Hybrid Planning

Theoretical Foundations and Practical Applications

PhD thesis outline

Pascal Bercher

Institute of Artificial Intelligence

June 2015

sfb transregio 62
Companion Technology

ulm university universität ulm

Companion Technology

- **SFB/Transregio 62**:
  Companion-Technology for Cognitive Technical Systems
  `http://www.companion-technology.org`
- Realizing "intelligent" technical systems, that provide, e.g., assistance for various tasks
- Based on *user-centered AI planning*!

Companion Technology – Hybrid Planning

- Hybrid Planning fuses:
  - Partial-Order Causal-Link (POCL) Planning with
  - Hierarchical Task Network (HTN) Planning
- POCL Planning enables to:
  - generate plans
  - repair plans in case of execution failures
  - explain plans based on causality
  - integrate the user (see paper/talk of Behnke at DC)
- *HTN Planning* enables to:
  - incorporate (procedural) expert knowledge
  - explain plans based on hierarchy
  - integrate the user (see Behnke)

Biundo et al. *Advanced user assistance based on AI planning*. Cognitive Systems Research '11

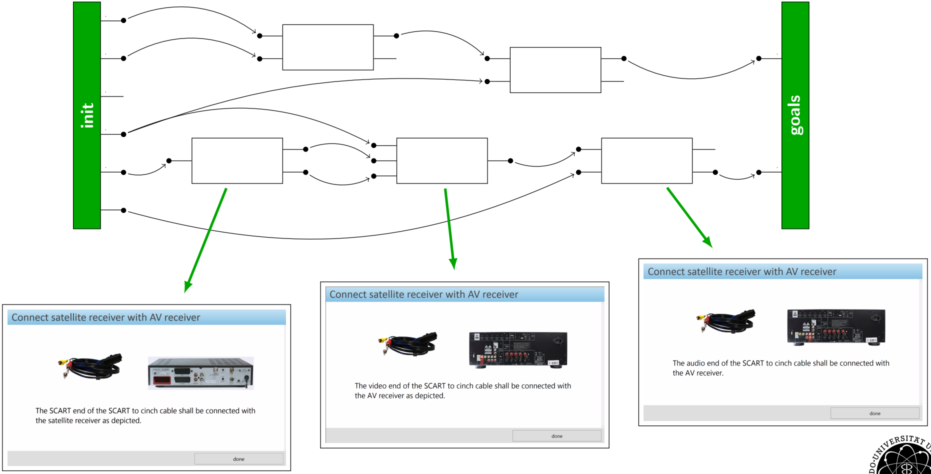
Companion Technology – Hybrid Planning

Example Scenario: Assembling a Home Entertainment System

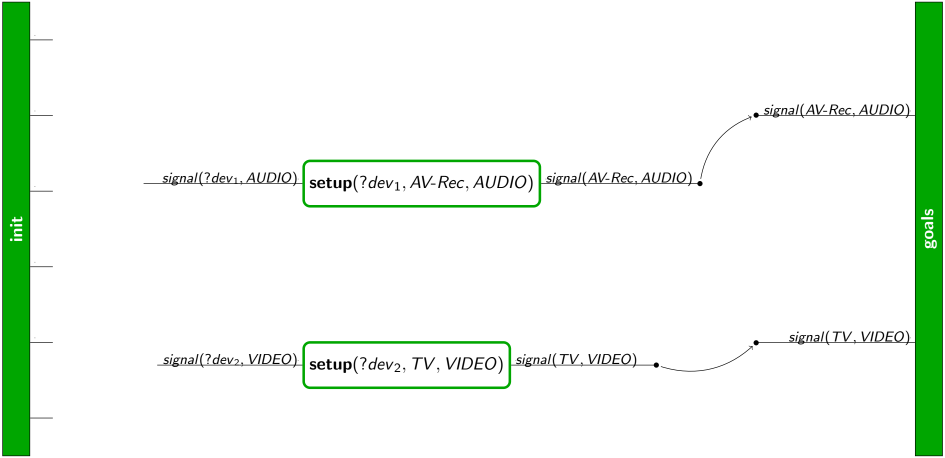
Bercher et al. *A Planning-based Assistance System for Setting Up a Home Theater*. AAAI '15.

Bercher et al. *Plan, Repair, Execute, Explain - How Planning Helps to Assemble your Home Theater*. ICAPS '14.

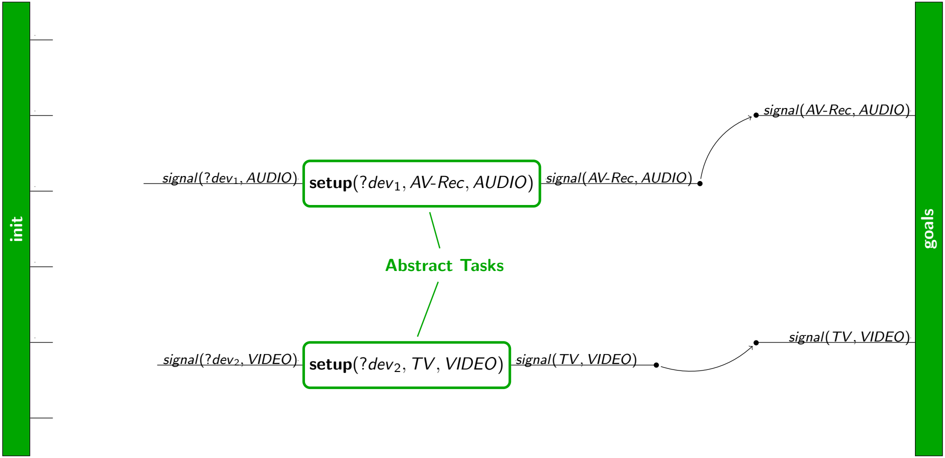
Companion Technology – Hybrid Planning



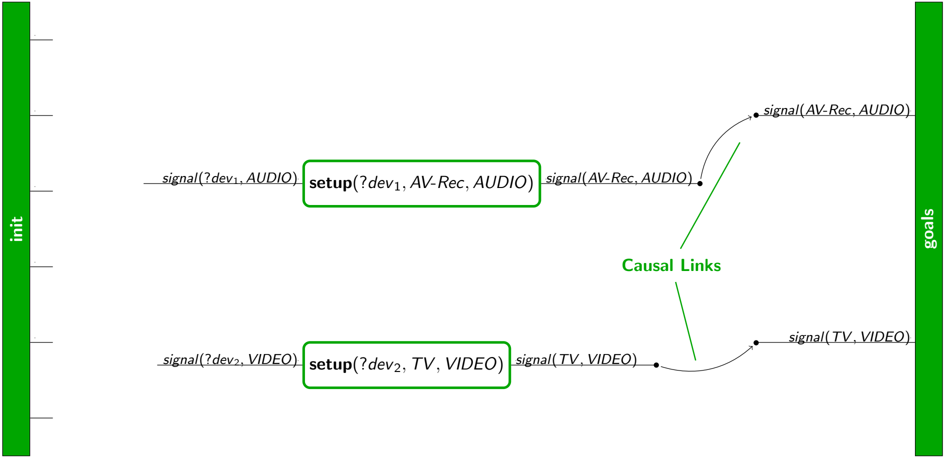
Planning Problem / Planning Procedure



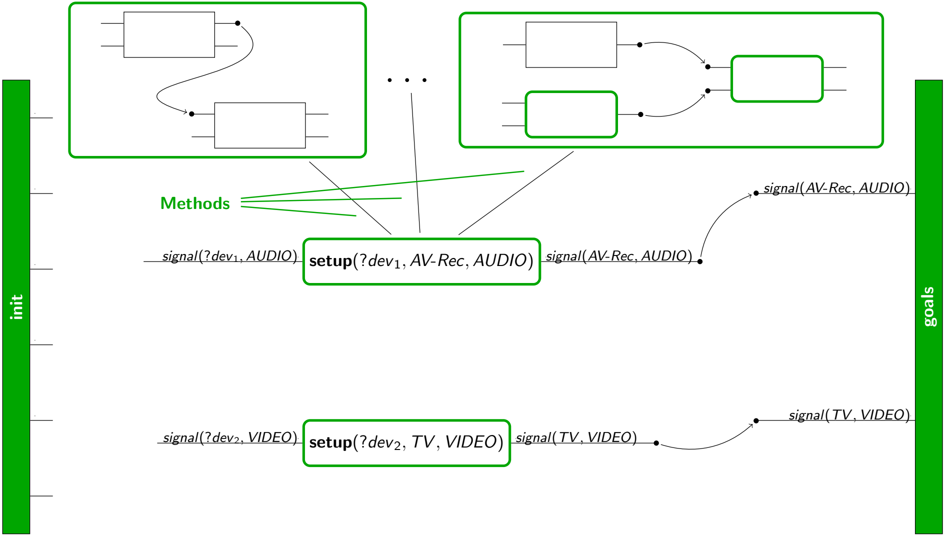
Planning Problem / Planning Procedure



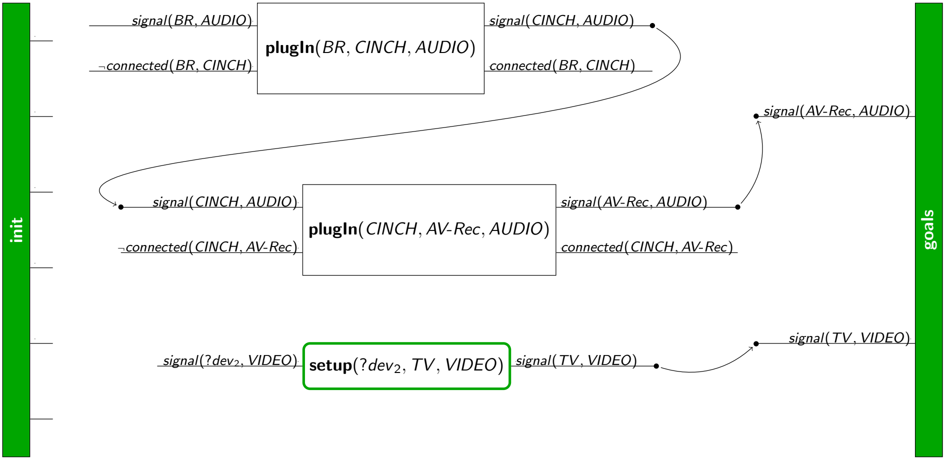
Planning Problem / Planning Procedure



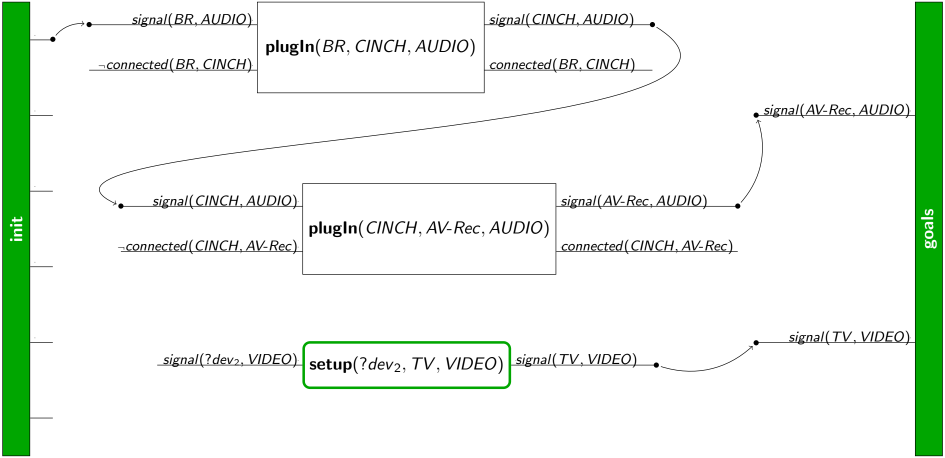
Planning Problem / Planning Procedure



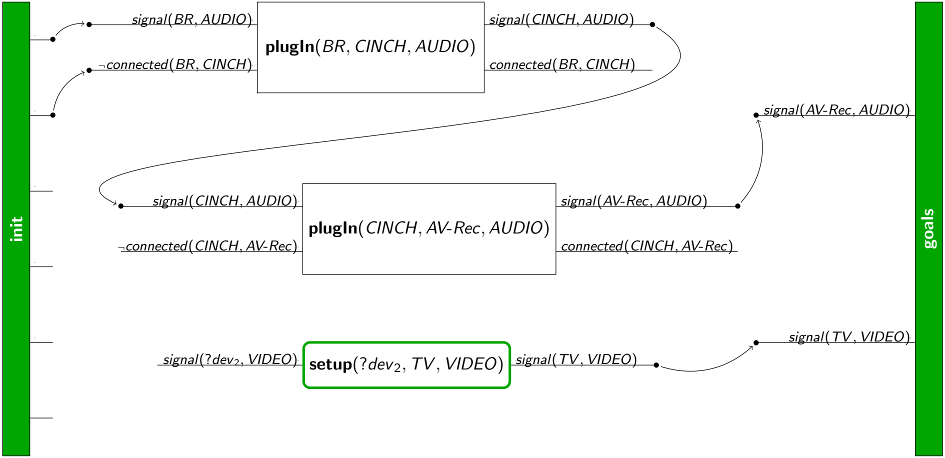
Planning Problem / Planning Procedure



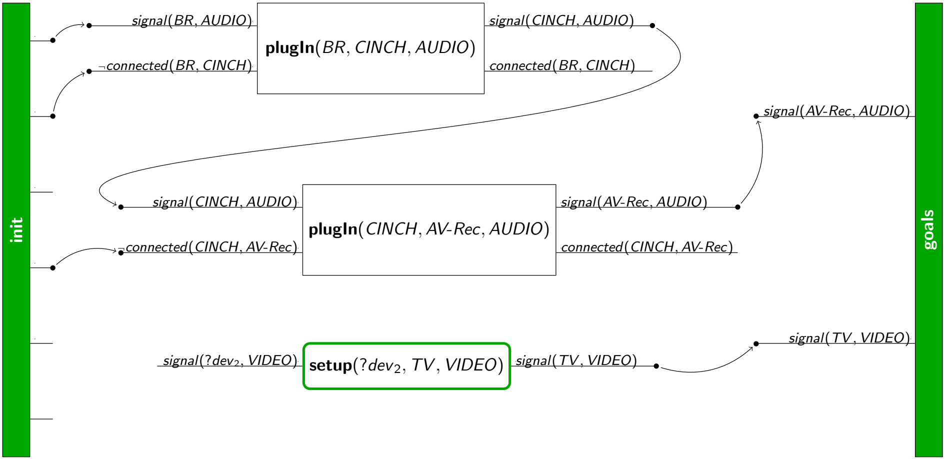
Planning Problem / Planning Procedure



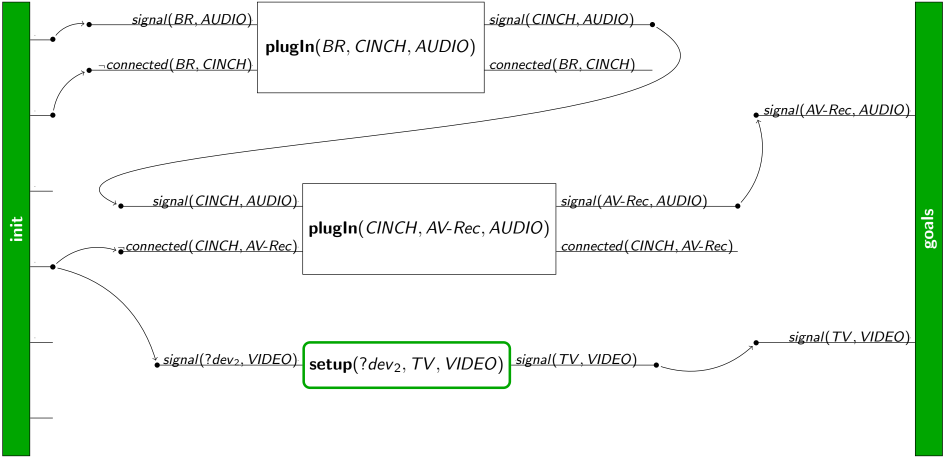
Planning Problem / Planning Procedure



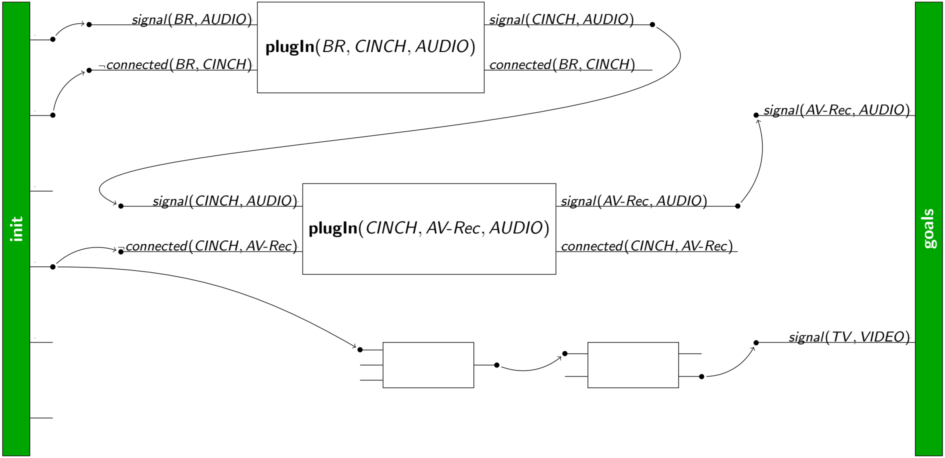
Planning Problem / Planning Procedure



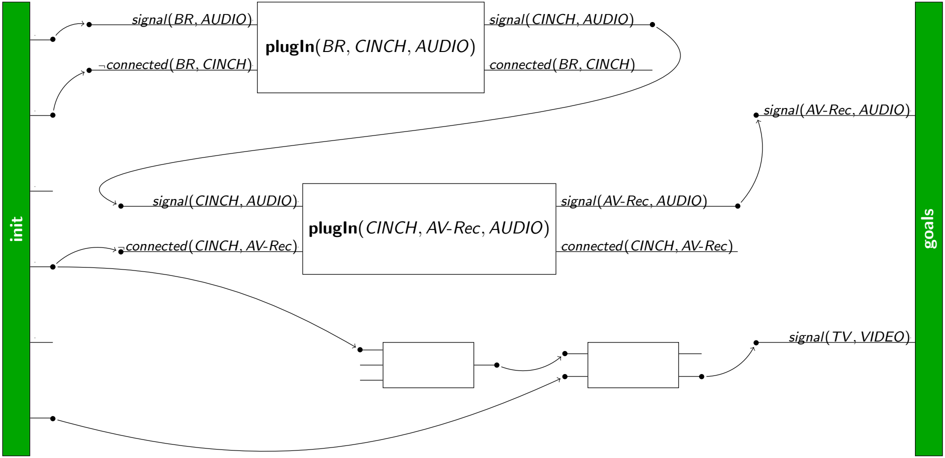
Planning Problem / Planning Procedure



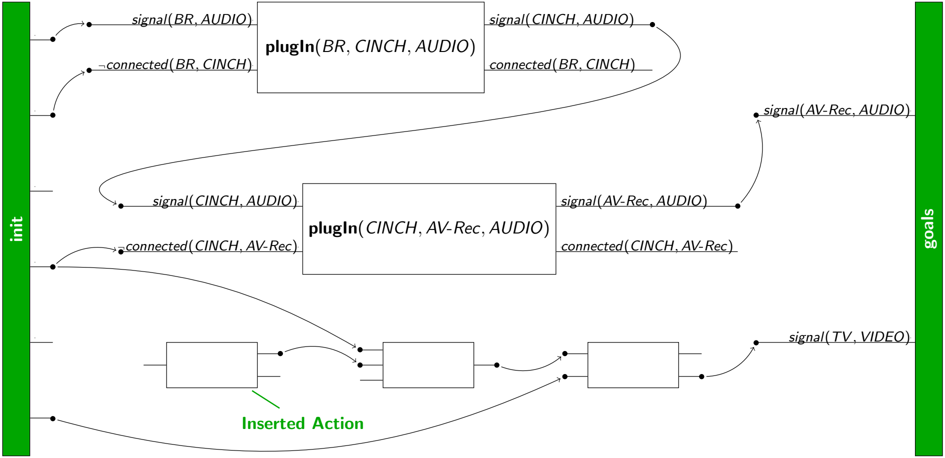
Planning Problem / Planning Procedure



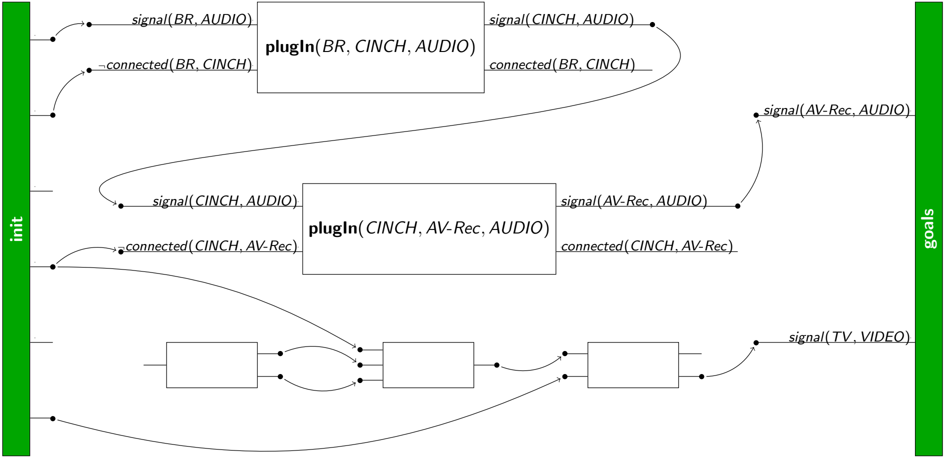
Planning Problem / Planning Procedure



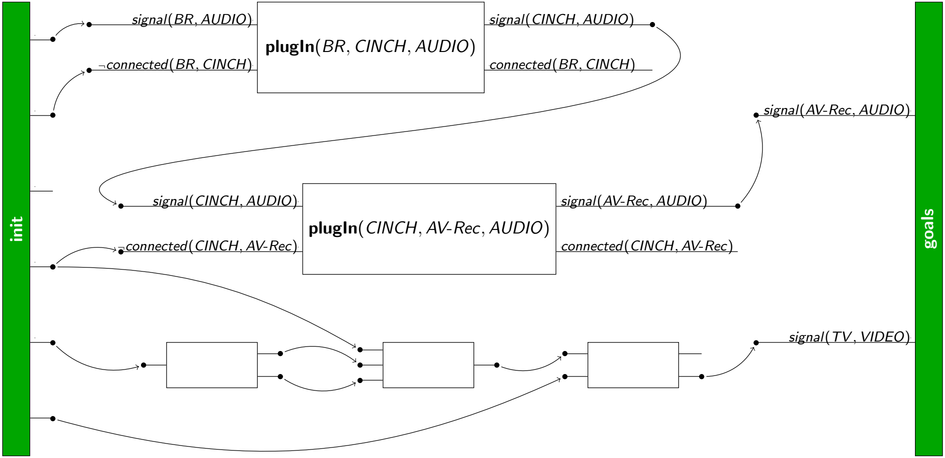
Planning Problem / Planning Procedure



Planning Problem / Planning Procedure



Planning Problem / Planning Procedure

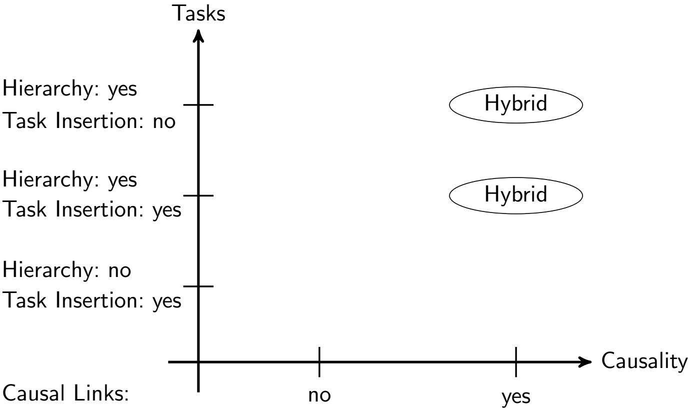


Different Kinds of Problem Classes

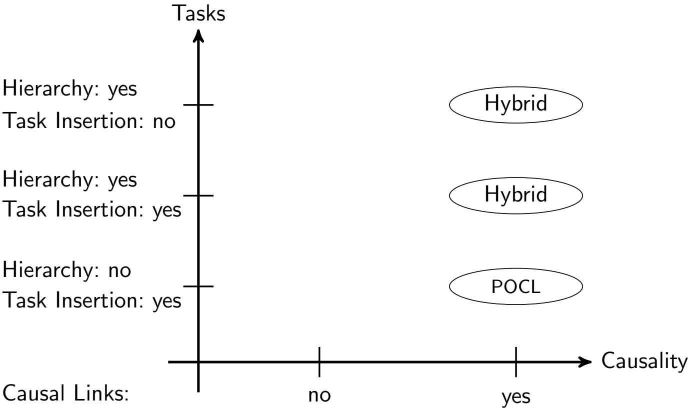
Various restrictions result in different problem classes:

- Is there a hierarchy? yes/no
- May actions be inserted? yes/no
- Are there causal links? yes/no

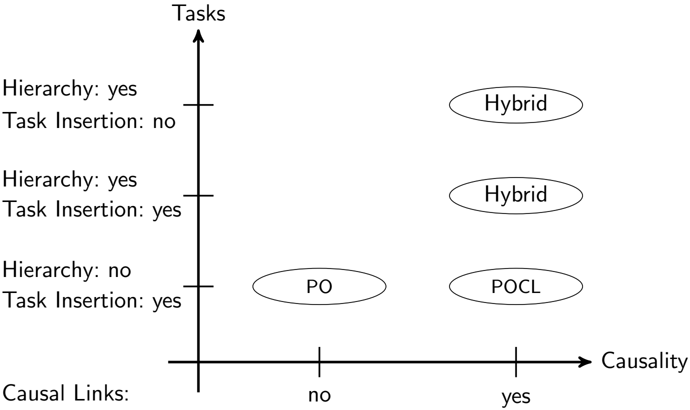
Different Kinds of Problem Classes



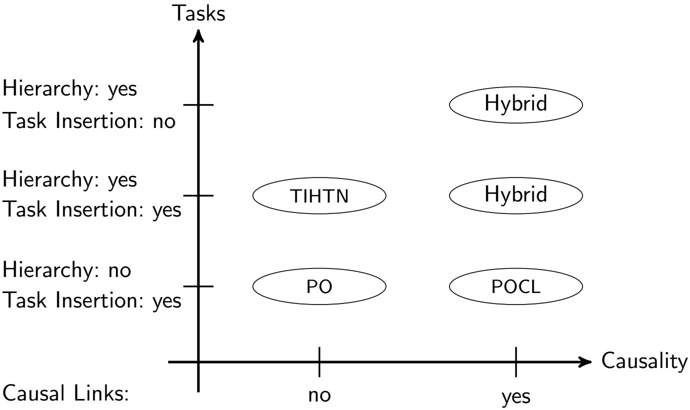
Different Kinds of Problem Classes



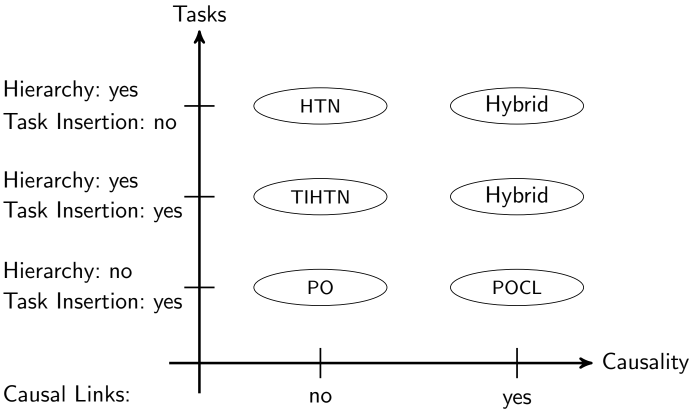
Different Kinds of Problem Classes



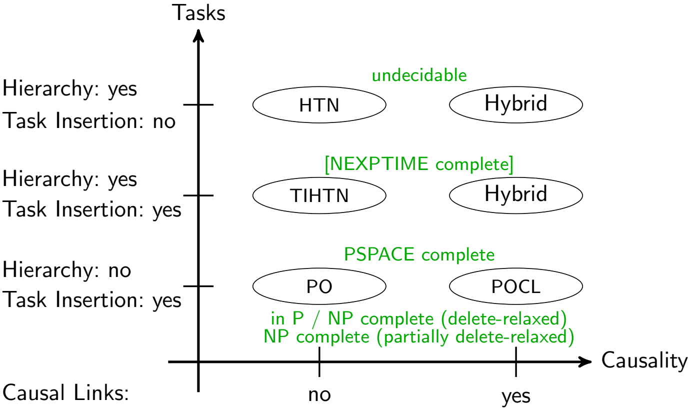
Different Kinds of Problem Classes



Different Kinds of Problem Classes



Overview: Complexity Results



Geier und Bercher. *On the Decidability of HTN Planning with Task Insertion*. IJCAI '11.

Bercher et al. *On Delete Relaxation in Partial-Order Causal-Link Planning*. ICTAI '13.

Alford, Bercher, und Aha. *Tight Bounds for HTN Planning*. ICAPS '15.

Alford, Bercher, und Aha. *Tight Bounds for HTN Planning with Task Insertion*. IJCAI '15.

Seach Algorithm PANDA$_2$

Input : $\texttt{Fringe} = \{P_{\text{init}}\}$
Output : A solution plan or `fail`.

while $\texttt{Fringe} \neq \emptyset$ **do**
  $P :=$ **PlanSel**$(\texttt{Fringe})$
  $F := \mathit{FlawDet}(P)$
  if $F = \emptyset$ **then return** P
  $f :=$ **FlawSel**(F)
  $\texttt{Fringe} := (\texttt{Fringe} \setminus \{P\}) \cup \mathit{Successors}(P, f)$
return `fail`

Bercher, Keen, und Biundo. *Hybrid Planning Heuristics Based on Task Decomposition Graphs*. SoCS '14.

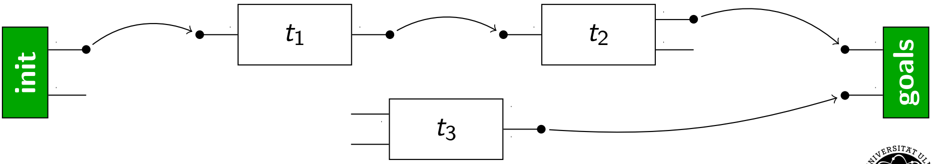
POCL Heuristic SampleFF

Partial order of plans makes delete-relaxation NP hard. Solution:

- sample n linearizations
- solve resulting sub problems, e.g., using the FF heuristic

Linearizations:

- $t_3 \rightarrow t_1 \rightarrow t_2$
- $t_1 \rightarrow t_3 \rightarrow t_2$
- $t_1 \rightarrow t_2 \rightarrow t_3$



Bercher et al. *On Delete Relaxation in Partial-Order Causal-Link Planning*. ICTAI '13.

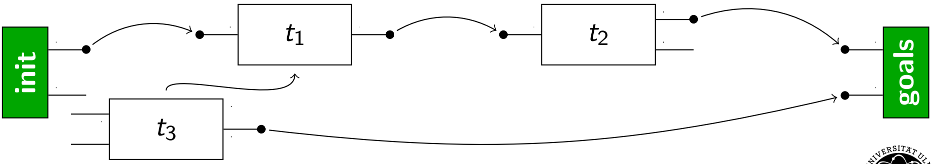
POCL Heuristic SampleFF

Partial order of plans makes delete-relaxation NP hard. Solution:

- sample *n* linearizations
- solve resulting sub problems, e.g., using the FF heuristic

Linearizations:

- $t_3 \rightarrow t_1 \rightarrow t_2$
- $t_1 \rightarrow t_3 \rightarrow t_2$
- $t_1 \rightarrow t_2 \rightarrow t_3$



Bercher et al. *On Delete Relaxation in Partial-Order Causal-Link Planning*. ICTAI '13.

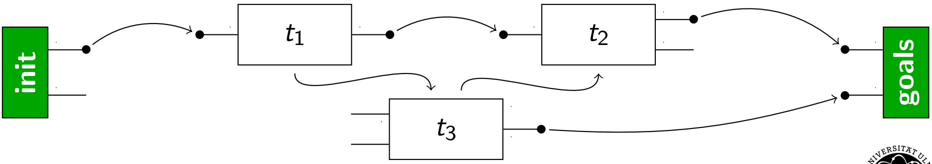
POCL Heuristic SampleFF

Partial order of plans makes delete-relaxation NP hard. Solution:

- sample *n* linearizations
- solve resulting sub problems, e.g., using the FF heuristic

Linearizations:

- $t_3 \rightarrow t_1 \rightarrow t_2$
- $t_1 \rightarrow t_3 \rightarrow t_2$
- $t_1 \rightarrow t_2 \rightarrow t_3$



Bercher et al. *On Delete Relaxation in Partial-Order Causal-Link Planning*. ICTAI '13.

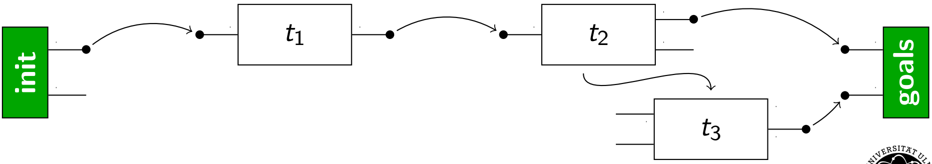
POCL Heuristic SampleFF

Partial order of plans makes delete-relaxation NP hard. Solution:

- sample n linearizations
- solve resulting sub problems, e.g., using the FF heuristic

Linearizations:

- $t_3 \rightarrow t_1 \rightarrow t_2$
- $t_1 \rightarrow t_3 \rightarrow t_2$
- $t_1 \rightarrow t_2 \rightarrow t_3$



Bercher et al. *On Delete Relaxation in Partial-Order Causal-Link Planning*. ICTAI '13.

POCL Heuristic SampleFF – Results

Empirical Results (IPC benchmarks)

- Add heuristic still better than SampleFF
- SampleFF competitive with Relax heuristic
- SampleFF was the only heuristic that was able to prove the unsolvability of an unsolvable planning instance

POCL Heuristics Based on State-based Heuristics

Translate planning problem $\mathcal{P} = \langle \mathcal{D}, P_{\text{init}} \rangle$ into $\mathcal{P}' = \langle \mathcal{D}', s_{\text{init}} \rangle$, such that:

solutions for $\mathcal{P}$	$\equiv$	solutions for $\mathcal{P}'$
goal distance for P_{init}	$\equiv$	goal distance for s_{init}

Then:
For existing state-based heuristic h' set $h(P) := h'(s_{\text{init}})$

Bercher et al. *Using State-Based Planning Heuristics for Partial-Order Causal-Link Planning*. KI '13.

POCL Heuristics Based on State-based Heuristics

Translate planning problem $\mathcal{P} = \langle \mathcal{D}, P_{\text{init}} \rangle$ into $\mathcal{P}' = \langle \mathcal{D}', s_{\text{init}} \rangle$, such that:

$$\begin{aligned} \text{solutions for } \mathcal{P} \quad &\equiv \quad \text{solutions for } \mathcal{P}' \\ \text{goal distance for } P_{\text{init}} \quad &\equiv \quad \text{goal distance for } s_{\text{init}} \end{aligned}$$

Then:
For existing state-based heuristic h' set $h(P) := h'(s_{\text{init}})$

Bercher et al. *Using State-Based Planning Heuristics for Partial-Order Causal-Link Planning*. KI '13.

POCL Heuristics Based on State-based Heuristics

Translate planning problem $\mathcal{P} = \langle \mathcal{D}, P_{\mathrm{init}} \rangle$ into $\mathcal{P}' = \langle \mathcal{D}', s_{\mathrm{init}} \rangle$, such that:

$$\begin{array}{lcl} \text{solutions for } \mathcal{P} & \equiv & \text{solutions for } \mathcal{P}' \\ \text{goal distance for } P_{\mathrm{init}} & \equiv & \text{goal distance for } s_{\mathrm{init}} \end{array}$$

Then:
For existing state-based heuristic h' set $h(P) := h'(s_{\mathrm{init}})$

Bercher et al. *Using State-Based Planning Heuristics for Partial-Order Causal-Link Planning*. KI '13.

POCL Heuristics Based on State-based Heuristics – Results

Empirical Results (IPC benchmarks)

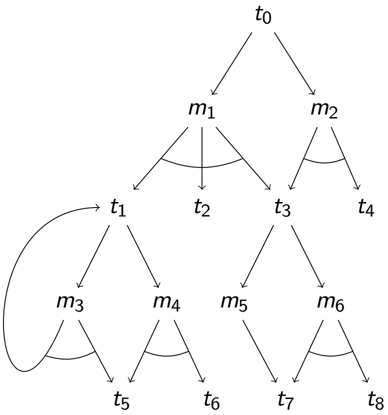
- many state-based heuristics do not work for encoding
- SampleFF with LM-cut better informed than Add and Relax heuristic

Theoretical Properties

- Encoding provides first admissible heuristic for POCL planning

MME Heuristic for Hybrid Planning

The Task Decomposition Graph (TDG) represents the Decomposition structure:



Task Decomposition Tree (TDT):

Elkawkagy et al. *Landmarks in hierarchical planning*. ECAI '10.

TDG:

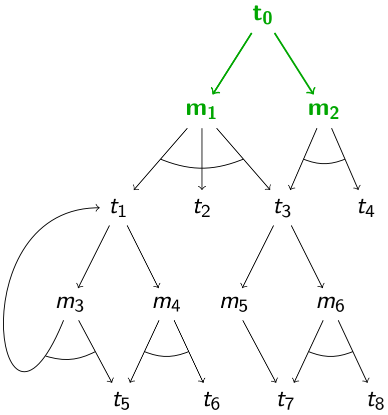
Elkawkagy et al. *Improving hierarchical planning performance by the use of landmarks*. AAAI '12.

TDG Heuristics:

Bercher, Keen, und Biundo. *Hybrid Planning Heuristics Based on Task Decomposition Graphs*. SoCS '14.

MME Heuristic for Hybrid Planning

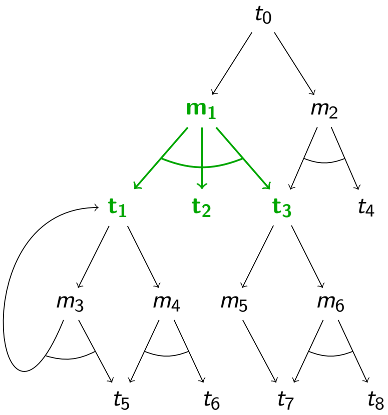
The Minimal Modification Effort Heuristic MME.



$$h(t_0) = 1 + min\,\{h(m_1), h(m_2)\}$$

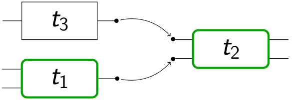
MME Heuristic for Hybrid Planning

The Minimal Modification Effort Heuristic MME.



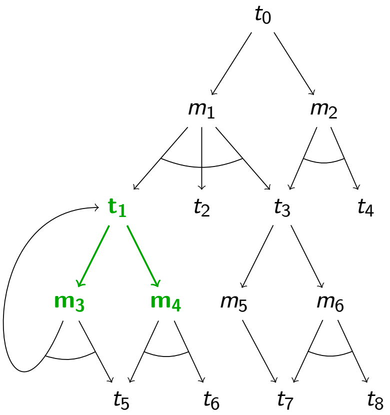
Method $m_1 = (t_0, P)$ with plan $P = (\{t_1, t_2, t_3\}, \prec, CL)$

$$h(m_1) = \sum_{t_i \in \{t_1, t_2, t_3\}} h(t_i) - |CL|$$



MME Heuristic for Hybrid Planning

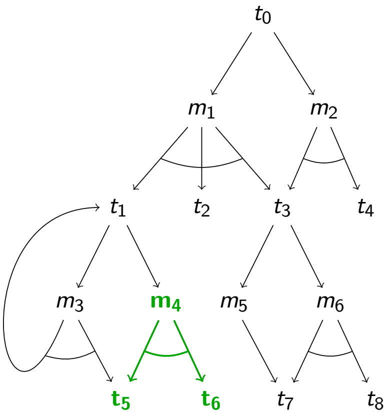
The Minimal Modification Effort Heuristic MME.



$$h(t_1) = 1 + min\,\{h(m_3), h(m_4)\}$$

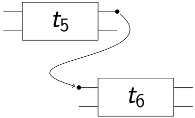
MME Heuristic for Hybrid Planning

The Minimal Modification Effort Heuristic MME.



Method $m_4 = (t_1, P)$ with
plan $P = (\{t_5, t_6\}, \prec, CL)$

$$\begin{aligned} h(m_4) &= h(t_5) + h(t_6) - |CL| \\ &= |pre(t_5)| + |pre(t_6)| - 1 \\ &= 2 + 2 - 1 = 3 \end{aligned}$$



MME Heuristic for Hybrid Planning– Results

Empirical Results (hybrid planning domains)

- Competitive with or better than all other heuristics for hybrid planning

Theoretical Properties

- First admissible heuristic for hybrid (and hierarchical) planning

Thesis Summary

- Identification of sub classes of hybrid planning and their theoretical analysis (plan existence)
- Development of hybrid planning algorithm ($PANDA_2$) to solve all mentioned sub classes
- Development of TDGs (grounded and lifted) as basis for hybrid planning heuristics
- Development of first admissible heuristic for hybrid and POCL planning
- Results have been deployed in practice (assistant system)

Thesis Summary

- Identification of sub classes of hybrid planning and their theoretical analysis (plan existence)
- Development of hybrid planning algorithm (PANDA_2) to solve all mentioned sub classes
- Development of TDGs (grounded and lifted) as basis for hybrid planning heuristics
- Development of first admissible heuristic for hybrid and POCL planning
- Results have been deployed in practice (assistant system)

Thesis Summary

- Identification of sub classes of hybrid planning and their theoretical analysis (plan existence)
- Development of hybrid planning algorithm (PANDA$_2$) to solve all mentioned sub classes
- Development of TDGs (grounded and lifted) as basis for hybrid planning heuristics
- Development of first admissible heuristic for hybrid and POCL planning
- Results have been deployed in practice (assistant system)

Thesis Summary

- Identification of sub classes of hybrid planning and their theoretical analysis (plan existence)
- Development of hybrid planning algorithm (PANDA_2) to solve all mentioned sub classes
- Development of TDGs (grounded and lifted) as basis for hybrid planning heuristics
- Development of first admissible heuristic for hybrid and POCL planning
- Results have been deployed in practice (assistant system)

Thesis Summary

- Identification of sub classes of hybrid planning and their theoretical analysis (plan existence)
- Development of hybrid planning algorithm (PANDA_2) to solve all mentioned sub classes
- Development of TDGs (grounded and lifted) as basis for hybrid planning heuristics
- Development of first admissible heuristic for hybrid and POCL planning
- Results have been deployed in practice (assistant system)

Formal Problem Description

Definition (HTN/TIHTN planning domain, propositional)

A *Planning Domain* $\mathcal{D} = (V, N_C, N_P, \gamma, M)$ consists of:

- V, finitely many *state variables*,
- N_C, finitely many *abstract task names*,
- N_P, finitely many *primitive task names*,
- $\gamma : N_P \to A$, bijective function, where A are *actions*,
- $M \subseteq N_C \times P_{N_C \cup N_P}$, finitely many *methods*.

Definition (state)

A *state* is a set $s \in 2^V$.

Formal Problem Description

Definition (HTN/TIHTN planning domain, propositional)

A *Planning Domain* $\mathcal{D} = (V, N_C, N_P, \gamma, M)$ consists of:

- V, finitely many *state variables*,
- N_C, finitely many *abstract task names*,
- N_P, finitely many *primitive task names*,
- $\gamma : N_P \to A$, bijective function, where A are *actions*,
- $M \subseteq N_C \times P_{N_C \cup N_P}$, finitely many *methods*.

Formal Problem Description

Definition (HTN/TIHTN planning domain, propositional)

A *Planning Domain* $\mathcal{D} = (V, N_C, N_P, \gamma, M)$ consists of:

- V, finitely many *state variables*,
- N_C, finitely many *abstract task names*,
- N_P, finitely many *primitive task names*,
- $\gamma : N_P \to A$, bijective function, where A are *actions*,
- $M \subseteq N_C \times P_{N_C \cup N_P}$, finitely many *methods*.

Definition (action)

An *action* $a = (prec^+, prec^-, eff^+, eff^-)$ consists of:

- $prec^+, prec^- \subseteq V$, the *preconditions* of a and
- $eff^+, eff^- \subseteq V$, the *effects* of a.

Formal Problem Description

Definition (HTN/TIHTN planning domain, propositional)

A *Planning Domain* $\mathcal{D} = (V, N_C, N_P, \gamma, M)$ consists of:

- V, finitely many *state variables*,
- N_C, finitely many *abstract task names*,
- N_P, finitely many *primitive task names*,
- $\gamma : N_P \rightarrow A$, bijective function, where A are *actions*,
- $M \subseteq N_C \times P_{N_C \cup N_P}$, finitely many *methods*.

Definition (plan)

A *plan* $P = (L, \prec, \alpha)$ consists of:

- the labels L,
- the partial order $\prec \subseteq L \times L$, and
- the labeling function $\alpha : L \rightarrow N_C \cup N_P$.

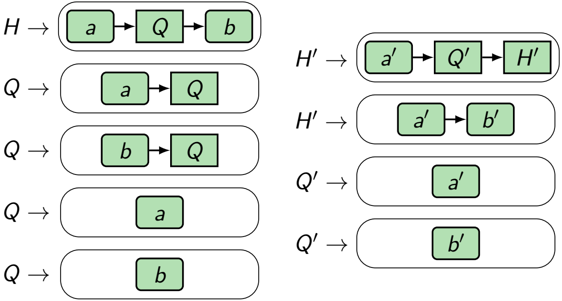
Formal Problem Description

Definition (planning problem)

A *planning problem* $\mathcal{P} = (\mathcal{D}, s_{\mathrm{init}}, P_{\mathrm{init}}, g)$ consists of:

- $\mathcal{D}$, the *planning domain*,
- $s_{\mathrm{init}} \in 2^V$, the *initial state*,
- $P_{\mathrm{init}} \subseteq P_{N_C \cup N_P}$, the *initial plan*, and
- $g = (g^+, g^-)$ with $g^+, g^- \subseteq V$, the *goal description*.

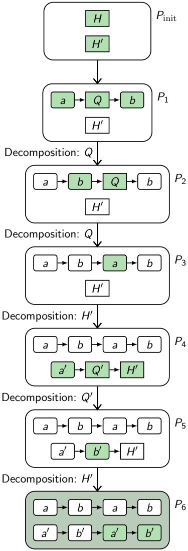
HTN Planning and Formal Grammars

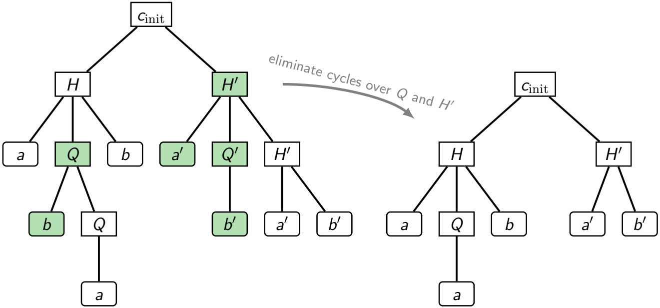


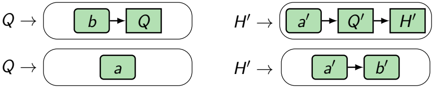
Grammars production rules can be expressed via methods.

Höller et al. *Language Classification of Hierarchical Planning Problems*. ECAI '14.

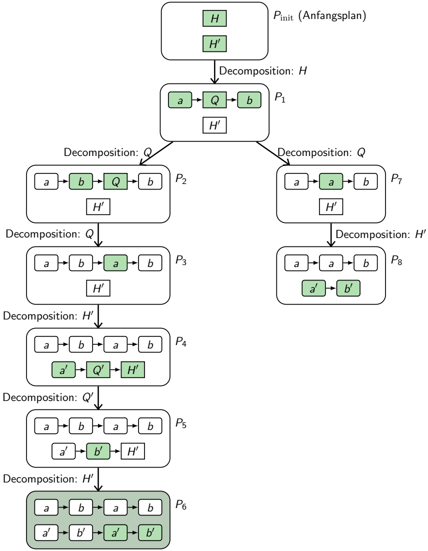
TIHTN Planning — Example







HTN Planning — Example



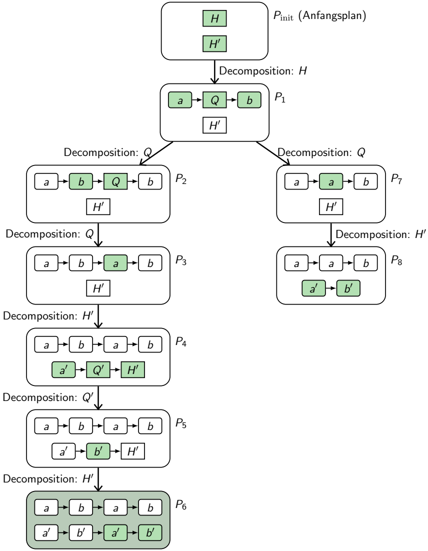
Notation:

row 1: words of grammar G
row 2: words of grammar G'

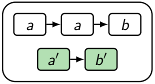
green actions: just inserted

green plans: solutions

HTN Planning — Example

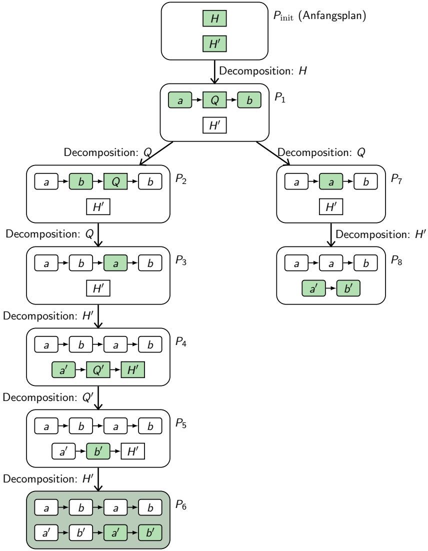


Plan P_8 ist no solution!

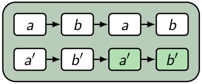


$aab \in L(G) = a(a|b)^+b$
$a'b' \in L(G') = (a'(a'|b'))^* a'b'$,
but $aab \not\cong a'b'$

HTN Planning — Example

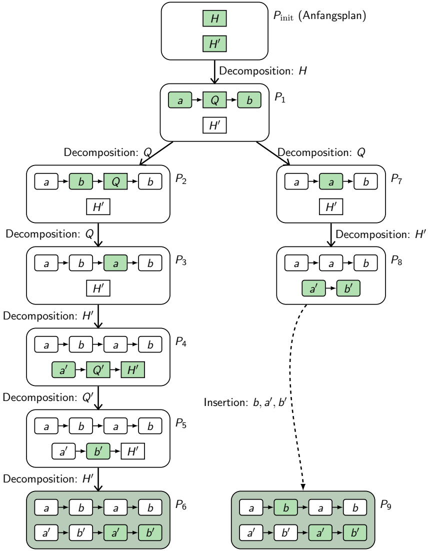


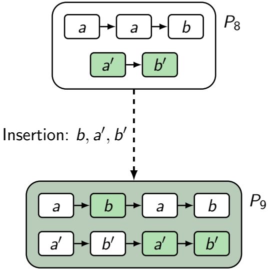
Plan P_6 is a solution!



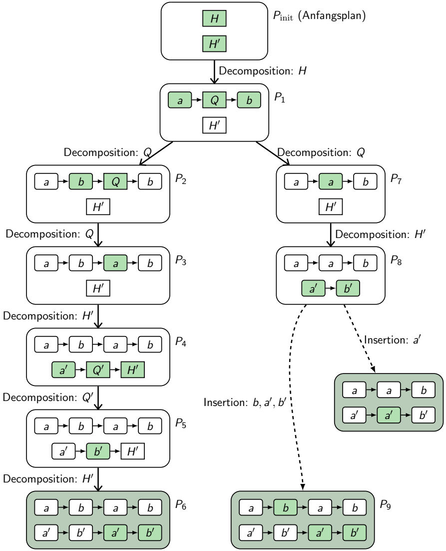
$abab \in L(G) = a(a|b)^+ b$
$a'b'a'b' \in L(G') = (a'(a'|b'))^* a'b'$,
and $abab \cong a'b'a'b'$

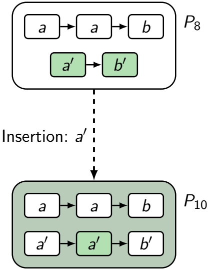
TIHTN Planning — Example





TIHTN Planning — Example





HTN Planning

Theorem: HTN Planning is undecidable.

Proof Idea:
Reduction of grammar intersection problem to HTN planning:
Given two context-free grammars G and G',
is there a word in $L(G) \cap L(G')$? $\square$

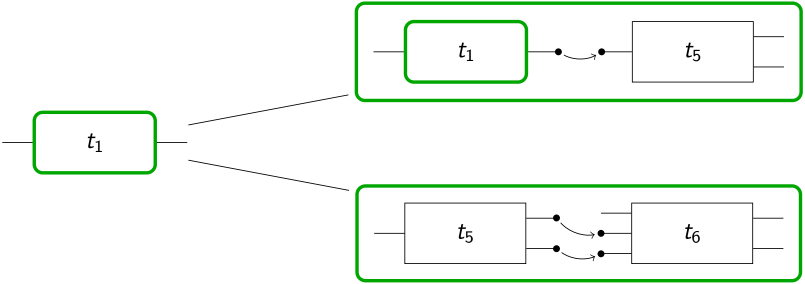
*Erol et al. *Complexity results for HTN planning*. Annals of Mathematics and Artificial Intelligence '96.

Geier und Bercher. *On the Decidability of HTN Planning with Task Insertion*. IJCAI '11.

TIHTN Planning

Theorem: HTN Planning with Task Insertion is decidable.

Proof Idea:



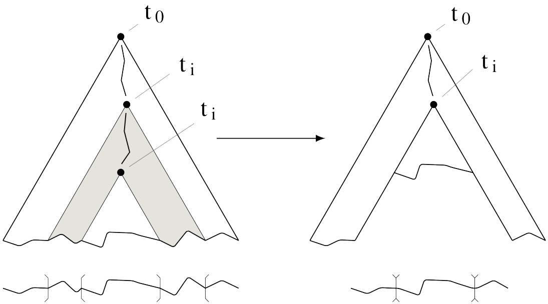
Geier und Bercher. *On the Decidability of HTN Planning with Task Insertion*. IJCAI '11.

Alford, Bercher, und Aha. *Tight Bounds for HTN Planning with Task Insertion*. IJCAI '15.

TIHTN Planning

Theorem: HTN Planning with Task Insertion is decidable.

Proof Idea:



Geier und Bercher. *On the Decidability of HTN Planning with Task Insertion*. IJCAI '11.

Alford, Bercher, und Aha. *Tight Bounds for HTN Planning with Task Insertion*. IJCAI '15.

PO and POCL Problems: No Relaxation

Theorem

PO and POCL problems are PSPACE complete.

Proof Idea

- *Hardness:* Both problem classes are generalizations of classical planning, which is PSPACE complete
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is PSPACE complete
  - Solutions to sub problems can be combined to complete solution □

PO and POCL Problems: No Relaxation

Theorem

PO and POCL problems are PSPACE complete.

Proof Idea

- *Hardness:* Both problem classes are generalizations of classical planning, which is PSPACE complete
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is PSPACE complete
  - Solutions to sub problems can be combined to complete solution □

PO and POCL Problems: No Relaxation

Theorem

PO and POCL problems are PSPACE complete.

Proof Idea

- *Hardness:* Both problem classes are generalizations of classical planning, which is PSPACE complete
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is PSPACE complete
  - Solutions to sub problems can be combined to complete solution □

PO and POCL Problems: No Relaxation

Theorem

PO and POCL problems are PSPACE complete.

Proof Idea

- *Hardness:* Both problem classes are generalizations of classical planning, which is PSPACE complete
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is PSPACE complete
  - Solutions to sub problems can be combined to complete solution □

PO and POCL Problems: No Relaxation

Theorem

PO and POCL problems are PSPACE complete.

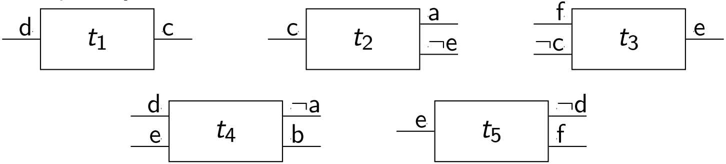
Proof Idea

- *Hardness:* Both problem classes are generalizations of classical planning, which is PSPACE complete
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is PSPACE complete
  - Solutions to sub problems can be combined to complete solution □

Excursion: Delete-Relaxation

Idee

Complexity reduction via delete-relaxation

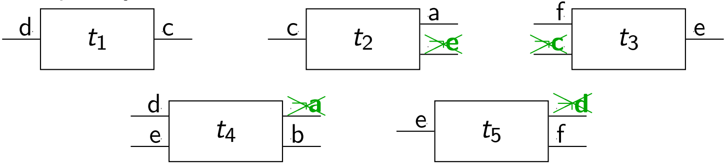


classical planning:

Excursion: Delete-Relaxation

Idee

Complexity reduction via delete-relaxation



classical planning: delete relaxation reduces complexity from PSPACE complete to NP or P:

- positive effects, positive preconditions: P
- positive effects, arbitrary preconditions: NP complete

PO and POCL Problems: Complete Delete-Relaxation

Theorem

Completely delete-relaxed PO and POCL problems are:

- in P (for positive preconditions)
- NP complete (for arbitrary preconditions)

Proof Idea (positive preconditions; in P)

Repeat until solution is found or fix point is reached:

- Apply all applicable actions in the domain
- Apply all applicable actions in the plan □

PO and POCL Problems: Complete Delete-Relaxation

Theorem

Completely delete-relaxed PO and POCL problems are:

- in P (for positive preconditions)
- NP complete (for arbitrary preconditions)

Proof Idea (arbitrary preconditions; NP complete)

- *Hardness:* problem class is generalization of the analog case in classical planning
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is NP complete
  - Solutions to sub problems can be combined to complete solution

PO and POCL Problems: Complete Delete-Relaxation

Theorem

Completely delete-relaxed PO and POCL problems are:

- in P (for positive preconditions)
- NP complete (for arbitrary preconditions)

Proof Idea (arbitrary preconditions; NP complete)

- *Hardness:* problem class is generalization of the analog case in classical planning
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is NP complete
  - Solutions to sub problems can be combined to complete solution □

PO and POCL Problems: Complete Delete-Relaxation

Theorem

Completely delete-relaxed PO and POCL problems are:

- in P (for positive preconditions)
- NP complete (for arbitrary preconditions)

Proof Idea (arbitrary preconditions; NP complete)

- *Hardness:* problem class is generalization of the analog case in classical planning
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is NP complete
  - Solutions to sub problems can be combined to complete solution □

PO and POCL Problems: Complete Delete-Relaxation

Theorem

Completely delete-relaxed PO and POCL problems are:

- in P (for positive preconditions)
- NP complete (for arbitrary preconditions)

Proof Idea (arbitrary preconditions; NP complete)

- *Hardness:* problem class is generalization of the analog case in classical planning
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is NP complete
  - Solutions to sub problems can be combined to complete solution □

PO and POCL Problems: Complete Delete-Relaxation

Theorem

Completely delete-relaxed PO and POCL problems are:

- in P (for positive preconditions)
- NP complete (for arbitrary preconditions)

Proof Idea (arbitrary preconditions; NP complete)

- *Hardness:* problem class is generalization of the analog case in classical planning
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is NP complete
  - Solutions to sub problems can be combined to complete solution □

PO and POCL Problems: Complete Delete-Relaxation

Theorem

Completely delete-relaxed PO and POCL problems are:

- in P (for positive preconditions)
- NP complete (for arbitrary preconditions)

Proof Idea (arbitrary preconditions; NP complete)

- *Hardness:* problem class is generalization of the analog case in classical planning
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is NP complete
  - Solutions to sub problems can be combined to complete solution □

PO and POCL Problems: Partial Delete-Relaxation

Delete-relax only actions in the domain, not those of the plan

Theorem

Partially delete-relaxed PO und POCL problems are NP-complete

- *Hardness:* problem class is generalization of the analog case in classical planning
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is NP complete
  - Solutions to sub problems can be combined to complete solution □

Bercher et al. *On Delete Relaxation in Partial-Order Causal-Link Planning*. ICTAI '13.

PO and POCL Problems: Partial Delete-Relaxation

Delete-relax only actions in the domain, not those of the plan

Theorem

Partially delete-relaxed PO und POCL problems are NP-complete

- *Hardness:* problem class is generalization of the analog case in classical planning
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is NP complete
  - Solutions to sub problems can be combined to complete solution □

Bercher et al. *On Delete Relaxation in Partial-Order Causal-Link Planning*. ICTAI '13.

PO and POCL Problems: Partial Delete-Relaxation

Delete-relax only actions in the domain, not those of the plan

Theorem

Partially delete-relaxed PO und POCL problems are NP-complete

- *Hardness:* problem class is generalization of the analog case in classical planning
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is NP complete
  - Solutions to sub problems can be combined to complete solution □

Bercher et al. *On Delete Relaxation in Partial-Order Causal-Link Planning*. ICTAI '13.

PO and POCL Problems: Partial Delete-Relaxation

Delete-relax only actions in the domain, not those of the plan

Theorem

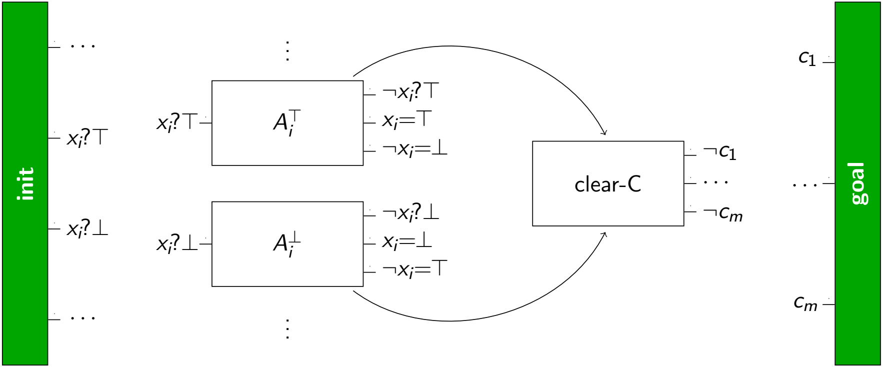
Partially delete-relaxed PO und POCL problems are NP-complete

- *Hardness:* problem class is generalization of the analog case in classical planning
- *Membership:*
  - Guess linearization of actions
  - Thus, we have a linear number of sub problems
  - Each sub problem is NP complete
  - Solutions to sub problems can be combined to complete solution □

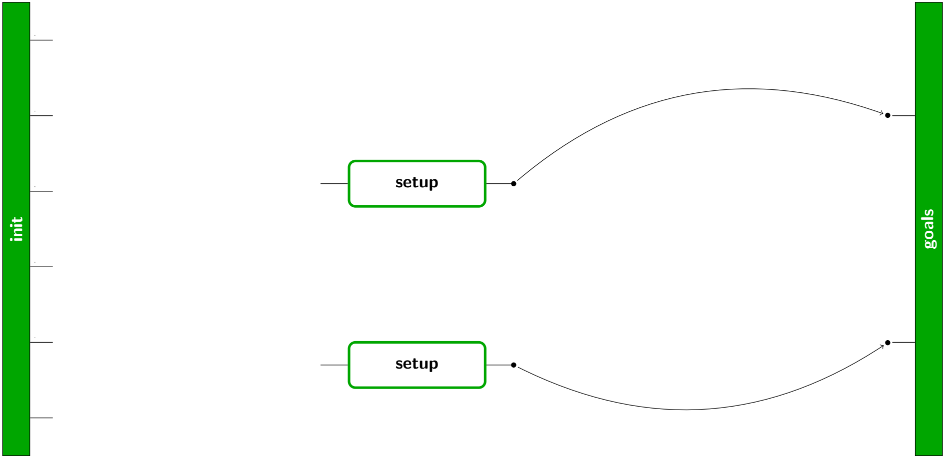
Bercher et al. *On Delete Relaxation in Partial-Order Causal-Link Planning*. ICTAI '13.

PO and POCL Problems: Partial Delete-Relaxation

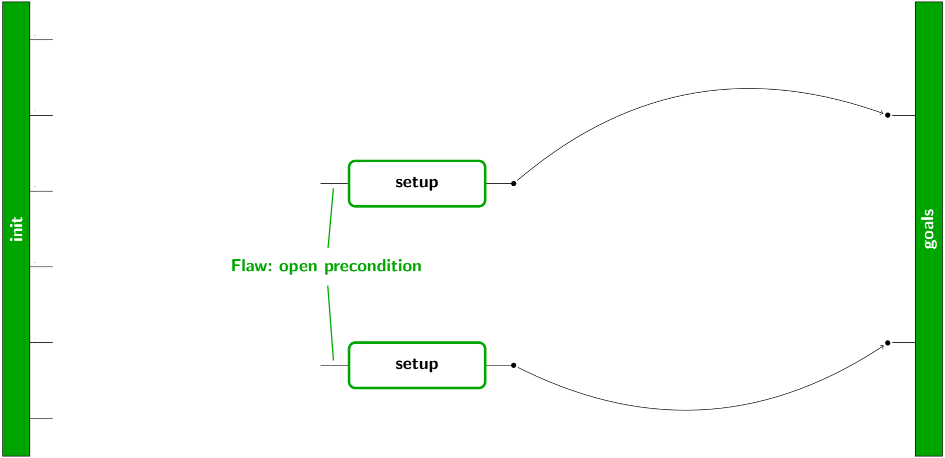
$x^1, \ldots, x^n$ are Boolen variables. $c^1, \ldots, c^m$ are clauses. Each c^j is a disjunction. For each $x^i \in c^j$ there is an action $\langle \{x_i = \top\}, \{c_j\}, \emptyset \rangle$.



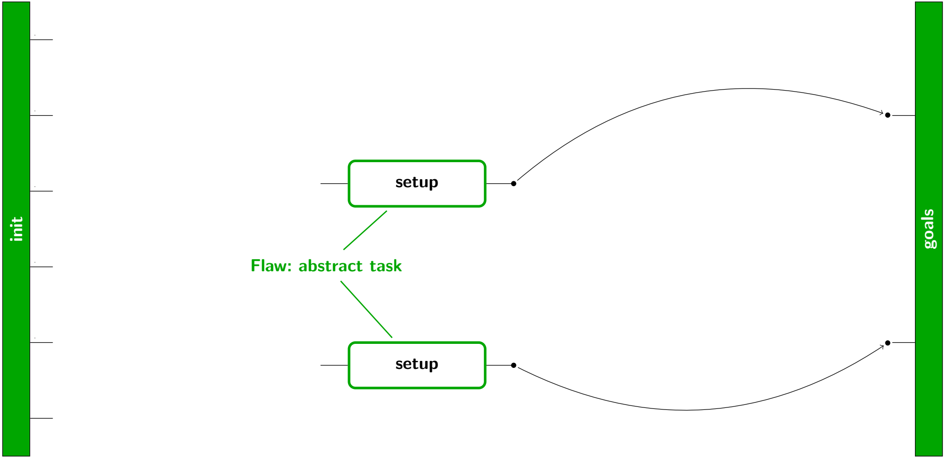
Seach Algorithm PANDA$_2$



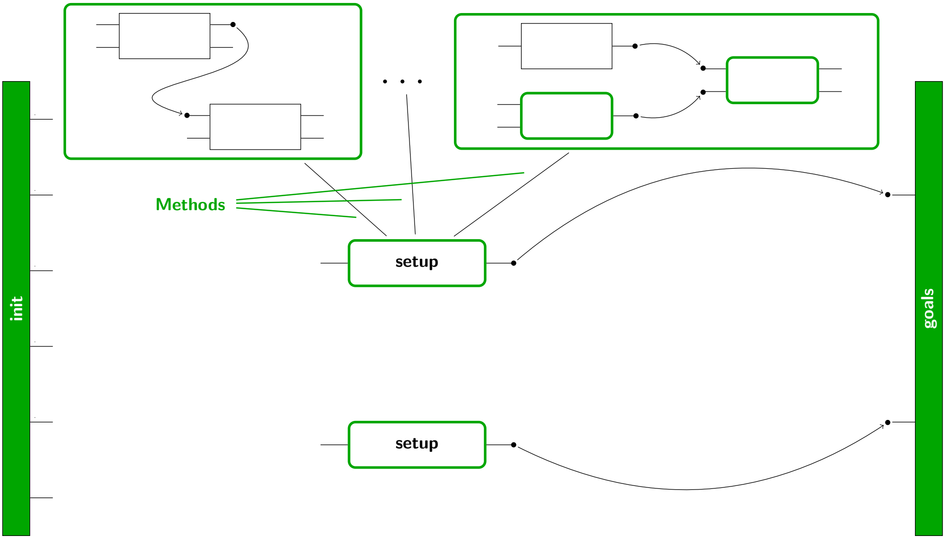
Seach Algorithm PANDA$_2$



Seach Algorithm PANDA$_2$



Seach Algorithm PANDA$_2$



Seach Algorithm PANDA$_2$

Possible flaws and their modifications:

Flaw	**Modification**
Abstract Task	Decomposition
Open Precondition	Causal Link insertion via: • existing action • action from domain (action insertion) • action from a method (decomposition)
Causal Threat	• Insertion of an ordering constraint • Insertion of a variable binding • Decomposition

Heuristik SampleFF – Evaluation

Evaluation with planning problems taken from IPC 1 to IPC 5.

Comparison of PANDA$_2$ with A* and:

- The **Add heuristik für POCL planning** (Younes & Simmons)
- The **Relax heuristic** (Nguyen & Kambhampati)
- 12 variants of **SampleFF heuristic**

POCL Heuristic SampleFF – Evaluation

results:

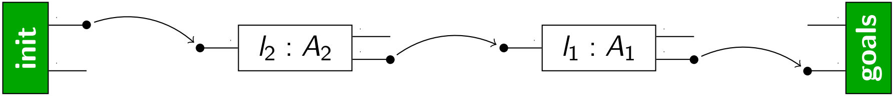
- Number of solved problem instances in 15 Minutes (of 446)
  Add: 292, Relax: 194, SampleFF: between 127 and 187
- For one of the unsolvable problems, its unsolvability could only be proved by SampleFF
- there is no relaxed solution among 30 samples for 36 % of all search nodes

POCL Heuristic SampleFF – Evaluation

Domain	n	**Add**	**Relax**	**SampleFF**											
				front: $\perp$ end: $\perp$				front: $\perp$ end: $\top$				front: $\top$ end: $\top$			
				1	3	10	30	1	3	10	30	1	3	10	30
grid	5	0	0	0	0	0	0	0	0	0	0	1	**1**	**1**	**1**
gripper	20	14	7	1	1	1	1	1	2	1	1	2	**3**	**3**	2
logistics	20	12	7	**8**	5	6	6	6	7	6	5	0	0	1	1
movie	30	30	30	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
mystery	20	8	9	10	11	9	9	10	11	10	9	**12**	**12**	11	11
mystery-prime	20	3	3	3	4	**6**	5	5	4	4	4	**6**	**6**	**6**	**6**
blocks	21	4	7	5	5	**6**	**6**	4	3	3	3	5	3	2	0
logistics	28	28	27	22	23	23	**24**	21	19	20	21	15	13	14	15
miconic	100	100	39	**40**	**40**	37	35	39	41	37	32	15	16	18	20
depot	22	2	1	1	1	1	1	0	1	1	1	2	2	**3**	2
driverlog	20	7	7	**11**	9	10	9	9	10	9	8	8	7	9	7
rover	20	20	19	13	14	**15**	**15**	11	11	12	11	7	9	9	9
zeno-travel	10	4	3	3	**5**	4	**5**	4	3	4	4	1	1	1	1
airport	20	18	9	10	**11**	**11**	**11**	7	10	10	10	7	8	6	4
pipesworld-noTankage	10	8	2	2	3	**5**	3	2	1	2	1	1	4	3	**5**
pipesworld-Tankage	10	1	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
satellite	20	16	7	**7**	5	6	6	5	5	7	5	1	2	3	3
pipesworld	10	1	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
storage	20	7	4	6	7	9	8	6	7	7	6	9	9	**10**	**10**
tpp	20	19	11	**8**	7	6	7	6	6	6	6	5	6	6	6
total	446	292	194	182	183	187	183	168	173	171	159	127	132	136	133

State-based Heuristics for POCL Planning

Let $\mathcal{P} = \langle \mathcal{D}, s_{init}, g \rangle$ mit $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$ and P be given by:

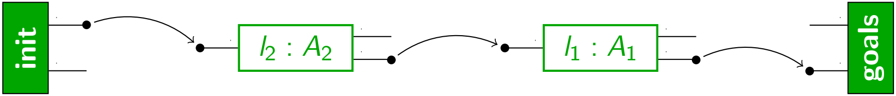


P is encoded in $\mathcal{P}' = \langle \mathcal{D}', s'_{init}, g' \rangle$ with $\mathcal{D}' = \langle \mathcal{V}', \mathcal{A}' \rangle$:

- $\mathcal{V}' := \mathcal{V} \cup \{l_1, l_2\}$
- $\mathcal{A}' := \mathcal{A} \cup \{enc(l_1 : A_1), enc(l_2 : A_2)\}$ with
  - $enc(l_1 : A_1) = \langle prec(A_1) \wedge \neg l_1 \wedge l_2, eff(A_1) \wedge l_1 \rangle$
  - $enc(l_2 : A_2) = \langle prec(A_2) \wedge \neg l_2, eff(A_2) \wedge l_2 \rangle$
- $s'_{\text{init}} := s_{\text{init}}$
- $g' := g \cup \{l_1, l_2\}$

State-based Heuristics for POCL Planning

Let $\mathcal{P} = \langle \mathcal{D}, s_{init}, g \rangle$ mit $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$ and P be given by:

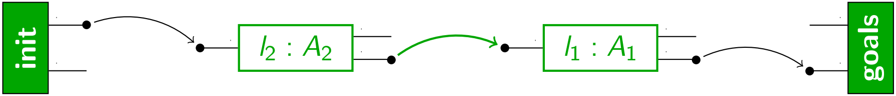


P is encoded in $\mathcal{P}' = \langle \mathcal{D}', s'_{init}, g' \rangle$ with $\mathcal{D}' = \langle \mathcal{V}', \mathcal{A}' \rangle$:

- $\mathcal{V}' := \mathcal{V} \cup \{l_1, l_2\}$
- $\mathcal{A}' := \mathcal{A} \cup \{enc(l_1 : A_1), enc(l_2 : A_2)\}$ with
  - $enc(l_1 : A_1) = \langle prec(A_1) \wedge \neg \mathbf{l_1} \wedge l_2, eff(A_1) \wedge \mathbf{l_1} \rangle$
  - $enc(l_2 : A_2) = \langle prec(A_2) \wedge \neg \mathbf{l_2}, eff(A_2) \wedge \mathbf{l_2} \rangle$
- $s'_{\text{init}} := s_{\text{init}}$
- $g' := g \cup \{l_1, l_2\}$

State-based Heuristics for POCL Planning

Let $\mathcal{P} = \langle \mathcal{D}, s_{init}, g \rangle$ mit $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$ and P be given by:

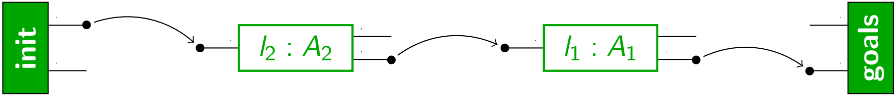


P is encoded in $\mathcal{P}' = \langle \mathcal{D}', s'_{init}, g' \rangle$ with $\mathcal{D}' = \langle \mathcal{V}', \mathcal{A}' \rangle$:

- $\mathcal{V}' := \mathcal{V} \cup \{l_1, l_2\}$
- $\mathcal{A}' := \mathcal{A} \cup \{enc(l_1 : A_1), enc(l_2 : A_2)\}$ with
  - $enc(l_1 : A_1) = \langle prec(A_1) \wedge \neg l_1 \wedge \mathbf{l_2}, eff(A_1) \wedge l_1 \rangle$
  - $enc(l_2 : A_2) = \langle prec(A_2) \wedge \neg l_2, eff(A_2) \wedge \mathbf{l_2} \rangle$
- $s'_{\text{init}} := s_{\text{init}}$
- $g' := g \cup \{l_1, l_2\}$

State-based Heuristics for POCL Planning

Let $\mathcal{P} = \langle \mathcal{D}, s_{init}, g \rangle$ mit $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$ and P be given by:



P is encoded in $\mathcal{P}' = \langle \mathcal{D}', s'_{init}, g' \rangle$ with $\mathcal{D}' = \langle \mathcal{V}', \mathcal{A}' \rangle$:

- $\mathcal{V}' := \mathcal{V} \cup \{l_1, l_2\}$
- $\mathcal{A}' := \mathcal{A} \cup \{enc(l_1 : A_1), enc(l_2 : A_2)\}$ with
  - $enc(l_1 : A_1) = \langle prec(A_1) \wedge \neg l_1 \wedge l_2, eff(A_1) \wedge \mathbf{l_1} \rangle$
  - $enc(l_2 : A_2) = \langle prec(A_2) \wedge \neg l_2, eff(A_2) \wedge \mathbf{l_2} \rangle$
- $s'_{\mathrm{init}} := s_{\mathrm{init}}$
- $g' := g \cup \{\mathbf{l_1}, \mathbf{l_2}\}$

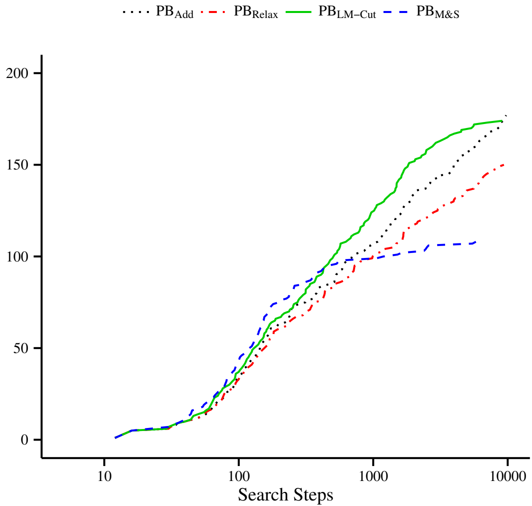
POCL Heuristics Based on State-based Heuristics – Eval.

Evaluation with planning problems taken from IPC 1 to IPC 5.

Comparison of PANDA_2 with A* and:

- The **Add heuristic for POCL planning** (Younes & Simmons)
- The **Relax heuristic** (Nguyen & Kambhampati)
- **Encoding + Lm-Cut** (Helmert & Domshlak)
- **Encoding + Merge & Shrink** (Helmert et al.)

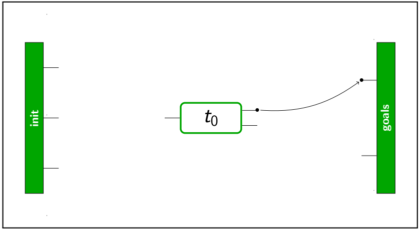
POCL Heuristics Based on State-based Heuristics – Eval.



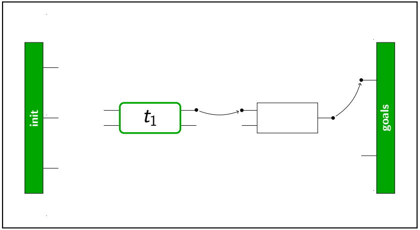
POCL Heuristics Based on State-based Heuristics – Eval.

Domain	n	PB_{Add}	PB_{Relax}	$PB_{LM\text{-}Cut}$	$PB_{M\&S}$	$SB_{LM\text{-}Cut}$	$SB_{M\&S}$
grid	5	**0**	**0**	**0**	**0**	2	2
gripper	20	14	**20**	1	1	20	8
logistics	20	**16**	15	6	0	16	1
movie	30	**30**	**30**	**30**	**30**	30	30
mystery	20	8	**10**	5	5	13	13
mystery-prime	20	3	**4**	2	1	12	12
blocks	21	2	3	3	**5**	21	21
logistics	28	**28**	**28**	27	5	28	15
miconic	100	**100**	53	65	29	100	68
depot	22	**2**	**2**	1	1	11	7
driverlog	20	7	**9**	3	3	15	12
freecell	20	**0**	**0**	**0**	**0**	6	6
rover	20	**20**	19	9	5	18	8
zeno-travel	20	4	4	3	**5**	16	13
airport	20	**18**	15	6	5	20	18
pipesworld-noTankage	20	**8**	5	1	1	18	19
pipesworld-Tankage	20	**1**	**1**	**1**	**1**	11	14
satellite	20	**18**	**18**	4	3	15	7
pipesworld	20	**1**	**1**	**1**	**1**	11	14
rover	20	**0**	**0**	**0**	**0**	18	8
storage	20	7	**9**	5	5	17	15
tpp	20	**9**	8	5	5	9	7

MME Heuristic for Hybrid Planning (Motivation)



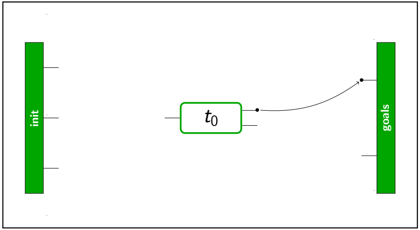
1 abstract tast
2 open precondition
$\geq$ 3 modifcations



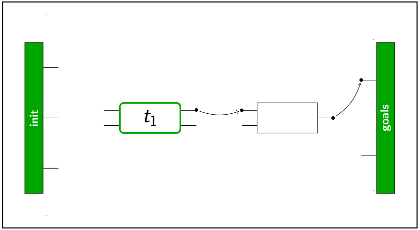
1 abstract task
4 open precondition
$\geq$ 5 modifications

... however, the sub-actions of the tasks t_0 and t_1 are ignored!

MME Heuristic for Hybrid Planning (Motivation)



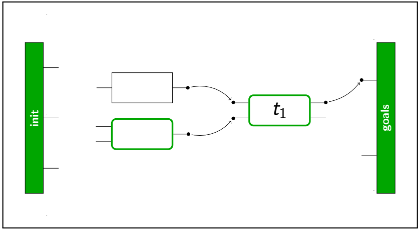
1 abstract tast
2 open precondition
$\geq$ 3 modifcations



1 abstract task
4 open precondition
$\geq$ 5 modifications

... however, the sub-actions of the tasks t_0 and t_1 are ignored!

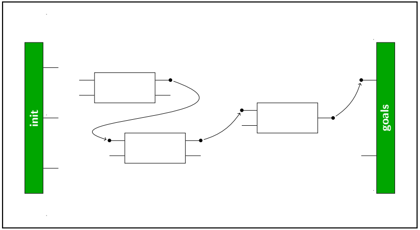
MME Heuristic for Hybrid Planning (Motivation)



2 abstract tasks

4 open preconditions

$\geq$ 6 modifications



0 abstract tasks

5 open preconditions

$\geq$ 5 modifications

... however, the sub-actions of the tasks t_0 and t_1 are ignored!

MME Heuristic for Hybrid Planning

Task node $n_t = \langle prec, eff \rangle$:

$$h_T(n_t) := \begin{cases} |prec^+(n_t)| + |prec^-(n_t)| & \text{if } n_t \text{ primitive} \\ 1 + \min\limits_{(n_t,n_m) \in E_{T \to M}} h_M(n_m) & \text{else} \end{cases}$$

Method node $n_m = (m, P)$ with $P = (PS, \prec, CL)$:

$$h_M(n_m) := \sum_{(n_m,n_t) \in E_{M \to T}} h_T(n_t) - |CL|$$

MME Heuristic for Hybrid Planning

Theoretical properties of MME heuristic:

- can be calculated in P
- first admissible heuristic for hierarchical and hybrid planning

MME Heuristic for Hybrid Planning – Evaluation

Hybrid planning domains:

- UM-Translog (logistics domain, original from 1995)
- Satellite (adapted IPC benchmark)
- SmartPhone (model of HTC smartphone)
- Woodworking (adapted IPC benchmark)

MME Heuristic for Hybrid Planning – Evaluation

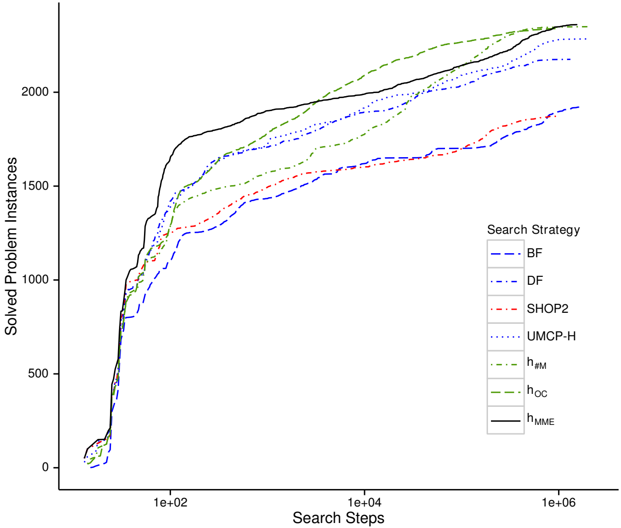
Baseline configurations:

- BF / DF (Breadth and Depth First search)
- SHOP2
- UMCP: BF, DF, heuristic version

Greedy suche with heuristics:

- $h_{\#M}$ (sum of modifications of all flaws)
- h_{OC} (number of open preconditions)
- h_{MME}

MME Heuristic for Hybrid Planning – Evaluation



MME Heuristic for Hybrid Planning – Evaluation