A Survey on Hierarchical Planning –
One Abstract Idea, Many Concrete Realizations

Pascal Bercher[1] and Ron Alford[2] and Daniel Höller[1]

[1]Institute of Artificial Intelligence, Ulm University, Germany
[2]MITRE, McLean, Virginia, USA

August 14, 2019

ulm university universität uulm

MITRE

Non-hierarchical Planning in a Nutshell

A classical problem description consists of:

- Given an initial state s_I,
- a portfolio of the system's available actions, and
- a goal description g.

Non-hierarchical Planning in a Nutshell

A classical problem description consists of:

- Given an initial state s_I,
- a portfolio of the system's available actions, and
- a goal description g.

What are we looking for?

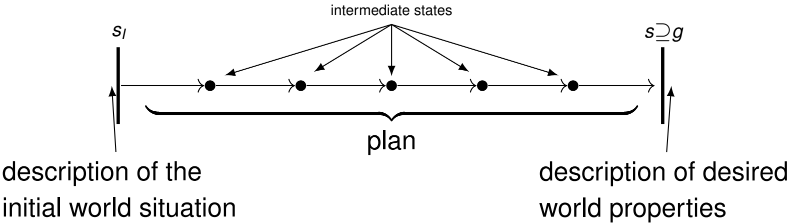
Non-hierarchical Planning in a Nutshell

A classical problem description consists of:

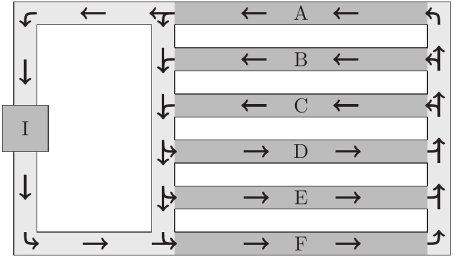
- Given an initial state s_I,
- a portfolio of the system's available actions, and
- a goal description g.

What are we looking for?

→ A *plan* that transforms s_I into g.



Planning Applications: Intelligent Factories



The Scanalyzer Domain

– automatic greenhouse logistic management

Pictures with kind permission from LemnaTec GmbH

Planning Applications: Intelligent Assistants

Sink devices:

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

Source devices:

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

Action Definition in the Home Theater Domain

Planning problems are usually defined in terms of a description language based on a first-order predicate logic.

- States are sets of (ground) propositions, e.g.,
  $s = \{HasPort(AMPLIFIER, HDMI),$
  $HasPort(AMPLIFIER, CINCH),$
  $HasPort(CABLE\_HDMI, HDMI)\}$

Action Definition in the Home Theater Domain

Planning problems are usually defined in terms of a description language based on a first-order predicate logic.

- States are sets of (ground) propositions, e.g.,
  $s = \{HasPort(AMPLIFIER, HDMI),$
  $HasPort(AMPLIFIER, CINCH),$
  $HasPort(CABLE\_HDMI, HDMI)\}$
- Actions are defined by preconditions and effects, e.g.,

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

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

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

  (Signal flow not shown for the sake of simplicity)

A Survey on *Hierarchical* Planning – Motivation

Hierarchical planning extends classical planning by hierarchies:

- on the tasks, which need to be decomposed, or
- on the state features, which need to be achieved (or decomposed)

A Survey on *Hierarchical* Planning – Motivation

Hierarchical planning extends classical planning by hierarchies:

- on the tasks, which need to be decomposed, or
- on the state features, which need to be achieved (or decomposed)

Why should we incorporate task or state hierarchies?

A Survey on *Hierarchical* Planning – Motivation

Hierarchical planning extends classical planning by hierarchies:

- on the tasks, which need to be decomposed, or
- on the state features, which need to be achieved (or decomposed)

Why should we incorporate task or state hierarchies? E.g., because:

- Domain experts might want to model it that way.
- Plans can be presented *more abstract* by relying on task hierarchies.
- Standard solutions or search advice can be modeled.
- More control on the generated solutions using structures similar to formal grammars.

A Survey on *Hierarchical* Planning – Motivation

Hierarchical planning extends classical planning by hierarchies:

- on the tasks, which need to be decomposed, or
- on the state features, which need to be achieved (or decomposed)

Why should we incorporate task or state hierarchies? E.g., because:

- Domain experts might want to model it that way.
- Plans can be presented *more abstract* by relying on task hierarchies.
- Standard solutions or search advice can be modeled.
- More control on the generated solutions using structures similar to formal grammars.

⇝ Our focus: Survey of recent hierarchical problem classes.

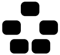
Problem Class: Hierarchical Task Network (HTN) Planning

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

- V a set of state variables.

Problem Class: Hierarchical Task Network (HTN) Planning

primitive
tasks



compound
tasks

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

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \rightarrow (2^V)^3$ the task name mapping.
- C a set of compound task names.

Problem Class: Hierarchical Task Network (HTN) Planning

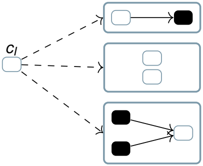
$$\mathcal{P} = (V, P, \delta, C, M, s_I, c_I)$$

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \to (2^V)^3$ 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,

Problem Class: Hierarchical Task Network (HTN) Planning



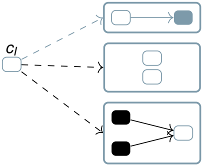
$$\mathcal{P} = (V, P, \delta, C, M, s_I, c_I)$$

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \to (2^V)^3$ 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

Problem Class: Hierarchical Task Network (HTN) Planning



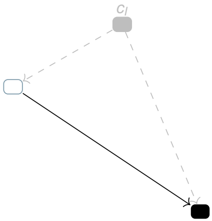
$\mathcal{P} = (V, P, \delta, C, M, s_I, c_I)$

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \rightarrow (2^V)^3$ 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

Problem Class: Hierarchical Task Network (HTN) Planning



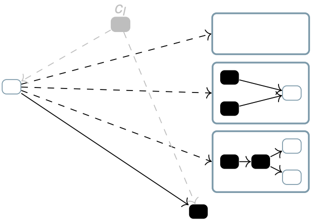
$$\mathcal{P} = (V, P, \delta, C, M, s_I, c_I)$$

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \rightarrow (2^V)^3$ 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

Problem Class: Hierarchical Task Network (HTN) Planning



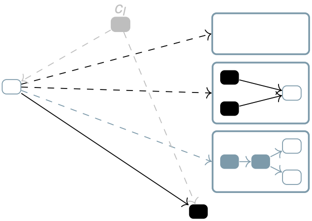
$$\mathcal{P} = (V, P, \delta, C, M, s_I, c_I)$$

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \rightarrow (2^V)^3$ 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

Problem Class: Hierarchical Task Network (HTN) Planning



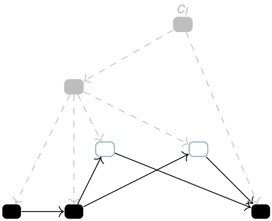
$$\mathcal{P} = (V, P, \delta, C, M, s_I, c_I)$$

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \rightarrow (2^V)^3$ 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

Problem Class: Hierarchical Task Network (HTN) Planning



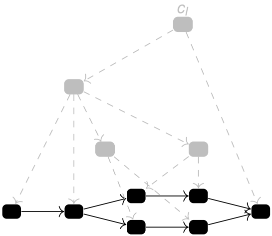
$$\mathcal{P} = (V, P, \delta, C, M, s_I, c_I)$$

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \to (2^V)^3$ 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

Problem Class: Hierarchical Task Network (HTN) Planning



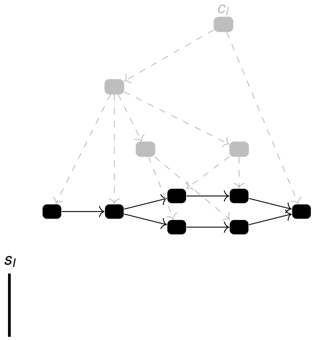
$$\mathcal{P} = (V, P, \delta, C, M, s_I, c_I)$$

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \rightarrow (2^V)^3$ 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

Problem Class: Hierarchical Task Network (HTN) Planning



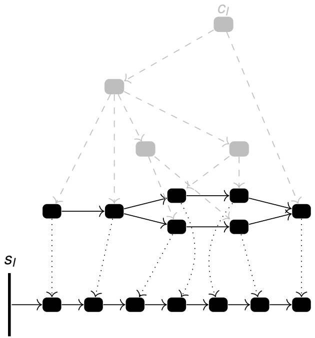
$\mathcal{P} = (V, P, \delta, C, M, s_I, c_I)$

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \rightarrow (2^V)^3$ 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.
- $s_I \subseteq V$ the initial state.

A solution task network *tn* must:

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

Problem Class: Hierarchical Task Network (HTN) Planning



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

- V a set of state variables.
- P a set of primitive task names.
- $\delta : P \to (2^V)^3$ 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.
- $s_I \subseteq V$ the initial state.

A solution task network *tn* must:

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

HTN Planning Problems – Some Notes

- In contrast to classical planning, there is no state-based goal.
- The objective is to decompose the initial abstract tasks into an executable primitive action sequence.

HTN Planning Problems – Some Notes

- In contrast to classical planning, there is no state-based goal.
- The objective is to decompose the initial abstract tasks into an executable primitive action sequence.
- In contrast to classical planning, the planner is *not* allowed to insert actions arbitrarily.

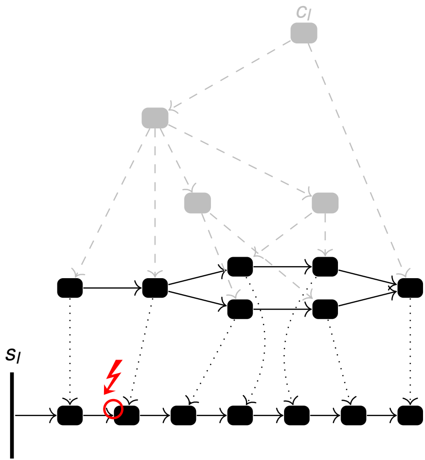
HTN Planning Problems – Some Notes

- In contrast to classical planning, there is no state-based goal.
- The objective is to decompose the initial abstract tasks into an executable primitive action sequence.
- In contrast to classical planning, the planner is *not* allowed to insert actions arbitrarily.
- Still, every classical problem can be encoded as an HTN problem, but HTN planning is more expressive.
- In the general case, it is undecidable to determine whether an HTN problem has a solution.

HTN Planning Problems – Some Notes

- In contrast to classical planning, there is no state-based goal.
- The objective is to decompose the initial abstract tasks into an executable primitive action sequence.
- In contrast to classical planning, the planner is *not* allowed to insert actions arbitrarily.
- Still, every classical problem can be encoded as an HTN problem, but HTN planning is more expressive.
- In the general case, it is undecidable to determine whether an HTN problem has a solution.
- The paper also – *very briefly* – explains all standard algorithms for solving HTN problems.

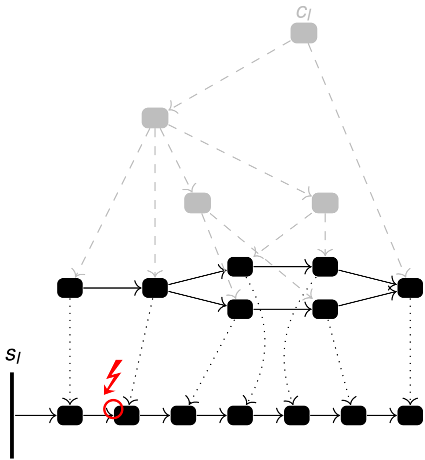
Optional Feature: Task Insertion



- Allowing *Task Insertion* means that actions can be inserted arbitrarily.

What to do when plans are not executable?

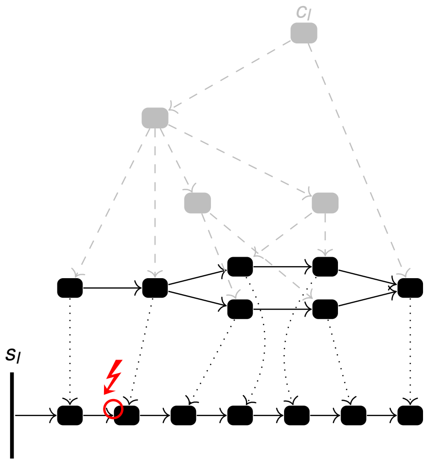
Optional Feature: Task Insertion



- Allowing *Task Insertion* means that actions can be inserted arbitrarily.
- This gives greater flexibility: certain parts do not need to be modeled via the hierarchy.

What to do when plans are not executable?

Optional Feature: Task Insertion

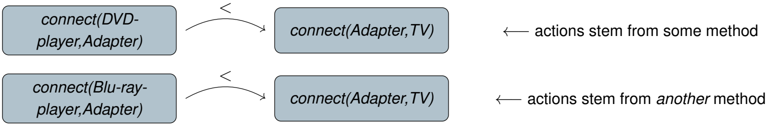


- Allowing *Task Insertion* means that actions can be inserted arbitrarily.
- This gives greater flexibility: certain parts do not need to be modeled via the hierarchy.
- Allowing Task Insertion in HTN planning results in the problem class called *TIHTN planning*.

What to do when plans are not executable?

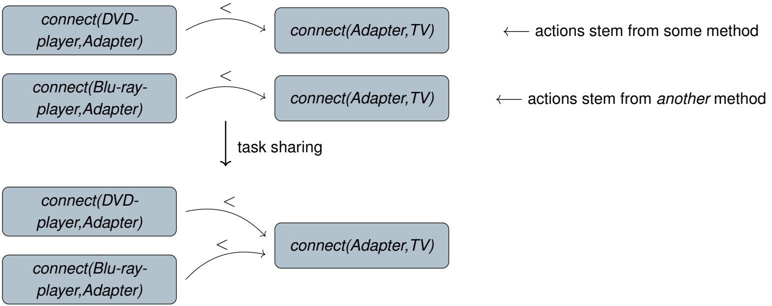
Optional Feature: Task Sharing

Task Sharing allows to eliminate duplicates that might just be modeling artifacts:



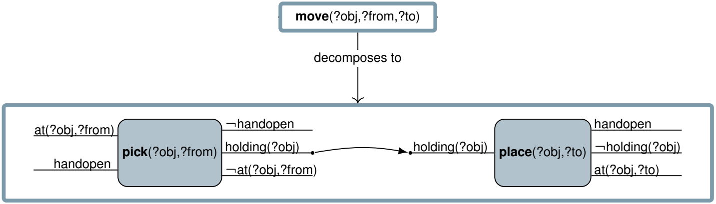
Optional Feature: Task Sharing

Task Sharing allows to eliminate duplicates that might just be modeling artifacts:



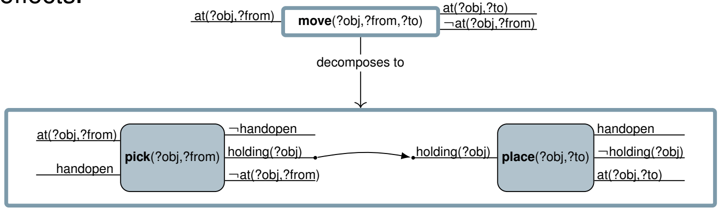
Problem Class: Decompositional and Hybrid Planning

- In many formalizations, abstract tasks show preconditions and effects.



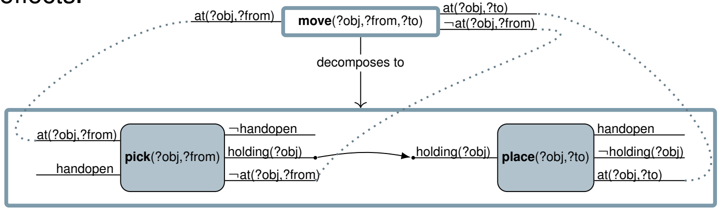
Problem Class: Decompositional and Hybrid Planning

- In many formalizations, abstract tasks show preconditions and effects.



Problem Class: Decompositional and Hybrid Planning

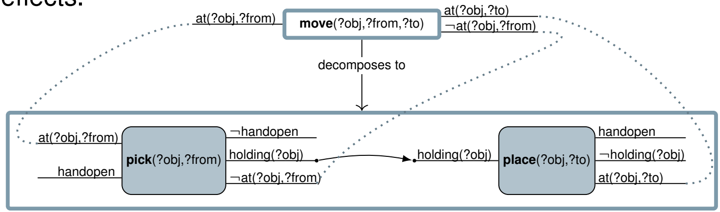
- In many formalizations, abstract tasks show preconditions and effects.



- They are e.g. used for checking implementation or legality criteria of methods,

Problem Class: Decompositional and Hybrid Planning

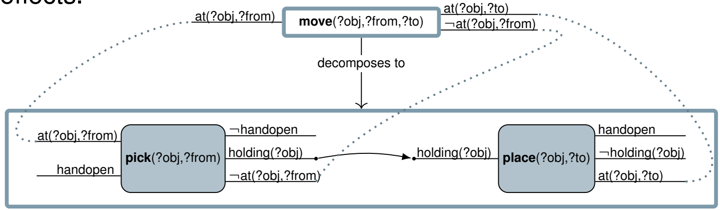
- In many formalizations, abstract tasks show preconditions and effects.



- They are e.g. used for checking implementation or legality criteria of methods,
- or to insert abstract tasks during planning.

Problem Class: Decompositional and Hybrid Planning

- In many formalizations, abstract tasks show preconditions and effects.



- They are e.g. used for checking implementation or legality criteria of methods,
- or to insert abstract tasks during planning.
- Some of these formalizations feature an initial task network, others do not.

Problem Class: HGN Planning

- Instead of restricting *what to do* (i.e., which action to perform), we can restrict *what to achieve* (i.e., which state features).

Problem Class: HGN Planning

- Instead of restricting *what to do* (i.e., which action to perform), we can restrict *what to achieve* (i.e., which state features).
- For this, *Hierarchical Goal Network (HGN) planning* defines a hierarchy on *goals*.

Problem Class: HGN Planning

- Instead of restricting *what to do* (i.e., which action to perform), we can restrict *what to achieve* (i.e., which state features).
- For this, *Hierarchical Goal Network (HGN) planning* defines a hierarchy on *goals*.
- Methods thus refine goals rather than tasks.

Problem Class: HGN Planning

- Instead of restricting *what to do* (i.e., which action to perform), we can restrict *what to achieve* (i.e., which state features).
- For this, *Hierarchical Goal Network (HGN) planning* defines a hierarchy on *goals*.
- Methods thus refine goals rather than tasks.
- Actions can be applied to a current state only if they contribute towards a goal without predecessors.

Problem Class: HGN Planning

- Instead of restricting *what to do* (i.e., which action to perform), we can restrict *what to achieve* (i.e., which state features).
- For this, *Hierarchical Goal Network (HGN) planning* defines a hierarchy on *goals*.
- Methods thus refine goals rather than tasks.
- Actions can be applied to a current state only if they contribute towards a goal without predecessors.
- Goals are achieved if they hold in the current state.

Core Contributions of the Survey

- We provided a survey on *hierarchical planning*.

Core Contributions of the Survey

- We provided a survey on *hierarchical planning*.
- We focused on various problem classes, most of which were developed or formalized within the last 10 years.

Core Contributions of the Survey

- We provided a survey on *hierarchical planning*.
- We focused on various problem classes, most of which were developed or formalized within the last 10 years.
- We discussed the impact of the different problem classes/features on the set of solutions.

Core Contributions of the Survey

- We provided a survey on *hierarchical planning*.
- We focused on various problem classes, most of which were developed or formalized within the last 10 years.
- We discussed the impact of the different problem classes/features on the set of solutions.
- We provided a brief explanation of all standard algorithms for solving HTN problems.