More than a Name?
On Implications of Preconditions and Effects of Compound HTN Planning Tasks

Pascal Bercher and Daniel Höller and Gregor Behnke and Susanne Biundo

Institute of Artificial Intelligence

September, 2016
Motivation

Several formalizations allow to specify preconditions and effects for abstract tasks. Why? To find abstract solutions, for search guidance, and to allow modeling assistance (restrict to legal methods).

However, most complexity results are only known for HTN planning, where abstract tasks are just names.

More than a Name? Complexity Results for Hybrid Planning (Bercher et al.)

September, 2016
Several formalizations allow to specify preconditions and effects for abstract tasks.
Several formalizations allow to specify preconditions and effects for abstract tasks. Why?

- To find abstract solutions,
- for search guidance, and
- to allow modeling assistance (restrict to legal methods).
Several formalizations allow to specify preconditions and effects for abstract tasks. Why?

- To find abstract solutions,
- for search guidance, and
- to allow modeling assistance (restrict to legal methods).
Several formalizations allow to specify preconditions and effects for abstract tasks. Why?

- To find abstract solutions,
- for search guidance, and
- to allow modeling assistance (restrict to legal methods).

However, most complexity results are only known for HTN planning, where abstract tasks are just names.
Hybrid Planning Framework

Hybrid planning fuses
- Hierarchical Task Network (HTN) planning with
- Partial-Order Causal-Link (POCL) Planning.
→ Here, also abstract tasks have preconditions and effects.

Planning problem and solutions:
- The planning problem is given in terms of an initial plan.
- Solutions are plans that
  - are executable and satisfy the goal and
  - they are refinements of the initial plan.
Legality Criteria

What’s the preconditions’ and effects’ impact on the complexity of:

- The plan verification problem. (“Is the plan $P$ a solution?”)
- The plan existence problem. (“Is there a solution at all?”)
Legality Criteria

What’s the preconditions’ and effects’ impact on the complexity of:

- The *plan verification* problem. ("Is the plan $P$ a solution?")
- The *plan existence* problem. ("Is there a solution at all?")

To this end: What do these preconditions and effects of abstract tasks *directly* entail?
Legality Criteria

What’s the preconditions’ and effects’ impact on the complexity of:

- The *plan verification* problem. ("Is the plan $P$ a solution?")
- The *plan existence* problem. ("Is there a solution at all?")

To this end: What do these preconditions and effects of abstract tasks *directly* entail? $\rightarrow$ They restrict the models! But how?
Legality Criteria: Are Methods correct Implementations?

What’s the preconditions’ and effects’ impact on the complexity of:
- The plan verification problem. (“Is the plan $P$ a solution?”)
- The plan existence problem. (“Is there a solution at all?”)

To this end: What do these preconditions and effects of abstract tasks directly entail? → They restrict the models! But how?
- So-called legality criteria define which methods are regarded legal implementations of their abstract task.
- Restrict planning model: only legal methods are allowed.
Legality Criteria: Are Methods correct Implementations?

What’s the preconditions’ and effects’ impact on the complexity of:
- The *plan verification* problem. ("Is the plan $P$ a solution?")
- The *plan existence* problem. ("Is there a solution at all?")

To this end: What do these preconditions and effects of abstract tasks *directly* entail? → They restrict the models! But how?
- So-called *legality criteria* define which methods are regarded legal implementations of their abstract task.
- Restrict planning model: only legal methods are allowed.
- Which legality criteria make sense? Which ones exist?
Legality Criteria: Are Methods correct Implementations?

What’s the preconditions’ and effects’ impact on the complexity of:
- The plan verification problem. (“Is the plan \( P \) a solution?”)
- The plan existence problem. (“Is there a solution at all?”)

To this end: What do these preconditions and effects of abstract tasks directly entail? → They restrict the models! But how?
- So-called legality criteria define which methods are regarded legal implementations of their abstract task.
- Restrict planning model: only legal methods are allowed.
- Which legality criteria make sense? Which ones exist?
  → Paper provides a survey and discussion.
Legality Criteria (cont’d)

Definition (Downward Compatible, Bercher et al., ECAI-2016)

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

- If $\varphi \in pre$, then $\varphi$ exists as precondition of a task in $P$ with no causal link pointing towards it.
- If $\varphi \in eff$, then $\varphi$ exists as effect of a task in $P$.

\[ n_a \xrightarrow{a} n_B \xrightarrow{d} n_C \xrightarrow{b} \]

\[ \neg c \rightarrow \neg d \rightarrow \neg d \]

(this method satisfies the criterion)
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 $eff$, $s \models eff$.

[Diagram showing decomposition of a method into $n_A$, $n_B$, $n_C$]
Legality Criteria (cont’d)

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 \( eff \), \( s \models eff \).

\[ \begin{array}{ccc}
\text{a} & \text{n}_A & \text{b} \\
\end{array} \quad \text{decomposes to} \quad \begin{array}{ccc}
\text{a} & \text{n}_B & \text{c} \\
\text{d} & \text{n}_C & \text{b} \\
\end{array} \]

(this method does not satisfy the 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.

\[
\begin{align*}
\text{a} & \quad \text{n}_A & \quad \text{b} & \quad \text{decomposes to} & \quad \text{a} & \quad \text{n}_B & \quad \neg c & \quad c & \quad \text{c} & \quad \text{n}_C & \quad \text{b} \\
\end{align*}
\]

(this method satisfies the criterion)
Legality Criteria (cont’d)

Definition (Young et al., 1994)

Let \( m = (n_c, P) \) be a method, \( n_c = (\text{pre}, \text{eff}) \) an abstract task, and \( P \) a plan.

- Any of \( n_c \)'s preconditions \( \text{pre} \) contributes to at least one of its effects \( \text{eff} \) via a chain of causal links
- \( \ldots \) and vice versa.

\[ \begin{align*}
\text{A} & \quad \text{b} \quad \text{decomposes to} \quad \text{C} \\
\text{a} & \quad \text{c} & \quad \text{d} & \quad \text{d}
\end{align*} \]

(this method satisfies the criterion)
Legality Criteria (cont’d)

Definition (Young et al., 1994)

Let \( m = (n_c, P) \) be a method, \( n_c = (\text{pre}, \text{eff}) \) an abstract task, and \( P \) a plan.

- Any of \( n_c \)'s preconditions \( \text{pre} \) contributes to at least one of its effects \( \text{eff} \) via a chain of causal links
- ... and vice versa.

\[
\begin{align*}
\d agency \quad a & \quad \overset{\text{decomposes to}}{\underset{\text{decomposes to}}{\text{Decomposes to}}} \\
\quad b & \\
\end{align*}
\]

\begin{align*}
\d agency \quad a & \quad \overset{\text{c}}{\underset{\text{d}}{\text{c}}} \\
\quad d & \quad \overset{\text{c}}{\underset{\text{d}}{\text{c}}} \\
\end{align*}

(this method does not satisfy the criterion)
Which impact have the legality criteria on the expressivity?

We show that every HTN problem $\pi$ can be transformed into a hybrid planning problem $\pi'$, such that:
- $\pi$ and $\pi'$ have the same set of solutions,
- $\pi'$ 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 \( T \).

More than a Name? Complexity Results for Hybrid Planning (Bercher et al.)
September, 2016
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 \( T \).
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 $\pi'$ satisfy all legality criteria.
Complexity Results (Plan Verification)

- **General case:**
  - Corresponds to standard HTN plan verification.
  - Is **NP-complete**, as in HTN planning.

- **Without hierarchy, i.e., no abstract tasks:**
  - Corresponds to standard POCL plan verification.
  - Is in **P** (commonly known).
  - Interestingly, this problem is **NP-hard** in HTN planning.
Theorem

Hybrid planning is strictly semi-decidable.

Proof.

semi-decidable:

- Enumerate all plans of a certain length (from 0 to $\infty$).
- Verify each plan in $\textbf{NP}$.
- Continue until a solution is found.

undecidable:

- Reduce the undecidable HTN plan existence problem to hybrid planning (using the encoding).
Corollary

Several sub classes of hybrid planning are as hard as in HTN planning: tail-recursive, acyclic, totally-ordered, and delete-relaxed.

Proof.

The class of a problem is preserved by the transformation.
Summary

- Provided formalization for HTN planning, where abstract tasks have preconditions and effects.

- Gave survey and discussion about legality criteria in hierarchical planning.

- Theoretically investigated their impact on:
  - The plan verification problem and
  - the plan existence problem.
Conclusions and Future Work

- Legality criteria were designed to give a clear semantics to methods – with respect to the tasks they implement.
Conclusions and Future Work

- Legality criteria were designed to give a clear semantics to methods – with respect to the tasks they implement.
- They can be used for modeling assistance.
Conclusions and Future Work

- Legality criteria were designed to give a clear semantics to methods – with respect to the tasks they implement.

- They can be used for modeling assistance.

- Their theoretical impact seems limited: the general case is as hard as in HTN planning – for all investigated legality criteria (Main reason for this: None of the given criteria enforces the specification of preconditions and effects).
Conclusions and Future Work

- Legality criteria were designed to give a clear semantics to methods – with respect to the tasks they implement.
- They can be used for modeling assistance.
- Their theoretical impact seems limited: the general case is as hard as in HTN planning – for all investigated legality criteria (Main reason for this: None of the given criteria enforces the specification of preconditions and effects).
- For many special cases, membership results are still missing. Thus, complete relationship between HTN and hybrid planning is yet unknown.
Conclusions and Future Work

- Legality criteria were designed to give a clear semantics to methods – with respect to the tasks they implement.
- They can be used for modeling assistance.
- Their theoretical impact seems limited: the general case is as hard as in HTN planning – for all investigated legality criteria (Main reason for this: None of the given criteria enforces the specification of preconditions and effects).
- For many special cases, membership results are still missing. Thus, complete relationship between HTN and hybrid planning is yet unknown.
- We did not yet investigate the impact of task insertion.