# Conformant Planning through Classical Planning

**Héctor Palacios**
Departamento de Tecnología
Universitat Pompeu Fabra
Pg Circunvalación, 8
08003 Barcelona, SPAIN
hector.palacios@upf.edu

**Héctor Geffner**
Departamento de Tecnología
ICREA & Universitat Pompeu Fabra
Pg Circunvalación, 8
08003 Barcelona, SPAIN
hector.geffner@upf.edu

**Abstract**

We define a general framework for translating a conformant planning problem into a classical planning one, that is solved by an off-the-shelf classical planner. The T0 planner works by detecting which configuration should be used depending on the problem. This lead to a fast and complete conformant planner.

## Introduction

Conformant planning is a form of planning where a goal is to be achieved when the initial situation is not fully known and actions may have non-deterministic effects (Goldman & Boddy 1996)[1]. Conformant planning is computationally harder than classical planning, as even under polynomial restrictions on plan length, plan verification remains hard (Turner 2002).

The T0 planner is based on a framework for the mapping conformant planning problems into classical planning problems than are then solved by an off-the-shelf classical planner (Palacios & Geffner 2007). We present such framework and explain how the planner T0 instantiate it. Some implementation details are outlined.

## A general mapping from Conformant into Classical Planning

Given a conformant planning problem $P = \langle F, O, I, G\rangle$, where $F$ stands for the fluents, $O$ for the actions, $I$ for the initial situation and $G$ for the goal, we want to obtain a planning problem. Every action $a$ has a precondition given by a set of fluent literals, and a set of conditional effects $C \rightarrow L$ where $C$ is a *set* of fluent literals and $L$ is a literal. Sometimes we write them as $a : C \rightarrow L$.

The translation $K_{T,M}$ depends on two parameters $T$ and $M$, a set of *tags* and a set of *merges*, respectively. A tag $t \in T$ is a set of literals $L$ from $P$ whose truth value is not known in the initial situation $I$. A merge $m \in M$ is a valid subset of tags, *i.e.* for a merge $m = \{t_1, \ldots, t_n\}$ the following holds:

$$I \models t_1 \vee \cdots \vee t_n$$

[1]We assume that actions are deterministic and all the uncertainty is on the initial state. This assumption does not lead to loss of expressivity.

The tagged literals $KL/t$ in the translation $K_{T,M}$, where $L$ is a literal in $P$ and $t \in T$ is a tag, capture the conditional 'it is known that if $t$ is true *initially*, then $L$ is true'.[2]

**Definition 1** ($K_{T,M}(P)$) *(Palacios & Geffner 2007) Let $P = \langle F, O, I, G\rangle$ be a conformant problem, then $K_{T,M}(P) = \langle F', I', O', G'\rangle$ is defined as:*

- $F' = \{KL/t, K\neg L/t \mid L \in F \text{ and } t \in T\}$
- $I' = \{KL/t \mid \text{if } I \models t \supset L\}$
- $G' = \{KL \mid L \in G\}$
- $O' = \{a : KC/t \rightarrow KL/t, a : \neg K\neg C/t \rightarrow \neg K\neg L/t \mid a : C \rightarrow L \text{ in } P\} \cup \{merge_{L,m} : KL/t_1 \wedge \cdots \wedge KL/t_n \rightarrow KL \mid t_i \in m \text{ and } m \in M\}$

*where $KL$ is a precondition of action $a$ in $K_{T,M}(P)$ if $L$ is a precondition of $a$ in $P$.*

For given $T$ and $M$, conformant plans for $P$ are get from classical plans of $K_{T,M}(P)$, by dropping $merge$ actions. But for some $T$ and $M$, $K_{T,M}(P)$ can be incomplete, not containing all the possible plans for $P$.

There are always $T$ and $M$ that lead to a complete $K_{T,M}(P)$, but in the worst case it will be exponentially bigger than $P$. Most current benchmarks can be solved with simple $T$ and $M$, getting a polynomial size increasing.

As we comment on (Palacios & Geffner 2007), the required $T$ and $M$ for a problem $P$ is related with the minimum number of clauses relevant to any precondition or goal $L$. This number is called the conformant width. Most of current benchmarks are solvable using $T$ and $M$ smaller than the suggested by the width.

### Example

Consider the problem of moving an object from an origin to a destination using two actions: $pick(l)$ picks up the object if the object is at $l$ and the hand is empty, while if the hand is not empty, $pick(l)$ just releases the object at $l$; $drop(l)$ that drops the object in a location if the object is being held.

Let us assume that these are conditional effects, and that there are no preconditions.

Given an instance $P$ where the object is at either $l_1$ or $l_2$, and must be moved to $l_3$, a conformant plan will be

$$\pi = \{pick(l_1), drop(l_3), pick(l_2), drop(l_3)\}$$

[2]*i.e.* tags are assumed to be true at *time-zero* (T0)

Let us consider now the translation $K_{T,M}(P)$ with $T = \{at_1, at_2\}$, and the single merge $m = \{at_1, at_2\}$, given that $at_1 \vee at_2$ is valid in $I$. We can show now that the plan $\pi'$

$$\{pick(l_1), drop(l_3), pick(l_2), drop(l_3), merge_{L,m}\}$$

for $L = at_3$ solves the classical problem $K_{T,M}(P)$, and that the plan $\pi$ obtained from $\pi'$ by dropping the merge action, is a valid conformant plan for $P$. We can see how some of the literals in $K_{T,M}(P)$ evolve as the actions in $\pi'$ are done:

| | |
|---|---|
| 0:$Kat_1/at_1, Kat_2/at_2$ | true in $I'$ |
| 1:$Khold/at_1, Kat_2/at_2$ | true after $pick(l_1)$ |
| 2:$Kat_3/at_1, Kat_2/at_2$ | true after $drop(l_3)$ |
| 3:$Kat_3/at_1, Khold/at_2$ | true after $pick(l_2)$ |
| 4:$Kat_3/at_1, Kat_3/at_2$ | true after $drop(l_3)$ |
| 5:$Kat_3$ | true after action $merge_{L,m}$ |

where $merge_{L,m}$ is the action with the conditional effect

$$Kat_3/at_1 \wedge Kat_3/at_2 \rightarrow Kat_3$$

whose condition is true before Step 5 producing $Kat_3$

## T0: a conformant planners based on $K_{T,M}(P)$

Given a conformant problem $P$, the planner T0 instantiates $K_{T,M}(P)$ with different $T$ and $M$, and use the FF classical planner to solve them (Hoffmann & Nebel 2001).

The T0 planner tries to use tags of size 0 or 1 to solve a problem $P$, getting $K_0(P)$ and $K_1(P)$ as defined in (Palacios & Geffner 2007). Otherwise, it generates tags with size $i > 1$, so that $K_i(P)$ is enough for obtaining any solution, but slowly and using more memory.

The classical PDDL $K_{T,M}(P)$ is enriched using ideas borrowed from (Palacios & Geffner 2006). First, a procedure call *action compilation* allows to add additional conditional effects to obtain more information after executing some actions. Second, T0 adds actions for reasoning over disjunctions true at the initial state, which value are not changed by any action. The combination of these techniques has dramatic effects on some domains without affecting the general performance.

The main optimization after selecting the tags $T$ and merges $M$ is not generating all the possibles $KL/t$ atoms, for any lit $L$ and any tag $t$. An atom $KL/t$ is generated only when $L$ is relevant to some goal or precondition $G$, $t$ is in a merge $m$ relevant to $G$, and $t$ is relevant to $L$. Otherwise, $KL/t$ can be safely replaced by $KL$.

The T0 planner is implemented in C++ and OCaml. The PDDL parser and some preprocessing is in C++[3]. After grounding and simple reachability analysis is done, the translator, implemented in OCaml, is invoked. It calculates bounds on the conformant width of the problem, deciding whether some instances of $K_{T,M}(P)$ as $K_0$, $K_1$ or $K_i$ for $i > 1$ can be feasible and useful. Then it generates the selected instances that, after a final reachability analysis, are saved in PDDL format. The classical planner FF is called upon them. Normally, FF detects very fast when some $K_i$ is not solvable, leading to T0 trying other instance. FF is never run more than three times.

A crafted version of FF was used, improved to deal better with huge grounded PDDL files (up to 100 MB). Most available classical planners are not well suited for dealing with grounded theories.

The performance of T0 depends on the fact that FF does not compile conditional effects into pure STRIPS. That would lead to further increasing in the size of the theories (Rintanen 2003). Translating conditional into STRIPS can be done by creating new actions, encoding each possible interaction of conditional effects in an action as a new action, that would case an exponential blow-up for our theories. Other option requires to introduce new literals and more actions, but that tend to perform bad with delete-free based heuristics planners (Bonet & Geffner 2001). Nowadays, good support of hundred of conditional effects is the main difficulty on using other classical planners with $K_{T,M}(P)$.

[3]based on code by Blai Bonet

## Acknowledgments

We thank Jörg Hoffmann for making FF available, and Alexandre Albore for useful discussion and helping us with improvements on FF.

## References

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Goldman, R. P., and Boddy, M. S. 1996. Expressive planning and explicit knowledge. In *Proc. AIPS-1996*.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Palacios, H., and Geffner, H. 2006. Compiling uncertainty away: Solving conformant planning problems using a classical planner (sometimes). In *Proc. AAAI-06*.

Palacios, H., and Geffner, H. 2007. From conformant into classical planning: Efficient translations that may be complete too. In *Proc. of the 17th Int. Conf. on Planning and Scheduling (ICAPS-07)*. AAAI Press.

Rintanen, J. 2003. Expressive equivalence of formalisms for planning with sensing. In *Proc. ICAPS'03*.

Turner, H. 2002. Polynomial-length planning spans the polynomial hierarchy. In *JELIA '02: Proc. of the European Conference on Logics in AI*, 111–124. Springer-Verlag.