On the revision of planning tasks

Andreas Herzig, Viviane Menezes, Leliane Nunes de Barros, Renata Wassermann

On the revision of planning tasks

Andreas Herzig, Viviane Menezes, Leliane Nunes de Barros, Renata Wassermann

Abstract. When a planning task cannot be solved then it can often be made solvable by modifying it a bit: one may change either the set of actions, or the initial state, or the goal description. We show that modification of actions can be reduced to initial state modification. We then apply Katsuno and Mendelzon's distinction between update and revision and show that the modification of the initial state is an update and the modification of the goal description is a revision. We consider variants of Forbus's update and Dalal's revision operation and argue that existing belief change operations do not apply as they stand because their inputs are boolean formulas, while plan task modification involves counterfactual statements. We show that they can be captured in Dynamic Logic of Propositional Assignments DL-PA.

1 INTRODUCTION

A classical planning task over a set of state variables $\mathbb{P}$ is a tuple $\Pi = (\mathtt{A}, \mathsf{s}_0, \mathsf{S}_g)$ where $\mathtt{A}$ is the set of actions (described in terms of preconditions and effects), $\mathsf{s}_0 \subseteq \mathbb{P}$ is the initial state, and $\mathsf{S}_g \subseteq 2^{\mathbb{P}}$ is the set of goal states. A solution to a classical planning task is a sequence of actions—a plan—that leads the agent from s_0 to a state in S_g. Sometimes there is no such plan: the task is *unsolvable*. The analysis of unsolvable tasks can help a knowledge engineer when modeling a new planning application. Possible explanations are: (i) the initial state s_0 is wrongly specified; (ii) the goal is over-constrained ('over-subscribed'); (iii) the action specifications are not sound. Consider a scenario with a room whose door is locked (¬Open), with the key in the room (InK), and a robot outside (¬InR) whose goal is to get into the room (InR) [5]. Several minimal modifications of the initial situation make the task solvable: the robot could be inside the room, the door could be open, or the key could be outside the room.

Several authors have recently studied how an unsolvable Π can be modified in a way such that it becomes solvable [16, 5, 14, 15]. In this paper we propose a powerful yet simple logical framework: Dynamic Logic of Propositional Assignments, abbreviated DL-PA [8, 1]. DL-PA is a simple instantiation of Propositional Dynamic Logic PDL [6]: instead of PDL's abstract atomic programs, its atomic programs are assignments of state variables to either true or false, written $p{\leftarrow}\top$ and $p{\leftarrow}\bot$. Its models are considerably simpler than PDL's Kripke models: valuations of classical propositional logic (alias states) are enough to interpret its programs and formulas. The assignment $p{\leftarrow}\top$ is interpreted as an update of the current state by p, while the assignment $p{\leftarrow}\bot$ is interpreted as an update by $\neg p$. These atomic programs can be combined by means of the PDL program operators: sequential and nondeterministic composition, finite iteration, and test. In the present paper we moreover make use of a program operator that is less frequently considered: the converse operator. The action enter whose precondition is $\neg\mathsf{InR} \wedge \mathsf{Open}$ and which adds InR can be captured in DL-PA by the program $\mathsf{pgm}(\mathsf{enter}) = \neg\mathsf{InR}\wedge\mathsf{Open}?; \mathsf{InR}{\leftarrow}\top$, and the action of opening the door by $\mathsf{pgm}(\mathsf{open}) = \mathsf{InR}{\leftrightarrow}\mathsf{InK}?; \mathsf{Open}{\leftarrow}\top$. Moreover, all possible finite combinations of these two actions can be captured by the DL-PA program $(\mathsf{pgm}(\mathsf{enter}){\cup}\mathsf{pgm}(\mathsf{open}))^*$. The program nondeterministically chooses a finite number of iterations n and chooses one of the two actions at each step, up to n.

Dynamic logics have not only programs describing the way the world evolves, but also formulas describing how the world is. When the set $\mathbb{P}$ of state variables is finite then e.g. the initial state s_0 of a planning task can be described by the DL-PA formula $\mathsf{Fml}(\mathsf{s}_0) = \Big(\bigwedge_{p\in\mathsf{s}_0} p\Big) \wedge \Big(\bigwedge_{p\in\mathbb{P}\setminus\mathsf{s}_0} \neg p\Big)$ and the set of goal states S_g by $\mathsf{Fml}(\mathsf{S}_g) = \bigvee_{s\in\mathsf{S}_g} \mathsf{Fml}(s)$. For our example we have $\mathsf{Fml}(\mathsf{s}_0) = \neg\mathsf{InR} \wedge \neg\mathsf{InK} \wedge \neg\mathsf{Open}$ and $\mathsf{Fml}(\mathsf{S}_g) = \mathsf{InR}$. Beyond such boolean formulas, DL-PA has modal formulas $\langle\pi\rangle\varphi$ and $[\pi]\varphi$ combining a program π and a formula φ. The formula $\langle\pi\rangle\varphi$ expresses that φ is true after *some* possible execution of π, and $[\pi]\varphi$ expresses that φ is true after *every* possible execution of π. For instance, that the set of states S_g where InR is true can be reached from the current state by means of the actions open and enter, denoted by $\mathsf{Reachable}(\mathsf{S}_g, \{\mathsf{open}, \mathsf{enter}\})$, is expressed by the DL-PA formula

$$\mathsf{Reachable}(\mathsf{S}_g, \{\mathsf{open}, \mathsf{enter}\}) = \langle(\mathsf{pgm}(\mathsf{open}) \cup \mathsf{pgm}(\mathsf{enter}))^*\rangle\mathsf{InR},$$

i.e., InR is true after some possible iteration of the program $\mathsf{pgm}(\mathsf{open}) \cup \mathsf{pgm}(\mathsf{enter})$. Then to decide whether our example task is solvable is the same as deciding validity of the DL-PA formula

$$(\neg\mathsf{InR}\wedge\neg\mathsf{InK}\wedge\neg\mathsf{Open}) \rightarrow \mathsf{Reachable}(\mathsf{S}_g, \{\mathsf{open}, \mathsf{enter}\}).$$

DL-PA provides an appropriate framework for studying planning task modification. We are going to take advantage of the recent embeddings of various change operations such as Winslett's and Forbus's update and Dalal's revision into DL-PA programs [7]. There, the idea is that an update by an input formula A can be captured by a DL-PA program $\mathsf{upd}(A)$, in the sense that the interpretation of $\mathsf{upd}(A)$ relates valuations to their update by A. Said differently, the update of the belief base B by A has the same models as the DL-PA formula $\langle\mathsf{upd}(A)^-\rangle B$, where $^-$ is the converse operator: being in a state where B has been updated by A is the same as being in a state that was attained by $\mathsf{upd}(A)$, before which B was true. Similarly, revision of B by A is captured by a program $\mathsf{rev}(A, B)$. All the operations satisfy the success postulate: both $\langle\mathsf{upd}(A)^-\rangle B \rightarrow A$ and $\langle\mathsf{rev}(A, B)^-\rangle B \rightarrow A$ are DL-PA valid.

We have seen above that a planning task $\Pi = (\mathtt{A}, \mathsf{s}_0, \mathsf{S}_g)$ is unsolvable exactly when $\mathsf{Fml}(\mathsf{s}_0) \rightarrow \mathsf{Reachable}(\mathsf{S}_g, \mathtt{A})$ is DL-PA invalid. We will show that the modification of the initial state such that the task is solvable is described by an update of $\mathsf{Fml}(\mathsf{s}_0)$ by the non-boolean formula $\mathsf{Reachable}(\mathsf{S}_g, \mathtt{A})$. So the modified initial states are described by

$$\Big\langle\mathsf{upd}(\mathsf{Reachable}(\mathsf{S}_g, \mathtt{A}))^-\Big\rangle\mathsf{Fml}(\mathsf{s}_0).$$

Indeed, the above formula implies $\mathsf{Reachable}(\mathsf{S}_g, \mathtt{A})$ by the success postulate for update, and therefore S_g can be reached from the modified initial state via $\mathtt{A}$. The modification depends on the update operation. In principle any operation satisfying the Katsuno-Mendelzon postulates might do. We here choose Forbus's update operation, which is based on the Hamming distance. Our main reason for that choice is that there is a closely related revision operation that is also based on the Hamming distance, viz. Dalal's revision operation, which allows for a uniform presentation. Note that the above formula does not describe a unique state: there might be several minimal modifications of s_0.

We also show that the modification of the goal such that the planning task is solvable is described by a revision of $\mathsf{Fml}(\mathsf{S}_g)$ by the DL-PA formula

$$\mathsf{Reachable}(\mathsf{s}_0, \mathtt{A}^-) = \Big\langle \big(\mathsf{pgm}(a_1)^- \cup \cdots \cup \mathsf{pgm}(a_n)^-\big)^* \Big\rangle \mathsf{Fml}(\mathsf{s}_0).$$

In words, we revise by a formula saying that the the actions in $\mathtt{A}$ can be executed 'the other way round' such that s_0 is reached. Then the revised goal can be described by the DL-PA formula

$$\Big\langle \mathsf{rev}\big(\mathsf{Reachable}(\mathsf{s}_0, \mathtt{A}^-), \mathsf{Fml}(\mathsf{S}_g)\big)^- \Big\rangle \top$$

where rev is a DL-PA program implementing a revision operation.

The rest of the paper is organized as follows. In Section 2 we set the stage. In Section 3 we formally define three kinds of task modification problems. In Section 4 we introduce DL-PA. In Section 5 we embed (variants of) Forbus's update and Dalal's revision into DL-PA. In Section 6 we show how task modification can be done in DL-PA.

2 BACKGROUND

We start by recalling the definitions of distances between states, classical planning, Forbus revision and Dalal update.

2.1 Propositional logic, distances between states

A *valuation*, alias a *state*, associates a truth value to each element of the finite set of state variables $\mathbb{P} = \{p, q, \ldots\}$. We identify states with subsets of $\mathbb{P}$ and use s, s_1, etc. to denote them. The set of all states is $2^{\mathbb{P}}$. We also write $s(p) = 1$ when $p \in s$ and $s(p) = 0$ when $p \notin s$.

Boolean formulas, also called *propositional formulas*, are built from state variables by means of the standard boolean connectives. We will in particular make use of the exclusive disjunction, noted $\oplus$. The set of boolean formulas is $\mathsf{Fml}_{\mathsf{bool}}$ and its elements are noted A, B, C, etc. Contrasting with that, *modal formulas*—to be defined in the next section—will be noted φ, ψ, etc.

A given state determines the truth value of the boolean formulas. A state where the boolean formula A is true is called a *model of* A or an A*-state*. The *set of* A*-states* is noted $\|A\|$. As $\mathbb{P}$ is finite, every state s can be described by means of a conjunction of literals $\mathsf{Fml}(s)$ as defined above: s is the only model of $\mathsf{Fml}(s)$, i.e., $\|\mathsf{Fml}(s)\| = \{s\}$.

The *symmetric difference* between two states s_1 and s_2 is the set of all those p such that $s_1(p) \neq s_2(p)$: $s_1 \dot{-} s_2 = (s_1 \setminus s_2) \cup (s_2 \setminus s_1)$. For example, $\{p, q\} \dot{-} \{q, r, s\} = \{p, r, s\}$. It is also called the *PMA distance*, referring to the so-called 'Possible Models Approach' update operation [17]. The *Hamming distance* between s_1 and s_2 is

$$\mathsf{h}(s_1, s_2) = \mathsf{card}(s_1 \dot{-} s_2) = \mathsf{card}(\{p \ : \ s_1(p) \neq s_2(p)\}).$$

For example, $\mathsf{h}(\{p, q\}, \{q, r, s\} = \mathsf{card}(\{p, r, s\}) = 3$.

2.2 Planning tasks and their modification

An *action* is a triple $a = (\mathsf{pre}_a, \mathsf{add}_a, \mathsf{del}_a)$, where $\mathsf{pre}_a \in \mathsf{Fml}_{\mathsf{bool}}$ is the precondition and $\mathsf{add}_a, \mathsf{del}_a \subseteq \mathbb{P}$ are the add list and the delete list. A given a determines a relation $\|a\|$ between states:

$$\|a\| = \{(s, s') \ : \ s \in \|\mathsf{pre}_a\| \text{ and } s' = (s \setminus \mathsf{del}_a) \cup \mathsf{add}_a\}$$

A state s is *reachable* from a state s_0 via a set of actions $\mathtt{A}$ if there is $n \geq 0$, a sequence of actions $(a_1, \ldots, a_n)$ and a sequence of states $(s_0, \ldots, s_n)$ such that $s_0 = \mathsf{s}_0$, $s_n = s$, and $(s_{k-1}, s_k) \in \|a_k\|$ for every k such that $1 \leq k \leq n$. A *classical planning task* is a tuple $\Pi = (\mathtt{A}, \mathsf{s}_0, \mathsf{S}_g)$ where $\mathtt{A}$ is a finite set of actions, $\mathsf{s}_0 \subseteq \mathbb{P}$, and $\mathsf{S}_g \subseteq 2^{\mathbb{P}}$. Π is *solvable* if at least one of the goal states in S_g is reachable from s_0 via $\mathtt{A}$. Else Π is *unsolvable*.

Suppose $\Pi = (\mathtt{A}, \mathsf{s}_0, \mathsf{S}_g)$ is unsolvable. What can be done in such a situation apart from resigning? According to [5] one may: (i) change the initial state s_0, (ii) change the goal description S_g (typically weakening it), or (iii) augment the set of actions $\mathtt{A}$. Several approaches exist in the literature in particular for the second kind of modification (partial satisfaction planning, alias oversubscription planning). None of them exploits the conceptual framework that is provided by the belief update and revision literature, which is what we do here. The next section recalls some basic definitions.

2.3 Update and revision

We now define two belief change operations that generalise Forbus's update operation and Dalal's revision operation. Beyond an initial belief base and an input formula they have a further argument: a set of variables P to be minimised (similarly to circumscription). They coincide with the original operations when P contains all variables occurring in the input formula.

Forbus's update operation [4] is based on minimisation of the Hamming distance between states. Let s_0 be a state, S a set of states, and P a set of variables. The *Forbus update of* s_0 *by* S *w.r.t.* P is the set of states $s \in S$ that is closest to s_0 w.r.t. the Hamming distance, where only variables from P can be changed. Formally:

$$s_0 \diamond_P^{\mathsf{forbus}} S = \{s \in S \ : \ s \dot{-} s_0 \subseteq P \text{ and there is no } s' \in S \text{ such that } \mathsf{h}(s_0, s') < \mathsf{h}(s_0, s)\}$$

So $s_0 \diamond_P^{\mathsf{forbus}} S$ is the set of all those states of S that are closest to s_0 w.r.t. the Hamming distance while differing in P. For example, $\emptyset \diamond_{\{p,q\}}^{\mathsf{forbus}} \{\{p\}, \{q\}, \{p, q\}\} = \{\{p\}, \{q\}\}$. Then the update of a set of states S_0 by a set of states S w.r.t. P is the collection of the state-wise updates of each element of S_0: $S_0 \diamond_P^{\mathsf{forbus}} S = \bigcup_{s_0 \in S_0} s_0 \diamond_P^{\mathsf{forbus}} S$.

Several other update operations have been proposed in the literature; e.g., the PMA update of s_0 by S is the set of states $s' \in S$ closest to s_0 w.r.t. symmetric difference. We refer to [10, 12] for an overview.

Dalal's belief change operation [2] is not an update operation but rather a revision operation, according to Katsuno and Mendelzon's distinction [11]. Just as Forbus's operation, Dalal's is based on the minimisation of the Hamming distance between states. However, we now minimise globally over all states and not state-by-state. We follow the usage in the literature and denote Dalal's revision operation by $*^{\mathsf{dalal}}$ (and not by $\diamond^{\mathsf{dalal}}$). Let S_0 and S be sets of states and P a set of variables. The *Dalal revision of* S_0 *by* S *w.r.t.* P is:

$$S_0 *_P^{\mathsf{dalal}} S = \{s \in S \ : \text{there is } s_0 \in S_0 \text{ such that } s_0 \dot{-} s \subseteq P \text{ and } \mathsf{h}(s_0, s) \leq \mathsf{h}(s_0', s') \text{ for all } s' \in S, s_0' \in S_0\}.$$

So the revision of an empty set is empty.[1]

For example, the revision of $\|p \oplus q\|$ by $\|p\|$ is $\|p \wedge \neg q\|$:

$$\{\{p\},\{q\}\} *^{\mathsf{dalal}}_{\{p,q\}} \{\{p\},\{p,q\}\} = \{\{p\}\}$$

This contrasts with the Forbus update of $p \oplus q$ by p (which is p) and illustrates that revision operations satisfy the preservation postulate.

We note that Dalal revision coincides with Forbus update if S_0 is a singleton: we have $\{\mathtt{s}_0\} *^{\mathsf{dalal}}_P S = \mathtt{s}_0 \diamond^{\mathsf{forbus}}_P S = \{\mathtt{s}_0\} \diamond^{\mathsf{forbus}}_P S$. Moreover, $\emptyset *^{\mathsf{dalal}}_P S$ is $\emptyset$.[2]

3 THREE KINDS OF TASK MODIFICATION: FORMAL DEFINITIONS

We now define initial state change as a particular update problem and goal change as a particular revision problem. Furthermore, we reduce the modification of the set of actions to initial state change.

3.1 Changing the initial state

Intuitively, given an unsolvable planning task $\Pi = (\mathtt{A}, \mathtt{s}_0, \mathtt{S}_g)$ we want to change the initial state $\mathtt{s}_0$ to a state $\mathtt{s}'_0$ such that $\Pi' = (\mathtt{A}, \mathtt{s}'_0, \mathtt{S}_g)$ is solvable and such that $\mathtt{s}'_0$ is as close to $\mathtt{s}_0$ as possible. It makes sense here to consider minimal change w.r.t. some given set of relevant variables. In our example we consider modifications of $\mathtt{s}_0$ w.r.t. InK and Open: to make InR true would be a trivial modification. More generally, it seems reasonable to exclude variables from the goal description.

Definition 1. *Let $\Pi = (\mathtt{A}, \mathtt{s}_0, \mathtt{S}_g)$ be a planning task and let $\mathtt{S}'_0$ be the set of initial states $\mathtt{s}'_0$ such that $(\mathtt{A}, \mathtt{s}'_0, \mathtt{S}_g)$ is solvable, i.e.,*

$$\mathtt{S}'_0 = \{\mathtt{s}'_0 \ : \ \textit{there is } \mathtt{s}_g \in \mathtt{S}_g \textit{ such that } \mathtt{s}_g \textit{ is reachable from } \mathtt{s}'_0 \textit{ via } \mathtt{A}\}.$$

Let $P \subseteq \mathbb{P}_{\mathtt{A}}$ be some of the variables occurring in $\mathtt{A}$. The minimal modification of $\mathtt{s}_0$ w.r.t. *P is the set of states from which $\mathtt{S}_g$ is reachable that only differ from $\mathtt{s}_0$ in P and that are closest to $\mathtt{s}_0$.*

It remains to clarify what closeness means. While in principle any distance between states can be used, there are some natural starting points: the PMA distance and the Hamming distance. We choose the latter because it is one of the most popular update operations and because it parallels Dalal's revision operation: it is also based on the Hamming distance and is the best known concrete revision operation. So we consider that the set of initial states closest to $\mathtt{s}_0$ from which $\mathtt{S}_g$ is reachable is $\mathtt{s}_0 \diamond^{\mathsf{forbus}}_P \mathtt{S}'_0$. In our example there are two candidate states that only differ from $\mathtt{s}_0$ in {InK, Open} and that are closest to $\mathtt{s}_0$, viz. $\emptyset$ and {InK, Open}. So the robot's task is solvable either when the key is outside or when the door is open.

Our definition also applies when the original task is solvable: in that case, the only possible update of the initial state $\mathtt{s}_0$ is $\mathtt{s}_0$ itself.

Observe that instead of Forbus update we could as well have employed Dalal's revision, given that the belief base to be modified is a singleton. Things will differ when it comes to changing the goal.

This is a minor difference with Dalal's original definition, which distinguishes the cases $S_0 \neq \emptyset$ and $S_0 = \emptyset$. In the latter case the result of the update by S is stipulated to be S itself. This guarantees that the result of the revision is nonempty as soon as S is.

[2] As said above, here our definition differs from Dalal's, where $\emptyset *^{\mathsf{dalal}}_P S = S$.

3.2 Changing the goal

Intuitively, we want to transform an unsolvable $\Pi = (\mathtt{A}, \mathtt{s}_0, \mathtt{S}_g)$ to a solvable $\Pi' = (\mathtt{A}, \mathtt{s}_0, \mathtt{S}'_g)$ where $\mathtt{S}_g$ and $\mathtt{S}'_g$ are as close as possible.

Definition 2. *Let $\Pi = (\mathtt{A}, \mathtt{s}_0, \mathtt{S}_g)$ be a planning task and let $\mathtt{S}'_g$ be the set of states $\mathtt{s}'_g$ such that $(\mathtt{A}, \mathtt{s}_0, \{\mathtt{s}'_g\})$ is solvable, i.e.,*

$$\mathtt{S}'_g = \{\mathtt{s}'_g \ : \ \mathtt{s}'_g \textit{ is reachable from } \mathtt{s}_0 \textit{ via } \mathtt{A}\}.$$

Let $P \subseteq \mathbb{P}_A$ be some variables occurring in A. The minimal modification of $\mathtt{S}_g$ w.r.t. *P is the set of states reachable from $\mathtt{s}_0$ that only differ from $\mathtt{S}_g$ in P and that are closest to $\mathtt{S}_g$.*

So we have to deal with the notion of closeness between sets of states: we have a case of distance-based belief revision. In the rest of the paper we consider that the set of goal states reachable from $\mathtt{s}_0$ that only differ in P and are closest to $\mathtt{S}_g$ is $\mathtt{S}_g *^{\mathsf{dalal}}_P \mathtt{S}'_g$.

3.3 Adding actions

Intuitively, given an unsovalble planning task $\Pi = (\mathtt{A}, \mathtt{s}_0, \mathtt{S}_g)$ we want to minimally augment the set $\mathtt{A}$ to a set $\mathtt{A}'$ such that $\Pi' = (\mathtt{A}', \mathtt{s}_0, \mathtt{S}_g)$ is solvable. We describe a way of reducing this to initial state update.

Suppose that there is a set of *all possible actions* $\mathtt{A}$ and a set of *currently available actions* $\mathtt{A}_{\mathtt{u}} \subseteq \mathtt{A}$. The elements of $\mathtt{A}_{\mathtt{u}}$ are currently available to build plans, and augmenting the set of plan operators consists in adding to $\mathtt{A}_{\mathtt{u}}$ some elements of $\mathtt{A} \setminus \mathtt{A}_{\mathtt{u}}$. The aim is to add a minimal number from the latter.

We replace each action $a = (\mathsf{pre}_a, \mathsf{add}_a, \mathsf{del}_a)$ in $\mathtt{A}$ by the action $a^+ = (\mathsf{pre}_a \wedge \mathtt{u}_a, \mathsf{add}_a, \mathsf{del}_a)$ where $\mathtt{u}_a$ is a fresh variable: it does not occur in any add list or delete list. Let $\mathtt{A}^+ = \{a^+ \ : \ a \in \mathtt{A}\}$ be the resulting set of actions. Observe that none of the actions in $\mathtt{A}^+$ changes any of the fresh variables $\mathtt{u}_a$. Furthermore, we encode in the initial state $\mathtt{s}_0$ that only the actions in $\mathtt{A}_{\mathtt{u}}$ are available: we replace $\mathtt{s}_0$ by $\mathtt{s}_0 \cup \mathtt{U}_{\mathtt{A}_{\mathtt{u}}}$, for $\mathtt{U}_{\mathtt{A}_{\mathtt{u}}} = \{\mathtt{u}_a \ : \ a \in \mathtt{A}_{\mathtt{u}}\}$.

Proposition 1. *Let $\mathtt{A}_{\mathtt{u}} \subseteq \mathtt{A}$ be sets of actions. Let $\mathtt{U}_{\mathtt{A}_{\mathtt{u}}} = \{\mathtt{u}_a \ : \ a \in \mathtt{A}_{\mathtt{u}}\}$. Then $(\mathtt{A}_{\mathtt{u}}, \mathtt{s}_0, \mathtt{S}_g)$ is solvable if and only if $(\mathtt{A}^+, \mathtt{s}_0 \cup \mathtt{U}_{\mathtt{A}_{\mathtt{u}}}, \mathtt{S}_g)$ is solvable.*

This can be established by showing that each of the two tasks is solvable iff $(\mathtt{A}_{\mathtt{u}}, \mathtt{s}_0 \cup \mathtt{U}_{\mathtt{A}_{\mathtt{u}}}, \mathtt{S}_g)$ is. For the first task this holds because the variables in $\mathtt{U}_{\mathtt{A}_{\mathtt{u}}}$ are fresh. For the second task, observe that no plan can contain an $a^+ \notin \mathtt{A}^+_{\mathtt{u}}$: the precondition of a^+ fails to be true at the initial state because $\mathtt{u}_a \notin \mathtt{s}_0 \cup \mathtt{U}_{\mathtt{A}}$, and it keeps on failing because $\mathtt{u}_a$ is fresh and does not occur in any add list.

Suppose $(\mathtt{A}_{\mathtt{u}}, \mathtt{s}_0, \mathtt{S}_g)$ is unsolvable. Finding a minimal set of actions to be added to $\mathtt{A}_{\mathtt{u}}$ amounts to finding a minimal modification of the initial state of the equally unsolvable $(\mathtt{A}^+, \mathtt{s}_0 \cup \mathtt{U}_{\mathtt{A}_{\mathtt{u}}}, \mathtt{S}_g)$ *that only adds the usability variables* $\mathtt{u}_a$. So when computing the minimal distance we should only consider the new variables $\mathtt{u}_a$, for $a \notin \mathtt{A}_{\mathtt{u}}$.[3] In other words, we minimally modify $\mathtt{s}_0 \cup \mathtt{U}_{\mathtt{A}_{\mathtt{u}}}$ w.r.t. $\mathtt{A}^+ \setminus \mathtt{A}_{\mathtt{u}}$. This results in zero, one or more new initial states $\mathtt{s}_0 \cup \mathtt{U}_{\mathtt{A}_{\mathtt{u}}} \cup U$, for some U subset of $\{\mathtt{u}_a \ : \ a \in \mathtt{A} \setminus \mathtt{A}_{\mathtt{u}}\}$ such that $(\mathtt{A}^+, \mathtt{s}_0 \cup \mathtt{U}_{\mathtt{A}_{\mathtt{u}}} \cup U, \mathtt{S}_g)$ is solvable. By Proposition 1, the latter task is solvable iff $(\mathtt{A}_{\mathtt{u}} \cup \{a \ : \ \mathtt{u}_a \in U\}, \mathtt{s}_0, \mathtt{S}_g)$ is. Therefore each of the above sets U is a candidate for a minimal augmentation of the set of actions solving the planning task. Note that contrarily to the two previous kinds of modifications, there is no guarantee here that we will make the problem solvable.

[3] Actually it suffices to only consider making them true. This can also be implemented in DL-PA.

3.4 The rest of the paper

We have formally defined three different task modification problems. In the rest of the paper we investigate these problems from a knowledge representation perspective. We therefore do not work with states but with formulas describing initial state and goal states and with programs describing actions.

How can we obtain a syntactical representation of the set of candidate initial states S'_0 of Definition 1 or of the set of modified goal states S'_g of Definition 2? Can we represent S'_0 and S'_g as a boolean formula? This is not so straightforward: the candidate initial states s'_0 are specified indirectly, and as noted in [5], what we update by is a *counterfactual*: s'_0 should be such that *if* we perform an appropriate sequence from $\mathtt{A}$ then S_g is achieved. Therefore Forbus's original operator cannot be used as it stands: it only allows boolean input formulas. The situation is the same for the goal modification task. To tackle the problem we need more linguistic resources. The logic DL-PA to be introduced next will provide them.

4 DYNAMIC LOGIC OF PROPOSITIONAL ASSIGNMENTS

Dynamic Logic of Propositional Assignments DL-PA. was studied in [1], and its applicability to various problems of reasoning about dynamics was demonstrated in several recent papers [8, 3, 9, 7].

The language of DL-PA is defined by the following grammar:

$$\begin{array}{lcl} \varphi & ::= & p \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi \\ \pi & ::= & p{\leftarrow}\top \mid p{\leftarrow}\bot \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \pi^- \mid \varphi? \end{array}$$

where p ranges over $\mathbb{P}$. So the *atomic programs* of the language of DL-PA are of the form $p{\leftarrow}\top$ and $p{\leftarrow}\bot$. The operators of sequential composition (";"), nondeterministic composition ("$\cup$"), finite iteration ("$(.)^*$", the so-called Kleene star), and test ("$(.)?$") are familiar from Propositional Dynamic Logic PDL. The operator "$(.)^-$" is the converse operator. The set of variables occurring in φ is noted $\mathbb{P}_\varphi$.

The *length* of a formula φ, noted $|\varphi|$, is the number of symbols used to write down φ, without "$\langle$", "$\rangle$", and parentheses. For example, $|\langle q{\leftarrow}\top\rangle(q \vee r)| = 3{+}3 = 6$. The length of a program π, noted $|\pi|$, is defined in the same way. For example, $|p{\leftarrow}\bot; p?| = 6$. We have $\mathsf{card}(\mathbb{P}_\varphi) \leq |\varphi|$ for every φ.

We abbreviate the logical connectives $\wedge$, $\rightarrow$, $\leftrightarrow$, and $\oplus$ in the usual way. Moreover, $[\pi]\varphi$ abbreviates $\neg\langle\pi\rangle\neg\varphi$. Several program connectives are familiar from PDL. First, skip abbreviates $\top?$ ("nothing happens"). Second, the conditional "if φ then π_1 else π_2" is expressed by $(\varphi?;\pi_1) \cup (\neg\varphi?;\pi_2)$. Third, the loop "while φ do π" is expressed by $(\varphi?;\pi)^*;\neg\varphi?$. Fourth, we recursively define for $n \geq 0$:

$$\pi^n \stackrel{\text{def}}{=} \begin{cases} \mathsf{skip} & \text{if } n = 0 \\ \pi;\pi^{n-1} & \text{if } n \geq 1 \end{cases}$$

$$\pi^{\leq n} \stackrel{\text{def}}{=} \begin{cases} \mathsf{skip} & \text{if } n = 0 \\ (\mathsf{skip} \cup \pi);\pi^{\leq n-1} & \text{if } n \geq 1 \end{cases}$$

The program π^n executes π n times, and $\pi^{\leq n}$ executes π at most n times. Let us moreover introduce assignments of literals to variables:

$$p{\leftarrow}q = \text{if } q \text{ then } p{\leftarrow}\top \text{ else } p{\leftarrow}\bot$$
$$p{\leftarrow}\neg q = \text{if } q \text{ then } p{\leftarrow}\bot \text{ else } p{\leftarrow}\top$$

The former assigns to p the truth value of q, while the latter assigns to p the truth value of $\neg q$. In particular, $p{\leftarrow}p$ does nothing (and is therefore equivalent to skip) and $p{\leftarrow}\neg p$ flips the truth value of p.

DL-PA programs are interpreted by means of a *relation between states*: the atomic programs $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ update states in the obvious way, and complex programs are interpreted just as in PDL by mutual recursion. The interpretation of the DL-PA connectives is by mutual recursion.

For formulas the interpretation function is:

$$\begin{aligned} \|p\| &= \{s \ : \ p \in s\} \\ \|\top\| &= 2^{\mathbb{P}} \\ \|\bot\| &= \emptyset \\ \|\neg\varphi\| &= 2^{\mathbb{P}} \setminus \|\varphi\| \\ \|\varphi \vee \psi\| &= \|\varphi\| \cup \|\psi\| \\ \|\langle\pi\rangle\varphi\| &= \{s \ : \ \text{there is } s_1 \text{ s.t. } (s,s_1) \in \|\pi\| \text{ and } s_1 \in \|\varphi\|\} \end{aligned}$$

and for programs it is:

$$\begin{aligned} \|p{\leftarrow}\top\| &= \{(s_1,s_2) \ : \ s_2 = s_1 \cup \{p\}\} \\ \|p{\leftarrow}\bot\| &= \{(s_1,s_2) \ : \ s_2 = s_1 \setminus \{p\}\} \\ \|\pi;\pi'\| &= \|\pi\| \circ \|\pi'\| \\ \|\pi \cup \pi'\| &= \|\pi\| \cup \|\pi'\| \\ \|\pi^*\| &= \bigcup_{k\in\mathbb{N}_0} (\|\pi\|)^k \\ \|\pi^-\| &= (\|\pi\|)^{-1} \\ \|\varphi?\| &= \{(s,s) \ : \ s \in \|\varphi\|\} \end{aligned}$$

We say that two formulas φ_1 and φ_2 are *formula equivalent* if $\|\varphi_1\| = \|\varphi_2\|$. Two programs π_1 and π_2 are *program equivalent* if $\|\pi_1\| = \|\pi_2\|$. In that case we write $\pi_1 \equiv \pi_2$. For example, we have $p?; p{\leftarrow}\top \equiv p?$, and $\mathsf{skip} \cup p{\leftarrow}\neg p \equiv p{\leftarrow}\top \cup p{\leftarrow}\bot$.

An *expression* is a formula or a program. When we say that two expressions are equivalent we mean program equivalence if we are talking about programs, and formula equivalence otherwise. Equivalence is preserved under replacement of a sub-expression by an equivalent expression [1, Proposition 7].

A formula φ is DL-PA *valid* if it is formula equivalent to $\top$, i.e., if $\|\varphi\| = 2^{\mathbb{P}}$. It is DL-PA *satisfiable* if it is not formula equivalent to $\bot$, i.e., if $\|\varphi\| \neq \emptyset$. For example, the formulas $\langle p{\leftarrow}\bot\rangle\top$ and $\langle p{\leftarrow}\top\rangle\varphi \leftrightarrow \neg\langle p{\leftarrow}\top\rangle\neg\varphi$ are DL-PA valid. Other examples of DL-PA validities are $\langle p{\leftarrow}\top\rangle p$ and $\langle p{\leftarrow}\bot\rangle\neg p$. The valid schemas $\varphi \rightarrow [\pi]\langle\pi^-\rangle\varphi$ and $\varphi \rightarrow [\pi^-]\langle\pi\rangle\varphi$ are inherited from converse PDL (they are called the conversion axioms). Moreover, $\varphi \rightarrow [\pi]\varphi$ is valid if and only if $\langle\pi^-\rangle\varphi \rightarrow \varphi$ is valid. (The two senses of the "if and only if" are called the conversion rules.)

Observe that if p does not occur in φ then both $\varphi \rightarrow \langle p{\leftarrow}\top\rangle\varphi$ and $\varphi \rightarrow \langle p{\leftarrow}\bot\rangle\varphi$ are valid. This is due to the following property.

Proposition 2. *Suppose $\mathbb{P}_\varphi \cap P = \emptyset$, i.e., none of the variables in P occurs in φ. Then $s \cup P \in \|\varphi\|$ iff $s \setminus P \in \|\varphi\|$.*

In PDL, all program operators can be eliminated except the Kleene star. In contrast, *all* program operators can be eliminated in DL-PA.

Proposition 3 ([1])**.** *Every* DL-PA *formula is equivalent to a boolean formula.*

In the rest of the section we introduce some DL-PA programs that are convenient to embed update, revision and task modification.

$$\mathsf{vary}(\{p_1,\ldots,p_n\}) \stackrel{\text{def}}{=} (p_1{\leftarrow}\top \cup p_1{\leftarrow}\bot);\cdots;(p_n{\leftarrow}\top \cup p_n{\leftarrow}\bot)$$

$$\mathsf{flip1}(\{p_1,\ldots,p_n\}) \stackrel{\text{def}}{=} p_1{\leftarrow}\neg p_1 \cup \cdots \cup p_n{\leftarrow}\neg p_n$$

If $n = 0$ then both programs equal skip. In order to alleviate notation we drop set parentheses and write $\mathsf{vary}(p)$ instead of $\mathsf{vary}(\{p\})$, etc. The program $\mathsf{vary}(P)$ nondeterministically changes the truth value of some of the variables in P and $\mathsf{flip1}(P)$ flips the truth value of exactly one of the variables in P. The former actually implements the operation of *forgetting* the variables in P [13]. Observe that the program $\mathsf{vary}(\mathbb{P}_\varphi);\varphi?$ relates states s to all φ-states where the variables outside φ have the same truth values as in s.

Note that for $m \leq n$, the recursively defined programs $\mathsf{flip1}(P)^m$ and $\mathsf{flip1}(P)^{\leq m}$ have length quadratic in n. Moreover, we have $\mathsf{vary}(P) \equiv \mathsf{flip1}(P)^{\leq n}$.

Proposition 4 (Proposition 4 of [7])**.** *The following hold:*

1. $(s_1, s_2) \in \|\mathsf{vary}(P)\|$ *iff* $\{p : s_1(p) \neq s_2(p)\} \subseteq P$.
2. $(s_1, s_2) \in \|\mathsf{flip1}(P)^1\|$ *iff* $\mathsf{h}(s_1, s_2) = 1$, *for* $P \neq \emptyset$.
3. $(s_1, s_2) \in \|\mathsf{flip1}(P)^{\leq m}\|$ *iff* $\mathsf{h}(s_1, s_2) \leq m$.

Note that the second item cannot be generalised: for $m \geq 2$ we may have $(s_1, s_2) \in \|\mathsf{flip1}(P)^m\|$ while $\mathsf{h}(s_1, s_2) < m$.

Proposition 5. *The program equivalences* $(\mathsf{vary}(P))^- \equiv \mathsf{vary}(P)$, $(\mathsf{flip1}(P)^1)^- \equiv \mathsf{flip1}(P)^1$, *and* $(\mathsf{flip1}(P)^{\leq m})^- \equiv \mathsf{flip1}(P)^{\leq m}$ *hold.*

Here are some useful DL-PA formulas.

$$\mathsf{Valid}(\varphi) \stackrel{\text{def}}{=} [\mathsf{vary}(\mathbb{P}_\varphi)]\varphi$$

$$\mathsf{Sat}(\varphi) \stackrel{\text{def}}{=} \langle \mathsf{vary}(\mathbb{P}_\varphi)\rangle\varphi$$

$$\mathsf{H}(\varphi, \geq m) \stackrel{\text{def}}{=} \begin{cases} \top & \text{if } m = 0 \\ \neg\langle \mathsf{flip1}(\mathbb{P}_\varphi)^{\leq m-1}\rangle\varphi & \text{if } m \geq 1 \end{cases}$$

The formula $\mathsf{Valid}(\varphi)$ expresses that the formula φ is valid and $\mathsf{Sat}(\varphi)$ expresses that the formula φ is satisfiable. The latter is equivalent to $\langle \mathsf{vary}(\mathbb{P}_\varphi);\varphi?\rangle\top$. The formula $\mathsf{H}(\varphi, \geq m)$ is true at a state s exactly when the closest φ-states in the sense of the Hamming distance differ in at least m variables from s. For example,

$$\begin{aligned}
\mathsf{H}(p, \geq 1) &= \neg\langle \mathsf{flip1}(p)^{\leq 0}\rangle p \\
&\leftrightarrow \neg p \\
\mathsf{H}(p \vee q, \geq 1) &= \neg\langle \mathsf{flip1}(p, q)^{\leq 0}\rangle(p \vee q) \\
&\leftrightarrow \neg p \wedge \neg q \\
\mathsf{H}(p \vee q, \geq 2) &= \neg\langle \mathsf{flip1}(p, q)^{\leq 1}\rangle(p \vee q) \\
&\leftrightarrow \neg\langle \mathsf{skip} \cup p{\leftarrow}\neg p \cup q{\leftarrow}\neg q\rangle(p \vee q) \\
&\leftrightarrow \neg(p \vee q \vee \langle p{\leftarrow}\neg p\rangle(p \vee q) \wedge \langle q{\leftarrow}\neg q\rangle(p \vee q)) \\
&\leftrightarrow \neg((p \vee q) \vee (\neg p \vee q) \vee (p \vee \neg q)) \\
&\leftrightarrow \bot
\end{aligned}$$

The first two items of the next proposition establish that both validity and satisfiability reduce to model checking.

Proposition 6. *Let φ be a formula and let s be any state.*

1. $s \in \|\mathsf{Valid}(\varphi)\|$ *iff* φ *is valid.*
2. $s \in \|\mathsf{Sat}(\varphi)\|$ *iff* φ *is satisfiable.*
3. $s \in \|\mathsf{H}(\varphi, \geq m)\|$ *iff* $\mathsf{h}(s, s_1) \geq m$ *for every state* $s_1 \in \|\varphi\|$.

The length of each of the above formulas is polynomial in the length of φ (linear for the first two and quadratic for the last).

Proposition 7. *For* $m \leq \mathsf{card}(\mathbb{P}_A)$, $(s_1, s_2) \in \|\mathsf{H}(A, \geq m)?;\mathsf{flip1}(\mathbb{P}_A)^m;A\|$ *iff* $s_2 \in \|A\|$, $\mathsf{h}(s_1, s_2) = m$, *and there is no* $s_2' \in \|A\|$ *such that* $\mathsf{h}(s_1, s_2') < m$.

5 EXPRESSING FORBUS UPDATE AND DALAL REVISION IN DL-PA

Following [7], we embed Forbus's update operation and Dalal's revision operation into DL-PA. All results are straightforward generalisations of those in [7].

5.1 Embedding Forbus's update operation

We polynomially transform update problems of the form $B \diamond_P^{\mathsf{forbus}} A$ into DL-PA: we define a family of update programs $\mathsf{upd}_P^{\mathsf{forbus}}(A)$ whose length is cubic in the length of A.

Proposition 8. *Let A, B be propositional formulas. Let $P \subseteq \mathbb{P}_A$. Let $\mathsf{upd}_P^{\mathsf{forbus}}(A)$ be the following program:*

$$\left(\bigcup_{0 \leq m \leq \mathsf{card}(P)} \mathsf{H}(A, \geq m)?;\mathsf{flip1}(P)^m\right);A?$$

Then $\|B\| \diamond_P^{\mathsf{forbus}} \|A\| = \|\langle(\mathsf{upd}_P^{\mathsf{forbus}}(A))^-\rangle B\|$.

The program $\mathsf{upd}_P^{\mathsf{forbus}}(A)$ nondeterministically selects an integer m, checks whether the Hamming distance to A is at least m, flips m variables from P, and checks whether A is true. Via the program equivalences for the converse operator it follows that

$$\|B\| \diamond_P^{\mathsf{forbus}} \|A\| = \|\langle A?;\Big(\bigcup_{0 \leq m \leq \mathsf{card}(P)} \mathsf{flip1}(P)^m;\mathsf{H}(A, \geq m)?\Big)\rangle B\|.$$

For example,

$$\begin{aligned}
\mathsf{upd}_{\{p\}}^{\mathsf{forbus}}(p) &= \Big((\mathsf{H}(p, \geq 0)?;\mathsf{flip1}(p)^0) \cup (\mathsf{H}(p, \geq 1)?;\mathsf{flip1}(p)^1)\Big);p? \\
&\equiv \Big((\top?;\mathsf{skip}) \cup (\neg\langle \mathsf{flip1}(p)^{\leq 0}\rangle p?;p{\leftarrow}\neg p)\Big);p? \\
&\equiv \Big(\mathsf{skip} \cup (\neg p?;p{\leftarrow}\neg p)\Big);p? \\
&\equiv p? \cup (\neg p?;p{\leftarrow}\neg p;p?) \\
&\equiv p{\leftarrow}\top
\end{aligned}$$

Therefore $\|B\| \diamond_{\{p\}}^{\mathsf{forbus}} \|p\| = \|\langle p{\leftarrow}\top^-\rangle B\|$. Here is another example:

$$\begin{aligned}
\mathsf{upd}_{\{p,q\}}^{\mathsf{forbus}}(p \vee q) \equiv\ & (\mathsf{H}(p \vee q, \geq 0)?;\mathsf{flip1}(p, q)^0;p \vee q?) \cup \\
& (\mathsf{H}(p \vee q, \geq 1)?;\mathsf{flip1}(p, q)^1;p \vee q?) \cup \\
& (\mathsf{H}(p \vee q, \geq 2)?;\mathsf{flip1}(p, q)^2;p \vee q?) \\
\equiv\ & (\top?;\mathsf{skip};p \vee q?) \cup \\
& (\neg(p \vee q)?;(p{\leftarrow}\neg p \cup q{\leftarrow}\neg q);p \vee q?) \cup \\
& (\bot?;\mathsf{flip1}(p, q)^2;p \vee q?) \\
\equiv\ & p \vee q? \cup (\neg(p \vee q)?;(p{\leftarrow}\neg p \cup q{\leftarrow}\neg q))
\end{aligned}$$

If $P = \mathbb{P}_A$ then $\langle(\mathsf{upd}_P^{\mathsf{forbus}}(A))^-\rangle B \rightarrow A$ is DL-PA valid: Forbus's operation satisfies the KM success postulate.

Observe that the length of $\mathsf{upd}_P^{\mathsf{forbus}}(A)$ is cubic in the length of A.

5.2 Embedding Dalal's revision operation

Our revision program $\mathsf{rev}_P^{\mathsf{dalal}}(A, B)$ not only depends on the input A, but also on the base B.

Proposition 9. *Let A, B be propositional formulas. Let $P \subseteq \mathbb{P}_A$. Let $\mathsf{rev}_P^{\mathsf{dalal}}(A, B)$ be the following program:*

$$\mathsf{vary}(\mathbb{P}_B)\ ;\ B?\ ;\left(\bigcup_{0 \leq m \leq \mathsf{card}(P)} [\mathsf{vary}(\mathbb{P}_B)\ ;\ B?]\mathsf{H}(A, \geq m)?\ ;\mathsf{flip1}(P)^m\right);A?$$

Then $\|B\| *_P^{\mathsf{dalal}} \|A\| = \|\langle(\mathsf{rev}_P^{\mathsf{dalal}}(A, B))^-\rangle\top\|$.

The program $\mathsf{rev}^{\mathsf{dalal}}_P(A, B)$ visits all B-states s_B via $\mathsf{vary}(\mathbb{P}_B)$; $B?$, failing if there is no such state. It then nondeterministically selects an integer m such that the Hamming distance between the B-states and the A-states is at least m, flips m of the variables in P, and checks whether A is true. For the case of atomic inputs we get:

$$\begin{aligned}\mathsf{rev}^{\mathsf{dalal}}_{\{p\}}(p, B) &= \mathsf{vary}(\mathbb{P}_B); B? \,; \\ &\quad \Big(([\mathsf{vary}(\mathbb{P}_B); B?]\mathsf{H}(p, \geq 0)? \,; \mathsf{flip1}(p)^0) \cup \\ &\quad\quad ([\mathsf{vary}(\mathbb{P}_B); B?]\mathsf{H}(p, \geq 1)? \,; \mathsf{flip1}(p)^1)\Big); p? \\ &\equiv \mathsf{vary}(\mathbb{P}_B); B? \,; \Big(p? \cup ([\mathsf{vary}(\mathbb{P}_B); B?]\neg p? \,; p{\leftarrow}\top)\Big)\end{aligned}$$

and likewise for $\|B\| *^{\mathsf{dalal}}_{\{p\}} \|\neg p\|$. So when $B{\wedge}p$ is consistent then $\mathsf{rev}^{\mathsf{dalal}}_{\{p\}}(p, B)$ goes to a $B{\wedge}p$-state, and updates by p otherwise.

Observe that the length of the program $\mathsf{rev}^{\mathsf{dalal}}_P(A, B)$ is cubic in the sum of $|A| + |B|$.

6 PLANNING TASKS AND THEIR MODIFICATION IN DL-PA

Let us finally embed planning tasks into DL-PA. Let $\Pi = (\mathtt{A}, \mathsf{s}_0, \mathsf{S}_g)$ be a planning task. To the initial state s_0 and the set of goal states S_g we associate the boolean formulas $\mathsf{Fml}(\mathsf{s}_0)$ and $\mathsf{Fml}(\mathsf{S}_g)$. Let $a = (\mathsf{pre}_a, \mathsf{add}_a, \mathsf{del}_a)$ be an action with $\mathsf{add}_a = \{p_1, \ldots, p_k\}$ and $\mathsf{del}_a = \{q_1, \ldots, q_l\}$. We associate to a the DL-PA program

$$\mathsf{pgm}(a) = \mathsf{pre}_a?; p_1{\leftarrow}\top; \cdots; p_k{\leftarrow}\top; q_1{\leftarrow}\bot; \cdots; q_l{\leftarrow}\bot$$

Given a set of actions $\mathtt{A} = \{a_1, \ldots, a_n\}$, the DL-PA program $\big(\mathsf{pgm}(a_1) \cup \cdots \cup \mathsf{pgm}(a_n)\big)^*$ describes all possible finite sequences of actions from $\mathtt{A}$. Similarly, $\big(\mathsf{pgm}(a_1)^- \cup \cdots \cup \mathsf{pgm}(a_n)^-\big)^*$ describes all possible finite sequences of the converse of actions from $\mathtt{A}$. Consider the formulas

$$\begin{aligned}\mathsf{Reachable}(\mathsf{S}_g, \mathtt{A}) &= \Big\langle\Big(\mathsf{pgm}(a_1) \cup \cdots \cup \mathsf{pgm}(a_n)\Big)^*\Big\rangle\mathsf{Fml}(\mathsf{S}_g) \\ \mathsf{Reachable}(\mathsf{s}_0, \mathtt{A}^-) &= \Big\langle\Big(\mathsf{pgm}(a_1)^- \cup \cdots \cup \mathsf{pgm}(a_n)^-\Big)^*\Big\rangle\mathsf{Fml}(\mathsf{s}_0)\end{aligned}$$

The former is true at all those states from which S_g can be reached, and the latter at all those states that can be reached from s_0.

Proposition 10. *Let $\Pi = (\mathtt{A}, \mathsf{s}_0, \mathsf{S}_g)$ be a planning task. Π is solvable if and only if $\mathsf{Fml}(\mathsf{s}_0) \to \mathsf{Reachable}(\mathsf{S}_g, \mathtt{A})$ is DL-PA valid.*

In the rest of the section we focus on task modification.

Proposition 11. *Let $\Pi = (\mathtt{A}, \mathsf{s}_0, \mathsf{S}_g)$ be a planning task.*

1. *The set of states from which S_g is reachable that only differ from s_0 in P and are closest to s_0 equals*

$$\mathsf{s}_0 \diamond^{\mathsf{forbus}}_P \big\|\mathsf{Reachable}(\mathsf{S}_g, \mathtt{A})\big\| = \big\|\big\langle\mathsf{upd}^{\mathsf{forbus}}_P(\mathsf{Reachable}(\mathsf{S}_g, \mathtt{A}))^-\big\rangle\mathsf{Fml}(\mathsf{s}_0)\big\|$$

2. *The set of states reachable from s_0 that only differ from S_g in P and are closest to S_g equals*

$$\mathsf{S}_g *^{\mathsf{dalal}}_P \big\|\langle\mathsf{pgm}(a_1)^- \cup \cdots \cup \mathsf{pgm}(a_n)^-)^*\rangle\mathsf{Fml}(\mathsf{s}_0)\big\| = \big\|\big\langle\mathsf{rev}^{\mathsf{dalal}}_P(\mathsf{Reachable}(\mathsf{s}_0, \mathtt{A}^-), \mathsf{Fml}(\mathsf{S}_g))^-\big\rangle\top\big\|$$

7 CONCLUSION

We have given logical definitions of three kinds of planning task modification and have shown how they can be embedded into DL-PA. Instead of Forbus update and Dalal revision we might as well use other belief change operations, such as Winslett's Standard Semantics update WSS, Winslett's Possible Models Approach update PMA, or Satoh's revision: each of them can be embedded into DL-PA [7].

We have supposed that the initial state is completely described. A natural generalisation of our approach is to allow for incomplete initial state descriptions, as done in conformant planning. The resulting initial state modification problem has the same characteristics as the goal modification problem: just as for the former, it can be argued that update is inappropriate and that it requires revision.

REFERENCES

[1] Philippe Balbiani, Andreas Herzig, and Nicolas Troquard, 'Dynamic logic of propositional assignments: a well-behaved variant of PDL', in *Proc. LICS*, ed., O. Kupferman, pp. 143–152. IEEE, (2013).

[2] Mukesh Dalal, 'Investigations into a theory of knowledge base revision: preliminary report', in *Proc. 7th Conf. on Artificial Intelligence (AAAI'88)*, pp. 475–479, (1988).

[3] Luis Fariñas del Cerro, Andreas Herzig, and Ezgi Iraz Su, 'Combining equilibrium logic and dynamic logic', in *LPNMR*, eds., Pedro Cabalar and Tran Cao Son, volume 8148 of *Lecture Notes in Computer Science*, pp. 304–316. Springer, (2013).

[4] Kenneth D. Forbus, 'Introducing actions into qualitative simulation', in *Proc. 11th Int. Joint Conf. on Artificial Intelligence (IJCAI'89)*, ed., N. S. Sridharan, pp. 1273–1278. Morgan Kaufmann Publishers, (1989).

[5] Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel, 'Coming up with good excuses: What to do when no plan can be found', in *ICAPS*, eds., R.I. Brafman, H. Geffner, J. Hoffmann, and H.A. Kautz, pp. 81–88. AAAI, (2010).

[6] David Harel, Dexter Kozen, and Jerzy Tiuryn, *Dynamic Logic*, MIT Press, 2000.

[7] Andreas Herzig, 'Belief change operations: a short history of nearly everything, told in dynamic logic of propositional assignments', in *Proc. KR 2014*, eds., C. Baral and G. De Giacomo. AAAI Press, (2014).

[8] Andreas Herzig, Emiliano Lorini, Frédéric Moisan, and Nicolas Troquard, 'A dynamic logic of normative systems', in *Proc. IJCAI*, ed., T. Walsh, pp. 228–233, Barcelona, (2011). IJCAI/AAAI.

[9] Andreas Herzig, Pilar Pozos Parra, and François Schwarzentruber, 'Belief merging in Dynamic Logic of Propositional Assignments', in *Proc. FolKS 2014*, eds., C. Beierle and C. Meghini, pp. 381–398. Springer, LNCS, (2014).

[10] Andreas Herzig and Omar Rifi, 'Propositional belief base update and minimal change', *AI Journal*, **115**(1), 107–138, (1999).

[11] Hirofumi Katsuno and Alberto O. Mendelzon, 'On the difference between updating a knowledge base and revising it', in *Belief revision*, ed., Peter Gärdenfors, 183–203, Cambridge University Press, (1992).

[12] Jérôme Lang, 'Belief update revisited', in *Proc. IJCAI 2007*, pp. 2517–2522, (2007).

[13] Jérôme Lang, Paolo Liberatore, and Pierre Marquis, 'Propositional independence: Formula-variable independence and forgetting', *J. AI Research (JAIR)*, **18**, 391–443, (2003).

[14] M Viviane Menezes, Leliane N de Barros, and Silvio do Lago Pereira, 'Planning task validation', *SPARK 2012*, 48, (2012).

[15] Vitaly Mirkis and Carmel Domshlak, 'Abstractions for oversubscription planning', in *Proc. ICAPS*, eds., D. Borrajo, S. Kambhampati, A. Oddi, and S. Fratini. AAAI, (2013).

[16] David E. Smith, 'Choosing objectives in over-subscription planning', in *ICAPS*, eds., Shlomo Zilberstein, Jana Koehler, and Sven Koenig, pp. 393–401. AAAI, (2004).

[17] Mary-Anne Winslett, 'Reasoning about action using a possible models approach', in *Proc. 7th Conf. on Artificial Intelligence (AAAI'88)*, pp. 89–93, St. Paul, (1988).