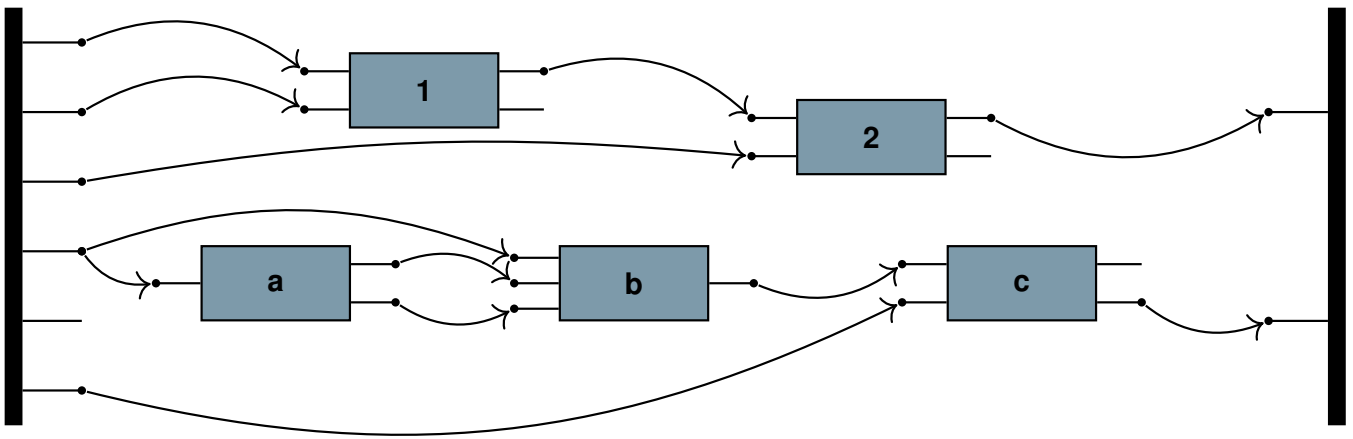
(A Glance at the) Planning Procedure



- 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, and
  - all causal threats must be resolved

Solutions / Summary

- Once all preconditions have a causal link and there are no causal threats, we have a solution:



- In a solution, every linearization that's compatible with the given ordering constraints is an executable action sequence.
- Therefore, POCL plans can represent up to an exponential number of sequential solutions!

So What?

Why did I introduce this?

- Because I'm interested in it! :) I'm an expert for POCL plans, notably complexity analyses. For example I studied, among others, how hard it is to decide whether:
  - a solution can be optimized by removing an action?
  - the makespan (parallel execution time) can be optimized by removing ordering constraints?
  - a given partial plan can be turned into a solution by ignoring certain information (like causal links)?

  (most problems are NP-complete)

So What?

Why did I introduce this?

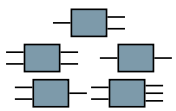
- Because I'm interested in it! :) I'm an expert for POCL plans, notably complexity analyses. For example I studied, among others, how hard it is to decide whether:
  - a solution can be optimized by removing an action?
  - the makespan (parallel execution time) can be optimized by removing ordering constraints?
  - a given partial plan can be turned into a solution by ignoring certain information (like causal links)?

  (most problems are NP-complete)
- Because it was used in the assembly assistant, notably for plan explanations (explaining why a specific action is in a plan)

HTN Planning

Introduction: 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: Introduction to HTN Planning

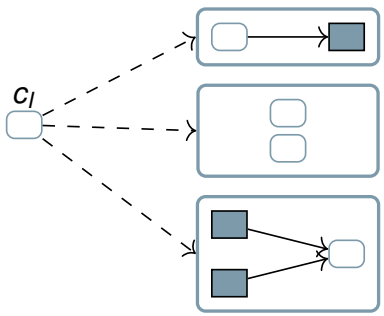
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- 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: Introduction to HTN Planning



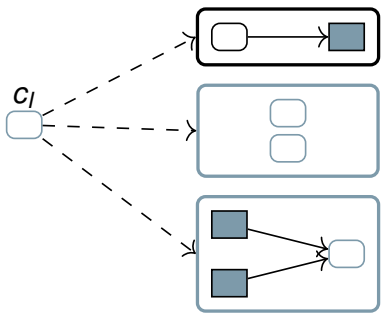
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- 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: Introduction to HTN Planning



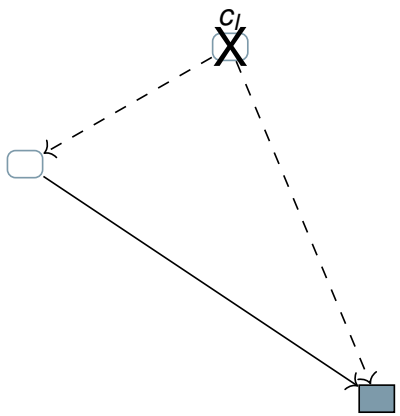
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- 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: Introduction to HTN Planning



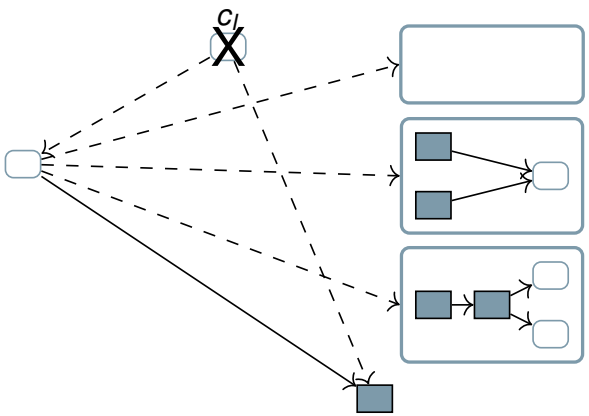
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- 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: Introduction to HTN Planning



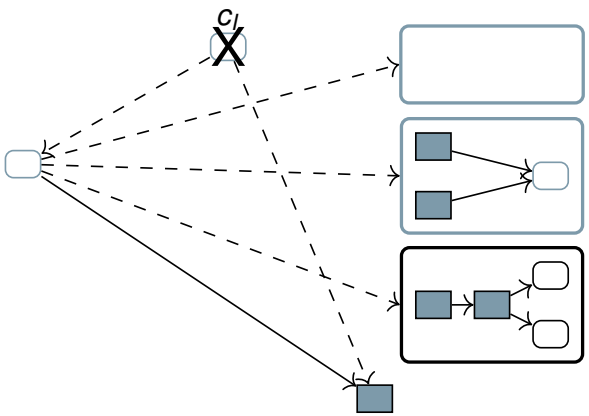
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- 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: Introduction to HTN Planning



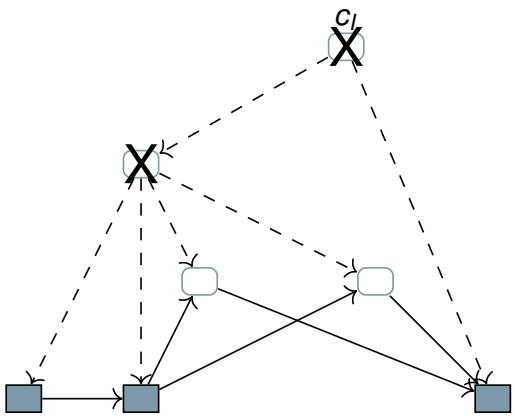
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- 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: Introduction to HTN Planning



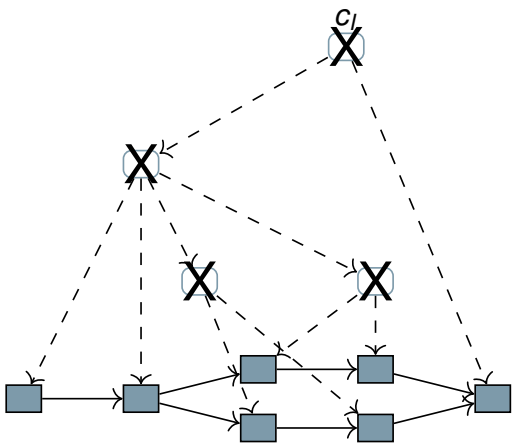
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- 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: Introduction to HTN Planning



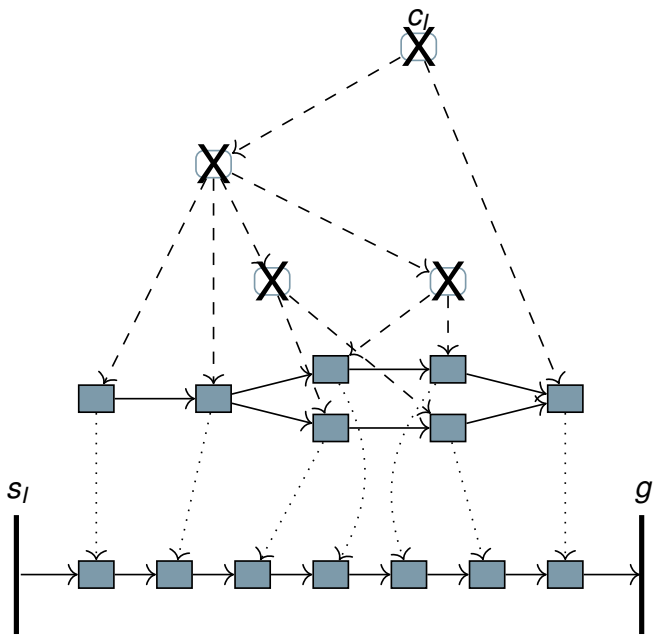
$$\mathcal{P} = (F, P, \delta, C, M, s_I, c_I, g)$$

- 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: 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
- $s_I \in 2^F$ 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.

Introduction: Motivation

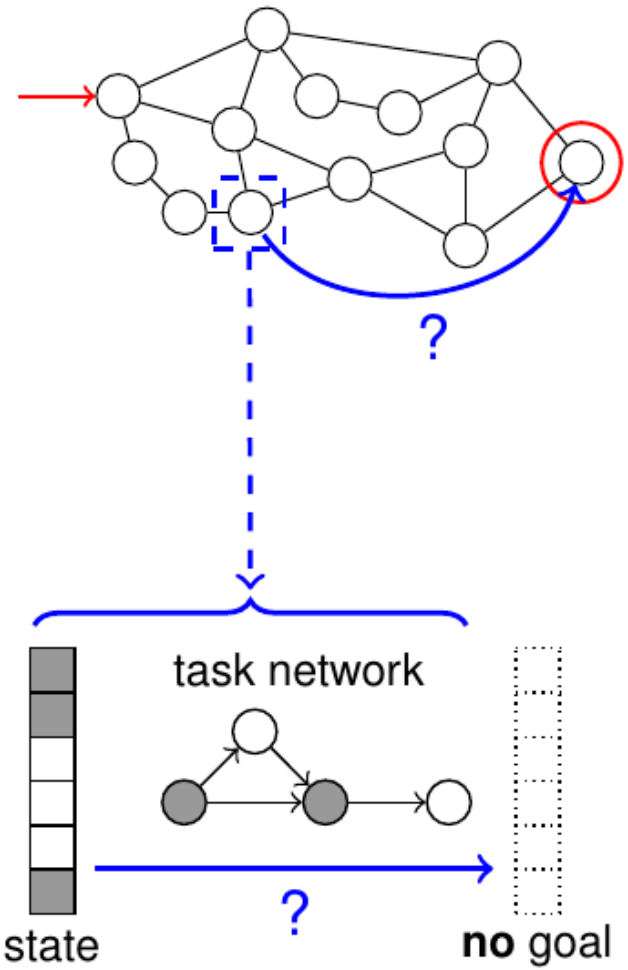
Why defining/solving a *Hierarchical Problem*?

- In many real-world applications, knowledge is given in form of control rules: we know the steps required to perform some task.
- More control on the generated plans, since all the “rules” need to be obeyed. We can *exclude* undesired plans!
- Plans can be presented *more abstract* by relying on task hierarchies.
- And many more!

Heuristic Search: Introduction

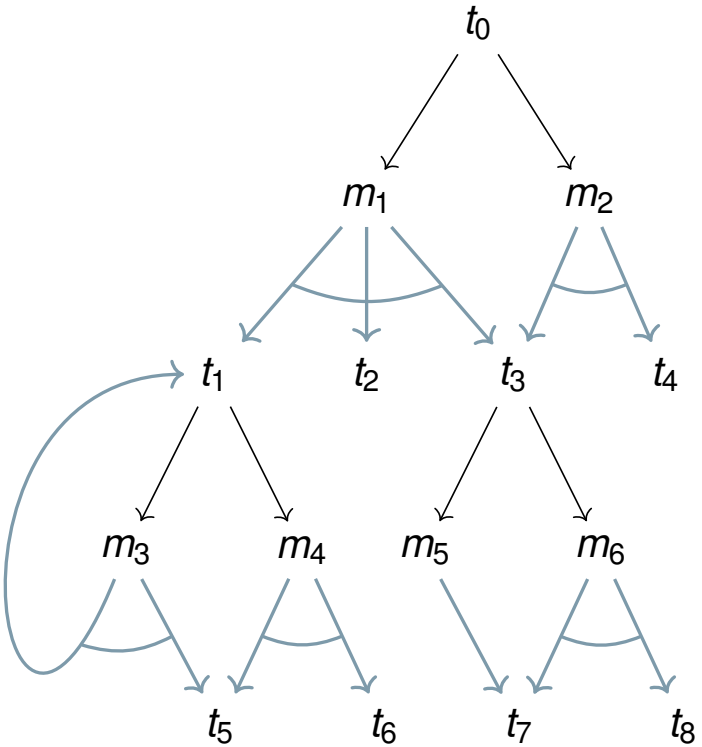
How does search work?

- We systematically “search” through the search space, i.e., we explore all “options” from an initial search node.
- In *heuristic* search, we explore search nodes according to their heuristic value.
- Heuristics estimate the distance from the current search node to the nearest goal.
- They do so by solving a simplified problem and using its cost as heuristic value. (Example: straight line distance!)



Heuristic Search: TDG-based Heuristics

A Task Decomposition Graph (TDG) represents the task hierarchy:



A TDG is a (possibly cyclic) bipartite graph $\mathcal{G} = \langle N_T, N_M, E_{(T,M)}, E_{(M,T)} \rangle$ with

- N_T, the task nodes,
- N_M, the method nodes,
- $E_{(T,M)}$, the task edges,
- $E_{(M,T)}$, the method edges.

P. Bercher et al. "An Admissible HTN Planning Heuristic". In: ***IJCAI 2017***

P. Bercher, S. Keen, and S. Biundo. "Hybrid Planning Heuristics Based on Task Decomposition Graphs". In: ***SoCS 2014***

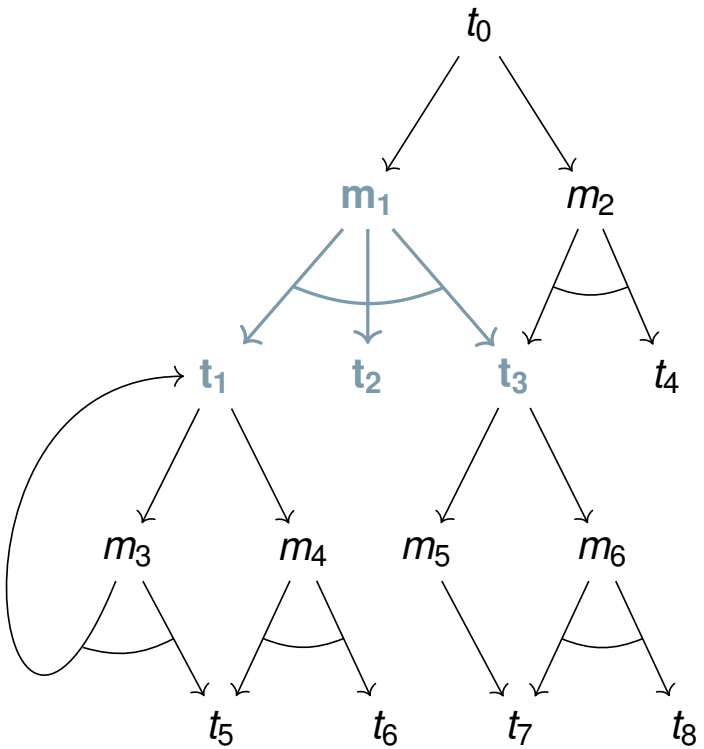
(Partially) based on the TDG as well: *Landmarks*

M. Elkawkagy, **P. Bercher**, B. Schattenberg, and S. Biundo. "Improving Hierarchical Planning Performance by the Use of Landmarks". In: ***AAAI 2012***

D. Höller and **P. Bercher**. "Landmark Generation in HTN Planning". In: ***AAAI 2021***

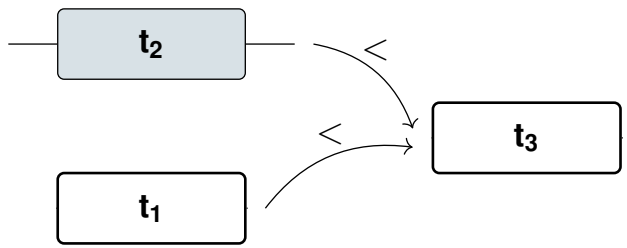
Heuristic Search: TDG-based Heuristics

A Task Decomposition Graph (TDG) represents the task hierarchy:



What to do?

- Minimize over all methods,
- add tasks within a method



$$h_M(m_1) = \sum_{t_i \in \{t_1, t_2, t_3\}} h_T(t_i)$$
$$= h_T(t_1) + cost(t_2) + h_T(t_3)$$

P. Bercher et al. "An Admissible HTN Planning Heuristic". In: ***IJCAI 2017***

P. Bercher, S. Keen, and S. Biundo. "Hybrid Planning Heuristics Based on Task Decomposition Graphs". In: ***SoCS 2014***

Heuristic Search: Further Work Related to HTN Heuristics

- Based on landmarks, i.e., tasks/methods/facts that are required in any solution.
- Encode delete- and ordering-relaxed HTN problem as Integer Linear Program (ILP).
- Encode search node (relaxed) into a non-hierarchical planning problem (and then use existing heuristics from that setting).

D. Höller and **P. Bercher**. “Landmark Generation in HTN Planning”. In: ***AAAI 2021***

D. Höller, **P. Bercher**, and G. Behnke. “Delete- and Ordering-Relaxation Heuristics for HTN Planning”. In: ***IJCAI 2020***

D. Höller, **P. Bercher**, G. Behnke, and S. Biundo. “HTN Planning as Heuristic Progression Search”. In: ***JAIR*** (2020)

– as above –. “On Guiding Search in HTN Planning with Classical Planning Heuristics”. In: ***IJCAI 2019***

– as above –. “A Generic Method to Guide HTN Progression Search with Classical Heuristics”. In: ***ICAPS 2018***

Planning-Based Assistance

Introduction: Planning-based Assistance Systems

Approach: Model the environment and task as planning problem.

- Then, automatically generate solution plans,
- choose user-friendly (i.e., meaningful) order for detailed instructions,
- repair plans if required (to deal with execution errors), and
- explain tasks or background knowledge.

Assembly Assistant: Problem Illustration

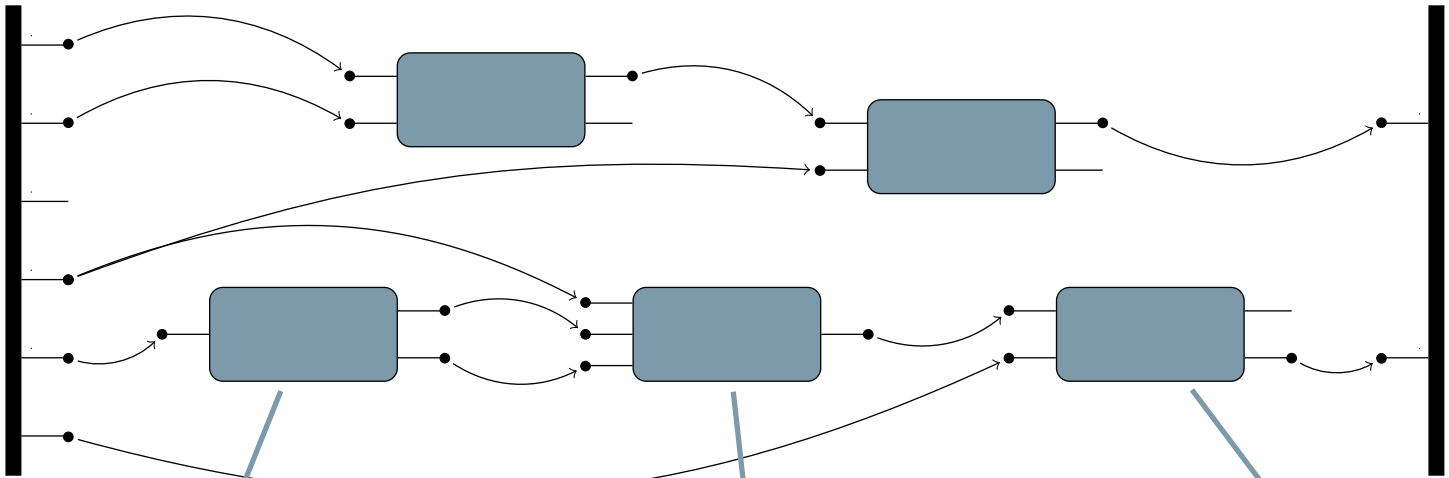
Sink devices:

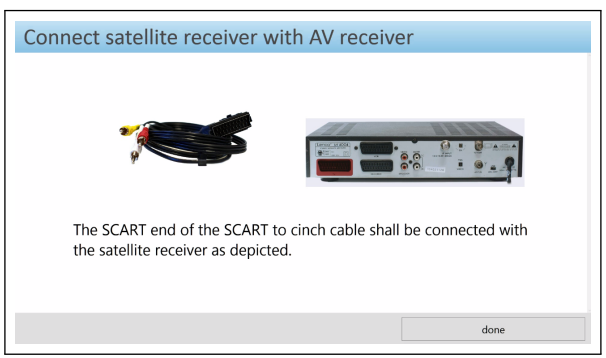
- Television (requires video)
- Amplifier (requires audio)

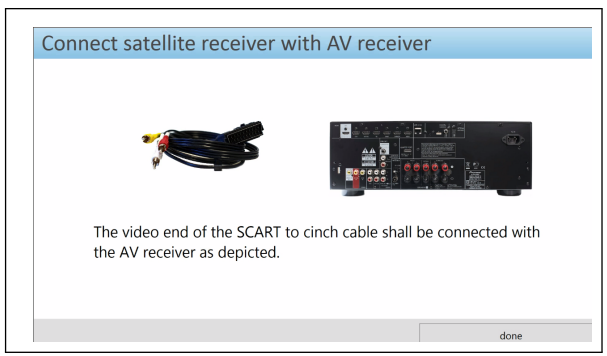
Source devices:

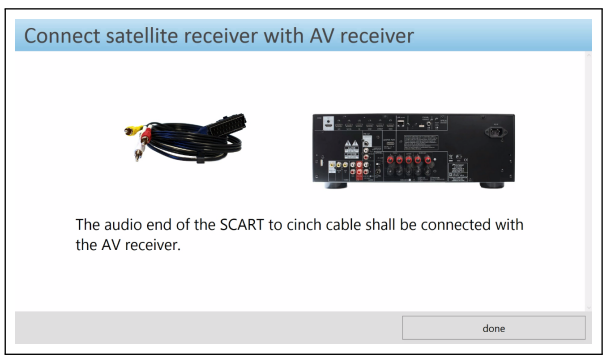
- Blu-ray player
- Satellite receiver
  (both produce audio & video)

Assembly Assistant: Solution Plan

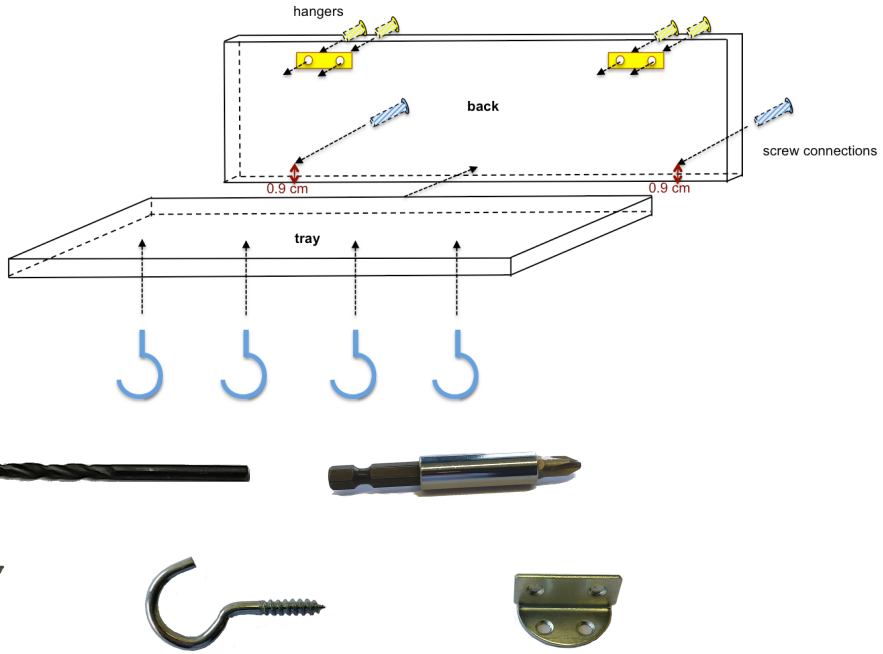








Do It Yoursef (DIY) Assistant: Problem Illustration



The material:

- Boards (need to be cut first)
- Electrical devices, e.g., drills, saws
- Attachments like drill bits and materials like nails

G. Behnke, **P. Bercher** et al. “New Developments for Robert – Assisting Novice Users Even Better in DIY Projects”. In: ***ICAPS 2020***

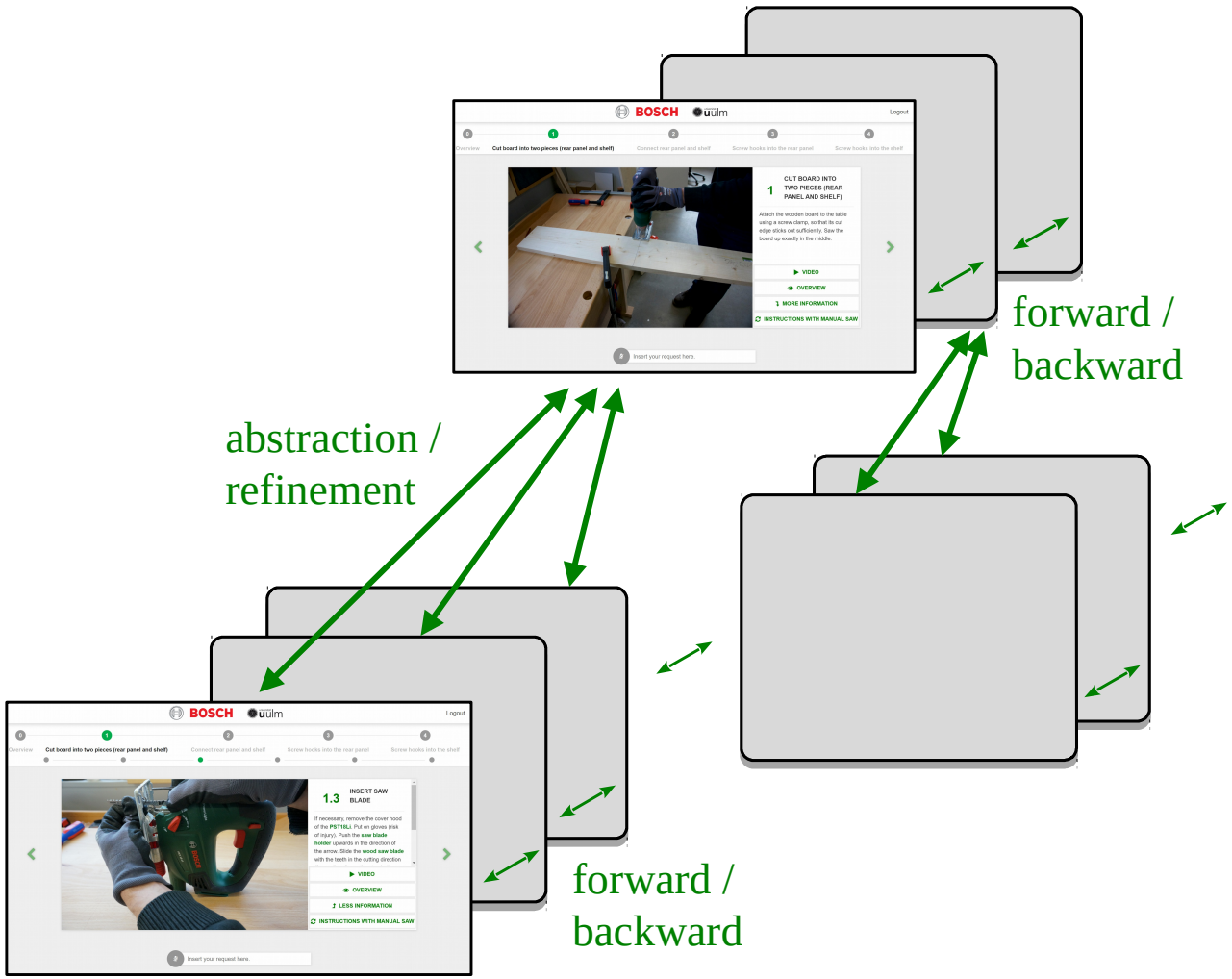
M. Kraus, M. Schiller, G. Behnke, **P. Bercher**, M. Dorna, M. Dambier, B. Glimm, and S. Biundo, W. Minker. “Was that successful? On Integrating Proactive Meta-Dialogue in a DIY-Assistant System using Multimodal Cues”. In: ***ICMI 2020***

G. Behnke, M. Schiller, M. Kraus, **P. Bercher**, M. Schmautz, M. Dorna, M. Dambier, W. Minker, B. Glimm, and S. Biundo. “Alice in DIY wonderland or: Instructing novice users on how to use tools in DIY projects”. In: ***AI Communications*** (2019)

– as above –. “Instructing Novice Users on How to Use Tools in DIY Projects”. In: ***IJCAI-ECAI 2018***

Do It Yoursef (DIY) Assistant: Exploit Task Hierarchy

Presentation of instructions on different levels of abstraction:



Summary

Main Research Interests

Problem classes:

- (Non-hierarchical) Partial Order Causal Link (POCL) Planning
- Hierarchical Task Network (HTN) Planning

Research areas:

- Complexity investigations
  (of various interesting problems)
- Heuristic search
  (algorithms, heuristics, etc.)
- Practical applications
  (or questions related to it)

} from theory to practice!

Appendix: Complexity Results

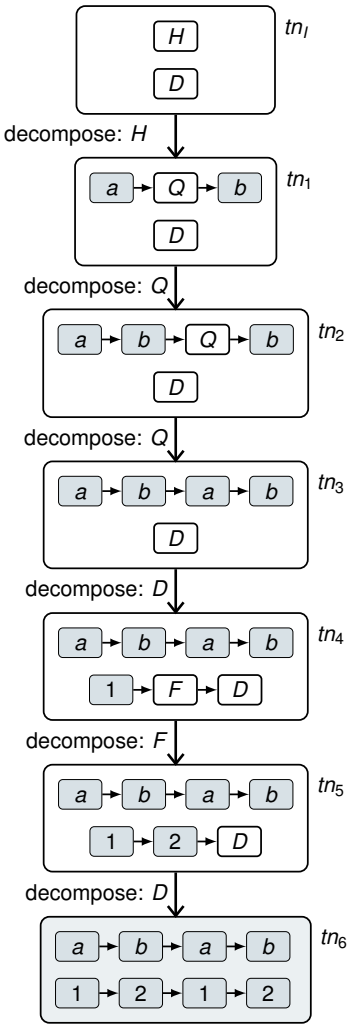
Plan Existence Complexity Results: HTN Planning versus Classical Planning

Theorem: HTN planning is **undecidable** *(Erol et al., 1994/1996)*

- In HTN planning, we can only "manipulate" task networks via decomposition: Pick a decomposition method (from the model) and replace its compound task by it.
- But what if we allowed the insertion of actions (primitive tasks) arbitrarily, just as in classical planning?
  - Will the problem become easier?
  - What are the consequences?
- We call the resulting formalism:
  TIHTN Planning: HTN Planning with Task Insertion

P. Bercher et al. "A Survey on Hierarchical Planning – One Abstract Idea, Many Concrete Realizations". In: ***IJCAI 2019***
R. Alford, **P. Bercher**, and D. Aha. "Tight Bounds for HTN planning with Task Insertion". In: ***IJCAI 2015***
T. Geier and **P. Bercher**. "On the Decidability of HTN Planning with Task Insertion". In: ***IJCAI 2011***

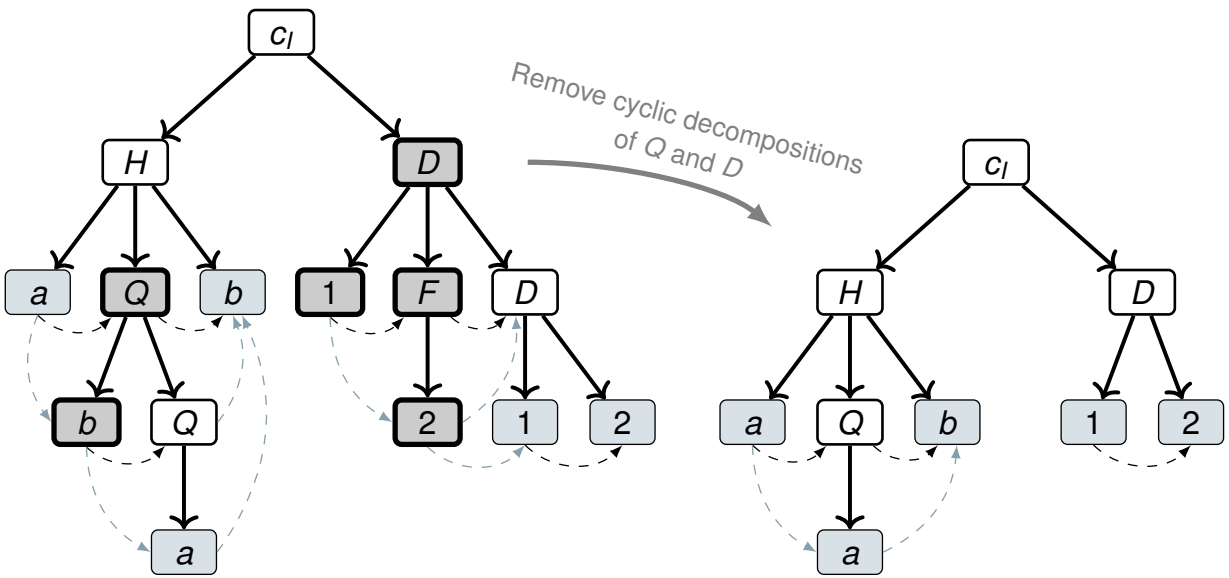
Plan Existence Complexity Results: HTN Planning with Task Insertion



Lemma: TIHTN planning doesn't need cyclic decompositions!

T. Geier and **P. Bercher**. "On the Decidability of HTN Planning with Task Insertion". In: ***IJCAI 2011***

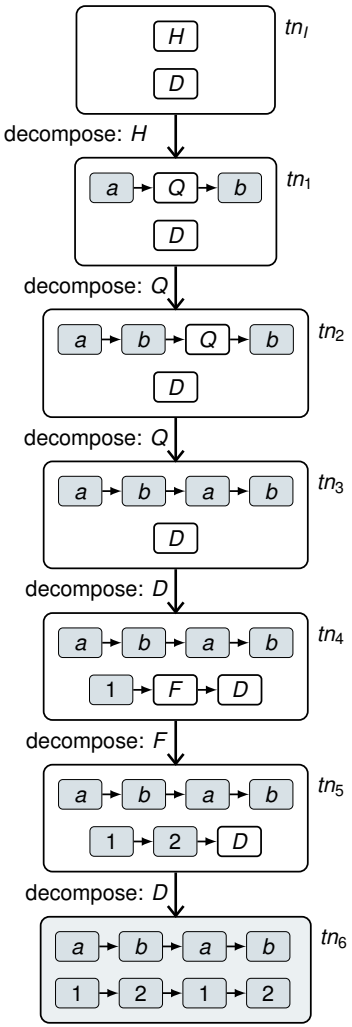
Restrict to *acyclic* decompositions, re-fill with task insertion!



The acyclic tree missed the actions b, 1, 2

But we can just re-insert them via task insertion!

Plan Existence Complexity Results: HTN Planning with Task Insertion



Theorem: TIHTN planning is in **NEXPTIME**

R. Alford, **P. Bercher**, and D. Aha. "Tight Bounds for HTN planning with Task Insertion". In: ***IJCAI 2015***

T. Geier and **P. Bercher**. "On the Decidability of HTN Planning with Task Insertion". In: ***IJCAI 2011***

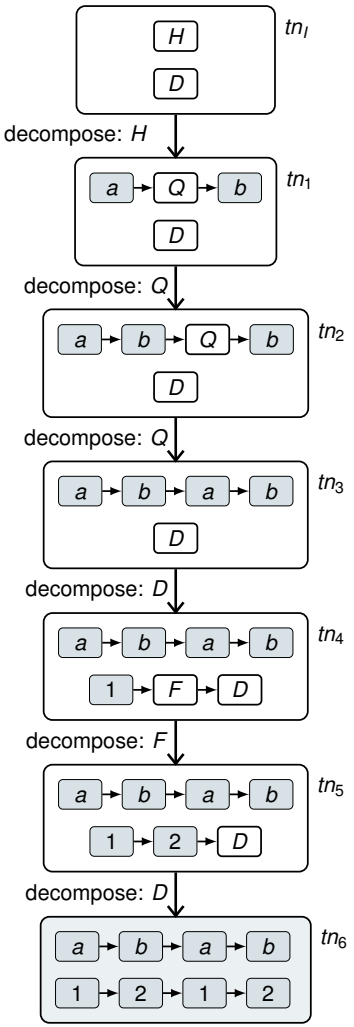
1. Step: Guess an acyclic decomposition:

A tree with acyclic decompositions describes at most $b^{|C|+1}$ decompositions.

(C = set of compound tasks)
(b = size of largest task network in the model)

Verify in $O(b^{|C|+1})$ whether the tree describes a correct sequence of decompositions.

Plan Existence Complexity Results: HTN Planning with Task Insertion



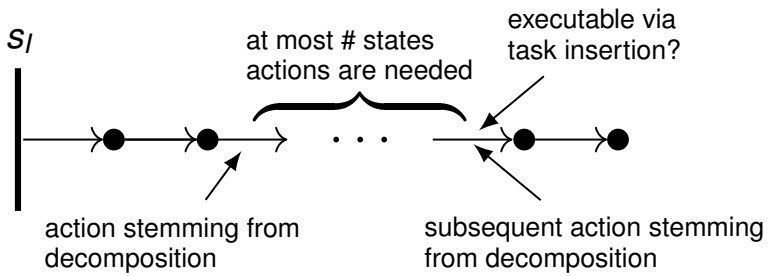
Theorem: TIHTN planning is in **NEXPTIME**

R. Alford, **P. Bercher**, and D. Aha. "Tight Bounds for HTN planning with Task Insertion". In: ***IJCAI 2015***

T. Geier and **P. Bercher**. "On the Decidability of HTN Planning with Task Insertion". In: ***IJCAI 2011***

2. Step: Guess the actions and orderings to be inserted.

The (guessed) decomposition tree results into a task network with at most $\leq b^{|C|+1}$ tasks.



Between each two actions, at most $2^{|V|}$ actions need to be inserted to achieve the next precondition.

($|V|$ = number of state variables, $2^{|V|}$ = number of states)

Plan Existence Complexity Results: Overview

Restrictions on Hierarchy	Task Networks Totally Ordered?	Task Insertion?	Complexity Class
Arbitrary	yes	yes	**PSPACE**
Arbitrary	yes	no	**EXPTIME**
Arbitrary	no	yes	**NEXPTIME**
Arbitrary	no	no	**undecidable** and **semi−decidable**
Tail-Recursive	yes	yes	**PSPACE**
Tail-Recursive	yes	no	**PSPACE**
Tail-Recursive	no	yes	**NEXPTIME**
Tail-Recursive	no	no	**EXPSPACE**
Regular	yes	yes	**PSPACE**
Regular	yes	no	**PSPACE**
Regular	no	yes	**PSPACE**
Regular	no	no	**PSPACE**
Acyclic	yes	yes	**PSPACE**
Acyclic	yes	no	**PSPACE**
Acyclic	no	yes	**NEXPTIME**
Acyclic	no	no	**NEXPTIME**
No Hierarchy	yes	yes	**PSPACE**
No Hierarchy	yes	no	in **P**
No Hierarchy	no	yes	**PSPACE**
No Hierarchy	no	no	**NP**

Results from the table:

P. Bercher et al. “More than a Name? On Implications of Preconditions and Effects of Compound HTN Planning Tasks”. In: ***ECAI 2016***

R. Alford, **P. Bercher**, and D. Aha. “Tight Bounds for HTN planning with Task Insertion”. In: ***IJCAI 2015***

R. Alford, **P. Bercher**, and D. Aha. “Tight Bounds for HTN Planning”. In: ***ICAPS 2015***

T. Geier and **P. Bercher**. “On the Decidability of HTN Planning with Task Insertion”. In: ***IJCAI 2011***

When ignoring orderings and delete effects:

D. Höller, **P. Bercher**, and G. Behnke. “Delete- and Ordering-Relaxation Heuristics for HTN Planning”. In: ***IJCAI 2020***

Loosely related: *Expressivity*

D. Höller, G. Behnke, and **P. Bercher**, and S. Biundo. “Language Classification of Hierarchical Planning Problems”. In: ***ECAI 2014***

D. Höller, G. Behnke, and **P. Bercher**, and S. Biundo. “Assessing the Expressivity of Planning Formalisms through the Comparison to Formal Languages”. In: ***ICAPS 2016***

Further Complexity Investigations

We investigated the computational complexity of related tasks:

- Is task/method/fact *t*/*m*/*f* a landmark, i.e., required in any solution? (As hard as the base problem)
- Is fact *f* a required precondition or guaranteed effect of a compound task? (As hard as the base problem)
- Plan verification (is a given task network a solution?) for a combination of HTN and POCL planning. (**NP-complete**)
- Can we perform change(s) X to a given solution – to obtain a solution again? (from **NP-complete** upwards)

D. Höller and **P. Bercher**. "Landmark Generation in HTN Planning". In: ***AAAI 2021***

C. Olz, S. Biundo, and **P. Bercher**. "Revealing Hidden Preconditions and Effects of Compound HTN Planning Tasks – A Complexity Analysis". In: ***AAAI 2021***

P. Bercher, D. Höller, G. Behnke, and S. Biundo. "More than a Name? On Implications of Preconditions and Effects of Compound HTN Planning Tasks". In: ***ECAI 2016***

G. Behnke, D. Höller, and **P. Bercher**, and S. Biundo. "Change the Plan – How hard can that be?" In: ***ICAPS 2016***