Planning with PDDL3.0 Preferences by Compilation into STRIPS with Action Costs

Percassi Francesco
University of Brescia
Department of Information Engineering
f.percassi@unibs.it

Abstract

The research community has sought to extend the classical planning problem following two strategies. The first one follows a top-down approach consisting in the development of solvers that support a more general class of problems; the second one follows a bottom-up approach consists in extending the applicability range of current classical planners. A possible interesting approach consists in compiling the new features offered by recent extension of planning language into a simpler target language such as STRIPS or ADL. PDDL 3.0, the official language in 2006 fifth IPC, introduced state-trajectory constraints and preferences in order to better characterize the solution quality. In this work I present a compilation schema, inspired by some previous works, for translating a STRIPS problem enriched with all kind of PDDL 3.0 preferences into an equivalent STRIPS problem with action cost.

Introduction and background

Given a problem described by an action domain, an initial state and a description of goal state, the aim of classic planning paradigm is finding a sequence of actions that can transform, if they are performed, the initial state into the target state. It is possible to distinguish which solution is preferable among the set of possible solutions evaluating the plan cost as the number of its actions. This approach has been extended to the minimal plan cost evaluated as the sum of the cost assigned to each contained action. A more sophisticated recent approach for characterizing when a solution is preferable: if they are performed, the initial state into the target state into penalizing their violation with a real value that is used to decrease the plan metric.

In PDDL 3.0 the following class of preferences can be expressed:

- always, which requires that a condition should hold in every reached state; this kind of preferences is very useful to express safety or maintenance conditions;
- sometime-before, which requires that a condition becomes true before a second condition becomes true;
- sometime, which requires that a condition becomes true at least once in the state trajectory of the plan;
- at-most-once, which requires that a condition becomes true at most once in the state trajectory of the plan;
- soft goal.

This work describes a compilation scheme which is an extension of what proposed in (Ceriani and Gerevini 2015) where only always preference and soft goal are considered.

STRIPS+ with preferences

A STRIPS+ problem is a tuple \( \langle F, I, O, G, c \rangle \) where \( \langle F, I, O, G \rangle \) is a STRIPS problem and \( c \) is a function that maps each \( o \in O \) to a non-negative real number. The cost of a plan \( \pi \) is defined as \( c(\pi) = \sum_{i=0}^{\vert \pi \vert -1} c(a_i) \), where \( c(a_i) \) represents the cost of the i-th action \( a_i \) in \( \pi \) and \( \vert \pi \vert \) is the plan length. Without loss of generality, we will assume that the condition of a preference \( P_t \) is expressed in conjunctive normal form, for example \( P_t = p_1 \land p_2 \land \ldots \land p_n \), where each \( p_j \) with \( j \in [1, \ldots, n] \) is a clause of \( P_t \) formed by literals over the problem fluents. We write \( \pi \models_{typ} P_t \) to indicate that plan \( \pi \) satisfies a type preference \( P_t \), where \( typ \) indicates its type among \( \{a, sb, st, amo, sg\} \) which abbreviating always, sometime-before, sometime, at-most-once and soft goal.

Definition 1 A STRIPS+ problem with preferences is a tuple \( \langle F, I, O, G, P, c, u \rangle \) where:

- \( \langle F, I, O, G, P, c, u \rangle \) is a STRIPS+ problem;
\( \mathcal{P} = \{ \text{AP} \cup \text{SBP} \cup \text{STP} \cup \text{AMOP} \cup \text{SG} \} \) is the set of the preferences of \( \Pi \) where \( \text{AP} \), \( \text{SBP} \), \( \text{STP} \), \( \text{AMOP} \) and \( \text{SG} \) contain respectively always, sometime-before, sometime, at-most-once and soft goal preferences;

- \( u \) is an utility function mapping each \( p \in \mathcal{P} \) to a value in \( \mathbb{R}_0^+ \)

In the following the class of STRIPS+ with a set of preferences is indicated with STRIPS+.  

**Definition 2** Let \( \Pi \) be a STRIPS+P problem with a set of different kind of preference \( \mathcal{P} \). The utility \( u(\pi) \) of a plan \( \pi \) solving \( \Pi \) is the difference between the total amount of utility of the preferences by the plan and its cost \( u(\pi) = \sum_{p \in \mathcal{P} : \pi = \pi_{\text{typ}}(p)} u(p) - c(\pi) \) where \( \text{typ} \) is a function that map each \( p \in \mathcal{P} \) to the respective type, i.e. \( \text{typ} : \mathcal{P} \to \{ a, \text{sb}, \text{st}, \text{amo}, \text{sg} \} \). 

A plan \( \pi \) with utility \( u(\pi) \) for a STRIPS+P problem is optimal when there is no plan \( \pi' \) such that \( u(\pi') > u(\pi) \). The definitions below are introduced to simplify the notation in the discussion. 

**Definition 3** Given a preference clause \( p = l_1 \lor l_2 \lor \ldots \lor l_n \), the set \( \mathcal{L}(p) = \{ l_1, l_2, \ldots, l_n \} \) is the equivalent set-based definition of \( p \) and \( \mathcal{L}_{\text{eff}}(p) = \neg l_1, \neg l_2, \ldots, \neg l_n \) is the literal complement set of \( \mathcal{L}(p) \). 

**Definition 4** Given an operator \( o \in \Omega \) of a STRIPS+P problem, \( Z(o) \) is the set of literals defined as: 
\[
Z(o) = (\text{prec}(o) \cap \{ p \mid \neg p \in \text{eff}(o)^+ \}) \cup \text{eff}(o)^+ \cup \text{eff}(o)^-.
\]
Note that set \( Z(o) \) represents the literals certainly true in the state resulting from the application of operator \( o \). 

**Preferences and Class of Operators**

In our compilation scheme of a STRIPS+P problem we have to distinguish, for each kind of preference, different class of operators that are specified in the following definitions. This distinction is important in order to specialize the operators compilation based on how they interact with the preferences of the problem. 

**Definition 5** Given an operator \( o \) and CNF formula \( \Phi \) of a preference \( P \) of a STRIPS+P problem, we say that \( o \) can make true \( \Phi \) if:

- there is at least a clause \( \varphi \) of \( \Phi \) such that \( \mathcal{L}(\varphi) \cap Z(o) \neq \emptyset \) and \( \mathcal{L}(\varphi) \not\subseteq \text{prec}(o) \); we indicate the set of clause which satisfy this condition as \( C(o, \Phi) \), the complementary set of the remaining clauses is defined as \( \overline{C}(o, \Phi) = \{ \varphi \in \Phi \mid \varphi \not\in C(o, \Phi) \} \)

- for each clause \( \varphi \not\in C(o, \Phi) \Rightarrow \mathcal{L}(\varphi) \not\subseteq Z(o) \).

The first condition in Definition 5 requires that exists at least a clause of the formula which contains some literals that become certainly true in the state resulting from the execution of \( o \) and that this clause is not true in the state where \( o \) is applied. The second condition requires that the other clauses of the formula, which are not contained in \( C(o, \Phi) \), are not falsified in the resulting state from the application of \( o \). 

**Always**

An always preference has the following PDDL syntax (always \( \Phi \)) where the formula \( \Phi \) has to hold in each reached state of the plan.

**Definition 6** Given an operator \( o \) and an always preference \( P = (\text{always} \ \Phi) \) of a STRIPS+P problem, \( o \) is a violation of \( \Phi \) if there is a clause \( \varphi \) of \( \Phi \) such that:
\[
\mathcal{L}(p) \subseteq Z(o) \land \mathcal{L}(p) \not\subseteq \text{prec}(o).
\]

If an operator violates a preference, the preference is unsatisfied independently from the state resulting from the application of the operator.

**Definition 7** Given an operator \( o \) and a always preference \( P \) of a STRIPS+P problem, \( o \) is a threat of \( P \) if it is not a violation and there exists a clause \( p \) of \( P \) such that:
\[
\mathcal{L}(p) \cap Z(o) \neq \emptyset \land \mathcal{L}(p) \cap Z(o) = \emptyset \land \mathcal{L}(p) \not\subseteq \text{prec}(o)
\]

A clause \( p \) of \( P \) satisfying the condition of the definition above is a threatened clause of \( P \). A threatened preference (clause) may be falsified by an operator depending on the state where the operator is applied. The expression \( \mathcal{L}(p) \not\subseteq \text{prec}(o) \) in Definition 5-6-7 is necessary to avoids that an operator \( o \) is considered a violation/threat when its precondition is already violated in the state where it is applied. The set of always preferences of \( \Pi \) which are threatened/violated by the operator \( o \) are denoted respectively \( T_{ag}(o) \) and \( V_{ag}(o) \).

**Definition 8** Given an operator \( o \) and a always preference \( P \) of a STRIPS+P problem, \( o \) is a safe for \( P \) if:

- for all clauses \( p \) of \( P \), \( L(p) \cap Z(o) \neq \emptyset \) or \( \mathcal{L}(p) \cap Z(o) = \emptyset \) holds;
- there exists a clause \( p \) such that \( \mathcal{L}(p) \subseteq \text{prec}(o) \).

**Sometime-Before**

A sometime-before constraint has the following PDDL syntax (sometime-before \( \Phi, \Psi \)), which in the following we abbreviate with \( \langle \Phi, \Psi \rangle \). The meaning of \( \langle \Phi, \Psi \rangle \) is that if \( \Phi \) is true in a state \( s \) then \( \Psi \) must have been true in state before \( s \).

**Definition 9** Given an operator \( o \) and a sometime-before preference \( P = \langle \Phi, \Psi \rangle \) of a STRIPS+P problem, \( o \) is a potential support for \( P \) if \( o \) can make \( \Psi \) true. 

An operator that satisfied Definition 9 is a potential support because its behaviour respect to the interested preference depends by the state where \( o \) is applied and consequently from the resulting state. We can distinguish two situations:

- if formula \( \Psi \) of \( P \) does not become true in the resulting state, then \( o \) is a neutral operator;
- if \( P \) is not violated in the state \( s \) where \( o \) is applied and the formula \( \Psi \) of \( P \) becomes true in the resulting state, then \( o \) is a real support operator.

The compilation scheme must take account of both these possibilities.

**Definition 10** Given an operator \( o \) and a sometime-before preference \( P = \langle \Phi, \Psi \rangle \) of a STRIPS+P problem, \( o \) is a potential threat for \( P \) if \( o \) could make true \( \Phi \).
Similarly to definition 9 also in this case the potential threat defines its behavior in correspondence of the consequences of its application. We distinguish the following situations:

- if formula Ψ of P does not become true in the resulting state, then o is a neutral operator;
- if formula Ψ of P becomes true in the resulting state and the formula Φ becomes true at least once in an earlier state, than s is neutral otherwise if the formula Φ has never become true, then o is a violation.

The set of sometime-before preferences of Π which are potentially threatened/supported by the operator o are denoted respectively with \(T_{sb}(o)\) and \(S_{sb}(o)\).

**Sometime**

A sometime preference has the following PDDL syntax (sometime Φ) where the formula Φ has to become true at least once in the plan state trajectory.

**Definition 11** Given an operator o and a sometime preference \(P = \langle \text{sometime } \Phi \rangle\) of a STRIPS+P problem, o is a potential satisfying operator for P if o can make Φ true. If an operator o can not make Φ then the operator is neutral for P.

The set of sometime-before preferences of Π which are potentially satisfied by the operator o are denoted with \(S_{st}(o)\).

**At-most-once**

An at-most-once preference has the following PDDL syntax (at most one Φ) where the formula Φ has to become true at most once in the plan state trajectory.

**Definition 12** Given an operator o and an at-most-once preference \(P = \langle \text{at-most-one } \Phi \rangle\) of a STRIPS+P problem, o is a potential threat operator for P if o could make Φ true.

We distinguish the following situations:

- if Φ has never become true in states earlier than the state s where o is applied and Φ becomes true in the state resulting from the application of o in s, then the corresponded compiled operator o′ has to take account this fact, otherwise, if Φ has become true in an earlier state, then o is a violation;
- if Φ does not become true in the state resulting from the application of o, then o is a neutral operator.

The set of at-most-once preferences of Π which are potentially threatened by the operator o are denoted with \(T_{amo}(o)\).

**Compilation intro STRIPS+**

**Definition 13** If an operator o ∈ O is safe for every always preference in P and neutral for every sometime-before, at-most-once and sometime preference in P then we say that o is neutral for the problem Π and we write this property with neutral(o). The set containing all the neutral operators for Π is defined as \(N(\Pi) = \{o \in O \mid \text{neutral}(o)\}\).

**Definition 14** Given an operator o ∈ O of a STRIPS+P problem Π \(I(op)\) is the following set: \(\{T_{a}(o) \cup T_{sb}(o) \cup S_{sb}(o) \cup T_{amo}(o) \cup S_{st}(o)\}\) which contains all the preferences \(p \in P\) of Π which are affected by the execution of o.

Given a STRIPS+P problem, an equivalent STRIPS+ problem can be derived by translation which has some similarities to what proposed by Keyder and Geffner for soft goals but also significant difference. The scheme proposed by Keyder and Geffner is considerably simpler than ours because it does not consider the interaction between actions and preferences such as threats, supports and violations. In order to simplify the compilation scheme we don’t consider the compilation of soft goals because it can be easily added using the same method of Keyder and Geffner. Moreover we assume that every always and sometime-before preference is satisfied in the problem initial state I.

Given a STRIPS+P problem \(\Pi = \langle F, I, O, G, P, c, u \rangle\), the compiled STRIPS+ problem of Π is \(\Pi' = \langle F', I', O', G', P', c' \rangle\) with:

- \(F' = F' \cup V_{a,sh,st,amo} \cup D \cup C \cup C' \cup \{\text{normal-mode}, \text{end-mode}, \text{pause}\}\);
- \(I' = I \cup C_{st} \cup V_{st} \cup \{\text{normal-mode}\}\);
- \(G' = G \cup C'\);
- \(O' = \{\text{collect}(st), \text{forgo}(st) \mid st \in ST \subseteq P\} \cup \{\text{end}\} \cup O_{comp}\);
- \(c'(o) = \begin{cases} u(st) & \text{if } \text{if } o = \text{forgo}(st), st \in ST \\ c(o) & \text{if } o \in N(\Pi) \\ c_{tv}(o) & \text{if } o \notin N(\Pi) \\ 0 & \text{otherwise} \end{cases}\)

where:

- \(V_{a,sh,st,amo} = \bigcup_{i=1}^{k} \{P_i \text{-violated}\}, k = |P|\);
- \(D = \bigcup_{i=1}^{n} \{P_i \text{-done}_{o_1},...,P_i \text{-done}_{o_m}\} \) where \(n = |O|\) and \(m = |I(o)|\);
- \(C'_{st} = \{ST'_i \mid ST_i \in ST \subseteq P\}\) and \(C_{st} = \{ST'_i \mid ST_i \in ST \subseteq P\}\);
- \(V_{st} \subseteq V_{a,sh,st,amo}\);
- \(\text{collect}(ST_i) = \{\text{end-mode}, \neg ST_i \text{-violated}, \overline{ST'_i}\}, \{ST'_i, \neg \overline{ST'_i}\}\}\);
- \(\text{forgo}(ST_i) = \{\text{end-mode}, ST_i \text{-violated}, \overline{ST'_i}\}, \{ST'_i, \neg \overline{ST'_i}\}\}\);
- \(\text{end} = \{\text{normal-mode}, \text{-pause}\}, \{\text{end-mode}, \text{normal-mode}\}\}
- \(O_{comp} = O_{\text{neutral}} \cup O_{\text{chained}} \cup O_{\text{violation}}\);
- \(O_{\text{neutral}} = \{\text{pre}(o) \cup \{\text{normal-mode}, \text{-pause}, ef(f(o))\} \mid o \in O \land o \in N(\Pi)\}\);
- \(O_{\text{chained}}\) and \(O_{\text{violation}}\) are the compiled operators sets generated by the transformation schema applied to the operators of Π that threaten, violate or interact with at least a
preference of \( \Pi \). An operator \( o \in O \) is compiled through the compilation schema if \( |I(o)| > 0 \); the compiled operators \( \Theta_{\text{chained}} \) of the non-neutral operators are defined as:

\[
\bigcup_{o \in O, o \in N(\Pi)} \text{chain}(o) \text{ where } \text{chain}(o) \text{ is a function defined further down;}
\]

- \( c_{tv}(o) \) is the cost of an operator \( o \not\in N(\Pi) \).

For each sometime preference \( ST \), the transformation of \( \Pi \) into \( \Pi' \) adds a dummy hard goal \( ST' \) to \( \Pi' \) which can be achieved in two ways: with action \( \text{collect}(ST) \), that has a cost 0 but requires that \( ST \) is satisfied, or with action \( \text{forgen}(ST) \), that has a cost equal to utility of \( ST \) and can be performed when \( ST \) is unsatisfied in \( s_n \). Note that the original initial state \( I \) is extended with the \( V_{st} \) set, which contains, for each \( ST \in STS \subseteq P \), a literal \( is-violated-ST \) stating that \( ST \) is violated until an \( o \in S_{st}(o) \) satisfies the associated formula. For each sometime preference exactly one of \( \{\text{collect}(ST), \text{forgen}(ST)\} \) appears in the plan. This approach is not used for every kind of preference, except sometime, whose violation is caught by the model during planning and not at the end of the planning.

The compilation schema

Each operator \( o \) such that \( |I(o)| > 0 \), or equivalently \( o \not\in N(\Pi) \), is compiled into a set of new operators. The set of the \( m \) preferences affected by \( o \) is \( I(o) = \{P_1, \ldots, P_m\} \). Then \( o \) is compiled into a set of operators \( \text{chain}(o) = \{\Theta(o, P_1), \ldots, \Theta(o, P_m)\} \) where each \( \Theta(o, P_i) \) for \( i \in [1, \ldots, m] \) is a set of operators, called stage, related to an affected preference \( P_i \in I(o) \). The definition of each stage \( \Theta(o, P_i) \) depends on the kind of preference \( \text{typ}(P_i) \) and the value of \( i \). Furthermore the stage set are built in order to execute the following operators sequence \( \omega_{\text{chain}}(o) = \langle o'_{P_1}, \ldots, o'_{P_m} \rangle \) where \( o'_{P_i} \), with \( i \in [1, \ldots, m] \), is selected from the \( i \)-th set \( \Theta(o, P_i) \).

Given a non-neutral operator \( o \) of \( \Pi \), the set of the compiled operators related to \( o \) for \( \Pi' \), called chain for \( o \), is defined as:

\[
\text{chain}(o) = \bigcup_{p_i \in I(o), i \in [1, \ldots, |I(o)|]} \Theta(o, p_i)
\]

This set is called chain because the operators in each stage are built in order to force the sequential execution of \( \omega_{\text{chain}}(o) \).

In this presentation I provide the detailed description for the compilation of an operator \( o \) that affects the \( i \)-th at-most-once preference in \( I(o) \).

**Definition 15** The compilation-method for the translation of a non-neutral operators \( o \) that affect the \( i \)-th at-most-once preference \( P_i = (\text{at-most-once } a_i) \) where \( a_i = \bigwedge_{j \in j} a_{ij} \) (where \( a_{ij} \) is a clause) of \( I(o) \) is:

- if \( i = 1 \) (init stage):
  \[
  \text{prec}(a_i) = \text{prec}(o) \cup \{\neg \text{pause}, \neg \text{is-violated-a}_1, \neg \text{seen-a}_1\} \cup \{\bigcup_{a_{ij} \in \mathcal{C}(o, a_i)} a_{ij}\}
  \]
  \[
  \text{ef}f(a_i) = \{\text{pause, seen-a}_1, a_1\text{-done}_o\}
  \]

  \[
  \text{prec}(\overline{a_i}) = \text{prec}(o) \cup \{\neg \text{pause}, \neg \text{is-violated-a}_1\} \cup \neg \{\bigwedge_{a_{ij} \in \mathcal{C}(o, a_i)} a_{ij}\}
  \]
  \[
  \text{ef}f(\overline{a_i}) = \{\text{pause, a}_1\text{-done}_o\}
  \]

- if \( 1 < i < m = |I(o)| \) (middle stage):
  \[
  \text{prec}(o_{a_i}) = \{\text{pause, is-violated-a}_i, \neg \text{seen-a}_i, a_{i-1}\text{-done}_o\} \cup \{\bigcup_{a_{ij} \in \mathcal{C}(o, a_i)} a_{ij}\}
  \]
  \[
  \text{ef}f(o_{a_i}) = \{\text{pause, seen-a}_i, \neg a_{i-1}\text{-done}_o, a_i\text{-done}_o\}
  \]

  \[
  \text{prec}(\overline{o_{a_i}}) = \{\text{pause, is-violated-a}_i, \neg \text{seen-a}_i, a_{i-1}\text{-done}_o\} \cup \neg \{\bigwedge_{a_{ij} \in \mathcal{C}(o, a_i)} a_{ij}\}
  \]
  \[
  \text{ef}f(\overline{o_{a_i}}) = \{\text{pause, a}_{i-1}\text{-done}_o, a_i\text{-done}_o\}
  \]

- if \( i = m \) (final stage):
  \[
  \text{prec}(o_{a_m}) = \{\text{pause, is-violated-a}_m, \neg \text{seen-a}_m, a_{m-1}\text{-done}_o\} \cup \left\{\bigcup_{a_{mj} \in \mathcal{C}(o, a_m)} a_{mj}\right\}
  \]
  \[
  \text{ef}f(o_{a_m}) = \text{ef}f(o) \cup \{\neg \text{pause, seen-a}_m, a_{m-1}\text{-done}_o\}
  \]

  \[
  \text{prec}(\overline{o_{a_m}}) = \{\text{pause, is-violated-a}_m, \neg \text{seen-a}_m, a_{m-1}\text{-done}_o\} \cup \left\{\bigcup_{a_{mj} \in \mathcal{C}(o, a_m)} a_{mj}\right\}
  \]
  \[
  \text{ef}f(\overline{o_{a_m}}) = \text{ef}f(o) \cup \{\neg \text{pause, a}_{m-1}\text{-done}_o\}
  \]

  \[
  \text{prec}(\overline{a_m}) = \{\text{pause, is-violated-a}_m, a_{m-1}\text{-done}_o\}
  \]
  \[
  \text{ef}f(\overline{a_m}) = \text{ef}f(o) \cup \{\neg \text{pause, a}_{m-1}\text{-done}_o, a_{m-1}\text{-done}_o\}
  \]

In accordance with Definition 12 the \( i \)-th stage \( \theta_{ama}(o, P_i) \) providing the following possible choices:

- \( o_{a_i} \) is a neutral operator for \( P_i \), which asserting that the related formula \( a_i \) has been seen for the first time:
• $\sigma_{a_i}$ is a violation of $P_i$ because the related formula $a_i$ has been true in a previous state;
• $\overline{\sigma}_{a_i}$ is a neutral operator for $P_i$ because it does not make $a_i$ true;
• $\overline{\overline{\sigma}}_{a_i}$ is a neutral operator for $P_i$ because it has already been violated in a previous state.

Conclusion
In my first years of PhD, I have worked on the compilation of PDDL 3.0 preferences into STRIPS with action costs. As a base I started from two works of (Keyder and Geffner 2009), for the compilation of soft goal, and (Ceriani and Gerevini 2015) for the compilation of always goal. I have developed a new complicative scheme for three type of preference which were not considered in the previous work. All the propose compitative methods have been implemented and preliminary experiments show that the investigated approach is competitive in terms of performance with other existing approaches to planning preferences.

References