Lecture *Hierarchical Planning*

Chapter:
Further Hierarchical Planning Formalisms

Dr. Pascal Bercher

Institute of Artificial Intelligence,
Ulm University, Germany

Winter Term 2018/2019

(Compiled on: February 20, 2019)

ulm university universität
uulm

Overview:

Introduction

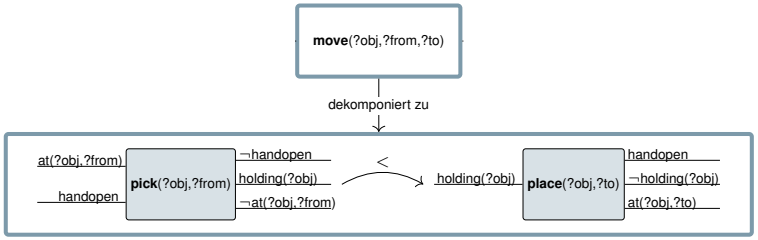
- So far, all investigations based upon a simplistic formalization of HTN planning.
- However, there several further hierarchical planning formalisms can be found in the literature.
- Some, we already covered so far:
  - TIHTN planning (covered in detail).
  - A lifted version of HTN and TIHTN planning (introduced rather informally).
  - A lifted HTN planning formalism that allows to express state constraints (briefly mentioned).
- But there are many more:
  - Some regard the task hierarchy *advice* rather than a *restriction*. Their goal is to solve *classical problems* rather than hierarchical ones (but do so hierarchically).
  - Others extend HTN planning to allow modeling support.
  - Others focus on the idea to refine/decompose *state features* rather than compound tasks.

Introduction

- Here, we deal with the *problem class* “hybrid planning”, not with the *algorithm* hybrid planning!
- To clarify: In the literature, the *Decomposition-based* planning approach (i.e., algorithm) for solving (TI)HTN problems via plan-based search using POCL techniques is also referred to as *hybrid planning* – so don’t confuse them
- Essentially, hybrid planning (from now on: the *problem class*!) is the same as (TI)HTN planning problem, but slightly extended:
  - Compound tasks can have preconditions and effects as well.
  - They can be used to define *implementation criteria* – they define the circumstances under which a decomposition method is regarded an *implementation* of its compound task.
  - Further, the model (and initial task network) can contain causal links.

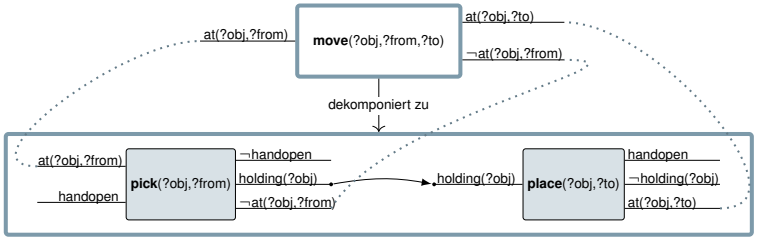
  → Consequently, the solution criteria also involve causal links.

Motivation



The hybrid planning formalism allows to define preconditions and effects for compound tasks. But why?

Motivation



The hybrid planning formalism allows to define preconditions and effects for compound tasks. But why?

- To generate abstract (i.e., non-primitive) solutions.
- To exploit it during search.
- To provide *modeling support* by restriction to legal models (test all decomposition methods).

Problem Description

- Here, we give only a rather informal description of hybrid problems via only mentioning the *differences* to (TI)HTN and POCL problems.
- The problem description extends (TI)HTN problems as follows:
  - Rather than task networks, the model uses *partial plans* known from POCL planning or from decomposition-based search (i.e., plan steps can be abstract as well and there might be causal links).
  - Compound tasks have the same syntactic form as actions, i.e., they allow for preconditions and effects.
  - Consequently, causal links are allowed to point to or from compound tasks as well.
  - Thus, we need to alter how causal links induce ordering constraints: In contrast to the previous definition (cf. first lecture) only those causal links induce an ordering that are defined between two primitive tasks.

Solution Criteria

- The solution criteria are essentially the same as for (TI)HTN planning, but extended in two ways:
  - To deal with the causal links pointing to or from a compound task, we demand that upon decomposition they get decomposed to any possible consumer (branching over all possibilities).
  - Solutions are defined as primitive plans without flaws. (Similar to the decomposition-based algorithm.)
- Similar to HTN vs. TIHTN problems, task insertion is an optional property of the the hybrid planning problem class.

Introduction

- *Legality criteria* (or *implementation criteria*) are criteria under which decomposition methods are assumed to be correct (or legal) implementations of their compound task.
- They all try to express “what it means that a compound task has preconditions or effects”.
- There are differently strong restrictions, though their theoretical impact is very limited.

Downward Compatible

Definition (Downward Compatible, Bercher et al. 2016)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task and P a partial plan.

- If $\varphi \in pre$, then there exists φ as precondition of a task in P without causal link, which points towards it.
- If $\varphi \in eff$, then exists φ as effect of a task in P.

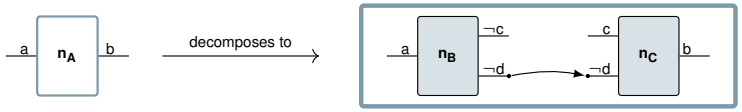
- Prevents/detects the most obvious modeling flaws.
- abstract tasks can always be decomposed – as it is the case in standard HTN planning

Downward Compatible

Definition (Downward Compatible, Bercher et al. 2016)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task and P a partial plan.

- If $\varphi \in pre$, then there exists φ as precondition of a task in P without causal link, which points towards it.
- If $\varphi \in eff$, then exists φ as effect of a task in P.



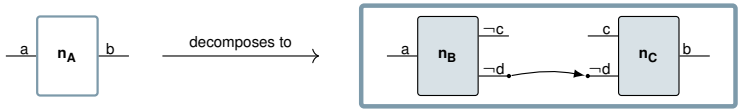
Does this method satisfy the criterion?

Downward Compatible

Definition (Downward Compatible, Bercher et al. 2016)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task and P a partial plan.

- If $\varphi \in pre$, then there exists φ as precondition of a task in P without causal link, which points towards it.
- If $\varphi \in eff$, then exists φ as effect of a task in P.



Does this method satisfy the criterion? Yes!

State-transition Semantics

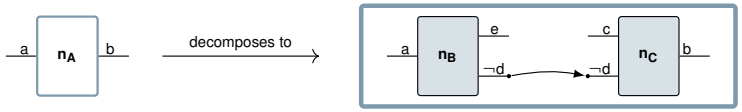
Definition (Biundo and Schattenberg, 2001)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a totally ordered plan.

- There needs to be a state s satisfying pre, $s \models pre$, such that P's task sequence $\bar{t}$ is executable in s.
- For all states satisfying the first criterion, $\bar{t}$ generates a state satisfying eff, $s \models eff$.

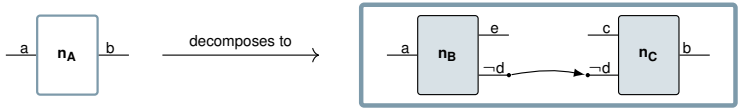
- State-transition semantics adopted to abstract tasks.
- Strong assumption: open preconditions need to be supported by one single state.

Introduction Hybrid Planning Decompositional Planning Summary
Legality Criteria
State-transition Semantics
Definition (Biundo and Schattenberg, 2001)
Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a totally ordered plan.
- There needs to be a state s satisfying pre, $s \models pre$, such that P's task sequence $\bar{t}$ is executable in s.
- For all states satisfying the first criterion, $\bar{t}$ generates a state satisfying eff, $s \models eff$.



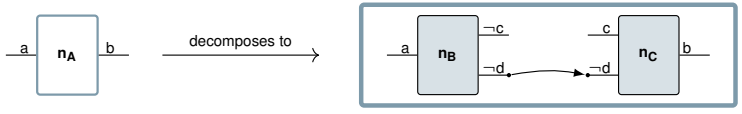
Does this method satisfy the criterion?

Introduction Hybrid Planning Decompositional Planning Summary
Legality Criteria
State-transition Semantics
Definition (Biundo and Schattenberg, 2001)
Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a totally ordered plan.
- There needs to be a state s satisfying pre, $s \models pre$, such that P's task sequence $\bar{t}$ is executable in s.
- For all states satisfying the first criterion, $\bar{t}$ generates a state satisfying eff, $s \models eff$.



Does this method satisfy the criterion? Yes!

Introduction Hybrid Planning Decompositional Planning Summary
Legality Criteria
State-transition Semantics
Definition (Biundo and Schattenberg, 2001)
Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a totally ordered plan.
- There needs to be a state s satisfying pre, $s \models pre$, such that P's task sequence $\bar{t}$ is executable in s.
- For all states satisfying the first criterion, $\bar{t}$ generates a state satisfying eff, $s \models eff$.



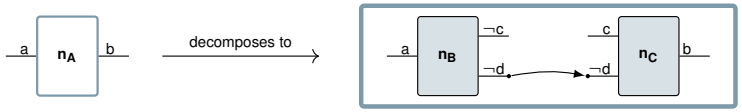
Does this method satisfy the criterion?

State-transition Semantics

Definition (Biundo and Schattenberg, 2001)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a totally ordered plan.

- There needs to be a state s satisfying pre, $s \models pre$, such that P's task sequence $\bar{t}$ is executable in s.
- For all states satisfying the first criterion, $\bar{t}$ generates a state satisfying eff, $s \models eff$.



Does this method satisfy the criterion? No!

Causal Threat Criterion

Definition (Yang, 1990)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a plan.

- *pre* and *eff* are actual preconditions and effects in P.
- There are no causal threats.

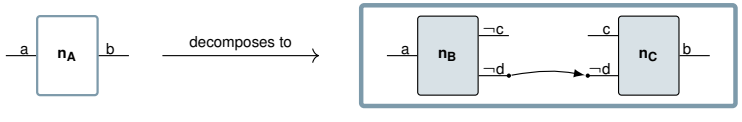
- Slightly weaker than the last criterion: open preconditions do not need to be achieved by one single state.

Causal Threat Criterion

Definition (Yang, 1990)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a plan.

- *pre* and *eff* are actual preconditions and effects in P.
- There are no causal threats.



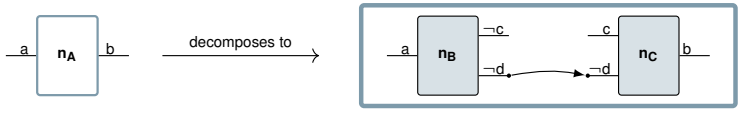
Does this method satisfy the criterion?

Causal Threat Criterion

Definition (Yang, 1990)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a plan.

- *pre* and *eff* are actual preconditions and effects in P.
- There are no causal threats.



Does this method satisfy the criterion? Yes!

Causal Link Chain Criterion

Definition (Young et al., 1994)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a plan.

- Any of n_c's preconditions *pre* contributes to at least one of its effects *eff* via a chain of causal links
- ... and vice versa.

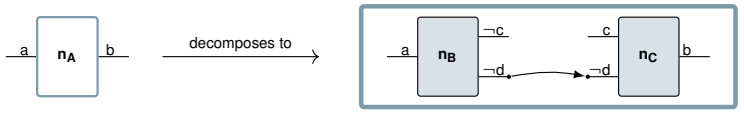
- Neither stronger, nor weaker than the previous criterion.

Causal Link Chain Criterion

Definition (Young et al., 1994)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a plan.

- Any of n_c's preconditions *pre* contributes to at least one of its effects *eff* via a chain of causal links
- ... and vice versa.



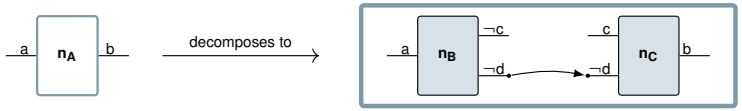
Does this method satisfy the criterion?

Causal Link Chain Criterion

Definition (Young et al., 1994)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a plan.

- Any of n_c's preconditions *pre* contributes to at least one of its effects *eff* via a chain of causal links
- ... and vice versa.



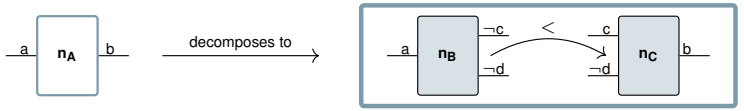
Does this method satisfy the criterion? Yes!

Causal Link Chain Criterion

Definition (Young et al., 1994)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a plan.

- Any of n_c's preconditions *pre* contributes to at least one of its effects *eff* via a chain of causal links
- ... and vice versa.



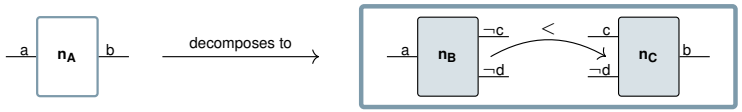
Does this method satisfy the criterion?

Causal Link Chain Criterion

Definition (Young et al., 1994)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a plan.

- Any of n_c's preconditions *pre* contributes to at least one of its effects *eff* via a chain of causal links
- ... and vice versa.



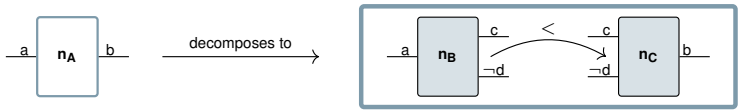
Does this method satisfy the criterion? No!

Causal Link Chain Criterion

Definition (Young et al., 1994)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a plan.

- Any of n_c's preconditions *pre* contributes to at least one of its effects *eff* via a chain of causal links
- ... and vice versa.



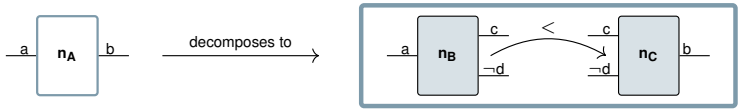
Does this method satisfy the criterion?

Causal Link Chain Criterion

Definition (Young et al., 1994)

Let $m = (n_c, P)$ be a method, $n_c = (pre, eff)$ an abstract task, and P a plan.

- Any of n_c's preconditions *pre* contributes to at least one of its effects *eff* via a chain of causal links
- ... and vice versa.



Does this method satisfy the criterion? No!

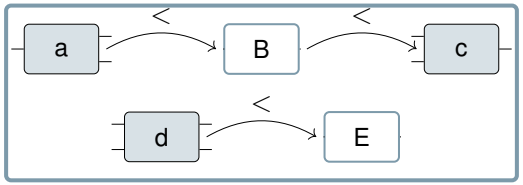
More than a Name?

- Which impact have the legality criteria on the expressivity?
- We show that every HTN problem $\mathcal{P}$ can be transformed into a hybrid planning problem $\mathcal{P}'$, such that:
  - $\mathcal{P}$ and $\mathcal{P}'$ have the same set of solutions (see discussion),
  - $\mathcal{P}'$ satisfies all legality criteria.

Encoding HTN Problems into Hybrid Problems

For each primitive task t, create an abstract copy T without preconditions and effects. Then:

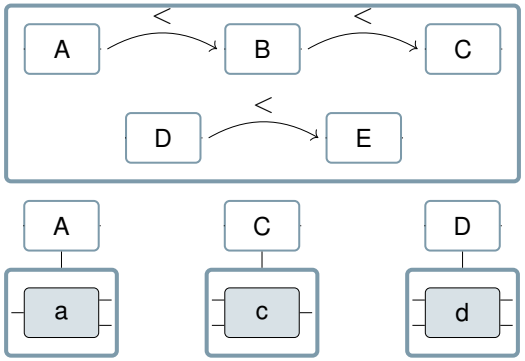
- Add a method $m = (T, P)$ with P containing exactly t.
- In each plan, replace t by T.



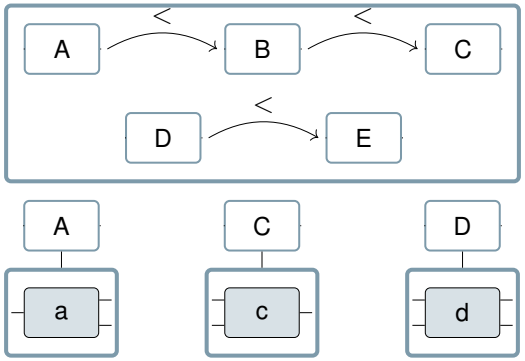
Encoding HTN Problems into Hybrid Problems

For each primitive task t, create an abstract copy T without preconditions and effects. Then:

- Add a method $m = (T, P)$ with P containing exactly t.
- In each plan, replace t by T.



Encoding HTN Problems into Hybrid Problems



Properties:

- All abstract tasks do not have preconditions or effects
- For all plans holds:
  - either there are only abstract tasks
  - or at most one.
- Thus, all methods in $\mathcal{P}'$ satisfy all legality criteria.

Introduction

- The problem definition and the solution criteria changed, so we have a new problem class with potentially different theoretical properties.
- So, how hard (or easy) is this problem?
- As we will see: we can express every HTN problem as a hybrid problem without violating any criterion.
- Thus, it's as hard as HTN planning.
- Membership results (lower bounds) are open, but probably identical to those from HTN planning as well.

Complexity Results (Plan Existence)

Theorem

Hybrid planning is strictly semi-decidable.

Proof:

semi-decidable:

- Perform BFS starting in the initial partial plan.

undecidable:

- Reduce the undecidable HTN plan existence problem to hybrid planning
- For this, we use the property shown before, i.e., that every HTN problem can be encoded by a hybrid problem that satisfies *all* legality criteria.

Hybrid Planning Algorithm

- To solve hybrid problems, we have to use the hybrid planning *algorithm* (i.e., decomposition-based search), because it's the only one being able to deal with causal links by now.
- That is, as soon as a compilation from hybrid models to standard (TI)HTN models is known, we can again use all standard techniques.
- Reminder: Now, partial plans can already contain causal links. Do we have to change the algorithm? Yes!
  - In some cases, we get fewer successors when decomposing, because we can use the effects of compound tasks as producer (see example from blackboard).
  - When decomposing an abstract task that's involved in causal links, we get, in general, much more successors than it's number of methods (see example from blackboard).

Motivation and Problem Definition

- *Decompositional planning* was created as a means to reduce search effort for solving classical problems via hierarchical planning techniques.
- That is, it is defined in exactly the same way as *hybrid planning with task insertion*, but without an initial partial plan. Why?
  - Now, also compound tasks can be inserted, which (hopefully) lead to solutions quicker than compared to relying on primitive task insertion completely.
  - Theory-wise, the problem is as expressive as classical planning, because we are not forced to insert compound tasks (and if there is a solution with compound task insertion then there is also one without).

Summary

- There are many hierarchical planning formalisms – more than just the HTN or TIHTN formalisms discussed so far.
- One of them is hybrid planning, which extends (TI)HTN planning with concepts known from POCL planning.
- Closely related is decompositional planning, which is essentially hybrid planning with task insertion, but without initial partial plan.
- Another formalism is HGN planning, Hierarchical Goal-Task Network planning (we only discussed this briefly).