On Hierarchical Task Networks

Andreas Herzig[1], Laurent Perrussel[1], Zhanhao Xiao[1,2]

[1] University of Toulouse, IRIT, France
[2] Dept. of CS, Western Sydney University, Australia

Abstract. In planning based on hierarchical task networks (HTN), plans are generated by refining high-level actions ('compound tasks') into lower-level actions, until primitive actions are obtained that can be sent to execution. While a primitive action is defined by its precondition and effects, a high-level action is defined by zero, one or several methods: sets of (high-level or primitive) actions decomposing it together with a constraint. We give a semantics of HTNs in terms of dynamic logic with program inclusion and propose a postulate guaranteeing soundness of action decomposition. We also discuss a completeness and a modularity postulates. We also show that hybrid planning can be analysed in the same dynamic logic framework.

Keywords: hierarchical task network; HTN planning; hybrid planning; dynamic logic

1 Introduction

The two main approaches to deterministic AI planning are classical state-based planning [15] and Hierarchical Task Network (HTN) planning [4]. The former is based on action preconditions and effects. The latter is based on domain-specific heuristics about the decomposition of high-level actions ('compound tasks') into lower-level actions, until primitive actions ('primitive tasks') are obtained. It has no generally agreed semantics [7]. We here propose a semantics in terms of an extension of Propositional Dynamic Logic PDL [8] by a program inclusion operator. This framework sheds light on a problem that had not been investigated before: the soundness of HTN domain descriptions.

Let us illustrate HTNs and the soundness issue by an abstract example. Suppose the only method for high-level action α is $\langle \alpha, \langle \{(\beta, t)\}, (t, p)\rangle\rangle$. The couple $\langle \{(\beta, t)\}, (t, p)\rangle$ is a *task network*: (β, t) instantiates the action β by the temporal label t, and the constraint (t, p) stipulates that p should be true immediately after t. So the only way to perform α is by performing β, with postcondition p. Suppose moreover that β is also a high-level action and that its only method is $\langle \beta, \langle \{(\gamma, t')\}, (t', \neg p)\rangle\rangle$. So the only way to perform β is to apply γ, with postcondition $\neg p$. No task involving α can ever be solved. We call such a HTN domain description *unsound*. It is reasonable to expect HTN domain descriptions not to contain unsound methods. This is a simple example, and more complex unsound

methods can be designed. In this paper we show that PDL provides a semantics where we can characterise sound domain descriptions. The PDL semantics also allows us to study whether the set of methods for a high-level action α is *complete*, in the sense that when the precondition of α is true then there is a method for α that is executable.

Beyond HTN planning, we can show that PDL with program inclusion also provides a semantics for hybrid planning [11]. In the latter, a domain description not only provides information about preconditions and effects of the primitive actions, but also of the high-level actions. The effect of a high-level action can be viewed as its main, *primary* effect [11]. Indeed, it is not obvious to describe all effects of a high-level action α, one of the reasons being that these effects may in particular be conditional on the refinement of α to be chosen. For example, the primary effect of the high-level action of building a house is that I have a house. Further effects may obtain, depending on whether I build the house myself or hire a builder: I either have a bad back, or an empty bank account. We therefore consider that non-primitive actions are not described by their effects but only by their postconditions.

Our paper is organised as follows. In Section 2 we introduce the language and semantics of PDL and briefly recall how classical planning can be defined in PDL. In Section 3 we introduce a PDL-based presentation of HTN planning domains. In Section 4 we recast the standard, operational definition of solutions of a HTN planning problem in PDL. In Section 5 we propose and discuss rationality principles for HTN planning domains: criteria of modularity and of soundness and completeness of action refinement. In Section **??** we represent hybrid planning problems. Section 6 discusses related work and concludes the paper.

2 Propositional Dynamic Logic PDL

In the present section we provide the necessary syntactic and semantic definitions for a version of Propositional Dynamic Logic PDL having intersection and inclusion of programs and having only boolean tests. This allows us to define classical planning problems and their solutions in PDL.

2.1 Language

Let $\mathtt{Prp}$ be a finite set of propositional variables, with typical elements $p, q, \ldots$ Let $\mathtt{Act}$ be a finite set of actions, with typical elements $\alpha, \beta, \ldots$ In examples we use capital letters for propositional variables (such as HasHouse) and small letters for actions (such as buildHouse).

Boolean formulas are defined as usual. The set of boolean formulas is noted $\mathtt{Fml}_{\mathsf{bool}}$.

Programs are defined by the following grammar:

$$\pi ::= \alpha \mid \pi;\pi \mid \pi \sqcup \pi \mid \pi \sqcap \pi \mid \pi^* \mid \varphi_0?$$

where α ranges over $\mathtt{Act}$ and φ_0 over $\mathtt{Fml}_{\mathsf{bool}}$. The programs $\pi;\pi'$, $\pi \sqcup \pi'$ and $\pi \sqcap \pi'$ are respectively the sequential, nondeterministic and parallel composition of the programs π and π'; π^* is bounded iteration of π and $\varphi_0?$ is the test of φ_0.

Then formulas are defined by:

$$\varphi ::= p \mid \bot \mid \varphi \to \varphi \mid \langle\pi\rangle\varphi \mid \pi \sqsubseteq \pi$$

The formula $\langle\pi\rangle\varphi$ reads "there is a possible execution of π after which φ is true". The formula $\pi' \sqsubseteq \pi$ reads "every execution of π' is also an execution of π", or "the effects of π' are implied by the effects of π".

Other connectives can be defined in the standard way as abbreviations, e.g. the formula $[\pi]\varphi$ abbreviates $\neg\langle\pi\rangle\neg\varphi$, and the program **while** φ_0 **do** π abbreviates $(\varphi_0?;\pi)^*;\neg\varphi_0?$.

The set of all PDL programs is noted $\mathtt{Pgm}_{\mathsf{PDL}}$ and the set of all PDL formulas is noted $\mathtt{Fml}_{\mathsf{PDL}}$. A *theory* is a set of formulas.

2.2 Semantics

A model is a triple $M = \langle W, R, V\rangle$ where W is a non-empty set of possible worlds, $R : \mathtt{Pgm}_{\mathsf{PDL}} \longrightarrow 2^{W\times W}$ associates an accessibility relation R_π to every program in $\mathtt{Pgm}_{\mathsf{PDL}}$, and $V : \mathtt{Prp} \longrightarrow 2^W$ associates a set $V(p) \subseteq W$ to every propositional variable p. The function R must satisfy some constraints:

$$\begin{aligned} R_{\pi_1;\pi_2} &= R_{\pi_1} \circ R_{\pi_2} \\ R_{\pi_1\sqcup\pi_2} &= R_{\pi_1} \cup R_{\pi_2} \\ R_{\pi_1\sqcap\pi_2} &= R_{\pi_1} \cap R_{\pi_2} \\ R_{\pi^*} &= (R_\pi)^* \\ R_{\varphi?} &= \{\langle w, w\rangle \ : \ M, w \Vdash \varphi\} \end{aligned}$$

Due to the last clause these constraints have to be defined by mutual recursion with the truth conditions for formulas:

$$\begin{array}{lll} M, w \Vdash p & \text{iff} & w \in V(p) \\ M, w \nVdash \bot & & \\ M, w \Vdash \varphi \to \varphi' & \text{iff} & M, w \nVdash \varphi \text{ or } M, w \Vdash \varphi' \\ M, w \Vdash \langle\pi\rangle\varphi & \text{iff} & M, v \Vdash \varphi \text{ for some } v \in R_\pi(w) \\ M, w \Vdash \pi \sqsubseteq \pi' & \text{iff} & R_\pi(w) \subseteq R_{\pi'}(w) \end{array}$$

where $R_\pi(w) = \{v \ : \ \langle w, v\rangle \in R_\pi\}$.

A set of formulas Γ is *true in a model* M, written $M \Vdash \Gamma$, if $M, w \Vdash \varphi$ for every $w \in W$ and $\varphi \in \Gamma$. A formula φ is a *consequence of* Γ, written $\Gamma \models \varphi$, if $M \Vdash \Gamma$ implies $M \Vdash \varphi$ for every model M. A formula φ is valid if it is a consequence of $\emptyset$. The following proposition will be useful.

Proposition 1. *The following formulas are valid:*

$$\pi' \sqsubseteq \pi \to (\langle \pi' \rangle \varphi \to \langle \pi \rangle \varphi) \qquad \pi' \sqsubseteq \pi \to (\pi' \sqcup \pi_2) \sqsubseteq (\pi \sqcup \pi_2)$$
$$(\varphi_0?; \pi) \sqsubseteq \pi' \leftrightarrow (\varphi_0 \to \pi \sqsubseteq \pi') \qquad \pi' \sqsubseteq \pi \to (\pi_1 \sqcup \pi') \sqsubseteq (\pi_1 \sqcup \pi)$$
$$\pi' \sqsubseteq \pi \to (\pi'; \pi_2) \sqsubseteq (\pi; \pi_2) \qquad \pi' \sqsubseteq \pi \to (\pi' \sqcap \pi_2) \sqsubseteq (\pi \sqcap \pi_2)$$
$$[\pi_1](\pi' \sqsubseteq \pi) \to (\pi_1; \pi') \sqsubseteq (\pi_1; \pi) \qquad \pi' \sqsubseteq \pi \to (\pi_1 \sqcap \pi') \sqsubseteq (\pi_1 \sqcap \pi)$$
$$\pi' \sqsubseteq \pi \to \big(\langle \pi' \rangle \top \to \langle \pi' \sqcap \pi \rangle \top\big)$$

Proof. We only prove the first item. For every model M and every $w \in W$, if $M, w \Vdash \pi \sqsubseteq \pi'$, then $R_\pi(w) \subseteq R_{\pi'}(w)$. Further, if $M, w \Vdash [\pi']\varphi$ then $M, v \Vdash \varphi$ for every $v \in R_\pi(w)$. So $M, w \Vdash [\pi]\varphi$ due to $R_\pi(w) \subseteq R_{\pi'}(w)$.

2.3 Classical planning

In classical planning, action preconditions are described by a mapping $\mathsf{Pre} : \mathtt{Act} \longrightarrow \mathtt{Fml}_{\mathsf{bool}}$ associating to each action a boolean precondition, and action effects are described by a mapping $\mathsf{Eff} : \mathtt{Act} \longrightarrow \mathtt{Fml}_{\mathsf{bool}}$ associating to each action its effect, where it is supposed that the effects are 'STRIPS-like': every $\mathsf{Eff}(\alpha)$ is of the form $\big(\bigwedge_{p \in \mathsf{eff}^+(\alpha)} p\big) \wedge \big(\bigwedge_{p \in \mathsf{eff}^-(\alpha)} \neg p\big)$, for some sets $\mathsf{eff}^+(\alpha)$ and $\mathsf{eff}^-(\alpha)$ such that $\mathsf{eff}^+(\alpha) \cap \mathsf{eff}^-(\alpha) = \emptyset$. The intended behavior of actions can be captured in PDL by the following theories:

$$\begin{aligned}
\mathtt{Fml}(\mathsf{Pre}) &= \{\mathsf{Pre}(\alpha) \leftrightarrow \langle \alpha \rangle \top \ : \ \alpha \in \mathtt{Act}\} \\
\mathtt{Fml}(\mathsf{Eff}) &= \{[\alpha]\mathsf{Eff}(\alpha) \ : \ \alpha \in \mathtt{Act}\} \cup \\
&\quad \{p \to [\alpha]p \ : \ p \notin \mathsf{eff}^-(\alpha)\} \cup \\
&\quad \{\neg p \to [\alpha]\neg p \ : \ p \notin \mathsf{eff}^+(\alpha)\}
\end{aligned}$$

The formulas in $\mathtt{Fml}(\mathsf{Pre})$ say that each action α is executable exactly when its precondition $\mathsf{Pre}(\alpha)$ is true. The formulas in $\mathtt{Fml}(\mathsf{Eff})$ say that the effects of each action α obtain, that the variables that are not negatively impacted by α remain true and that the variables that are not positively impacted by α remain false. Note that $\mathtt{Fml}(\mathsf{Eff})$ is finite because $\mathtt{Prp}$ is so.

A *classical planning domain* is a couple $\mathcal{D}_{\mathsf{class}} = \langle \mathsf{Pre}, \mathsf{Eff} \rangle$. We define its theory by

$$\mathtt{Fml}(\mathcal{D}_{\mathsf{class}}) = \mathtt{Fml}(\mathsf{Pre}) \cup \mathtt{Fml}(\mathsf{Eff})$$

When $M \Vdash \mathtt{Fml}(\mathcal{D}_{\mathsf{class}})$ then we say that M is a *model of* $\mathcal{D}_{\mathsf{class}}$.

A *classical planning problem* is a triple $\mathcal{P}_{\mathsf{class}} = \langle \mathcal{D}_{\mathsf{class}}, \mathsf{Init}, \mathsf{Goal} \rangle$ where $\mathcal{D}_{\mathsf{class}}$ is a classical planning domain and $\mathsf{Init}, \mathsf{Goal} \in \mathtt{Fml}_{\mathsf{bool}}$ are boolean formulas. (It is usually supposed that Init completely describes a classical valuation, but we do not need this hypothesis here.) A *solution* to $\mathcal{P}_{\mathsf{class}}$ is a sequence $\alpha_1; \cdots; \alpha_n$ of actions such that $\mathtt{Fml}(\mathcal{D}_{\mathsf{class}}) \models \mathsf{Init} \to \langle \alpha_1; \cdots; \alpha_n \rangle \mathsf{Goal}$. The following proposition rephrases solution in terms of more general consequence:

Proposition 2. *There exists a solution to* $\mathcal{P}_{\mathsf{class}}$ *if and only if*

$$\mathtt{Fml}(\mathcal{D}_{\mathsf{class}}) \models \mathsf{Init} \to \Big\langle \Big(\bigsqcup_{\alpha \in \mathtt{Act}} \alpha \Big)^* \Big\rangle \mathsf{Goal}.$$

3 HTN planning domains in PDL

HTN planning presupposes that the set of actions $\mathtt{Act}$ is partitioned into two sets: the set of primitive actions $\mathtt{Act}_0$ and the set of high-level actions $\mathtt{Act} \setminus \mathtt{Act}_0$. We use α, β for arbitrary elements of $\mathtt{Act}$ as before and use $a, b, \ldots$ for typical elements of $\mathtt{Act}_0$. A *primitive plan* is a sequence of primitive actions. Similarly, a *primitive program* is a program where only elements of $\mathtt{Act}_0$ occur.

As we have said in the introduction, standard presentations of HTN planning contain explicit descriptions of preconditions and effects only for the primitive actions. We here suppose that *all* actions have preconditions and effects. While we require the effects to be STRIPS-like for the primitive actions, we allow high-level actions to have any boolean formula as an effect. For example, the effect of the high-level action of travelling abroad could be a formula such as $\neg \mathsf{InFrance} \wedge (\mathsf{InGermany} \vee \mathsf{InChina} \vee \ldots)$.

3.1 HTN planning domains

An *HTN planning domain* is a couple $\mathcal{D}_{\mathsf{htn}} = \langle \mathsf{Pre}, \mathsf{Eff}, \mathsf{Ref} \rangle$ where

$$\begin{aligned} \mathsf{Pre} &: \mathtt{Act} \longrightarrow \mathtt{Fml}_{\mathsf{bool}} \\ \mathsf{Eff} &: \mathtt{Act} \longrightarrow \mathtt{Fml}_{\mathsf{bool}} \\ \mathsf{Ref} &: \mathtt{Act} \longrightarrow 2^{\mathtt{Pgm}_{\mathsf{PDL}}} \end{aligned}$$

and where the effect function Eff is STRIPS-like for primitive actions: for every $a \in \mathtt{Act}_0$, $\mathsf{Eff}(a)$ is of the form $\big(\bigwedge_{p \in \mathsf{eff}^+(a)} p\big) \wedge \big(\bigwedge_{p \in \mathsf{eff}^-(a)} \neg p\big)$, for some sets $\mathsf{eff}^+(a)$ and $\mathsf{eff}^-(\alpha)$ such that $\mathsf{eff}^+(a) \cap \mathsf{eff}^-(a) = \emptyset$. We also suppose that there is no action α such that $\mathsf{Eff}(\alpha)$ is equivalent to $\bot$. The refinement function Ref associates to each high-level action α the ways α can be accomplished: the set of programs $\mathsf{Ref}(\alpha)$ refining α. The refinement function Ref must be such that primitive actions cannot be refined: we suppose that $\mathsf{Ref}(a) = \emptyset$ for every $a \in \mathtt{Act}_0$.

Example 1. Let us take up the abstract example of the introduction where the high-level action α has postcondition p and its refinement β has postcondition $\neg p$. Let us suppose for simplicity that there are no other postcondition.

$$\begin{array}{lll} \mathsf{Pre}(\alpha) = \varphi_\alpha & \mathsf{Eff}(\alpha) = p & \mathsf{Ref}(\alpha) = \{\beta\} \\ \mathsf{Pre}(\beta) = \varphi_\beta & \mathsf{Eff}(\beta) = \neg p & \mathsf{Ref}(\beta) = \{\pi_\beta\} \end{array}$$

When φ_α is satisfiable then we expect the above HTN specification to be unsound (whatever π_β is). That is, there is some logical inconsistency between the effects of the actions. When φ_α is unsatisfiable then the specification is sound, but action α can never be refined: so $\mathsf{Ref}(\alpha)$ could as well be empty, and the specification has a redundant element.

Example 2. A typical example that can be found in almost all papers on HTN planning is that of an agent travelling from A to B. Ours here is inspired from

$\mathsf{Pre}(\mathsf{goAB}) = \mathsf{AtA}$	$\mathsf{Ref}(\mathsf{goAB}) = \{\mathsf{goFootAB}, \mathsf{goTaxiAB}\}$
$\mathsf{Eff}(\mathsf{goAB}) = \mathsf{AtB}$	
$\mathsf{Pre}(\mathsf{goFootAB}) = \mathsf{AtA}$	$\mathsf{Ref}(\mathsf{goFootAB}) = \{\textbf{while } \neg\mathsf{AtB} \textbf{ do } \mathsf{walk}\}$
$\mathsf{Eff}(\mathsf{goFootAB}) = \mathsf{AtB}$	
$\mathsf{Pre}(\mathsf{walk}) = \neg(\mathsf{AtA} \wedge \mathsf{AtB})$	$\mathsf{Ref}(\mathsf{walk}) = \{\mathsf{walkA}, \mathsf{walkB}\}$
$\mathsf{Eff}(\mathsf{walk}) = \mathsf{AtA} \vee \mathsf{AtB}$	
$\mathsf{Pre}(\mathsf{goTaxiAB}) = \mathsf{AtA}$	$\mathsf{Ref}(\mathsf{goTaxiAB}) = \{(\mathsf{rideTaxiAB}; \mathsf{pay}), (\mathsf{Flat?}; \mathsf{pay}; \mathsf{rideTaxiAB})\}$
$\mathsf{Eff}(\mathsf{goTaxiAB}) = \mathsf{AtB}$	

$\mathsf{Pre}(\mathsf{pay}) = \mathsf{Money}$	$\mathsf{Eff}(\mathsf{pay}) = \neg\mathsf{Money}$	$\mathsf{Ref}(\mathsf{pay}) = \emptyset$
$\mathsf{Pre}(\mathsf{rideTaxiAB}) = \mathsf{AtA}$	$\mathsf{Eff}(\mathsf{rideTaxiAB}) = \mathsf{AtB}$	$\mathsf{Ref}(\mathsf{rideTaxiAB}) = \emptyset$
$\mathsf{Pre}(\mathsf{walkA}) = \top$	$\mathsf{Eff}(\mathsf{walkA}) = \mathsf{AtA}$	$\mathsf{Ref}(\mathsf{walkA}) = \emptyset$
$\mathsf{Pre}(\mathsf{walkB}) = \top$	$\mathsf{Eff}(\mathsf{walkB}) = \mathsf{AtB}$	$\mathsf{Ref}(\mathsf{walkB}) = \emptyset$

Table 1. A HTN planning domain $\mathcal{D}^{\mathsf{AB}}_{\mathsf{htn}}$: traveling from A to B

[5] and is described by the domain of Table 1. The primitive actions are $\mathtt{Act}_0 = \{\mathsf{pay}, \mathsf{rideTaxiAB}, \mathsf{walkA}, \mathsf{walkB}\}$. The only refinement of going by foot from A to B is to walk until the destination is reached, and going by taxi has two possible refinements: either ride and then pay, or, if there is a flat rate, first pay and then ride. Observe that the effects of the goAB action do not mention the possible effect $\neg$Money, which is conditional on whether it is refined to goFootAB or to goTaxiAB.

Observe that the refinement of goFootAB in Example 2 takes advantage of the rich language of programs of PDL: it cannot be expressed in standard presentations of HTN planning. It is clear that HTN methods with linear constraints can be expressed by sequential composition and tests. As to more general constraints, we have to leave the exact correspondence to future work and just note that the PDL program operators are expressive enough to capture the standard examples in the literature.

3.2 Expressing HTN planning domains in PDL

The intended behavior of actions is captured in PDL by the following theories:

$$\begin{aligned}
\mathtt{Fml}(\mathsf{Pre}) &= \{\langle\alpha\rangle\top \leftrightarrow \mathsf{Pre}(\alpha) \ : \ \alpha \in \mathtt{Act}\} \\
\mathtt{Fml}(\mathsf{Eff}) &= \{[\alpha]\mathsf{Eff}(\alpha) \ : \ \alpha \in \mathtt{Act}\} \cup \\
&\quad \{p \to [a]p \ : \ a \in \mathtt{Act}_0 \text{ and } p \notin \mathsf{eff}^-(a)\} \cup \\
&\quad \{\neg p \to [a]\neg p \ : \ a \in \mathtt{Act}_0 \text{ and } p \notin \mathsf{eff}^+(a)\} \\
\mathtt{Fml}(\mathsf{Ref}) &= \{\langle\alpha\rangle\top \to \pi \sqsubseteq \alpha \ : \ \alpha \in \mathtt{Act}, \pi \in \mathsf{Ref}(\alpha)\}
\end{aligned}$$

Note that all these sets are finite because $\mathtt{Act}$ and $\mathtt{Prp}$ are so. So primitive actions behave like STRIPS actions, while the description of high-level actions are less constrained, leaving room for side effects and conditional effects. We define the theory of an HTN planning domain by:

$$\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) = \mathtt{Fml}(\mathsf{Pre}) \cup \mathtt{Fml}(\mathsf{Eff}) \cup \mathtt{Fml}(\mathsf{Ref}).$$

When $M \Vdash \mathtt{Fml}(\mathcal{D}_{\mathsf{htn}})$ then we say that M is a *model of* $\mathcal{D}_{\mathsf{htn}}$.

We now list some properties of HTN theories that will be useful in the sequel.

Proposition 3. *Let $\alpha \in \mathtt{Act}$. Then*

$$\mathtt{Fml}(\mathsf{Eff}) \models \pi \sqsubseteq \alpha \to \big[\pi\big]\mathsf{Eff}(\alpha).$$

Proof. Suppose M is a model of $\mathtt{Fml}(\mathsf{Eff})$ and $M, w \Vdash \pi \sqsubseteq \alpha$, i.e., $R_\pi(w) \subseteq R_\alpha(w)$. As $M, w \Vdash [\alpha]\mathsf{Eff}(\alpha)$ we have $M, v \Vdash \mathsf{Eff}(\alpha)$ for every $v \in R_\pi(w)$. So $M, w \Vdash [\pi]\mathsf{Eff}(\alpha)$.

The next result says that primitive actions behave in a deterministic way.

Proposition 4. *Let $a \in \mathtt{Act}_0$ be primitive and let $\varphi_0 \in \mathtt{Fml}_{\mathsf{bool}}$ be boolean. Then*

$$\mathtt{Fml}(\mathsf{Eff}) \models \langle a\rangle\varphi_0 \to [a]\varphi_0.$$

Proof. Suppose M is a model of $\mathtt{Fml}(\mathsf{Eff})$. Then, for any w in M:

$$M, w \Vdash [a]\Big(\big(\bigwedge_{p\in\mathsf{eff}^+(a)} p\big) \wedge \big(\bigwedge_{p\in\mathsf{eff}^-(a)} \neg p\big)\Big) \wedge \big(\bigwedge_{p\notin\mathsf{eff}^-(a)} p \to [a]p\big) \wedge \big(\bigwedge_{p\notin\mathsf{eff}^+(a)} \neg p \to [a]\neg p\big)$$

It is easy to check that the valuation associated to each possible world $v \in R_a(w)$ is unique: if $p \in \mathsf{eff}^+(a) \cup \mathsf{eff}^-(a)$ then the truth value of p is defined with respect to the effect functions; if $p \notin \mathsf{eff}^+(a) \cup \mathsf{eff}^-(a)$ then the truth value of p in v is determined by its truth value at w. So if $M, v \Vdash \varphi_0$ holds for some $v \in R_a(w)$ then $M, v \Vdash \varphi_0$ holds for every $v \in R_a(w)$.

Observe that due to Proposition 1,

$$\mathtt{Fml}(\mathsf{Eff}) \models (a_1;\cdots;a_n)\sqsubseteq\pi \to \big(\langle a_1;\cdots;a_n\rangle\top \to \langle (a_1;\cdots;a_n) \sqcap \pi\rangle\top\big).$$

always holds. The converse implication fails to hold: $\mathtt{Fml}(\mathsf{Eff}) \not\models \langle (a_1;\cdots;a_n) \sqcap \pi\rangle\top \to (a_1;\cdots;a_n)\sqsubseteq\pi$ because a possible world may have more than one $R_{a_1;\cdots a_n}$-successor.

4 HTN planning problems and their solutions in PDL

We now define HTN planning problems and their solutions.

4.1 HTN planning problems

A *HTN planning problem* is a triple $\mathcal{P}_{\mathsf{htn}} = \langle \mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi \rangle$ where $\mathcal{D}_{\mathsf{htn}}$ is an HTN planning domain, $\mathsf{Init} \in \mathtt{Fml}_{\mathsf{bool}}$ is a boolean formula, and $\pi \in \mathtt{Pgm}_{\mathsf{PDL}}$ is a program. (It is usually supposed that Init is a complete description of a state, but we do not need that requirement here.) An example of a HTN planning problem is a triple $\langle \mathcal{D}^{\mathsf{AB}}_{\mathsf{htn}}, \mathsf{Init}, \mathsf{goAB} \rangle$ where $\mathsf{Init} = \mathsf{AtA} \wedge \neg\mathsf{AtB} \wedge \mathsf{Money} \wedge \neg\mathsf{Flat}$.

4.2 Solutions of a HTN planning problem

The definitions of solutions of HTN planning problems that can be found in the literature are mainly in terms of fixed-points [4]. These solutions are basically obtained by refining the 'goal program' step-by-step until a primitive program is obtained. We now recast this definition in PDL, in three steps.

First, for a primitive program π_0 we can define its *completion* as follows:

$$\mathsf{compl}(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi_0) = \{a_1;\cdots;a_n \ : \ \mathtt{Fml}(\mathsf{Eff}) \models \mathsf{Init} \rightarrow \langle (a_1;\cdots;a_n) \sqcap \pi_0 \rangle \top\}$$

So the completion of a primitive program is the set of all primitive plans that are compatible with it.

Second, the *reduction* of a program π in an HTN planning domain $\mathcal{D}_{\mathsf{htn}}$ is defined by:

$$\mathsf{red}(\mathcal{D}_{\mathsf{htn}}, \pi) = \{\pi^{\alpha}_{\mathsf{Pre}(\alpha)?;\pi'} \ : \ \alpha \text{ occurs in } \pi \text{ and } \pi' \in \mathsf{Ref}(\alpha)\}$$

where $\pi^{\alpha}_{\mathsf{Pre}(\alpha)?;\pi'}$ is obtained from π by replacing some occurrence of α in π by $\mathsf{Pre}(\alpha)?;\pi'$. Observe that when π is a primitive program then $\mathsf{red}(\mathcal{D}_{\mathsf{htn}}, \pi) = \emptyset$. The function $\mathsf{red}(\mathcal{D}_{\mathsf{htn}}, \pi)$ behaves in a way such that, under $\mathcal{D}_{\mathsf{htn}}$, only programs included in π are produced. This is stated as the next result.

Proposition 5. *If* $\pi' \in \mathsf{red}(\mathcal{D}_{\mathsf{htn}}, \pi)$ *then* $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \pi' \sqsubseteq \pi$.

Proof. The proof is by induction on the structure of π and uses Proposition 1. For the base case π is one action α and $\alpha' \in \mathsf{Ref}(\alpha)$ and then $\pi' = \mathsf{Pre}(\alpha)?;\alpha'$. As $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \langle\alpha\rangle\top \rightarrow \alpha' \sqsubseteq \alpha$ and $\mathsf{Pre}(\alpha) \leftrightarrow \langle\alpha\rangle\top$, then $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Pre}(\alpha) \rightarrow \alpha' \sqsubseteq \alpha$. By Proposition 1, $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Pre}(\alpha)?;\alpha' \sqsubseteq \alpha$. So $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \pi' \sqsubseteq \pi$.

For the induction step, we only prove the case $\pi = \pi_1;\pi_2$ such that $\pi'_1;\pi_2 \in \mathsf{red}(\mathcal{D}_{\mathsf{htn}}, \pi)$. Suppose the induction hypothesis holds that if $\pi''_1 \in \mathsf{red}(\mathcal{D}_{\mathsf{htn}}, \pi_1)$ then $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \pi''_1 \sqsubseteq \pi_1$. According to the definition of red, we have $\pi'_1 \in \mathsf{red}(\mathcal{D}_{\mathsf{htn}}, \pi_1)$. Then $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \pi'_1 \sqsubseteq \pi_1$. Then by Proposition 1, $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \pi'_1;\pi_2 \sqsubseteq \pi_1;\pi_2$.

Finally, the *solutions* of an HTN planning problem are primitive plans that defined recursively as follows:

$$\mathsf{sol}^1(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi) = \begin{cases} \mathsf{compl}(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi) & \text{if } \pi \text{ is primitive} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathsf{sol}^{k+1}(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi) = \mathsf{sol}^k(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi) \cup \bigcup_{\pi' \in \mathsf{red}(\mathcal{D}_{\mathsf{htn}}, \pi)} \mathsf{sol}^k(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi')$$

$$\mathsf{sol}(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi) = \bigcup_k \mathsf{sol}^k(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi)$$

Now we connect solutions of HTN planning problems and consequence in PDL:

Theorem 1. *Let* $\mathcal{P}_{\mathsf{htn}} = \langle \mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi \rangle$ *be an HTN planning problem. Then*

$$a_1;\cdots;a_n \in \mathsf{sol}(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi) \quad \textit{implies} \quad \mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Init} \to \langle (a_1;\cdots;a_n) \sqcap \pi \rangle \top.$$

Proof. The proof is by induction on the number of iterations that are required to obtain the solution $a_1;\cdots;a_n$. For the base case $k = 1$, when $a_1;\cdots;a_n \in \mathsf{sol}^1(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi)$ then π must be a primitive plan. It follows by the definition of compl that $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Init} \to \langle (a_1;\cdots;a_n) \sqcap \pi \rangle \top$.

For the induction step, suppose the induction hypothesis holds up to k steps, i.e., $a_1;\cdots;a_n \in \mathsf{sol}^k(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi)$ implies $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Init} \to \langle (a_1;\cdots;a_n) \sqcap \pi \rangle \top$. Suppose $a_1;\cdots;a_n$ is in $\mathsf{sol}^{k+1}(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi)$, but not in $\mathsf{sol}^k(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi)$. Then by definition of sol^k there must exist a $\pi' \in \mathsf{red}(\mathcal{D}_{\mathsf{htn}}, \pi)$ such that $a_1;\cdots;a_n$ is in $\mathsf{sol}^k(\mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi')$. By induction hypothesis we have $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Init} \to \langle (a_1;\cdots;a_n) \sqcap \pi' \rangle \top$. As $\pi' \in \mathsf{red}(\mathcal{D}_{\mathsf{htn}}, \pi)$, by Proposition 5 we have $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \pi' \sqsubseteq \pi$. Therefore by Proposition 1 we have $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Init} \to \langle (a_1;\cdots;a_n) \sqcap \pi \rangle \top$.

Corollary 1. *Let* $\mathcal{P}_{\mathsf{htn}} = \langle \mathcal{D}_{\mathsf{htn}}, \mathsf{Init}, \pi \rangle$ *be an HTN planning problem. If* $\mathcal{P}_{\mathsf{htn}}$ *has a solution then*

$$\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Init} \to \langle \big(\bigsqcup \mathtt{Act}_0 \big)^* \sqcap \pi \rangle \top.$$

Proof. By the valid formula $\langle a_1;\cdots;a_n \rangle \varphi \to \langle (\bigsqcup \mathtt{Act}_0)^* \rangle \varphi$.

The converse of Theorem 1 and its corollary do not hold. To see this consider Example 2, where $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}})^{\mathsf{AB}} \models \mathsf{AtA} \to \langle \mathsf{rideTaxiAB} \sqcap \mathsf{goAB} \rangle \top$ but $\mathsf{rideTaxiAB} \notin \mathsf{sol}(\mathcal{D}_{\mathsf{htn}}^{\mathsf{AB}}, \mathsf{AtA}, \mathsf{goAB})$.

5 Rationality postulates for HTN planning

We now formulate and discuss several postulates that, we claim, reasonable HTN planning domains should satisfy.

5.1 Modularity

Intuitively, planning domains should satisfy several properties that can be related to the concept of modularity. Such principles were studied in the reasoning about actions literature [9, 10, 19]. One of these principles says that Pre contains all information about action executability. (This hypothesis of complete information is also made by Reiter w.r.t. his Poss predicate specifying action preconditions [18]). Our Explicit Executability Constraint (**EE**) states this principle as follows:

$$\text{If } \mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \langle\alpha\rangle\top \leftrightarrow \varphi \text{ then } \mathtt{Fml}(\mathsf{Pre}) \models \langle\alpha\rangle\top \leftrightarrow \varphi. \qquad (\mathbf{EE})$$

Let us revisit our introduction example to illustrate this constraint.

Example 3. The planning domain of Example 1 is captured by

$$\begin{aligned}\mathtt{Fml}(\mathsf{Pre}) &= \{\langle\alpha\rangle\top \leftrightarrow \varphi_\alpha,\ \langle\beta\rangle\top \leftrightarrow \varphi_\beta\}\\ \mathtt{Fml}(\mathsf{Eff}) &= \{[\alpha]p,\ [\beta]\neg p\}\\ \mathtt{Fml}(\mathsf{Ref}) &= \{\langle\alpha\rangle\top \to \beta\sqsubseteq\alpha,\ \langle\beta\rangle\top \to \pi_\beta\sqsubseteq\beta\}\end{aligned}$$

Let $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) = \mathtt{Fml}(\mathsf{Pre}) \cup \mathtt{Fml}(\mathsf{Eff}) \cup \mathtt{Fml}(\mathsf{Ref})$ and M be a model such that $M \Vdash \mathtt{Fml}(\mathcal{D}_{\mathsf{htn}})$. Suppose M contains a possible world w such that $M, w \Vdash \varphi_\alpha$. Then $R_\alpha(w)$ is non-empty due to formula $\langle\alpha\rangle\top \leftrightarrow \varphi_\alpha$ in $\mathtt{Fml}(\mathsf{Pre})$. Due to formula $\langle\alpha\rangle\top \to \beta\sqsubseteq\alpha$ in $\mathtt{Fml}(\mathsf{Ref})$ we moreover have $M, w \Vdash \beta\sqsubseteq\alpha$, and therefore $R_\alpha(w) \subseteq R_\beta(w)$. However, such a w cannot exist due to $\mathtt{Fml}(\mathsf{Eff})$. It follows that for every model M such that $M \Vdash \mathtt{Fml}(\mathcal{D}_{\mathsf{htn}})$ we have $M \Vdash \neg\varphi_\alpha$, that is, $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \langle\alpha\rangle\top \leftrightarrow \bot$. So $\mathcal{D}_{\mathsf{htn}}$ violates Constraint (**EE**) as we have $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \langle\alpha\rangle\top \leftrightarrow \bot$ but we don't have $\mathtt{Fml}(\mathsf{Pre}) \models \langle\alpha\rangle\top \leftrightarrow \bot$.

A second principle that should hold is that information about refinement should not be relevant for the status of primitive formulas, i.e., formulas that only concern primitive actions. A *primitive formula* φ_0 is a $\mathtt{Fml}_{\mathsf{PDL}}$-formula where all programs are primitive. Primitive Modularity Constraint (**PM**) states this principle as follows:

$$\forall\varphi_0 \in \mathtt{Fml}_{\mathsf{bool}},\ \text{if } \mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \varphi_0 \text{ then } \mathtt{Fml}(\mathsf{Pre}) \cup \mathtt{Fml}(\mathsf{Eff}) \models \varphi_0. \qquad (\mathbf{PM})$$

The following example shows that Constraint (**PM**) is not always satisfied.

Example 4. Let $\mathtt{Act}_0 = \{a\}$ and let $\mathtt{Act} = \mathtt{Act}_0 \cup \{\alpha, \beta\}$. Let $\mathcal{D}_{\mathsf{htn}}$ be a planning domain as captured by the following formulas:

$$\begin{aligned}\mathtt{Fml}(\mathsf{Pre}) &= \{\langle\alpha\rangle\top \leftrightarrow \top,\ \langle\beta\rangle\top \leftrightarrow \top,\ \langle a\rangle\top \leftrightarrow \top\}\\ \mathtt{Fml}(\mathsf{Eff}) &= \{[a]\top,\ [\alpha]p,\ [\beta]\neg p\} \cup \{p \to [a]p\ :\ p \in \mathtt{Prp}\} \cup \{\neg p \to [a]\neg p\ :\ p \in \mathtt{Prp}\}\\ \mathtt{Fml}(\mathsf{Ref}) &= \{\langle\alpha\rangle\top \to a\sqsubseteq\alpha,\ \langle\beta\rangle\top \to a\sqsubseteq\beta\}\end{aligned}$$

We have $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models a\sqsubseteq\alpha \wedge a\sqsubseteq\beta$ due to $\mathtt{Fml}(\mathsf{Pre})$ and $\mathtt{Fml}(\mathsf{Ref})$. By Proposition 1 we then have $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models [a]p \wedge [a]\neg p$ due to $\mathtt{Fml}(\mathsf{Eff})$, and therefore $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \bot$ due to $\mathtt{Fml}(\mathsf{Pre})$. However, it is not the case that $\mathtt{Fml}(\mathsf{Pre}) \cup \mathtt{Fml}(\mathsf{Eff}) \models \bot$ and consequently Constraint (**PM**) is violated.

5.2 Sound refinability

We now formulate a soundness postulate on refinements. It requires that when α is executable then every possible refinement of α guarantees that the effects of α obtain. This is conditional on the precondition of α: if they are false then there is no point in refining α and π may have arbitrary consequences.

Definition 1. *Given a model M and world w in M, we say that α is* soundly refinable at (M, w) *if and only if either $M, w \not\Vdash \mathsf{Pre}(\alpha)$ or for every $\pi \in \mathsf{Ref}(\alpha)$ and $v \in R_\pi(w)$, $M, v \Vdash \mathsf{Eff}(\alpha)$.*

Theorem 2. *Let $\mathcal{D}_{\mathsf{htn}}$ be an HTN domain. An action $\alpha \in \mathtt{Act}$ is soundly refinable at every pointed model (M, w) if and only if*

$$\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Pre}(\alpha) \to \big[\bigsqcup \mathsf{Ref}(\alpha)\big]\mathsf{Eff}(\alpha).$$

Proof. Suppose M is a model of $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}})$ and $M, w \Vdash \mathsf{Pre}(\alpha)$. For every $\pi \in \mathsf{Ref}(\alpha)$, we have $R_\pi(w) \subseteq R_\alpha(w)$ because of $\mathtt{Fml}(\mathsf{Ref})$. As for every $v \in R_\alpha(w)$, $M, v \Vdash \mathsf{Eff}(\alpha)$, we have $M, w \Vdash [\pi]\mathsf{Eff}(\alpha)$ for every $\pi \in \mathsf{Ref}(\alpha)$. So $M, w \Vdash \big[\bigsqcup \mathsf{Ref}(\alpha)\big]\mathsf{Eff}(\alpha)$.

The other way round .....

Clearly, a reasonable HTN domain should require every action is soundly refinable. Notice that soundness is not related to the modularity principles: executability conditions may be explicit but unsound at the same time.

5.3 Complete refinability

Symmetrically to sound refinability, one may formulate a postulate of complete refinability: when the precondition of a high-level action is true then it should be refinable in some way.

Definition 2. *Given a model M and world w in M, we say that high-level action $\alpha \in \mathtt{Act} \setminus \mathtt{Act}_0$ is* completely refinable at (M, w) *if and only if either $M, w \not\Vdash \mathsf{Pre}(\alpha)$ or there is a $\pi \in \mathsf{Ref}(\alpha)$ such that $R_\pi(w)$ is not empty.*

In other words, in every possible, as long as the precondition of α is true, then one of the programs refining α is executable.

Theorem 3. *Let $\mathcal{D}_{\mathsf{htn}}$ be an HTN domain. An action $\alpha \in \mathtt{Act} \setminus \mathtt{Act}_0$ is completely refinable at every pointed model (M, w) if and only if*

$$\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Pre}(\alpha) \to \big\langle \bigsqcup \mathsf{Ref}(\alpha)\big\rangle \top.$$

Proof. Suppose M is a model of $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}})$. Assume α is completely refinable at (M, w) and $M, w \Vdash \mathsf{Pre}(\alpha)$. Then there is a $\pi \in \mathsf{Ref}(\alpha)$ such that $R_\pi(w)$ is not empty. So $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \langle \pi \rangle \top$. By the existence of π, we have $\big\langle \bigsqcup \mathsf{Ref}(\alpha)\big\rangle \top$.

Assume $\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \mathsf{Pre}(\alpha) \to \big\langle \bigsqcup \mathsf{Ref}(\alpha)\big\rangle \top$. If $M, w \not\Vdash \mathsf{Pre}(\alpha)$ then α is completely refinable at (M, w). Else as $\big\langle \bigsqcup \mathsf{Ref}(\alpha)\big\rangle \top$, we have one $\pi \in \mathsf{Ref}(\alpha)$ such that $M, w \Vdash \langle \pi \rangle \top$. It means $R_\pi(w)$ is not empty.

As discussed in [11], even when some refinement is physically possible, there may exist reasons for not including them in the Ref function. There are two possible such reasons: either the refinement is legally impossible, or it is not preferred. This former case of incompleteness can be illustrated with the help of Example 2. Plan rideTaxiAB of just taking the taxi without paying also achieves the effect of goAB. However, the domain designer did not want to allow such a refinement and deliberately omitted rideTaxiAB in $\mathsf{Ref}(\mathsf{goAB})$[3].

[4] Actually, Myers [14] was the first to advocate augmenting HTN planning with hard constraints to choose the refinement of actions. Some HTN preference-based planners that find a most preferred solution provided an HTN problem with user's preference have been developed subsequently [16, 12, 17, 13, 2].

We conclude this section by showing that complete refinability can be weakened, viz. by requiring that an executable high-level α must be refinable *unless there is no primitive plan achieving the effect of* α. In formulas, we require

$$\mathtt{Fml}(\mathcal{D}_{\mathsf{htn}}) \models \big(\mathsf{Pre}(\alpha) \wedge \big\langle \big(\bigsqcup \mathtt{Act}_0\big)^* \big\rangle \mathsf{Eff}(\alpha)\big) \rightarrow \big\langle \bigsqcup \mathsf{Ref}(\alpha) \big\rangle \top.$$

This is similar to what is called *planner completeness* in [11], which, as we understand it, requires that every solution that can be obtained by a classical planner is also obtainable by the HTN planner. This requirement is different from what is called *schema completeness* in [11], which is difficult to capture formally: it basically requires that Ref lists all refinements that are intuitively desirable.

6 Discussion and Conclusion

Our work is related to the representation of HTN in Situation Calculus by Baral and Son [3] who extended the high-level action programming language ConGolog by adding a special HTN construct. Later Gabaldon [6] encoded HTN planning problem by Golog and ConGolog by means of the their **proc** operator, without adding a HTN construct. In both cases, they did not question the properties of the HTN planning domain by taking advantage of the reasoning potential of the Situation Calculus. More recently Goldman gave a semantics in terms of ConGolog [7]. In [1], a restricted class of HTN planning domains was translated into PDDL: the constraint about the decomposition of a high-level action concerns the total sequential execution of lower-level actions. [5]

In this paper we proposed a representation of HTN in PDL augmented by program inclusion. Taking advantage of the PDL program operators, we can represent more general HTN decomposition methods compared with the standard

[3] It may be argued that non-refinability of goAB to unique action rideTaxiAB without pay could as well be achieved by adding Money to the precondition of rideTaxiAB, making it thus a legal taxi riding action.

[4] **AH:** paragraph to be improved (Reviewer 3); drop? (not clear how to integrate these preferences over programs into PDL)

[5] **AH:** improve formulation

HTN planner, such as the iteration of actions. Moreover, we discussed postulates that a reasonable HTN planning domain should own and gave soundness and completeness principles for HTN planning domains and formalise them in PDL.

Still, many questions are open. First, we have proposed some rationality principles and some of them deserve additional investigation. Our aim is to exhibit representation theorems [6] for the proposed constraints and also for soundness and completeness; the interplay between all these rationality principles should be investigated. Second, we want to define further rationality principles in order to handle preference-based HTN planning.

Acknowledgements

Thanks are due to the JELIA 2016 reviewers whose comments allowed us to improve the presentation of the paper.

References

1. Alford, R., Kuter, U., Nau, D.S.: Translating htns to pddl: A small amount of domain knowledge can go a long way. In: IJCAI. pp. 1629–1634 (2009)
2. Baier, J.A., Sohrabi, S., McIlraith, S.A.: Htn planning with preferences. In: 21st Int. Joint Conf. on Artificial Intelligence. pp. 1790–1797 (2009)
3. Baral, C., Son, T.C.: Extending congolog to allow partial ordering. In: International Workshop on Agent Theories, Architectures, and Languages. pp. 188–204. Springer (1999)
4. Erol, K., Hendler, J., Nau, D.S.: HTN planning: Complexity and expressivity. In: AAAI. vol. 94, pp. 1123–1128 (1994)
5. Erol, K., Hendler, J., Nau, D.S.: Complexity results for htn planning. Annals of Mathematics and Artificial Intelligence 18(1), 69–93 (1996)
6. Gabaldon, A.: Programming hierarchical task networks in the situation calculus. In: AIPS02 Workshop on On-line Planning and Scheduling (2002)
7. Goldman, R.P.: A semantics for HTN methods. In: Gerevini, A., Howe, A.E., Cesta, A., Refanidis, I. (eds.) Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009. AAAI (2009), http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/750
8. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)
9. Herzig, A., Perrussel, L., Varzinczak, I.J.: Elaborating domain descriptions. In: Brewka, G., Coradeschi, S., Perini, A., Traverso, P. (eds.) ECAI 2006, 17th European Conference on Artificial Intelligence, August 29 - September 1, 2006, Riva del Garda, Italy, Including Prestigious Applications of Intelligent Systems (PAIS 2006), Proceedings. Frontiers in Artificial Intelligence and Applications, vol. 141, pp. 397–401. IOS Press (2006)
10. Herzig, A., Varzinczak, I.J.: Metatheory of actions: Beyond consistency. Artif. Intell. 171(16-17), 951–984 (2007), http://dx.doi.org/10.1016/j.artint.2007.04.013

[6] **AH:** not clear to me how such theorems would look like. Aren't Thm 1 etc enough?

11. Kambhampati, S., Mali, A., Srivastava, B.: Hybrid planning for partially hierarchical domains. In: AAAI/IAAI. pp. 882–888 (1998)
12. Kuter, U., Sirin, E., Parsia, B., Nau, D., Hendler, J.: Information gathering during planning for web service composition. Web semantics: science, services and agents on the World Wide Web 3(2), 183–205 (2005)
13. Lin, N., Kuter, U., Sirin, E.: Web service composition with user preferences. In: European Semantic Web Conference. pp. 629–643. Springer (2008)
14. Myers, K.L.: Planning with conflicting advice. In: AIPS. pp. 355–362 (2000)
15. Nau, D., Ghallab, M., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)
16. Nau, D.S., Au, T.C., Ilghami, O., Kuter, U., Murdock, J.W., Wu, D., Yaman, F.: Shop2: An htn planning system. J. Artif. Intell. Res.(JAIR) 20, 379–404 (2003)
17. Rabideau, G., Engelhardt, B., Chien, S.A.: Using generic preferences to incrementally improve plan quality. In: AIPS. pp. 236–245 (2000)
18. Reiter, R.: Knowledge in action: logical foundations for specifying and implementing dynamical systems. MIT press (2001)
19. Varzinczak, I.J.: On action theory change. J. Artif. Intell. Res. (JAIR) 37, 189–246 (2010), http://dx.doi.org/10.1613/jair.2959