Compiling Uncertainty Away in Non-Deterministic Conformant Planning

Alexandre Albore[1] and **Hector Palacios**[2] and **Hector Geffner**[3]

Abstract. It has been shown recently that deterministic conformant planning problems can be translated into classical problems that can be solved by off-the-shelf classical planners. In this work, we aim to extend this formulation to non-deterministic conformant planning. We start with the well known observation that non-deterministic effects can be eliminated by using hidden conditions that must be introduced afresh each time a non-deterministic action is applied. This observation, however, leads to translations that have to be recomputed as the search for plans proceeds. We then introduce other translations, that while incomplete, appear to be quite effective and result in classical planning problems that need to be solved only once. A number of experimental results over existing and new domains are reported.

1 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 [7, 14]. While few practical problems are purely conformant, the ability to find conformant plans is needed in contingent planning where conformant situations are a special case. It has also been shown that solutions to contingent problems can benefit from conformant planning methods [8, 1], and more recently, that a large class of contingent problems can be translated into conformant ones [4].

The problem of conformant planning can be formulated as a path-finding problem in belief space, where a sequence of actions that map an initial belief state into a target belief state is sought [3]. A belief state represents the set of states that are deemed possible, and actions, whether deterministic or not, map one belief state into another. This formulation, that underlies most current conformant planners [9, 5, 6], must address the representation of beliefs, and the derivation and use of effective belief heuristics.

An alternative approach has been pursued recently in the form of a family of translations $K_{T,M}$, where T and M are two parameters, such that the plans for a *deterministic* conformant problem P are obtained from the plans of the classical problem $K_{T,M}(P)$, that are computed by off-the-shelf classical planners [12]. These $K_{T,M}$ translations are sound, and for suitable choices of the parameters — the set of tags T and merges M — are also complete. For example, the translation K_i is exponential in the non-negative integer i and is complete for problems with conformant width bounded by i. The conformant width gives an idea of the structural complexity of the problem, and is related to the size of the tags required for a complete translation. The translation K_1 has turned out to be particularly practical as it can be computed very fast and yields solutions to most existing deterministic benchmarks.

In this work, we aim to extend the translation-based approach to conformant problems featuring *non-deterministic* effects. We start with the well known observation that non-deterministic effects can be eliminated by using hidden conditions that must be introduced afresh each time a non-deterministic action is applied. This observation, however, leads to translations that have to be recomputed as the search for plans proceeds. We then introduce other translations, that while incomplete, appear to be quite effective and involve calling a classical planner only once.

The paper is organized as follows. We recall first the definition of non-deterministic conformant problems and the translation of deterministic conformant problems into classical ones. We then consider a standard deterministic relaxation of non-deterministic problems and its use for defining three types of translation-based planners able to handle non-deterministic actions. We then test these planners and summarize the results.

2 Non-Deterministic Conformant Planning

Conformant planning problems P are represented as tuples of the form $P = \langle F, I, O, G\rangle$ where F stands for the fluent symbols in the problem, I is a set of clauses over F defining the initial situation, O stands for a set of (ground) operators or actions a, and G is a set of literals over F defining the goal. Every action a has a precondition $Pre(a)$ given by a set of fluent literals, and a set of conditional effects or rules $a : C \rightarrow C_1 \,|\, C_2 \,|\, \cdots \,|\, C_n$, $n \geq 1$, where C and C_i stand for sets (conjunctions) of literals, and C can be empty. The effect is *deterministic* if $n = 1$, and *non-deterministic* otherwise. When convenient, we take a deterministic effect $C \rightarrow C'$ as the set of effects $C \rightarrow L$ for each $L \in C'$. We write the complement of a literal L as $\neg L$.

The semantics of a non-deterministic problem $P = \langle F, I, O, G\rangle$ is defined in terms of the state trajectories that are possible. A state is a set of literals representing a truth assignment over the fluents in F. An action $a \in O$ is applicable in s, if $Pre(a) \subseteq s$, and s' is a *possible successor state* of s given a if for each of the conditional effects

$$C^i \rightarrow C^i_1 \,|\, \cdots \,|\, C^i_{n_i}$$

associated with the action a, s' is the single successor state of s given an action a' that is like a but with the *deterministic* conditional effects $C^i \rightarrow C^i_{f(i)}$, where $1 \leq f(i) \leq n_i$ is a function that selects one outcome $C^i_{f(i)}$ from the set of possible outcomes. We assume that this successor state is always well defined, and hence, that if two

[1] Universitat Pompeu Fabra, Spain, email: alexandre.albore@upf.edu.
[2] Universidad Simón Bolívar, Venezuela, email: hlp@ldc.usb.ve.
[3] ICREA & Universitat Pompeu Fabra, Barcelona, Spain, email: hector.geffner@upf.edu.

outcomes $C^i_{f(i)}$ and $C^k_{f(k)}$ are complementary, the bodies of such effects C^i and C^k can't be reached jointly from any possible initial state. The state trajectories $s_0, \ldots, s_{n+1}$ that are possible given an action sequence $a_0, \ldots, a_n$ are the ones that start in a possible initial state s_0 and s_{i+1} is a possible successor state of s_i given a_i.

An action sequence $a_0, \ldots, a_n$ is a *conformant plan* for P if each action a_i, $i = 1, \ldots, n$, is applicable in the state s_i of all the state trajectories $s_0, \ldots, s_i$ that are possible given the preceding action sequence $a_0, \ldots, a_{i-1}$, and s_{n+1} is a goal state.

Alternatively, if b_0 is the set of initial states that are possible, and b_{i+1} is the set of states that are possible given an action a_i applicable in each state in b_i, $a_0, \ldots, a_n$ is a conformant plan for P if it maps the initial set b_0 into a final set b_{n+1} of goal states. The sets of states b_i, $i = 0, \ldots, n$, are called *belief states.*

3 Translation of Deterministic Problems

The translation-based approach to conformant planning maps conformant problems P into classical problems $K(P)$ that can be solved by off-the-shelf planners. In this section, we review the main ideas [12]. The simplest translation, called K_0, replaces the literals L in P by literals KL and $K\neg L$ that are aimed at capturing whether L is known to be true and known to be false respectively.

Definition 1 (K_0) *For a* deterministic *conformant problem $P = \langle F, I, O, G\rangle$, the translation $K_0(P) = \langle F', I', O', G'\rangle$ is the classical planning problem with*

- $F' = \{KL, K\neg L \mid L \in F\}$,
- $I' = \{KL \mid L$ *is a unit clause in* $I\}$,
- $G' = \{KL \mid L \in G\}$, *and*
- $O' = O$ *with each precondition L for $a \in O$ replaced by KL, and each conditional effect $C \rightarrow L$ replaced by $KC \rightarrow KL$ and $\neg K\neg C \rightarrow \neg K\neg L$.*

The expressions KC and $\neg K\neg C$ for $C = L_1, L_2 \ldots$ are abbreviations for $KL_1, KL_2 \ldots$ and $\neg K\neg L_1, \neg K\neg L_2 \ldots$ respectively.

The intuition behind the translation is simple: first, the literal KL is true in I' if L is known to be true in I; otherwise it is false. This removes all uncertainty from $K_0(P)$, making it into a classical planning problem. In addition, for soundness, each rule $a : C \rightarrow L$ in P is mapped into *two* rules: a **support rule** $a : KC \rightarrow KL$, that ensures that L is known to be true when the condition is known to be true, and a **cancellation rule** $a : \neg K\neg C \rightarrow \neg K\neg L$ that guarantees that $K\neg L$ is deleted (prevented to persist) when action a is applied and C is not known to be false.

The translation $K_0(P)$ is sound as every classical plan that solves $K_0(P)$ is a conformant plan for P, but incomplete, as not all conformant plans for P are classical plans for $K_0(P)$.

The more general translation schema $K_{T,M}$ builds on the K_0 translation using two parameters: a set T of *tags* t and a set M of merges m. The tags and the merges are used to account for conformant plans that reason by cases; indeed, the tags are used to introduce assumptions about the initial situation that are eliminated via the merges. The new literals KL/t in the translation aim to express that L is known to be true if t is true in the initial situation. A merge m is a non-empty collection of tags t in T that stands for the Disjunctive Normal Form (DNF) formula $\bigvee_{t\in m} t$. A merge m is *valid* when one of the tags $t \in m$ must be true in I. The translation assumes all merges to be valid. A merge m for a literal L in P translates then into a 'merge action' with single effect

$$\bigwedge_{t\in m} KL/t \;\rightarrow\; KL \;.$$

The set of 'merge actions' associated with the set of merges M is referred to as O_M. The translation $K_{T,M}(P)$ is the basic translation $K_0(P)$ 'conditioned' with the tags t in T and extended with the set O_M of actions. The literals KL are assumed to stand for the literals KL/t where t is the 'empty tag'. The empty tag expresses no assumption about the initial situation and is assumed implicitly in every set T.

Definition 2 ($K_{T,M}$) *Let $P = \langle F, I, O, G\rangle$ be a conformant problem, then $K_{T,M}(P)$ is the* classical planning problem *$K_{T,M}(P) = \langle F', I', O', G'\rangle$ with*

- $F' = \{KL/t, K\neg L/t \mid L \in F$ *and* $t \in T\}$,
- $I' = \{KL/t \mid\ I, t \models L\}$,
- $G' = \{KL \mid L \in G\}$, *and*
- $O' = O_M \cup O$ *with each precondition L for $a \in O$ replaced by KL, and each conditional effect $C \rightarrow L$ replaced by $KC/t \rightarrow KL/t$ and $\neg K\neg C/t \rightarrow \neg K\neg L/t$.*

The translation $K_{T,M}(P)$ reduces to the basic translation $K_0(P)$ when M is empty and T contains only the empty tag. For suitable choices of T and M, the translation $K_{T,M}(P)$ can be both *sound* and *complete*: sound, meaning that the classical plans for $K_{T,M}(P)$ are all conformant plans for P once merge actions are removed, and complete, meaning that all conformant plans for P yield classical plans for $K_{T,M}(P)$ once merge actions are added.

One way to get a complete translation is by associating the tags in T with the set of possible initial states of P. However, complete translations that are compact are also possible [12]. The core of such translations is given by the set $C_I(L)$ of clauses in I, including possibly tautologies of the form $L' \vee \neg L'$, that are *relevant* to a literal L. Assuming that I is in prime implicate form, a translation can be shown to be *complete* when, for each precondition and goal literal L, it includes a 'covering merge': this is a merge $m = \{t_1, \ldots, t_n\}$ that corresponds to the valid DNF formula $t_1 \vee \cdots \vee t_k$ in I such that each term t_i *satisfies* $C_I(L)$. Such tags and merges can be computed in time that is exponential in a parameter associated with the set $C_I(L)$ that is called the *width* of L.

$K_{S0}(P)$ is the translation $K_{T,M}(P)$ obtained by setting T to the set of possible initial states S_0 and having one merge in M equal to S_0 for each precondition and goal literal. An alternative complete translation is $K_{models}(P)$, where the merge for L is defined as the set of models of the clauses $C_I(L)$ in I relevant to L. More interestingly, the $K_i(P)$ translation for a fixed integer $i \geq 0$, is polynomial and complete for problems P with conformant width bounded by i. The merges for L in $K_i(P)$ are defined by converting subsets of i clauses from $C_I(L)$ into DNF.

As an illustration, consider the conformant problem P with initial situation $I = \{x_1 \vee \cdots \vee x_m\}$, goal $G = L$, and actions a_i, $i = 1, \ldots, m$, each with effect $x_i \rightarrow L$. For this problem, there is a single goal or precondition literal L, and the set of clauses in I relevant to L, $C_I(L)$, contains the single clause $x_1 \vee \cdots \vee x_m$. For this problem, $K_{S0}(P)$ contains the merge $m = S_0$ for L, where S_0 stands for the set of possible initial states of P. Since, L is initially unknown, and there are $2^m - 1$ satisfying assignments over the x_i variables, there are $2 \cdot (2^m - 1)$ possible initial states, and hence $2 \cdot (2^m - 1) + 1$ tags. The K_{models} translation associates the merge m for L with the

models of $C_I(L)$ that involve only the x_i variables, and hence results in $(2^m - 1) + 1 = 2^m$ tags. Last, the $K_i(P)$ translation features merges for L obtained by translating sets of i-clauses from $C_I(L)$ into DNF. In particular, K_1 contains the merge $m = \{x_1, \ldots, x_m\}$ obtained from translating the single clause $x_1 \vee \cdots \vee x_m$ in $C_I(L)$, which is already in DNF. Since the width of P is trivially 1 — there is a single goal or precondition literal with a singleton $C_I(L)$ set — the translation K_1, while polynomial, is guaranteed to be complete. The fluents in this translation are of the form KL'/t, where L' is L, x_i, or their negations, and t is x_i or the empty tag, $i = 1, \ldots, m$. The conformant plans for P are the classical plans for $K_1(P)$ with the merge actions removed.

4 Deterministic Relaxation

The translations $K_{T,M}$ above are for deterministic conformant problems. For translating *non-deterministic* problems, one possibility is to take advantage of a well known transformation that maps each one of the *non-deterministic* effects

$$C^i \rightarrow C^i_1 \mid C^i_2 \mid \cdots \mid C^i_{n_i(a)} \tag{1}$$

of an action a with $n_i(a)$ possible outcomes, $i = 1, \ldots, n(a)$, into $n_i(a)$ *deterministic* effects

$$C^i, h^i_k(a) \rightarrow C^i_k \tag{2}$$

$k = 1, \ldots, n_i(a)$, along with 'oneof' expressions

$$oneof(h^i_1(a), \ldots, h^i_{n_i}(a)) \tag{3}$$

added to I. In this transformation, uncertainty in the state transitions is converted into uncertain conditions $h^i_k(a)$ in the initial situation.

The *deterministic* conformant problem that follows from applying this transformation, that we call P_d, is not equivalent to the original *non-deterministic* conformant problem P in general, but is equivalent to P over plans that do not involve non-deterministic actions more than once:

Proposition 3 *Let π be an action sequence that involves each non-deterministic action from P at most once. Then π is a plan for the* non-deterministic *conformant problem P iff π is a plan for the* deterministic *conformant plan P_d.*

When non-deterministic actions are applied more than once, the difference between P and P_d is that the hidden h conditions in P_d establish correlations among the possible outcomes of the same action during the execution of a plan. In particular, the possible outcome of an action a in P can be C^i_k the first time, and $C^i_{k'}$ the second time, with $k' \neq k$, but this is not possible in the deterministic relaxation P_d.

For example, if $move$ is a non-deterministic action in P with non-deterministic effect

$$x(i), y(j) \;\rightarrow\; x(i+1), \neg x(i) \;\Big|\; y(j+1), \neg y(j)\ ,$$

a sequence of two $move$ actions starting at $\langle x(0), y(0)\rangle$ will result in three possible locations: $\langle x(2), y(0)\rangle$, $\langle x(0), y(2)\rangle$, and $\langle x(1), y(1)\rangle$. On the other hand, only the first two locations are possible in P_d; the first follows from states where the first hidden condition $h^1_1(move)$ is true; the second, from states where the second hidden condition $h^1_2(move)$ is true.

The transformation of P into P_d can be used however to obtain an incomplete non-deterministic conformant planner in a simple manner. A *classical* planner is called over the translation $K(P_d)$ for some $K = K_{T,M}$, and if no non-deterministic action from P appears twice in the plan returned, from the soundness of the translation and Proposition 3, the plan with the merge actions removed must be a plan for P.

Below we will use the relaxation P_d, the translations $K_{T,M}$, and off-the-shelf classical planners to define various types of non-deterministic conformant planners. None is complete, but some, as we will see, are pretty effective.

Before proceeding with the description of such planners, it will be useful to consider first a generalization of the deterministic relaxation P_d that works on a variant of P that we call P^m, which is exactly like P but with each non-deterministic action a copied m times, with different names $a^1, \ldots, a^m$. These copies make no difference to P, as the problems P and P^m are equivalent, but make a difference in the relaxations P_d and P^m_d of P and P^m respectively that are not equivalent. Indeed, while the relaxation P_d can capture plans for P that include each non-deterministic action of P *at most once*, the relaxation P^m_d can capture plans for P where each non-deterministic action is done *at most m times.* Indeed, for a sufficiently large m, P^m_d will necessarily account for a plan that solves P, and even for a plan that solves P optimally.

We can actually modify the relaxation P^m_d slightly so that the translation $K(P^m_d)$ generates sound plans only; namely plans where each non-deterministic action is applied up to m times and no more. The change is very simple: we create new fluents $blocked(a^k)$, for each copy a^k of a non-deterministic action a in P^m_d, $k = 1, \ldots, m$, and 1) set all these atoms true initially, except for $blocked(a^1)$, that is false initially, 2) add the literal $\neg blocked(a^k)$ to the precondition of the action a^k, and 3) add the literals $\neg blocked(a^{k+1})$ and $blocked(a^k)$ to its effects. Notice that that the relaxation P_d is P^m_d with $m = 1$.

5 Three Non-Deterministic Conformant Planners

We focus now on the definition of three types of non-deterministic conformant planners, all of which rely on the use of classical planners. They are all based on the relaxations above for eliminating non-deterministic effects, and on the various translations K for mapping deterministic conformant problems into classical ones.

5.1 A K-Replanner

The K-Replanner is a *lazy* but incomplete implementation of a planner based on the translation K of the deterministic relaxation P^m_d of P, for an arbitrary m, where the last copy a^k of each non-deterministic action a does not get blocked and can be used more than once. It exploits the property that if such actions a^k are not used more than once in the classical plan π returned for $K(P^m_d)$, then π is a conformant plan for P (once the merge actions are dropped). On the other hand, the situation where a plan $\pi = a_0, a_1, \ldots, a_k$ is returned for $K(P^m_d)$ such that $\pi_{i+1} = a_0, \ldots, a_{i+1}$ is the first prefix of π that violates this condition, constitutes a *flaw*, that is 'repaired' by preserving the 'flawless' prefix π_i, while merging it with a plan tail π' obtained recursively from the resulting state s_{i+1} over an encoding, that is like $K(P^m_d)$ except that the 'faulty' action copy a^k is split into two: the action a^k itself, that is blocked in s_{i+1}, and a new copy a^{k+1} with its own fresh hidden h variables, that is not. The resulting planning algorithm is *dynamic* in the sense that each time a

classical plan with a 'flaw' is returned, a plan tail is computed (recursively) over a classical problem that is slightly different, and includes more fluents (the h and $blocked$ fluents), more initial conditions (involving the new $blocked$ fluents and one-of h expressions), and more actions (the merges resulting from the new one-of expressions). Basically, if P_i is the deterministic relaxation of P before the flaw, and P_{i+1} is the deterministic relaxation after the flaw, the classical problems before and after the flaw are $K(P_i)$ and $K(P_{i+1})$ respectively. Notice that for all translation schemas $K = K_{T,M}$, the translation $K(P_{i+1})$ can be computed incrementally with minor modifications from the translation $K(P_i)$ of the previous deterministic relaxation P_i.

We call this planning schema able to handle non-deterministic actions starting with $P_1 = P_d$, the K-replanner. The K-replanner is incomplete even if the translation K is complete for deterministic problems. The incompleteness is a result of the commitment to the 'flawless' plan prefixes that are extended after each iteration, and which may render a solvable problem P, unsolvable. The schema, however, is sound:

Proposition 4 *If the K-replanner returns an action sequence π, then π with the merge actions removed is a plan for the non-deterministic conformant problem P.*

5.2 K-Reset Planner

The second and third planning schemas are simpler and require calling a classical planner only once. The classical plan returned is a solution to the original non-deterministic conformant problem. Both schemas are thus sound, and while neither one is complete, they turn out to be more effective than the K-replanner.

The K-reset planner uses the translation $K(P_d)$ of the deterministic relaxation P_d extended with the *blocked* fluents that prevent a non-deterministic action from being applied more than once. Yet, the classical encoding $K(P_d)$ is extended with reset actions, $reset(a)$, one for each non-deterministic action a, that can be used to unblock these actions and use them multiple times *in a sound manner.*

The definition of the $reset(a)$ action takes advantage of the structure of the translations $K = K_{T,M}$, and it allows multiple occurrences of non-deterministic actions without having to introduce multiple action copies. It is based on an idea that we express first in the language of conformant planners that search in belief space. Assume a belief space planner that represents beliefs as sets (conjunctions) of formulas. Actions in such a setting, deterministic or not, map one set of formulas F_i into another set F_{i+1}. Likewise, an action sequence $a_0, \ldots, a_n$ is a plan if it maps the initial set of formulas F_0 into a final set F_{n+1} that implies the goal. Now consider a version of such a planner that drops some of the formulas in F_{i+1} and thus maps a set of formulas F_i into a weaker set F'_{i+1}. Such a planner is still sound but possibly incomplete. The relevant observation here is that one such incomplete planner over the deterministic relaxation P_d can accommodate plans with multiple occurrences of non-deterministic actions a provided that, before new occurrences of the same action a are applied in P_d, *all beliefs (formulas) involving hidden conditions $h^i_k(a)$ are dropped.*

This is precisely what the $reset(a)$ action does: it unblocks the action a while deleting all beliefs involving the hidden conditions $h^i_k(a)$ associated with a. This is achieved by having the literal $blocked(a)$ as a precondition of $reset(a)$, and the literals $\neg blocked(a)$ and $\neg KL/t$ as its effects, for all L and tags t that include a hidden condition $h^i_k(a)$.[4]

If P_d is the deterministic relaxation of P extended with the $blocked(a)$ fluents, and $K(P_d)$ is the translation extended with the $reset(a)$ actions, it can be proved that the classical plans for $K(P_d)$ are all plans for P:

Proposition 5 *Any plan π returned by a classical planner from the translation $K(P_d)$ extended with the reset actions, is a plan for the non-deterministic conformant problem P, once the merge and reset actions are removed.*

5.3 $\langle K, K_0 \rangle$ Planner

The third non-deterministic planner is a special case of the K-reset planner that uses a particular type of K translation for getting rid of the $reset(a)$ actions and the $blocked(a)$ fluents. Indeed, it follows from the arguments above that the reset actions and blocking fluents are not needed in the K-reset planner when the translation K is such that it does not generate tags involving the hidden conditions $h^i_k(a)$. For example, the K-reset planner for $K = K_0$ does not require blocking fluents and reset actions as it does not generate any tags at all, except for the empty tag. The K_0-reset planner, however, is just too weak.

A much larger family of translations that do not generate tags involving the hidden conditions $h^i_k(a)$ can be defined in analogy to the family of deterministic translations K_{models}, and K_i for $i \geq 0$. Recall that these translations are instances of the general $K_{T,M}$ translation defined by the manner in which they map subsets of clauses of $C_I(L)$ into merges for L. For a translation K_x, let us denote its set of merges for each literal L, as $m_x(C_I(L))$. A class of translations $\langle K_x, K_0 \rangle$ can then be defined for the deterministic relaxation P_d of P by simply splitting the set of clauses in $C_I(L)$ into two sets: the clauses $C^h_I(L)$ that involve hidden conditions $h^i_k(a)$ for some action a, and the clauses $C^o_I(L)$ that do not. The translation $\langle K, K_0 \rangle$ for $K = K_x$ can then be defined by discarding the clauses $C^h_I(L)$ involving the hidden conditions, and hence applying the K_x translation to the remaining set of clauses $C^o_I(L)$ only. Namely, the merges in the translations $\langle K, K_0 \rangle$ are simply the merges $m_x(C^o_I(L))$ for the goal and precondition literals L, and the resulting set of tags is the set of tags in all such merges (along with the empty tag). It is clear that the translation $\langle K, K_0 \rangle$ does not generate tags involving the hidden conditions $h^i_k(a)$, and hence nor beliefs that are conditional over such conditions. It does not require therefore to block or to reset (non-deterministic) actions a whose effects depend on them, as it does not generate any such action. It can then be shown that:

Proposition 6 *Any plan π returned by a classical planner from the translation $\langle K, K_0 \rangle(P_d)$ is a plan for the non-deterministic conformant problem P once the merge actions are dropped.*

Since the $\langle K, K_0 \rangle$ translation does not capture disjunctive reasoning over the hidden h-conditions, we extend it with two types of general inference rules from [11], implemented as additional actions in the classical encoding that capture some of those patterns.

The first is the *static-or* rule, that is based on the disjunctions $x_1 \vee \cdots \vee x_n$ in the problem P that are *invariant*, meaning that are

[4] It is actually not necessary to delete all literals KL/t involving tags featuring the hidden conditions $h^i_k(a)$ before applying the action a a new time; it suffices to delete all such literals when KL does not hold. Otherwise, all literals KL/t can be maintained. This refinement is often convenient and is part of the K-reset planner tested below.

true in all reachable belief states. The associated action has n conditional effects $C_i \rightarrow Kx_i$, $i = 1, \ldots, n$, where C_i is the conjunction of all literals $K\neg x_k$, $k \neq i$. For example, in a grid $n \times m$, the disjunctions $x_1 \vee \cdots \vee x_n$ and $y_1 \vee \cdots \vee y_m$ encoding the possible x and y locations are invariant and therefore result in two actions of this type.

The second rule, called *action compilation* [11], makes explicit effects that are otherwise implicit. For example, the action move-right with conditional effects $x_i \rightarrow x_{i+1}$ in P, for $i = 1, \ldots, n-1$, generates effects like $Kx_i \rightarrow Kx_{i+1}$ in all $K_{T.M}(P)$ translations. Inspection of the action, however, reveals other effects like $K\neg x_i \rightarrow K\neg x_{i+1}$. Action compilation obtains such implicit effects in polynomial time by considering each action in isolation, as a preprocessing step.

6 Experimental Results

We evaluate the performance of the $\langle K, K_0 \rangle$ and K-reset planners using LAMA [13] and FF [10] as the base classical planners. We do not include results for the K-Replanner, as the experiments using a preliminary implementation suggest that it does not scale up. We compare with MBP and KACMBP [6, 2], which to the best of our knowledge are the only other (qualitative) conformant planners that deal with non-deterministic actions. The results are shown in Table 1. The table shows times in seconds, including preprocessing, translation, and search, and plan costs, as measured in the number of actions in the plan. The data has been generated on PCs running Linux at 2.33GHz with 8GB of RAM, with a cutoff time of 2 hours, and a memory bound of 2.1GB.

Many of the domains are from the MBP/KACMBP and T_0 distributions [6, 2, 12]. These include *bmtuc*, *btuc*, *nondet-ring* and *nondet-ring-1key*, from the former, and *sgripper* from the latter. The first two are non-det variations of the bomb-in-the-toilet problem, the second two are variations of the deterministic ring domain, and the last one is a variation of the classical gripper domain.

The other domains are new. *Mouse-and-cat-n* is about a mouse that must collect one of m cheeses in known locations over a $n \times n$ grid. The initial position of the cat is known, but every time the mouse moves, the cat moves non-deterministically in one of the four possible directions. The mouse can move or grab a cheese only if the cat is not in that position. An instance has a solution if the mouse can get to a cheese, reaching each position before the cat does.

The domains *nd-coins* and *nd-uts* are non-deterministic variations of the *coins* and *uts* domains used in the conformant track of a previous IPC. In nd-coins, the lift non-deterministically closes its doors when the agent steps in or out. The lift can't move if a door is open, and an action is available to shut the doors. In nd-uts, the traveller can forget his passport in the plane after each leg of the trip, and there is an action to recover the passport that is necessary to travel.

Trail-follow-n is a about an agent moving in a grid from $x = 0$, $y = n/2$ to $x = n$, $y = n/2$. There are actions for moving 1 unit along the x-axis with noise over the y coordinate that can be $+1$, -1, or 0. In addition, there is an action 'back-to-trail' that moves the agent 1 unit up or down, or none, according to whether the agent is below, above, or at the $y = n/2$ row (the trail).

Last, *move-pkgs-n-m* is about moving m objects from their initial locations to their final locations over a $n \times n$ grid. The possible actions involve picking-up or putting-down an object, and moving from a location to an adjacent one. The action 'move' has the non-deterministic effect that the object being held may drop at the target location.

The best results in the table are for the KACMBP and $\langle K, K_0 \rangle$ planners. The $\langle K, K_0 \rangle$ planners produce much shorter plans then KACMBP or MBP. The $\langle K, K_0 \rangle$ planner is used by trying first the $\langle K_0, K_0 \rangle$ translation, then $\langle K_1, K_0 \rangle$, and finally, $\langle K_{models}, K_0 \rangle$. A translation is assumed to fail when the classical planner reports an infinite heuristic for the initial state. The $\langle K, K_0 \rangle$ translation with $K = K_0$ produces solutions for *mouse-and-cat*, *sgripper*, *trail-follow*, and *move-pkgs*, and with $K = K_1$, solutions for all the other domains except for *nondet-ring-1key*. For the K-reset planner, $K = K_1$ was used in all cases, and this explains why it reports the *nondet-ring-1key* instances as unsolvable. KACMBP is best on the two *nondet-ring* domains. In the version with the key, because, even leaving the non-deterministic actions aside, the problem has width higher than 1 and the K_1 translation does not render it solvable. Hence, the K_{models} translation ends up being used in the $\langle K, K_0 \rangle$ planner, whose size grows exponentially with the number of rooms. On the other hand, in the version without the key, the difficulties arise in the classical planners: LAMA times out while ordering the landmarks, and FF gets lost in the search. Last, in *mouse-and-cat*, the problem is in our translators, that time out. This problem, however, should be fixable with a better implementation.

In the table, the classical planner used is LAMA, except for *move-pkgs* ($\langle K, K_0 \rangle$), *btuc*, *trail-follow*, *sgripper*, where results with FF have been reported. The classical planners FF and LAMA provide roughly a similar coverage over these domains, with most failures arising not during the search but during preprocessing, as neither planner has been designed to handle the huge grounded PDDL files that result from the translations. For instance, in the K_1-reset translation, where tags are added for each of the hidden h conditions, LAMA times out in the translation into SAS in domains like *nd-coins* and *move-pkgs*, while in the same two domains, FF's parser breaks down. Likewise, in *nondet-ring*, FF runs out of memory in the search, while LAMA times out while processing the landmarks.

7 Summary

We have considered extensions of the translation-based approach to conformant planning for settings where some of the actions have non-deterministic effects, making use of a deterministic relaxation that is correct as long as the non-deterministic actions are executed at most once. We then considered several incomplete translation schemas and planners that use this relaxation, some of which appear to be quite effective and map non-deterministic conformant problems into classical ones. Two of these planners, based on the K-reset and $\langle K, K_0 \rangle$ translations, are compatible with any translation K, and in particular, the $\langle K, K_0 \rangle$ translation applied successively for $K = K_0$ and $K = K_1$ appears to be quite effective. The empirical results of these translations are encouraging, even if the resulting planners do not always perform better than existing ones.

One theoretical issue for the future, involves studying the conditions under which some of these incomplete translations are either strongly or weakly complete. A translation $K(P)$ is strongly complete if it captures all plans for P, and is weakly complete if it captures some plans. In the latter case, the translation is useful too, as it can be used to obtain a plan for P. The K_1 translation for deterministic conformant planning is strongly complete for problems with width bounded by 1, and yet it is often effective (weakly complete) for problems with higher widths. These characterizations are still to be worked out for the incomplete translations proposed.

The problems that cannot be solved by the $\langle K, K_0 \rangle$ and K-reset planners are problems that involve non-trivial disjunctive reasoning

	$\langle K, K_0 \rangle$		K_1-reset		MBP		KACMBP	
	time	#acts	time	#acts	time	#acts	time	#acts
bmtuc-10-10	0.0	20	0.0	20	65.9	29	0.2	20
bmtuc-50-50	1.3	100	1.5	100	$>$ 2h		2722.4	100
bmtuc-100-10	12.0	200	12.7	200	$>$ 2h		25.1	200
bmtuc-100-50	5.6	200	6.1	200	$>$ 2h		$>$ 2h	
bmtuc-100-100	**10.6**	200	**11.9**	200	$>$ 2h		$>$ 2h	
btuc-100	2.7	200	2.8	200	$>$ 2h		2.0	200
btuc-200	20.3	400	21.4	400	$>$ 2h		16.9	400
btuc-300	**70.6**	600	**72.8**	600	$>$ 2h		**62.1**	600
nondet-ring-20	103.4	78	105.5	78	$>$ 2h		7.3	422
nondet-ring-30	430.1	206	440.9	206	$>$ 2h		21.1	349
nondet-ring-40	1698.4	276	1729.4	276	$>$ 2h		67.6	469
nondet-ring-50	SMF, PTL		SMF, PTL		$>$ 2h		**603.1**	2552
nondet-ring-1key-10	12.6	77	unsolvable		11.2	122	4.0	197
nondet-ring-1key-15	101.9	272	unsolvable		$>$ 2h		33.7	375
nondet-ring-1key-20	SM		unsolvable		$>$ 2.1 GB		**246.5**	1104
sgripper-20	0.6	97	7.6	116	$>$ 2h		5.4	148
sgripper-30	2.5	147	34.7	176	$>$ 2h		23.3	228
sgripper-50	**16.0**	247	255.1	296	$>$ 2h		155.6	388
mouse-and-cat-20	5.2	37	1031.7	37	1.8	37	0.2	37
mouse-and-cat-30	23.3	57	KT		38.8	57	0.9	57
mouse-and-cat-40	KT		KT		49.2	77	**2.2**	77
nd-coins-08	0.0	26	0.0	26	882.1	24	2.4	52
nd-coins-10	0.0	21	0.0	21	$>$ 2h		3.8	106
nd-coins-20	**0.1**	88	**0.1**	88	$>$ 2h		$>$ 2h	
nd-uts-04	0.0	23	0.1	27	12.2	40	18.8	42
nd-uts-06	0.1	35	0.4	40	$>$ 2h		$>$ 2h	
nd-uts-07	**0.2**	41	**0.6**	44	$>$ 2h		$>$ 2h	
trail-follow-100	0.8	198	PMF, PTL		0.2	198	0.1	198
trail-follow-150	1.3	298	PMF, PTL		0.4	298	0.1	298
trail-follow-200	1.9	398	PMF, PTL		0.7	398	**0.2**	398
move-pkgs-nd-4-1	0.0	8	34.4	8	0.0	8	0.0	8
move-pkgs-nd-4-3	0.2	28	PMF, PTL		48.3	27	1797.0	37
move-pkgs-nd-5-1	0.2	19	PMF, PTL		0.0	25	0.1	19
move-pkgs-nd-5-3	**0.4**	22	PMF, PTL		$>$ 2h		398.6	26

Table 1. Performance of the non-deterministic conformant planners based on the $\langle K, K_0 \rangle$ and K_1-reset translations, using LAMA and FF in comparison with MBP and KACMBP. The best times for each domain shown in **bold**. In legends, KT means translation time out, PMF means FF preprocessor memory-out, PTL means preprocessor times out in LAMA, SMF means that search memory-out in FF. 'Unsolvable' means that the translation results in classical planning problem with an unreachable goal ($h(s_0) = \infty$). Times in seconds and rounded to the closest decimal. Plan quality expressed as number of actions in plan.

patterns over the hidden conditions h. For example, a problem where a goal $x = n + 1$ is to be achieved starting from $x = 0$ and $y = 0$ with an action that increases x one by one up to $x = n$, and increase y non-deterministically by either 1 or 0. If there are then n actions $enter(i)$ to move from $x = n$ to $x = n + 1$, $i = 1, \ldots, n$, each with condition $y = i$, the plan that increases x n times, followed by the actions $enter(1), \ldots, enter(n)$ solves the problem, but can't be captured by the $\langle K, K_0 \rangle$ and K-reset planners for any K if $n > 2$.

Last, while the results show that the translation-based approach is feasible and competitive in the non-deterministic setting, they also suggest that scalability could be improved by integrating the classical planner and the translators more tightly. Moreover, tags in the translation play two roles: keeping track of the 'conditional beliefs', and producing the heuristic for guiding the search over beliefs. It seems also that scalability could be improved by separating these two roles, and implementing them in different ways.

REFERENCES

[1] A. Albore, H. Palacios, and H. Geffner, 'A translation-based approach to contingent planning', in *Proc. IJCAI-2009*, pp. 1623–1628, (2009).

[2] P. Bertoli and A. Cimatti, 'Improving heuristics for planning as search in belief space', in *Proc. AIPS-2002*, eds., M. Ghallab, J. Hertzberg, and P. Traverso, pp. 143–152. AAAI Press, (2002).

[3] B. Bonet and H. Geffner, 'Planning with incomplete information as heuristic search in belief space', in *Proc. of AIPS-2000*, (2000).

[4] B. Bonet, H. Palacios, and H. Geffner, 'Automatic derivation of memoryless policies and finite-state controllers using classical planners', in *Proceedings of ICAPS-2009*, pp. 34–41, (2009).

[5] D. Bryce, S. Kambhampati, and D. E. Smith, 'Planning graph heuristics for belief space search', *Journal of Artificial Intelligence Research – JAIR*, **26**, 35–99, (2006).

[6] A. Cimatti, M. Roveri, and P. Bertoli, 'Conformant planning via symbolic model checking and heuristic search', *Artificial Intelligence*, **159**, 127–206, (2004).

[7] R. P. Goldman and M. S. Boddy, 'Expressive planning and explicit knowledge', in *Proc. AIPS-1996*, pp. 110–117, (1996).

[8] J. Hoffmann and R. Brafman, 'Contingent planning via heuristic forward search with implicit belief states', in *Proceedings of ICAPS 2005*, pp. 71–80. AAAI, (2005).

[9] J. Hoffmann and R. Brafman, 'Conformant planning via heuristic forward search: A new approach', *Artificial Intelligence*, **170**(6-7), 507–541, (2006).

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

[11] H. Palacios and H. Geffner, 'Compiling uncertainty away: Solving conformant planning problems using a classical planner (sometimes)', in *Proc. AAAI-06*, pp. 900–905, (2006).

[12] H. Palacios and H. Geffner, 'Compiling uncertainty away in conformant planning problems with bounded width', *Journal of Artificial Intelligence Research – JAIR*, **35**, 623–675, (2009).

[13] S. Richter, M. Helmert, and M. Westphal, 'Landmarks revisited', in *Proceedings of AAAI-08*, pp. 975–982, (2008).

[14] D. Smith and D. Weld, 'Conformant graphplan', in *Proceedings AAAI-98*, pp. 889–896. AAAI Press, (1998).