# 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 traslating 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 intial 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 better than others is based on the notion of preference, which are properties that a plan has to satisfy to increase its quality. Planning with preferences concerns the generation of plans for problems involving soft goal or soft state-trajectory constraints, called preference in PDDL 3.0 (Gerevini et al. 2009), which are preferable to satisfy, but that they are not necessary to hold in a valid plan.

In PDDL 3.0 has been proposed some new features in order to increase the expressive power about the quality solution specification. The new introduced constructs include *soft state-trajectory constraints*, which are constraints that should be satisfied in the state trajectory to increase the quality plan, and *soft problem goal*. An approach to assign a priority to each *preference* (hereafter we indicate indistinctly a soft goal or a soft state-trajectory as preference) consists 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 $\Psi$ has become true before a second condition $\Phi$ 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 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 $c(\pi) = \sum_{i=0}^{|\pi|-1} c(a_i)$, where $c(a_i)$ represents the cost of the i-th action $a_i$ in $\pi$ and $|\pi|$ is the plan lenght. Without loss of generality, we will assume that the condition of a preference $P_i$ is expressed in conjuctive normal form, for example $P_i = p_1 \wedge p_2 \wedge ... \wedge p_n$, where each $p_j$ with $j \in [1, ..., n]$ is a clause of $P_i$ formed by literals over the problem fluents. We write $\pi \models_{typ} P_i$ to indicate that plan $\pi$ satisfies a $typ$ preference $P_i$ where $typ$ indicate 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 $\langle F, I, O, G, \mathcal{P}, c, u\rangle$ where:*

- *$\langle F, I, O, G, c\rangle$ is a STRIPS+ problem;*

- $\mathcal{P} = \{AP \cup SBP \cup STP \cup AMOP \cup SG\}$ *is the set of the preferences of* $\Pi$ *where* $AP$*,* $SBP$*,* $STP$*,* $AMOP$ *and* $SG$ *contain respectively always, sometime-before, sometime, at-most-once and soft goal prefences;*
- $u$ *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+P.

**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* $u(\pi) = \sum_{P \in \mathcal{P}: \pi \models_{typ(P)} P} u(P) - c(\pi)$ *where* $typ$ *is a function that map each* $P \in \mathcal{P}$ *to the respective type, i.e.* $typ : \mathcal{P} \rightarrow \{a, sb, st, amo, sg\}$

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* $p = l_1 \vee l_2 \vee ... l_n$*, the set* $L(p) = \{l_1, l_2, ..., l_n\}$ *is the equivalent set-based definition of* $p$ *and* $\overline{L}(p) = \{\neg l_1, \neg l_2, ..., \neg l_n\}$ *is the literal complement set of* $L(p)$*.*

**Definition 4** *Given an operator* $o \in O$ *of a STRIPS+P problem,* $Z(o)$ *is the set of literal defined as:* $Z(o) = (prec(o) \setminus \{p \mid \neg p \in eff(o)^-\}) \cup eff(o)^+ \cup eff(o)^-$*. 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* $L(\varphi) \cap Z(o) \neq \emptyset$ *and* $L(\varphi) \not\subseteq prec(o)$*; we indicate the set of clause which satisfy this condition as* $C(o, \Phi)$*; the complementary set of the remaining clauses is defined as* $\overline{C}(o, \Phi) = \{\varphi \in \Phi \mid \varphi \notin C(o, \Phi)\}$
- *for each clause* $\varphi \notin C(o, \Phi) \Rightarrow \overline{L}(\varphi) \not\subseteq Z(o)$*.*

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* $P = (always\ \Phi)$ *of a STRIPS+P problem,* $o$ *is a* ***violation*** *of* $P$ *if there is a clause* $\phi$ *of* $\Phi$ *such that:* $\overline{L}(p) \subseteq Z(o) \wedge \overline{L}(p) \not\subseteq prec(o)$*.*

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:* $\overline{L}(p) \cap Z(o) \neq \emptyset \wedge L(p) \cap Z(o) = \emptyset \wedge \overline{L}(p) \not\subseteq prec(o)$

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 $\overline{L}(p) \not\subseteq prec(o)$ in Defintion 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* $\overline{L}(p) \cap Z(o) = \emptyset$ *holds;*
- *there exists a clause* $p$ *such that* $\overline{L}(p) \subseteq prec(o)$*.*

### Sometime-Before

A sometime-before constraint has the following PDDL syntax $(sometime\text{-}before\ \Phi\ \Psi)$, 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 $\Psi$ of $P$ does not become true in the resulting state, then $o$ is a *neutral* operator;
- if formula $\Psi$ of $P$ becomes true in the resulting state and the formula $\Phi$ has become true at least once in a earlier state, than $s$ is *neutral* otherwise if the formula $\Phi$ has never become true, then $o$ is a *violation*.

The set of sometime-before preferences of $\Pi$ which are potentially threatned/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 $(sometime\ \Phi)$ where the formula $\Phi$ has to become true at least once state in the plan state trajectory.

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

The set of sometime preferences of $\Pi$ 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 $(at - most - once\ \Phi)$ where the formula $\Phi$ has to become true at most once in the plan state trajectory.

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

We distinguish the following situations:

- if $\Phi$ has never become true in states earlier than the state $s$ where $o$ is applied and $\Phi$ becomes true in the state resulting from the application of $o$ in $s$, then the corrispondent compiled operator $o'$ has to take account this fact, otherwise, if $\Phi$ has become true in a earlier state, then $o$ is a *violation*;
- if $\Phi$ 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 $\Pi$ which are potentially threatned by the operator $o$ are denoted with $T_{amo}(o)$.

## Compilation intro STRIPS+

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

**Definition 14** *Given an operator $o \in O$ of a STRIPS+P problem $\Pi$ $I(op)$ 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 $\Pi$ 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 considerable simpler than ours because it does not to 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 $\Pi$ is $\Pi' = \langle F,' I', O', G', P', c' \rangle$ with:

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

  where:
- $V_{a,sb,st,amo} = \cup_{i=1}^{k} \{P_i\text{-}violated\}, k = |P|$;
- $D = \cup_{i=1}^{n} \{P_i\text{-}done_{o_1}, ..., P_i\text{-}done_m\}$ where $n = |O|$ and $m = |I(o)|$;
- $C'_{st} = \{ST'_i \mid ST_i \in ST \subseteq P\}$ and $\overline{C'}_{st} = \{\overline{ST'_i} \mid ST_i \in ST \subseteq P\}$
- $V_{st} \subseteq V_{a,sb,st,amo}$;
- $collect(ST_i) = \langle \{end\text{-}mode, \neg ST_i\text{-}violated, \overline{ST'_i}\}, \{ST', \neg \overline{ST'}\} \rangle$
- $forgo(ST_i) = \langle \{end\text{-}mode, ST_i\text{-}violated, \overline{ST'_i}\}, \{ST', \neg \overline{ST'}\} \rangle$
- $end = \langle \{normal\text{-}mode, \neg pause\}, \{end\text{-}mode, normal\text{-}mode\} \rangle$
- $O_{comp} = O_{neutral} \cup O_{chained} \cup O_{violation}$
- $O_{neutral} = \{\langle pre(o) \cup \{normal\text{-}mode, \neg pause\}, eff(o)\} \rangle \mid o \in O \wedge o \in N(\Pi)$;
- $O_{chained}$ and $O_{violation}$ are the compiled operators sets generated by the transformation schema applied to the operators of $\Pi$ 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 $o_{chained}$ of the non-neutral operators are defined as: $\bigcup_{o \in O, o \notin N(\Pi)} chain(o)$ where $chain(o)$ is a function defined further down;

- $c_{tv}(o)$ is the cost of an operator $o \notin N(\Pi)$.

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 $collect(ST)$, that has a cost 0 but requires that $ST$ is satisfied, or with action $forgo(ST)$, 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 $ST \in STS \subseteq \mathcal{P}$, a literal $is\text{-}violated\text{-}ST$ stating that $ST$ is violated until a $o \in S_{st}(o)$ satisfies the associated formula. For each sometime preference exactly one of $\{collect(ST), forgo(ST)\}$ appears in the plan.
This approach is not used for every other kind of preference, except sometime, whose violation is catched 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 $o \notin N(\Pi)$, is compiled into a set of new operators. The set of the $m$ preferences affected by $o$ is $I(o) = \{P_1, ..., P_m\}$. Then $o$ is compiled into a set of operators $chain(o) = \{\Theta(o, P_1), ..., \Theta(o, P_m)\}$ where each $\Theta(o, P_i)$ for $i \in [1, ..., m]$ 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 $typ(P_i)$ and the value of $i$. Furthermore the stage set are built in order to execute the following operators sequence $\omega_{chain(o)} = \langle o'_{P_1}, ..., o'_{P_m} \rangle$ where $o'_{P_i}$, with $i \in [1, ...m]$, is selected from the i-th set $\Theta(o, P_i)$.

Given a not-neutral operator $o$ of $\Pi$, the set of the compiled operators related to $o$ for $\Pi'$, called chain for $o$, is defined as:

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

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

In this presentation I provide the detailed descripion 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* $P_i = (\textit{at-most-once } a_i)$ *where* $a_i = \bigwedge_j a_{ij}$ *(where* $a_{ij}$ *is a clause) of* $I(o)$ *is:*

- *if* $i = 1$ *(init stage):*
$prec(o_{a_1}) = prec(o) \cup \{\neg pause, \neg is\text{-}violated\text{-}a_1, \neg seen\text{-}a_1\} \cup \{\bigcup_{a_{1j} \in \overline{C}(o, a_i)} a_{1j}\}$
$eff(o_{a_1}) = \{pause, seen\text{-}a_1, a_1\text{-}done_o\}$

$prec(\overline{o}_{a_1}) = prec(o) \cup \{\neg pause, \neg is\text{-}violated\text{-}a_1, seen\text{-}a_1\} \cup \{\bigcup_{a_{1j} \in \overline{C}(o, a_1)} a_{1j}\}$
$eff(\overline{o}_{a_1}) = \{pause, is\text{-}violated\text{-}a_1, a_1\text{-}done_o\}$

$prec(\overline{\overline{o}}_{a_1}) = prec(o) \cup \{\neg pause, \neg is\text{-}violated\text{-}a_1\} \cup \neg\{\bigwedge a_{1j} \in \overline{C}(o, a_i) a_{1j}\}$
$eff(\overline{\overline{o}}_{a_1}) = \{pause, a_1\text{-}done_o\}$

$prec\left(\overline{\overline{\overline{o}}}_{a_1}\right) = prec(o) \cup \{\neg pause, is\text{-}violated\text{-}a_1\}$
$eff\left(\overline{\overline{\overline{o}}}_{a_1}\right) = \{pause, a_1\text{-}done_o\}$

- *if* $1 < i < m = |I(o)|$ *(middle stage):*
$prec(o_{a_i}) = \{pause, \neg is\text{-}violated\text{-}a_i, \neg seen\text{-}a_i, a_{i-1}\text{-}done_o\} \cup \{\bigcup_{a_{ij} \in \overline{C}(o, a_i)} a_{ij}\}$
$eff(o_{a_i}) = \{pause, seen\text{-}a_i, \neg a_{i-1}\text{-}done_o, a_i\text{-}done_o\}$

$prec(\overline{o}_{a_i}) = \{pause, \neg is\text{-}violated\text{-}a_i, seen\text{-}a_i, a_{i-1}\text{-}done_o\} \cup \{\bigcup_{a_{ij} \in \overline{C}(o, a_i)} a_{ij}\}$
$eff(\overline{o}_{a_i}) = \{pause, is\text{-}violated\text{-}a_i, \neg a_{i-1}\text{-}done_o, a_i\text{-}done_o\}$

$prec(\overline{\overline{o}}_{a_i}) = \{pause, \neg is\text{-}violated\text{-}a_i, a_{i-1}\text{-}done_o\} \cup \neg\{\bigwedge a_{ij} \in \overline{C}(o, a_i) a_{ij}\}$
$eff(\overline{\overline{o}}_{a_i}) = \{pause, \neg a_{i-1}\text{-}done_o, a_i\text{-}done_o\}$

$prec\left(\overline{\overline{\overline{o}}}_{a_i}\right) = \{pause, is\text{-}violated\text{-}a_i, a_{i-1}\text{-}done_o\}$
$eff\left(\overline{\overline{\overline{o}}}_{a_i}\right) = \{pause, a_{i-1}\text{-}done_o, \neg a_{i-1}\text{-}done_o, a_i\text{-}done_o\}$

- *if* $i = m$ *(final stage):*
$prec(o_{a_m}) = \{pause, \neg is\text{-}violated\text{-}a_m, \neg seen\text{-}a_m, a_{m-1}\text{-}done_o\} \cup \{\bigcup_{a_{mj} \in \overline{C}(o, a_m)} a_{mj}\}$
$eff(o_{a_m}) = eff(o) \cup \{\neg pause, seen\text{-}a_m, \neg a_{m-1}\text{-}done_o\}$

$prec(\overline{o}_{a_m}) = \{pause, \neg is\text{-}violated\text{-}a_m, seen\text{-}a_m, a_{m-1}\text{-}done_o\} \cup \{\bigcup_{a_{mj} \in \overline{C}(o, a_m)} a_{mj}\}$
$eff(\overline{o}_{a_m}) = eff(o) \cup \{\neg pause, is\text{-}violated\text{-}a_m, \neg a_{m-1}\text{-}done_o\}$

$prec(\overline{\overline{o}}_{a_m}) = \{pause, \neg is\text{-}violated\text{-}a_m, a_{m-1}\text{-}done_o\} \cup \neg\{\bigwedge a_{mj} \in \overline{C}(o, a_m) a_{mj}\}$
$eff(\overline{\overline{o}}_{a_m}) = eff(o) \cup \{\neg pause, \neg a_{m-1}\text{-}done_o\}$

$prec\left(\overline{\overline{\overline{o}}}_{a_m}\right) = \{pause, is\text{-}violated\text{-}a_m, a_{m-1}\text{-}done_o\}$
$eff\left(\overline{\overline{\overline{o}}}_{a_m}\right) = eff(o) \cup \{\neg pause, a_{m-1}\text{-}done_o, \neg a_{m-1}\text{-}done_o\}$

In accordance with Definition 12 the i-th stage $\theta_{amo}(o, P_i)$ 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;

- $\overline{o}_{a_i}$ is a *violation* of $P_i$ because the related formula $a_i$ has been true in a previous state;
- $\overline{\overline{o}}_{a_i}$ is a *neutral* operator for $P_i$ because it does not make $a_i$ true;
- $\overline{\overline{\overline{o}}}_{a_i}$ 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 compilative scheme for three type of preference which were not considered in the previous work. All the propose compilative 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


Ceriani, L., and Gerevini, A. E. 2015. Planning with always preferences by compilation into strips with action costs. In *Eighth Annual Symposium on Combinatorial Search*.

Gerevini, A. E.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners. *Artificial Intelligence* 173(5):619–668.

Keyder, E., and Geffner, H. 2009. Soft goals can be compiled away. *Journal of Artificial Intelligence Research* 547–556.