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** [*]

Abstract. There are several formalizations for hierarchical planning. Many of them allow to specify preconditions and effects for compound tasks. They can be used, e.g., to assist during the modeling process by ensuring that the decomposition methods' plans "implement" the compound tasks' intended meaning. This is done based on so-called *legality criteria* that relate these preconditions and effects to the method's plans and pose further restrictions. Despite the variety of expressive hierarchical planning formalisms, most theoretical investigations are only known for standard HTN planning, where compound tasks are just names, i.e., no preconditions or effects can be specified. Thus, up to now, a direct comparison to other hierarchical planning formalisms is hardly possible and fundamental theoretical properties are yet unknown. To enable a better comparison between such formalisms (in particular with respect to their computational expressivity), we first provide a survey on the different legality criteria known from the literature. Then, we investigate the theoretical impact of these criteria for two fundamental problems to planning: *plan verification* and *plan existence*. We prove that the plan verification problem is at most $\mathbf{NP}$-$\mathbf{complete}$, while the plan existence problem is in the general case both $\mathbf{semi\text{-}decidable}$ and $\mathbf{undecidable}$, independent of the demanded criteria. Finally, we discuss our theoretical findings and practical implications.

1 Introduction

Hierarchical planning approaches are often chosen when it comes to practical real-world planning applications [33]. Examples include composition of web services [29], real-time strategy games [23, 35], robotics [15, 28], or user assistance [11, 10]. While there are several different formalizations for hierarchical planning, it is apparent that most of the *theoretical* investigations are done for a standard formalization (called hierarchical task network (HTN) planning), where compound (or abstract) tasks are just names or symbols [16, 19] – they thus neither show preconditions nor effects. Those investigations include the complexity of the plan existence problem ("Is the problem solvable?") [16, 19, 5, 2, 3], plan verification ("Is the given plan a solution to the problem?") [9], changes to plans ("Is the plan still a solution if I change X?") [8], and expressivity analysis ("What plan structures can be expressed using different language features?") [21, 22]. The answers to such questions, besides being of theoretical interest, are highly relevant to come up with tractable problem relaxations for heuristics [5] or for problem compilations [4, 1].

For mainly practically motivated reasons, such as providing modeling assistance or generating abstract solutions, several researchers developed hierarchical planning formalizations in which compound tasks are allowed to have preconditions and effects [40, 25, 37, 41, 18, 26, 12, 14, 30, 11, 7, 15] (cf. example given in Fig. 1). However, the only theoretical investigations for such a formalization that we are aware of are about the upward and downward refinement properties [40, 6, 30]. So, up to now, for many of such formalisms it is not even clear how hard the respective problems are (given the typical HTN solution criteria), since the plan existence problem was not studied for such a formalization. To close this gap, we investigate the plan existence and the plan verification problems for our formalization that allows to specify preconditions and effects for compound tasks. We survey several legality criteria that define which decomposition methods may be specified for which compound task, depending on its preconditions and effects and take these criteria into account in our complexity analysis. We conclude the paper by discussing our findings and its implications.

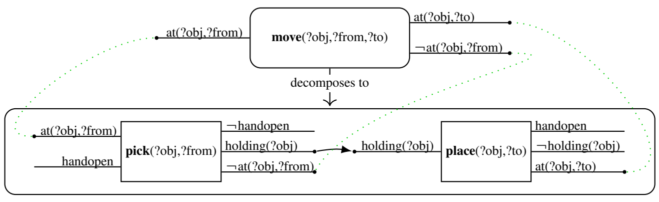

Figure 1. Example for a compound task (*move*) with preconditions and effects and one of its methods. The method's plan consists of two primitive tasks (*pick* and *place*) and a causal link protecting the condition *holding*. The dotted green lines indicate how the preconditions and effects of the tasks in the plan's method are related to their more abstract representation.

2 Problem Formalization

We introduce a hierarchical planning formalism that allows to specify preconditions and effects for compound tasks. In our complexity analysis, we take into account several restrictions on the relationship between compound tasks and the plans that are associated with them via their decomposition methods. The investigated restrictions, called legality criteria, are taken from the literature. To formally study their impact on the complexity results, the formalism needs to be rich enough to be capable of expressing these criteria. Most of the formalisms that define them [40, 41, 37, 12] fuse standard HTN planning [16, 20, 19] with Partial-Order Causal-Link (POCL) planning [31, 36, 20]. We assume this is because the concept of causal links makes it easy to express the desired criteria. Thus, we also use such an hybridization for our investigations. A variety of formalisms fuse HTN with POCL planning [40, 25, 37, 41, 18, 14, 7, 15, 11].

[*]Institute of Artificial Intelligence, Ulm University, D-89069 Ulm, Germany, {*forename.surname*}@uni-ulm.de

However, none of these formalizations is both rich enough to allow expressing all legality criteria while being simple enough to easily serve as a basis for proofs. Since all HTN complexity results that are relevant for the sake of this paper have been shown or reproduced in the simplistic (propositional) HTN formalism by Geier and Bercher [19], extending it allows an easy comparison. We therefore extend it by the necessary POCL concepts. In accordance to the literature [26, 11], we refer to the resulting formalism as *hybrid planning*.

Let V be a finite set of *state variables* (or *proposition symbols*). In POCL planning, actions are typically 2-tuples consisting of a precondition and effect, both being conjunctions of literals. Here, we use an equivalent set-based formalization: Actions (or *primitive tasks*) are 4-tuples $(\mathit{prec}^+, \mathit{prec}^-, \mathit{eff}^+, \mathit{eff}^-)$, where prec^+ and prec^- denote the positive and negative *preconditions* and eff^+ and eff^- denote the positive and negative *effects*. They describe single state transitions as usual. The *compound* (or *abstract*) tasks have a different underlying meaning, as they need to be decomposed into predefined plans by relying on so-called *decomposition methods*. Despite that fact, we allow compound tasks to use preconditions and effects as well (cf. Fig. 1 for an example). Every task has a *task name*. The set of names for the primitive tasks is given by N_p and those of the compound ones by N_c. We define $N := N_p \dot{\cup} N_c$. The mapping between task names and their actual tasks (i.e., 4-tuples) is established using the function $\delta : N \to (2^V)^4$. For convenience, we also write $\mathit{prec}^+(n)$, $\mathit{prec}^-(n)$, $\mathit{eff}^+(n)$, and $\mathit{eff}^-(n)$ to refer to n's positive and negative preconditions and effects, respectively[†]. We call a sequence of tasks $\delta(n_1), \ldots, \delta(n_k)$ *executable in a state* $s_0 \in 2^V$ if and only if there is a corresponding sequence of states $s_0, \ldots, s_k$, such that for all $1 \leq i \leq k$ holds $\mathit{prec}^+(n_i) \subseteq s_{i-1}$ and $\mathit{prec}^-(n_i) \cap s_{i-1} = \emptyset$ as well as $s_i = (s_{i-1} \setminus \mathit{eff}^-(n_i)) \cup \mathit{eff}^+(n_i)$. The state s_k is called *the state generated by* $\delta(n_1), \ldots, \delta(n_k)$.

In non-linear planning approaches, *plans* are only partially ordered. For a set of ordering constraints $\prec$, we denote its transitive closure by $\prec^*$. To differentiate multiple occurrences of the same task within a plan, the partial order is defined over a set of so-called *plan steps* PS, which then map to the actual task name, $\alpha : PS \to N$. When we mention a *linearization* of the plan steps of a plan, we refer to a total order of PS that does not violate the partial order $\prec$. Plans may also contain so-called *causal links*. A causal link $(ps, v, ps') \in PS \times V \times PS$ indicates that the precondition v of the *consumer* plan step ps' is *supported* by the *producer* plan step ps. The condition v is also said to be *protected* by the causal link. This means, if v is a positive (resp. negative) precondition of ps', then no task with v as negative (resp. positive) effect is allowed to be ordered between ps and ps'.

Definition 1 (Plan)**.** *A* plan *P over a set of task names N is a 4-tuple $(PS, CL, \prec, \alpha)$, where:*

- *PS is a finite (possibly empty) set of plan steps,*
- *$CL \subseteq PS \times V \times PS$ is a set of causal links. If $(ps, v, ps') \in CL$, then $v \in \mathit{prec}^+(ps')$ and $v \in \mathit{eff}^+(ps)$ or $v \in \mathit{prec}^-(ps')$ and $v \in \mathit{eff}^-(ps)$. We also require that every precondition variable of all plan steps is protected by at most one causal link,*
- *$\prec\ \subseteq PS \times PS$ is a strict partial order. If $(ps, v, ps') \in CL$ with $\alpha(ps)$ and $\alpha(ps')$ being primitive, then $(ps, ps') \in \prec^{‡}$,*
- *$\alpha : PS \to N$ labels every plan step with its task name.*

$\mathcal{P}_N$ denotes the set of all plans over the task names N. Two plans are called isomorphic if they are identical except for plan step renaming.

Based on the concept of plans, we define *decomposition methods*. The set of all decomposition methods $M \subseteq N_c \times \mathcal{P}_N$ is given in the planning domain and defines how compound tasks can be decomposed. That is, a method $(n_c, P) \in M$ indicates that the compound task (name) n_c can be decomposed into the plan P.

Definition 2 (Hybrid Planning Problem)**.** *A* hybrid planning problem *is a 6-tuple $\pi = (V, N_c, N_p, \delta, M, P^i)$, where:*

- *V is a finite set of* state variables*,*
- *we require $N_c \cap N_p = \emptyset$ and define $N := N_c \cup N_p$, where:*
  - *N_c is a finite set of* compound task names*,*
  - *N_p is a finite set of* primitive task names*,*
  - *$\{\mathit{init}, \mathit{goal}\} \subseteq N_p$ denote two special primitive task names,*
- *$\delta : N \to (2^V)^4$ is a function mapping the task names to their preconditions and effects*[†][§]*,*
- *$M \subseteq N_c \times \mathcal{P}_{N \setminus \{\mathit{init}, \mathit{goal}\}}$ is a finite set of* (decomposition) methods*, and*
- *$P^i = (PS^i, CL^i, \prec^i, \alpha^i) \in \mathcal{P}_N$, is the* initial plan*. We require that there are plan steps $ps, ps' \in PS^i$ such that:*
  - *$\alpha^i(ps) = \mathit{init}$ and $\alpha^i(ps') = \mathit{goal}$, and*
  - *$ps \prec ps'$ and for all $ps'' \in PS^i$ with $\{ps''\} \cap \{ps, ps'\} = \emptyset$ holds $ps \prec ps'' \prec ps'$.*

The actual problem that one would like to have solved is given in terms of the initial plan P^i. As done in POCL planning, this plan contains two artificial actions that encode the initial state and the goal description, respectively[§]. Since hybrid planning is a hierarchical setting, P^i usually contains a set of compound tasks for which one needs to find an executable refinement. We added the specification of a goal description for practical reasons: It is especially interesting if one allows the arbitrary insertion of tasks into the plan apart from decomposing compound tasks [19] (not considered in this paper), since hybrid planning then directly captures both classical and POCL planning. Independent of whether task insertion is allowed or not, adding a goal description is not required from a purely theoretical point of view, since one can easily simulate it [19, Sec. 2].

In HTN planning, only those plans are regarded solutions that can be obtained from the initial plan by successively applying decomposition methods to compound tasks. We thus need to define how applying a method transforms one plan into another. Since the decomposed task might serve as a producer or consumer of causal links, we have to decide how such links will be passed down to sub tasks and whether this is mandatory or not (i. e., we could even allow that such links may be deleted upon decomposition). Adding a causal link to a compound task means to commit that the state variable of that link is protected for the complete sequence of states over which this link spans. Allowing to remove that link upon decomposition would remove this constraint and violate the refinement principle [24]. If

[†]To simplify upcoming definitions, we require that for all primitive tasks n_p, $\mathit{prec}^+(n_p) \cap \mathit{prec}^-(n_p) = \mathit{eff}^+(n_p) \cap \mathit{eff}^-(n_p) = \emptyset$.

[‡]As usual in POCL planning, any causal link between primitive tasks implies an ordering. If one of these tasks was compound, this would not be reasonable: Consider the example depicted in Fig. 1 and assume there is a causal link from *move*'s effect $\neg at$ to another task's precondition. If that link would imply an ordering, then after *move* was decomposed, the consumer task had to be ordered behind *place*, which is overly restrictive.

[§]The initial state and goal description are specified in terms of the task names init and goal and their tuple representation using δ. As usual in POCL planning, the action for init does not show a precondition and uses the initial state as effect and, analogously, the action for goal has no effects and uses the goal description as precondition.

causal links do have to be passed down to sub tasks, then the respective formalism satisfies the so-called *monotonic property* [27]. If this property does not hold, hierarchical planning systems cannot exploit such causal links to prune plans from the search space, as the constraint imposed by these links could disappear upon decomposition [14]. We hence require that causal links are not allowed to disappear upon decomposition. *How* causal links are passed down has yet to be decided – and different conventions exist [25, p. 204]. We follow the canonical approach by Yang [40] and pass down every causal link to each "compatible" sub task. In other words, if there is a link to a compound task's precondition/effect v, then for each precondition/effect v in its sub tasks, one successor plan is generated in which the link is passed down to the respective task. In case there is more than one matching sub task, it is not required that the causal link is duplicated to support all these tasks (or a sub set thereof), since the respective plan can be obtained via link insertions from the other plans.

The following definitions formally capture the decomposition of compound tasks and the inheritance of causal links. We first define two functions $in_P, out_P : PS \to 2^{CL}$ that return the set of incoming, respectively outcoming, causal links of a plan step $ps \in PS$ in a plan $P = (PS, CL, \prec, \alpha)$ as $in_P : ps \mapsto \{(ps', v, ps) \in CL \mid ps' \in PS\}$ and $out_P : ps \mapsto \{(ps, v, ps') \in CL \mid ps' \in PS\}$.

Definition 3 (Decomposition). *A method $m = (n_c, P) \in M$ decomposes a plan $P' = (PS', CL', \prec', \alpha')$ into another plan P'' by replacing plan step $ps \in PS'$ with $\alpha'(ps) = n_c$ if and only if:*

- *there is a plan $\widetilde{P} = (\widetilde{PS}, \widetilde{CL}, \widetilde{\prec}, \widetilde{\alpha})$ that is isomorphic to P, such that $\widetilde{PS} \cap PS' = \emptyset$,*
- *for each causal link $(ps', v, ps) \in in_{P'}(ps)$ there is a plan step $\widetilde{ps}_{(ps',v,ps)} \in \widetilde{PS}$, such that:*
  - *$v \in prec^+(\widetilde{\alpha}(\widetilde{ps}_{(ps',v,ps)}))$ in case $v \in prec^+(\alpha'(ps))$, or*
  - *$v \in prec^-(\widetilde{\alpha}(\widetilde{ps}_{(ps',v,ps)}))$ in case $v \in prec^-(\alpha'(ps))$,*
- *for each causal link $(ps, v, ps') \in out_{P'}(ps)$ there is a plan step $\widetilde{ps}_{(ps,v,ps')} \in \widetilde{PS}$, such that:*
  - *$v \in eff^+(\widetilde{\alpha}(\widetilde{ps}_{(ps,v,ps')}))$ in case $v \in eff^+(\alpha'(ps))$, or*
  - *$v \in eff^-(\widetilde{\alpha}(\widetilde{ps}_{(ps,v,ps')}))$ in case $v \in eff^-(\alpha'(ps))$,*
- *$P'' = (PS'', CL'', \prec'', \alpha'')$ is given as follows:*

$$
\begin{aligned}
PS'' &:= (PS' \setminus \{ps\}) \cup \widetilde{PS} \\
CL'' &:= (CL' \setminus (in_{P'}(ps) \cup out_{P'}(ps))) \cup \widetilde{CL} \\
&\qquad \cup \{(ps', v, \widetilde{ps}_{(ps',v,ps)}) \mid (ps', v, ps) \in in_{P'}(ps)\} \\
&\qquad \cup \{(\widetilde{ps}_{(ps,v,ps')}, v, ps') \mid (ps, v, ps') \in out_{P'}(ps)\} \\
\prec'' &:= (\prec_1 \cup \widetilde{\prec} \cup \prec_2 \cup \prec_3)^*, \text{ with} \\
&\prec_1 := (\prec' \setminus \{(ps', ps'') \in \prec' \mid \{ps\} \cap \{ps', ps''\} \neq \emptyset\}) \\
&\prec_2 := \{(ps', ps'') \in PS' \times \widetilde{PS} \mid (ps', ps) \in \prec'\} \cup \\
&\qquad \{(ps', ps'') \in \widetilde{PS} \times PS' \mid (ps, ps'') \in \prec'\} \\
&\prec_3 := \{(ps', \widetilde{ps}_{(ps',v,ps)}) \mid (ps', v, \widetilde{ps}_{(ps',v,ps)}) \in CL'' \\
&\qquad\qquad \text{and } \{\alpha(ps'), \alpha(\widetilde{ps}_{(ps',v,ps)})\} \subseteq N_p\} \cup \\
&\qquad \{(\widetilde{ps}_{(ps,v,ps')}, ps') \mid (\widetilde{ps}_{(ps,v,ps')}, v, ps') \in CL'' \\
&\qquad\qquad \text{and } \{\alpha(\widetilde{ps}_{(ps,v,ps')}), \alpha(ps')\} \subseteq N_p\} \\
\alpha'' &:= (\alpha' \setminus \{(ps, n_c)\}) \cup \widetilde{\alpha}
\end{aligned}
$$

As noted, we require that causal links involving the decomposed plan step ps (i.e., $in_{P'}(ps)$ and $out_{P'}(ps)$) are passed down upon decomposition (cf. CL''). For this, any compatible precondition that is not yet protected by a causal link inside the method's plan $\widetilde{P}$ can be used. The definition of the ordering constraints of the new plan P'', $\prec''$ comprises all ordering constraints of the original plan P' except the ones involving the decomposed plan step ps, $\prec_1$. It further contains all ordering constraints of the method's plan $\widetilde{P}$, $\widetilde{\prec}$, as well as those that are inherited from the orderings involving ps, $\prec_2$. All new causal links only involving primitive tasks are responsible for adding further orderings, $\prec_3$. Apart from the necessary extensions to handle causal links, our definition of decomposition is identical to the one from HTN planning [19, Def. 3].

In HTN planning, any solution to a planning problem (1) needs to be obtainable from the initial task network via the application of a sequence of decompositions and (2) needs to contain an executable sequence of its actions [19, Def. 5, 6]. We consider the second criterion as impractical: One is usually interested in executable action *sequences*, but finding one from a "solution" task network is still **NP-hard** [34, Thm. 15], [16, Thm. 8]. Further, such a sequence itself is in general not regarded a solution (but just the task network in which this sequence occurs), which we regard contra-intuitive. Instead, we require that *all* linearizations are executable and that each executable linearization is considered a solution as well. To support this stronger notion of solutions, we also allow the insertion of causal links and ordering constraints.

Definition 4 (Causal Link Insertion). *Let $P = (PS, CL, \prec, \alpha)$ and $P' = (PS, CL', \prec', \alpha)$ be plans. P' can be obtained from P by insertion of a causal link $(ps, v, ps') \notin CL$ with $ps, ps' \in PS$ if and only if:*

- *$v \in eff^+(\alpha(ps))$ and $v \in prec^+(\alpha(ps'))$ or $v \in eff^-(\alpha(ps))$ and $v \in prec^-(\alpha(ps'))$,*
- *$CL' = CL \cup \{(ps, v, ps')\}$, and*
- *$\prec' = (\prec \cup \{(ps, ps') \mid \{\alpha(ps), \alpha(ps')\} \subseteq N_p\})^*$*

Definition 5 (Ordering Insertion). *Let $P = (PS, CL, \prec, \alpha)$ and $P' = (PS, CL, \prec', \alpha)$ be plans. P' can be obtained from P by insertion of an ordering constraint $(ps, ps') \notin \prec$, $ps, ps' \in PS$ if and only if $\prec' = (\prec \cup \{(ps, ps')\})^*$.*

Definition 6 (Solution). *A plan $P = (PS, CL, \prec, \alpha)$ is a solution to a planning problem π, if and only if:*

1. *P is a refinement of P^i. That is, there is a sequence of decompositions (cf. Def. 3), causal link insertions (cf. Def. 4), and ordering constraint insertions (cf. Def. 5) transforming P^i into P,*
2. *P is primitive, i.e., $\{\alpha(ps) \mid ps \in PS\} \subseteq N_p$, and*
3. *P is executable in the standard POCL sense:*
   - *There are no* unprotected preconditions. *A precondition $v \in prec^+(\alpha(ps))$ (resp. $v \in prec^-(\alpha(ps))$) of a plan step $ps \in PS$ is called unprotected, if and only if there is no plan step $ps' \in PS$ with a causal link (ps', v, ps) for $v \in eff^+(\alpha(ps))$ (resp. $v \in eff^-(\alpha(ps))$).*
   - *There are no* causal threats. *A plan contains a causal threat if and only if there is a causal link $(ps, v, ps') \in CL$ with $v \in prec^+(\alpha(ps'))$ (resp. $v \in prec^-(\alpha(ps'))$) and a plan step $ps'' \in PS$ with $v \in eff^-(\alpha(ps''))$ (resp. $v \in eff^+(\alpha(ps''))$), such that neither $(ps'', ps) \in \prec$ nor $(ps', ps'') \in \prec$ holds.*

The first solution criterion corresponds to the standard HTN criterion that requires every solution to be in the refinement space of the initial plan. The second solution criterion demands the respective plan to be primitive, as only primitive plans are typically regarded executable. The third criterion requires executability as it is done in POCL planning; these criteria ensure that every linearization of the plan steps corresponds to a sequence of tasks that is executable in the initial state and generates a state satisfying the goal description.

3 Legality Criteria – A Survey and Discussion

Provided the planning domain allows to specify preconditions and effects for compound tasks, there should be a clearly-defined criterion stating which decomposition methods are allowed to be specified for such compound tasks [17, 14]. When considering a compound task as an abstraction of a certain plan, it does not seem to make much sense to specify a precondition or effect of that task if it does not occur anywhere in the plan. So, if the domain modeler decides to specify preconditions and effects for a compound task, he or she has a certain idea on how the plans of the respective decomposition methods should look like. Thus, several researchers have formalized possible relations between a compound task's preconditions and effects and its methods' plans. We call these criteria *legality criteria* and plans that respect them *implementations* of their compound task. For each criterion, we give a small example illustrating it. Further, we want to note that the example depicted in Fig. 1 satisfies all of the legality criteria discussed in this section[¶].

The first and weakest criterion that we investigate is closely related to the criteria that ensure that a compound task can be decomposed (cf. Def. 3). We restrict to models where the plan of a compound task's method makes use of the task's preconditions and effects.

Definition 7 (Downward Compatible). *A method $(n_c, P) \in M$ with $P = (PS, CL, \prec, \alpha)$ is called* downward compatible *if and only if:*

- *for each $v \in prec^+(n_c)$ (resp. $v \in prec^-(n_c)$) there is a plan step $ps \in PS$ with an unprotected precondition $v \in prec^+(\alpha(ps))$ (resp. $v \in prec^-(\alpha(ps))$).*
- *for each $v \in eff^+(n_c)$ (resp. $v \in eff^-(n_c)$) there is a plan step $ps \in PS$ with the same effect $v \in eff^+(\alpha(ps))$ (resp. $v \in eff^-(\alpha(ps))$).*

Downward compatible methods (n_c, P) are always applicable to a plan as long as it contains n_c – a property shared with standard HTN planning (without method preconditions). If the method was not downward compatible, causal links involving n_c influence its applicability, which also causes unintended "strange" behavior during search: Let us assume a plan P' contains a plan step ps with the name n_c that has an unprotected effect $v \in V$. Now, assume that (n_c, P) does violate the legality criterion. Whether this method can be used to generate a successor plan now depends on the planner's choice whether it first decomposes ps (this would work since all causal links can be correctly passed down to sub tasks in P') or first adds a causal link from $ps \in PS$ protecting v (then, ps can not be decomposed because the newly inserted link cannot be passed down, which violates the decomposition criteria given in Def. 3).

While this criterion clearly ensures that the "most obvious" modeling errors are prevented, it is not yet clear whether the user's intent about the relationship between the compound task and its methods' plans is actually satisfied. This is due to the fact that there are various possibilities what the preconditions and effects of the compound task are meant to entail. For example, if a primitive action n_p is within a plan, we know that – assuming that plan is executable – there are also states s and s', such that s satisfies n_p's precondition and s' satisfies n_p's effects. For compound tasks, this is not necessarily the case. In particular for the downward compatibility, it is clear that the compound task's effects do not need to be true in one single state, but its state variables might only hold in different states.

[¶]For the sake of readability, the example is given with variables, whereas our formalization assumes a ground (i.e., propositional) representation.

Several more restrictive criteria have been proposed in the literature. They can be categorized into two classes: one *enforces* that compound tasks have non-empty preconditions/effects under certain circumstances, and one where the specification thereof is *optional*.

We are only aware of one legality criterion that falls into the first class: It was proposed by Russell and Norvig for their fusion of HTN with POCL planning [37]. For each method (n_c, P) and every effect of n_c, they require that *"it [is] asserted by at least one step of P and is not denied by some other, later step"*. Further, they require that *"every precondition of the steps in P must be achieved by a step in P or be one of the preconditions of n_c"*. This implies that every open precondition in P needs to be a precondition of the compound task. This, in turn, has further consequences: Consider the case where the task n_c has two decomposition methods (n_c, P) and (n_c, P'). According to this criterion, n_c must use all open preconditions of both P and P'. As a consequence, both the downward compatibility and hence the monotonic property [27] may be violated. As argued in Sect. 2 (motivation for Def. 3), in this paper, we do not allow that commitments on an abstract level may be removed upon decomposition. We are thus not investigating this criterion in more detail.

The next criterion that we discuss was proposed by Biundo and Schattenberg [12]. As their planning formalism shows substantial differences to the one in this paper, we do not capture every detail, but only the main ideas. Their formalism is based upon a many-sorted first-order logic that features abstract state variables and so-called decomposition axioms defining them. They enable abstract tasks to make use of abstract state variables thereby fusing action abstraction with state abstraction. Further, their definition of methods only featured totally ordered plans: there, a method contains a set of task sequences, each of which is an implementation of the compound task. So, each method containing n sequences is a compact representation of n methods with totally ordered plans. Allowing for methods with partially ordered plans strictly increases expressivity (with respect to the plan existence problem [16, 2] and the generated solutions [21]), so legality should also be defined for such methods. We hence adapt their definition to partially ordered plans.

They require that if a method's task sequence is executable in a state satisfying the compound task's precondition, then it generates a state that satisfies the compound task's effects. They further require the compound task's precondition to be an abstraction of the task sequence's first task's precondition. Our generalization to partial orders states that this property must hold for every linearization that is induced by the given plan. However, we further demand that there also needs to be a state in which all linearizations are executable. As discussed earlier, causal links involving compound tasks do not (directly) induce ordering constraints (cf. Def. 1). However, since we consider a plan legal if all its linearizations are legal (which in turn need to respect the causal links), we here interpret causal links as additional ordering constraints. We thus define $\prec^+ = (\prec \cup \{(ps_i, ps_j) \mid (ps_i, v, ps_j) \in CL\})^*$ and only consider linearizations with respect to $\prec^+$. Then, a plan step linearization is said to *respect* a set of causal links CL if none of the protected conditions is violated by the induced state sequence.

Definition 8 (along Biundo and Schattenberg [12]). *A method $(n_c, P) \in M$, $P = (PS, CL, \prec, \alpha)$, is called* legal *if and only if:*

- *(n_c, P) is downward compatible,*
- *let $N_{first} := \{\alpha(ps) \mid ps \in PS$ and there is no $ps' \in PS$ with $(ps', ps) \in \prec^+\}$. Then, $prec^+(n_c) \subseteq \bigcap_{n \in N_{first}} prec^+(n)$ and $prec^-(n_c) \subseteq \bigcap_{n \in N_{first}} prec^-(n)$,*
- *for all states $s \in 2^V$ holds: if (a) $s \supseteq prec^+(n_c)$ and $s \cap prec^-(n_c) = \emptyset$, (b) every task sequence $\bar{t}$ corresponding to a plan step linearization $ps_1, \ldots, ps_{|PS|}$ with respect to $\prec^+$ is executable in s, and (c) respects CL, then (d) $\bar{t}$ generates a state s', such that $eff^+(n_c) \subseteq s'$ and $eff^-(n_c) \cap s' = \emptyset$. Further, there needs to be a state $s \in 2^V$, such that (a) to (d) hold.*

Concerning the intention behind the compound task's preconditions and effects, this criterion is already much stronger than the simple downward compatibility criterion. As opposed to that criterion, we here get the property that in any solution plan that is obtained from decomposing a compound task n_c, there is a *single* state in which n_c's preconditions hold. This trivially follows from the fact that the preconditions of n_c are abstractions of any first task and any compound task needs to be decomposed into a primitive plan. However, the criteria do not imply that there is a single state in which the compound task's effects hold – despite the restrictions imposed on the relationship between n_c's effects and its methods' plans. The reason for this is that the criterion does not take into account additional effects of the compound tasks that are in the method's plan, other than those explicitly specified in their preconditions and effects. This issue is addressed by Marthi et al.'s *possible* effects [30]. They are, however, restricting to totally ordered methods.

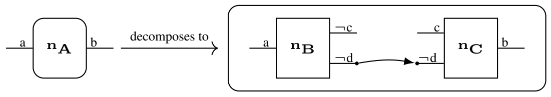

Figure 2. Illustration of a method that is supposed to implement the compound task n_A. The tasks n_B and n_C are primitive.

Concerning executability, Def. 8 requires that there is at least one state in which the respective plan's linearizations are executable, which might be considered too restrictive. The example given in Fig. 2 is not legal with respect to Def. 8, since – due to the variable c – there cannot be a state in which n_A's method's plan is executable. Since other tasks could be ordered in between n_B and n_C to support n_C's precondition c, one could also relax this criterion. This is done by the next legality criterion (for which the example is legal), as it allows that open preconditions of a method's plan may be supported by tasks at an arbitrary "position" within a plan (i.e., it does not require that a single state enables the execution of the plan's tasks).

The criterion was proposed by Yang [40, p. 14] and consists of three sub criteria. The first two imply downward compatibility, but require more. Criterion 1 demands, similar to Def. 8, that none of the tasks in the plan invalidates the compound task's effects. Criterion 2 requires that every precondition variable of the compound task has to occur in the plan in such a way that none of its sub tasks can be used to establish it. Criterion 3 is closely related to the concept of causal threats. Each precondition within the plan might not be violated by a converse effect of another task. Note that this is neither equivalent to stating that the respective plan must be free of causal threats (as the criterion must also hold in the absence of any causal links) nor that the plan is executable in any way (as none of the criteria ensures preconditions to be supported).

Definition 9 (Yang [40]). *A method $(n_c, P) \in M$ with $P = (PS, CL, \prec, \alpha)$ is called* legal *if and only if:*

1. *for all $v \in eff^+(n_c)$ (resp. $v \in eff^-(n_c)$) there exists a $ps \in PS$ with $v \in eff^+(\alpha(ps))$ (resp. $v \in eff^-(\alpha(ps))$), such that for all $ps' \in PS$, $ps' \neq ps$ holds: if $v \in eff^-(\alpha(ps'))$ (resp. $v \in eff^+(\alpha(ps'))$), then $(ps', ps) \in \prec$.*
2. *for all $v \in prec^+(n_c)$ (resp. $v \in prec^-(n_c)$) there exists a $ps \in PS$ with $v \in prec^+(\alpha(ps))$ (resp. $v \in prec^-(\alpha(ps))$), such that for all $ps' \in PS$, $ps' \neq ps$ holds: if $v \in eff^+(\alpha(ps'))$ (resp. $v \in eff^-(\alpha(ps'))$), then $(ps, ps') \in \prec$.*
3. *for all $ps \in PS$, for all $v \in prec^+(\alpha(ps))$ (resp. $v \in prec^-(\alpha(ps))$), and for all $ps' \in PS$ with $ps' \neq ps$ and $(ps, ps') \notin \prec$ it holds: if $v \in eff^-(\alpha(ps'))$ (resp. $v \in eff^+(\alpha(ps'))$), then there exists a $ps'' \in PS$, such that $\{(ps', ps''), (ps'', ps)\} \subseteq \prec$ and $v \in eff^+(\alpha(ps''))$ (resp. $v \in eff^-(\alpha(ps''))$).*

The next legality criterion is by Young et al. [41]. They argue that Yang's criterion of threat-free plans was too strong. Instead, their only requirement is that any of the compound task's preconditions "contributes" to at least one of its effects (and vice versa), which they ensure by requiring the existence of a chain of causal links within the plans connecting them with each other [41, p. 191]. Thus, our example illustrated in Fig. 2 also satisfies this criterion, but it would not do so if both the variable d and the respective causal link were not present (while assuming that n_C is still ordered behind n_B).

They model their criterion by including artificial start and end actions, which use the compound task's precondition and effect as effect and precondition, respectively, and require the causal link chains between them. Upon decomposition, those actions disappear, but the causal links involving them are reused to be linked to the plan steps that share causal links with the compound task. Those actions thus do not imply that there are states, in which the compound task's preconditions and effects hold. In the following definition, we therefor did not include the artificial start and end tasks.

Definition 10 (Young et al. [41]). *A method $(n_c, P) \in M$ with $P = (PS, CL, \prec, \alpha)$ is called* legal *if and only if:*

- *(n_c, P) is downward compatible and*
- *for each $v \in eff^+(n_c)$ (resp. $v \in eff^-(n_c)$), there is a sequence of plan steps $ps_1, \ldots, ps_k$ with $ps_i \in PS$ for $1 \leq i \leq k$, $v \in eff^+(\alpha(ps_k))$ (resp. $v \in eff^-(\alpha(ps_k))$), $v' \in (prec^+(\alpha(ps_1)) \cup prec^-(\alpha(ps_1))) \cap (prec^+(n_c) \cup prec^-(n_c))$, and a chain of causal links connecting them.*
- *for each $v \in prec^+(n_c)$ (resp. $v \in prec^-(n_c)$), there is a sequence of plan steps $ps_1, \ldots, ps_{k'}$ with $ps_i \in PS$ for $1 \leq i \leq k'$, $v \in prec^+(\alpha(ps_1))$ (resp. $v \in prec^-(\alpha(ps_1))$), $v' \in (eff^+(\alpha(ps_{k'})) \cup eff^-(\alpha(ps_{k'}))) \cap (eff^+(n_c) \cup eff^-(n_c))$, and a chain of causal links connecting them.*

Due to space restrictions, we cannot include all the work related to formalisms that allow to specify preconditions and effects for compound tasks. Concerning the surveyed legality criteria, we restricted the presentation to approaches which *explicitly* mention the demanded criteria. We further want to mention the work by Castillo et al. [14], which also fuses HTN planning with POCL planning. Since they infer the methods automatically, their plans also fulfill certain criteria, which seem to be closely related to Def. 8. Further attention deserves the *angelic semantics* by Marthi et al. [30]. Their conditions ("high-level action descriptions") take all states into account that are generated by any primitive plan that is reachable by decomposing the respective compound task. In contrast, the legality criteria surveyed before relate a compound task's preconditions and effects directly to its methods' plans (in particular to the preconditions and effects of its tasks).

4 Complexity Results

In this section we investigate the complexity of the plan verification and the plan existence problem for the hybrid planning formalism.

For some of our hardness results, we reduce a certain set of HTN planning problems to hybrid problems. Since all required results are proved or reproduced in the HTN planning framework by Geier and Bercher [19] (or based upon it), we first state a proposition that every HTN planning problem can be expressed as a hybrid planning problem with the same set of solutions and without violating any of the legality criteria for decomposition methods. Concerning notation, note that a *task network* $(T, \prec, \alpha)$ [19, Def. 1] in HTN planning is a special case of plans (cf. Def. 1), as task networks do not contain causal links. What we call plan steps is referred to as *tasks* T in HTN planning. We use both notations, depending on the context.

Theorem 1. *Let $\mathcal{P}$ be an HTN planning problem according to Def. 2 by Geier and Bercher [19]. Then, $\mathcal{P}$ can be transformed into a hybrid planning problem π according to Def. 2 that satisfies the legality criteria in Defs. 7, 8, 9, and 10, such that:*

1. ***Refinement Correspondence.*** *Let tn be a (not necessarily primitive) task network that can be obtained via decomposition in $\mathcal{P}$. Then, the plan P that is isomorphic to tn can be obtained via decomposition in π. Conversely, let P be a primitive plan that can be obtained via decomposition in π. Then, the task network tn that is isomorphic to P can be obtained via decomposition in $\mathcal{P}$.*
2. ***Solution Correspondence.*** *Let tn be a solution task network for $\mathcal{P}$. Then, for each executable task sequence $t_1, \ldots, t_n$ thereof there is a solution plan P to π containing exactly this task sequence. Conversely, let P be a solution plan for π. Then, for all linearizations $\overline{ps}$ of the plan steps of P there is a task network tn that is a solution for $\mathcal{P}$, such that $\overline{ps}$ is an executable linearization of the tasks in tn.*

Proof. Let $\mathcal{P} = (L, C, O, M, c_I, s_I)$ with L being a finite set of proposition symbols and C and O finite sets of compound and primitive task name symbols, respectively. Each primitive task name $o \in O$ has a unique operator $(prec(o), add(o), del(o)) \in (2^L)^3$ corresponding to an action without negative preconditions. M is a finite set of methods mapping compound tasks to task networks. $c_I \in C$ is the initial compound task name and $s_I \in 2^L$ the initial state.

Transformation. We transform the HTN planning problem $\mathcal{P}$ into a hybrid planning problem $\pi = (V, N_c, N_p, \delta, M', P^i)$. We define $V := L$. The initial compound task c_I and the initial state s_I of $\mathcal{P}$ are encoded in π by the initial plan $P^i = (PS^i, CL^i, \prec^i, \alpha^i)$. That is, $PS^i = \{ps_{init}, ps, ps_{goal}\}$, $CL^i = \emptyset$, $\prec^i = \{(ps_{init}, ps), (ps, ps_{goal})\}^*$, and $\alpha^i = \{(ps_{init}, init), (ps, c_I), (ps_{goal}, goal)\}$. The actions of $init$ and $goal$ are given by $\delta(init) = (\emptyset, \emptyset, s_I, V \setminus s_I)$ and $\delta(goal) = (\emptyset, \emptyset, \emptyset, \emptyset)$. We are now defining the tasks and methods of π in such a way that all methods in M' satisfy all legality criteria. We construct a model in which no compound task has preconditions or effects and all methods contain only compound tasks or at most one task. We do this by introducing an additional compound task o_{clone} for every primitive task $o \in O$ and replace all primitive tasks in every decomposition method's plan by the respective compound task. To ensure that this does not change the set of solutions, each of these compound tasks o_{clone} has exactly one decomposition method with a plan containing exactly o: Let $N_c := C \cup \{o_{clone} \mid o \in O\}$ and for all $n_c \in N_c$, let $\delta(n_c) := (\emptyset, \emptyset, \emptyset, \emptyset)$. For the primitive tasks, let $N_p := O \cup \{init, goal\}$ and for all $o \in O$, let $\delta(o) = (prec(o), \emptyset, add(o), del(o))$. The methods M' are given by

$$\begin{aligned} M' := \{(c, (T, \emptyset, \prec, \alpha')) \mid (c, (T, \prec, \alpha)) \in M, \text{ with} \\ \alpha' := \{(t, c) \mid (t, c) \in \alpha, c \in C\} \cup \\ \{(t, o_{clone}) \mid (t, o) \in \alpha, o \in O\}\} \\ \cup \{(o_{clone}, (\{ps\}, \emptyset, \emptyset, \{(ps, o)\})) \mid o \in O\} \end{aligned}$$

Legality. The downward compatibility (Def. 7) and legality criterion that requires chains of causal links (Def. 10) trivially hold for all methods, since none of the compound tasks have preconditions or effects (so, the methods' plans are not further restricted). The other legality criteria (Defs. 8 and 9) require further restrictions on the plans even if the (parent) compound task does not have preconditions or effects. It is easy to see that all these restrictions hold, since – by construction – plans only contain compound tasks or at most one task. In the first case, there are no preconditions and effects, hence all criteria hold. In the second, the preconditions or effects of the single task do not violate any of the criteria as well.

Refinement Correspondence. Let $tn_1, \ldots, tn_k$ be a sequence of task networks, such that $tn_1 = (\{t\}, \emptyset, \{(t, c_I)\})$, $tn_k = tn$, and any task network tn_j can be obtained from tn_{j-1}, $1 < j \leq k$, via decomposition. Let the corresponding sequence of decomposed task names be $c_1, \ldots, c_{k-1}$. Since no compound task $c \in C$ was removed from any of the decomposition methods' task networks, and since none of them contains causal links, there is also a sequence of plans $P_1, \ldots P_k$, such that $P_1 = P^i$ and the plan P_j results from decomposing c_{j-1} in P_{j-1}, $1 < j \leq k$. The resulting plan P_k is isomorphic to the task network tn_k except that P_k contains a compound task $o_{clone} \in N_c \setminus C$ for any primitive task $o \in O$ in tn_k. Thus, using the methods for those $o_{clone} \in N_c \setminus C$, there is a sequence of decompositions that transform P_k into P'_k with P'_k being isomorphic to tn_k. For the other direction, let P be a primitive plan that can be obtained via decomposition in π. Because the decomposition of a compound task $o_{clone} \in N_c \setminus C$ does not introduce further compound tasks, we can assume that first tasks $c_1, \ldots, c_{k-1}$, $c_i \in C$, $1 \leq i < k$ are decomposed leading to a plan P_k and then only tasks in $o_{clone} \in N_c \setminus C$ leading to a primitive plan P'_k. When decomposing $c_1, \ldots, c_{k-1}$ in $\mathcal{P}$, we obtain a task network tn_k that is isomorphic to P'_k.

Solution Correspondence. Let tn be a solution to $\mathcal{P}$. It is thus reachable via decomposition in $\mathcal{P}$. Due to the refinement correspondence, P being isomorphic to tn can be obtained via decomposition in π. Thus, for any executable linearization of the tasks of tn, P can be turned into a totally ordered solution plan P' containing exactly that sequence via ordering and causal link insertions. For the other direction, let P be a solution for π. Without loss of generality we can assume that P can be generated by first decomposing, then inserting causal links, then ordering constraints. Let P' be the last primitive plan before any ordering or causal link insertion. Due to the refinement correspondence, tn being isomorphic to P' is reachable via decomposition in $\mathcal{P}$. Then, tn contains all linearizations of the plan steps of P' (in particular the executable ones). □

Plan Verification. We now investigate how hard it is to verify whether a given plan is a solution to a hybrid planning problem. While in classical, non-hierarchical planning, this question can be answered in linear time w.r.t. the size of the input plan [16, Thm. 8], the corresponding problem is much harder in the HTN setting. Behnke et al. [9] proved that the HTN plan verification problem is $\mathbf{NP}$-$\mathbf{complete}$ even under several restrictions.

We first investigate the special case where there is no hierarchy. In HTN planning, this means to decide whether a primitive plan has an executable linearization, which is already $\mathbf{NP}$-$\mathbf{complete}$ [34, Thm. 15], [16, Thm. 8]. In hybrid planning, *all* linearizations need to be executable. The respective problem is hence equivalent to verifying whether a POCL plan is a solution to a POCL planning problem, which is commonly known to be tractable.

Theorem 2. *Let P be a plan and $\pi = (V, N_c, N_p, \delta, M, P^i)$ a hybrid planning problem without hierarchy, i.e., $N_c = M = \emptyset$. Deciding whether P is a solution to π is in* $\mathbf{P}$.

Proof. Checking that every precondition is supported by exactly one causal link can be done in linear time w.r.t. the number of all preconditions and causal links. Checking the absence of causal threats can be done in quadratic time. Let $P = (PS, CL, \prec, \alpha)$. For each causal link $(ps, v, ps') \in CL$ iterate over all plan steps $ps'' \in PS$, $ps'' \notin \{ps, ps'\}$. If none of these ps'' has an effect conflicting with v and can be ordered between ps and ps' without violating $\prec$, continue to the next causal link, otherwise fail. □

Please note that the reason why this verification problem is easier than in HTN planning cannot be attributed to the fact that compound tasks show preconditions and effects, but to the fact that in hybrid (and POCL) planning, all linearizations need to be executable, whereas HTN planning only requires that there exists one.

Next we consider the general case, in which there are no restrictions on the hierarchy. We start by showing $\mathbf{NP}$ membership.

Lemma 1. *Let P be a plan and π a hybrid planning problem. Independently of the demanded legality criteria (Def. 7 to 10), it is in* $\mathbf{NP}$ *to decide whether P is a solution to π.*

Proof sketch: For HTN planning, we showed that the corresponding verification problem is in $\mathbf{NP}$ [9, Thm. 1]. We show that the two main proof steps (not emphasizing some special cases due to lack of space) remain applicable despite the extension of the formalism to hybrid planning: First, we show that any plan that can be obtained via decomposition can be obtained by a polynomial number of methods. Second, we give a guess-and-verify algorithm that runs in $\mathbf{NP}$.

The first main step consists of five sub steps. First, we construct a so-called ε-extended planning problem π' from π that has the same set of solutions as π, but allows for shorter decomposition sequences. We call methods that contain an empty plan ε-methods. Now, for any compound task name that can be transformed into an empty plan by an arbitrary number of decomposition methods (due to ε-methods already present in M), we introduce an additional ε-method in M' of π'. For HTN planning, this can be done in $\mathbf{P}$ [9, after Def. 1]. Since causal links are not allowed to disappear upon decomposition, the same result applies to hybrid planning. Second, given a sequence $\overline{m}_1$ of methods in M' leading from the initial plan to a plan P', the methods are reordered as follows: all ε-methods immediately follow the method that inserted the plan step they erase into the plan, resulting in the sequence $\overline{m}_2$. Note that reordering these decomposition methods is possible, since all legality criteria satisfy Def. 7, downward compatibility (cf. footnote on page 4). Third, if for any non-ε-method in $\overline{m}_2$, all plan steps of its plan are thereafter erased, all those methods are replaced by one singe ε-method resulting in a shorter sequence $\overline{m}_3$. Let $\overline{P} = P_1, \ldots, P_n$ with $P_1 = P^i$ and $P_n = P'$ be the corresponding sequence of plans. Its subsequence $\overline{P}'$ that consists of the plans to which non-ε-methods are applied and P' forms a sequence with non-decreasing plan step size. Due to plateaus (which can be caused, e. g., by so-called unit-methods, which decompose a compound task into a single other task), $\overline{P}'$ can still be arbitrary long. Step four is a preparation to obtain a bound on their lengths: We reorder methods between the plateaus such that in every plateau only methods remain that decompose the plan step that is decomposed last in the respective plateau (as this step ends the plateau) resulting into $\overline{m}_4$. As argued before, reordering is also possible in the hybrid planning setting. In step five we can now shorten the new sequence of plans corresponding to $\overline{m}_4$. Every plateau in the new plan sequence can now be limited to at most $|N_c|$ plans, as otherwise cycles must occur. Because there are at most k plateaus (k being $|PS|$ of P'), we can state that the final sequence of methods $\overline{m}_5$ has at most $2k(|N_c|+1)$ non-ε-methods[‖] and thus $2k(|N_c|+1)\Delta$ methods in total, Δ being the maximal number of plan steps of the plans in the methods and P^i. We thereby conclude that if a plan P' can be obtained via decomposition in π, it can be obtained by a polynomial number of methods in π'.

For the second main step, we guess a polynomially bounded sequence of methods in π' and calculate the resulting plan P'. We then guess additional ordering constraints (bounded by k^2) and causal links (bounded by $k|V|$), insert them into P' resulting in P'', guess a bijection between the plan steps in P'' and P and verify isomorphism. We then verify executability of P in $\mathbf{P}$ (Thm. 2). □

We now show that the plan verification problem is also $\mathbf{NP}$-$\mathbf{hard}$.

Theorem 3. *Let P be a plan and π a hybrid planning problem. Independently of the demanded legality criteria (Def. 7 to 10), it is* $\mathbf{NP}$-$\mathbf{complete}$ *to decide whether P is a solution to π.*

Proof. Membership is stated in Lem. 1. For hardness, we use a corollary of HTN plan verification [9, Cor. 5]. According to this, it is $\mathbf{NP}$-$\mathbf{complete}$ to verify, given a sequence of tasks $\bar{t}$ and a totally unordered precondition- and effect-free HTN problem $\mathcal{P}$, whether there is a solution task network tn, such that $\bar{t}$ is an (executable) linearization of tn's tasks. (Note that the complexity of the problem does not stem from finding an *executable* linearization, as there are no preconditions and effects, but from finding the right decompositions leading to the desired plan. The original proof reduces vertex cover to HTN plan verification.)

Let π be a hybrid planning problem that is constructed from $\mathcal{P}$ with the properties stated in Thm. 1. Further, let P be a totally ordered plan containing $\bar{t}$ as plan step sequence. If there is a solution tn of $\mathcal{P}$, such that $\bar{t}$ is an executable linearization of tn, then we can conclude that P is a solution to π (Thm. 1). Conversely, if P is a solution to π, then there exists a solution tn to $\mathcal{P}$, such that P's plan step linearization is an executable linearization of tn (Thm. 1). □

Plan Existence. In general, HTN planning is $\mathbf{undecidable}$ [16, 19]. We now show that this also holds for hybrid planning.

Theorem 4. *Hybrid planning is* $\mathbf{undecidable}$. *That is, it is undecidable to determine whether a hybrid planning problem has a solution – no matter, which of the legality criteria of Def. 7 to 10 hold.*

Proof. Since we can encode any HTN planning problem into a solution-conserving hybrid planning problem that satisfies all legality criteria (Thm. 1), we can reduce the undecidable plan existence problem for HTN planning [19, Thm. 1] to hybrid planning. □

[‖] We previously stated a bound of only $|k|(|N_c| + 1)$ [9, Lem. 1], as we handled a special case wrong (details omitted due to space restrictions).

From this theorem we can conclude that hybrid planning is as expressive as HTN planning, since it allows to encode undecidable problems. However, HTN planning is also known to be semi-decidable (or recursively enumerable, $\mathbf{RE}$) [16, Thm. 1], which implies that for any HTN planning problem, a solution can be eventually found if one exists (while the undecidability prevents one from proving – in general – that there is no solution in case there actually is none). We now show that this is also true for hybrid planning.

Theorem 5. *Hybrid planning is* $\mathbf{semi\text{-}decidable}$*. That is, the set of all hybrid planning problems that possess a solution is in* $\mathbf{RE}$*.*

Proof. We give a partial recursive function f that, given a hybrid planning problem π, returns *true* if π has a solution and that may not halt, otherwise. We define f as the algorithm that enumerates all plans and verifies whether they solve π. It may run infinitely long, but as soon as it finds a solution, f returns *true*. Although there are infinitely many plans, enumeration is possible by starting with all plans of length two (the only tasks of which are the artificial *init* and *goal* actions) and then successively incrementing plan length. For each plan, verify in $\mathbf{NP}$ whether it is a solution (Thm. 3). □

As a further corollary from Thm. 1, it also follows that many sub classes of hybrid planning are as hard as the respective problem classes in HTN planning. Such restrictions include syntactical ones (such as totally ordered task networks [2] or delete-relaxed actions [5]) and structural restrictions on the hierarchy (such as tail-recursive or acyclic problems [2]). To formally prove this, we would have to show that the respective restrictions still hold in the hybrid planning problem after the translation process done in the proof of Thm. 1.

5 Discussion

As a corollary from the last section's results, we can observe that for the studied legality criteria, allowing preconditions and effects for compound tasks does neither increase nor decrease the expressivity of the formalism with regard to the plan existence problem in the general case. We want to emphasize that this is caused by the fact that none of the studied criteria (Def. 7 to 10) *enforces* to specify preconditions or effects for compound tasks (such as the one by Russell and Norvig [37]). Legality criteria that enforce to specify such preconditions or effects might influence the respective results and therefore also reduce expressivity, as they might prevent to specify computationally hard problems. Having *the option* to model such preconditions and effects still serves several practically relevant purposes, however. We shortly discuss some of them in this section and give pointers to the literature for further details.

As argued by Fox [17, p. 196], *"one of the strongest motivations for using some form of abstraction in planning is the observation that people use it to great effect in their problem-solving"*, which is also backed up by psychological studies [13]. Consequently, people should already be supported during the process of constructing (hierarchical) planning domains. Modeling support has attracted increased interest during the last years, which is one of the reasons that lead to the establishment of the *International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS)*[**]. Nevertheless, there is still only very little research to automatically support the modeling process, which is particularly true for hierarchical models. The tool by McCluskey and Kitchin [32] as well as GIPO [39] for hierarchical models expressed in the modeling language OCL$_h$ checks certain properties and reports violations. In analogy to the respective properties that these tools verify, the legality criteria allow to automatically verify the relationship between compound tasks and their methods' plans. As Fox points out, *"abstract plans have intentional meaning"* – and so do compound tasks. Thus, adhering some desired legality criterion is a possible way to automatically verify whether the model complies with the user's intent.

[**] http://www.icaps-conference.org/index.php/Main/Competitions

Apart from providing modeling assistance, another main purpose of being able to specify preconditions and effects for compound tasks is to exploit them during search. Reasoning about these preconditions and effects may result in a smaller search space, as irresolvable flaws, such as open preconditions, can be detected earlier, i.e., before the respective compound task is decomposed. This further allows to generate "solution" plans on different levels of abstraction. Such plans look like ordinary (primitive) solutions with the difference that some tasks are still compound. When the model fulfills further prerequisites, it is guaranteed that such abstract solutions can be refined into a primitive one [40, 30]. Then, the model is said to fulfill the downward refinement property [6].

Preconditions and effects of compound tasks can also improve plan explanations. Plan explanations as developed by Seegebarth et al. [38] give a justification about the purpose of a primitive action questioned by the user. It is based upon a sequence of arguments, each being of the form (a) "action a is required as it supports a precondition variable of another action a' by a causal link" or (b) "action a is required as it was introduced via decomposition of a compound task c". Necessity of the other task a' (resp. c) is proved similarly. Preconditions and effects of compound tasks now allow to combine these two argument types [38]. Then, the causal chain argument (a) can be extended from just primitive actions to compound tasks, as they show preconditions and effects as well, which allows for much shorter and more abstract explanations.

6 Conclusion

To finally answer the question whether compound tasks with preconditions and effects are more than just names: We can state *no* in the sense that for the criteria we studied in more detail, we were able to show that in the general case, the formalism is equally expressive (with respect to the plan existence problem) than the HTN formalism, in which compound tasks are just names. For many sub classes, however, we only showed lower bounds – upper bounds still need to be proved. It might also be that other, more restrictive, legality criteria influence the hardness of the problem, in which case we also had to state *yes*. We can already answer the question with *yes* with regard to practical considerations, such as modeling assistance: The preconditions and effects, when combined with a desired legality criterion, can be exploited to provide assistance to ensure that the methods comply with the user's intent – or at least to rule out some of the modeling flaws.

ACKNOWLEDGEMENTS

We thank Thomas Geier for discussions that helped improving this paper as well as Kathi Krammer for proof reading an early draft. This work was done within the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

We also thank Mario Schmautz for contributing to this revised version (date: 17.11.2017), most notably in Defs. 3 and 10.

REFERENCES

[1] Ron Alford, Gregor Behnke, Daniel Höller, Pascal Bercher, Susanne Biundo, and David Aha, 'Bound to plan: Exploiting classical heuristics via automatic translations of tail-recursive HTN problems', in *Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 20–28. AAAI Press, (2016).

[2] Ron Alford, Pascal Bercher, and David Aha, 'Tight bounds for HTN planning', in *Proc. of the 25th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 7–15. AAAI Press, (2015).

[3] Ron Alford, Pascal Bercher, and David Aha, 'Tight bounds for HTN planning with task insertion', in *Proc. of the 25th Int. Joint Conf. on AI (IJCAI)*, pp. 1502–1508. AAAI Press, (2015).

[4] Ron Alford, Ugur Kuter, and Dana S. Nau, 'Translating HTNs to PDDL: A small amount of domain knowledge can go a long way', in *Proc. of the 21st Int. Joint Conf. on AI (IJCAI)*, pp. 1629–1634. AAAI Press, (2009).

[5] Ron Alford, Vikas Shivashankar, Ugur Kuter, and Dana Nau, 'On the feasibility of planning graph style heuristics for HTN planning', in *Proc. of the 24th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 2–10. AAAI Press, (2014).

[6] Fahiem Bacchus and Qiang Yang, 'Downward refinement and the efficiency of hierarchical problem solving', *Artificial Intelligence*, **71**(1), 43 – 100, (1994).

[7] Patrick Bechon, Magali Barbier, Guillaume Infantes, Charles Lesire, and Vincent Vidal, 'HiPOP: Hierarchical partial-order planning', in *Proc. of the 7th Europ. Starting AI Researcher Symposium (STAIRS)*. IOS Press, (2014).

[8] Gregor Behnke, Daniel Höller, Pascal Bercher, and Susanne Biundo, 'Change the plan – how hard can that be?', in *Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 38–46. AAAI Press, (2016).

[9] Gregor Behnke, Daniel Höller, and Susanne Biundo, 'On the complexity of HTN plan verification and its implications for plan recognition', in *Proc. of the 25th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 25–33. AAAI Press, (2015).

[10] Pascal Bercher, Susanne Biundo, Thomas Geier, Thilo Hörnle, Florian Nothdurft, Felix Richter, and Bernd Schattenberg, 'Plan, repair, execute, explain - how planning helps to assemble your home theater', in *Proc. of the 24th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 386–394. AAAI Press, (2014).

[11] Susanne Biundo, Pascal Bercher, Thomas Geier, Felix Müller, and Bernd Schattenberg, 'Advanced user assistance based on AI planning', *Cognitive Systems Research*, **12**(3-4), 219–236, (2011). Special Issue on Complex Cognition.

[12] Susanne Biundo and Bernd Schattenberg, 'From abstract crisis to concrete relief – a preliminary report on combining state abstraction and HTN planning', in *Proc. of the 6th Europ. Conf. on Planning (ECP)*, pp. 157–168. AAAI Press, (2001).

[13] Richard Byrne, 'Planning meals: Problem solving on a real data-base', *Cognition*, **5**, 287–332, (1977).

[14] Luis A. Castillo, Juan Fernández-Olivares, and Antonio González, 'On the adequacy of hierarchical planning characteristics for real-world problem solving', in *Proc. of the 6th Europ. Conf. on Planning (ECP)*, pp. 169–180. AAAI Press, (2001).

[15] Filip Dvořk, Arthur Bit-Monnot, Flix Ingrand, and Malik Ghallab, 'A flexible ANML actor and planner in robotics', in *Proc. of the 2nd Workshop on Planning and Robotics (PlanRob)*, pp. 12–19, (2014).

[16] Kutluhan Erol, James A. Hendler, and Dana S. Nau, 'Complexity results for HTN planning', *Annals of Mathematics and Artificial Intelligence*, **18**(1), 69–93, (1996).

[17] Maria Fox, 'Natural hierarchical planning using operator decomposition', in *Proc. of the 4th Europ. Conf. on Planning (ECP)*, pp. 195–207. Springer, (1997).

[18] Maria Fox and Derek Long, 'Hierarchical planning using abstraction', *IEE Proc. – Control Theory Appl.*, **142**(3), 197–210, (1995).

[19] Thomas Geier and Pascal Bercher, 'On the decidability of HTN planning with task insertion', in *Proc. of the 22nd Int. Joint Conf. on AI (IJCAI)*, pp. 1955–1961. AAAI Press, (2011).

[20] Malik Ghallab, Dana S. Nau, and Paolo Traverso, *Automated Planning: Theory and Practice*, Morgan Kaufmann, 2004.

[21] Daniel Höller, Gregor Behnke, Pascal Bercher, and Susanne Biundo, 'Language classification of hierarchical planning problems', in *Proc. of the 21st Europ. Conf. on AI (ECAI)*, pp. 447–452. IOS Press, (2014).

[22] Daniel Höller, Gregor Behnke, Pascal Bercher, and Susanne Biundo, 'Assessing the expressivity of planning formalisms through the comparison to formal languages', in *Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 158–165. AAAI Press, (2016).

[23] Éric Jacopin, 'Game AI planning analytics: The case of three first-person shooters', in *Proc. of the 10th AI and Interactive Digital Entertainment Conf. (AIIDE)*, pp. 119–124. AAAI Press, (2014).

[24] Subbarao Kambhampati, 'Refinement planning as a unifying framework for plan synthesis', *AI Magazine*, **18**(2), 67–98, (1997).

[25] Subbarao Kambhampati and James A. Hendler, 'A validation-structure-based theory of plan modification and reuse', *Artificial Intelligence*, **55**, 193–258, (1992).

[26] Subbarao Kambhampati, Amol Mali, and Biplav Srivastava, 'Hybrid planning for partially hierarchical domains', in *Proc. of the 15th Nat. Conf. on AI (AAAI)*, pp. 882–888. AAAI Press, (1998).

[27] Craig A. Knoblock, 'Automatically generating abstractions for planning', *Artificial Intelligence*, **68**, 243–302, (1994).

[28] Raphaël Lallement, Lavindra De Silva, and Rachid Alami, 'HATP: An HTN planner for robotics', in *Proc. of the 2nd Workshop on Planning and Robotics (PlanRob)*, pp. 20–27, (2014).

[29] Naiwen Lin, Ugur Kuter, and Evren Sirin, 'Web service composition with user preferences', in *Proc. of the 5th Europ. Semantic Web Conf. (ESWC)*, pp. 629–643. Springer, (2008).

[30] Bhaskara Marthi, Stuart J. Russell, and Jason Wolfe, 'Angelic semantics for high-level actions', in *Proc. of the 17nd Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 232–239. AAAI Press, (2007).

[31] David McAllester and David Rosenblitt, 'Systematic nonlinear planning', in *Proc. of the 9th Nat. Conf. on AI (AAAI)*, pp. 634–639. AAAI Press, (1991).

[32] Thomas Lee McCluskey and Diane E. Kitchin, 'A tool-supported approach to engineering HTN planning models', in *In Proc. of 10th IEEE Int. Conf. on Tools with AI (ICTAI)*, pp. 272–279. IEEE, (1998).

[33] Dana S. Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, Dan Wu, Fusun Yaman, Héctor Muñoz-Avila, and J. William Murdock, 'Applications of SHOP and SHOP2', *Intelligent Systems, IEEE*, **20**, 34–41, (2005).

[34] Bernhard Nebel and Christer Bäckström, 'On the computational complexity of temporal projection, planning, and plan validation', *Artificial Intelligence*, **66**(1), 125–160, (1994).

[35] Santiago Ontañón and Michael Buro, 'Adversarial hierarchical-task network planning for complex real-time games', in *Proc. of the 24th Int. Conf. on AI (IJCAI)*, pp. 1652–1658. AAAI Press, (2015).

[36] J. Scott Penberthy and Daniel S. Weld, 'UCPOP: A sound, complete, partial order planner for ADL', in *Proc. of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pp. 103–114. Morgan Kaufmann, (1992).

[37] Stuart Russell and Peter Norvig, *Artificial Intelligence – A modern Approach*, chapter 12: Practical Planning, 367–376, Prentice-Hall, 1 edn., 1994.

[38] Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo, 'Making hybrid plans more clear to human users – a formal approach for generating sound explanations', in *Proc. of the 22nd Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 225–233. AAAI Press, (2012).

[39] Ron M. Simpson, Diane E. Kitchin, and Thomas Lee McCluskey, 'Planning domain definition using GIPO', *The Knowledge Engineering Review*, **22**, 117–134, (2007).

[40] Qiang Yang, 'Formalizing planning knowledge for hierarchical planning', *Computational Intelligence*, **6**(1), 12–24, (1990).

[41] Robert Michael Young, Martha E. Pollack, and Johanna D. Moore, 'Decomposition and causality in partial-order planning', in *Proc. of the 2nd Int. Conf. on AI Planning Systems (AIPS)*, pp. 188–193. AAAI Press, (1994).