Compiling Uncertainty Away: Solving Conformant Planning Problems Using a Classical Planner (Sometimes)

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

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

Abstract

Even under polynomial restrictions on plan length, conformant planning remains a very hard computational problem as plan verification itself can take exponential time. This heavy price cannot be avoided in general although in many cases conformant plans are verifiable efficiently by means of simple forms of disjunctive inference. This raises the question of whether it is possible to identify and use such forms of inference for developing an efficient but incomplete planner capable of solving non-trivial problems quickly. In this work, we show that this is possible by mapping conformant into classical problems that are then solved by an off-the-shelf classical planner. The formulation is sound as the classical plans obtained are all conformant, but it is incomplete as the inverse relation does not always hold. The translation accommodates 'reasoning by cases' by means of an 'split-protect-and-merge' strategy; namely, atoms L/X_i that represent conditional beliefs 'if X_i then L' are introduced in the classical encoding, that are combined by suitable actions to yield the literal L when the disjunction $X_1 \vee \cdots \vee X_n$ holds and certain invariants in the plan are verified. Empirical results over a wide variety of problems illustrate the power of the approach.

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; Smith & Weld 1998). Conformant planning is computationally harder than classical planning, as even under polynomial restrictions on plan length, plan verification remains hard (Haslum & Jonsson 1999; Baral, Kreinovich, & Trejo 2000; Turner 2002; Rintanten 2004). This additional complexity cannot be avoided in general, although often conformant plan verification can be done efficiently by means of simple forms of disjunctive inference.

For example, simple rules suffice to show that a robot that systematically scans a grid, collecting the objects in each of the cells, will pick up all the objects in the grid, regardless of their original locations. Or similarly, that a robot that moves n times to the right in an empty grid of size n, will necessarily end up in the rightmost column.

This raises the question of whether it is possible to identify and use such forms of inference for developing an efficient but incomplete conformant planner capable of solving non-trivial problems quickly. In this work, we show that this is possible by formulating a suitable translation of conformant problems into classical problems which are then solved by an off-the-shelf classical planner. The translation is sound as the classical plans are all conformant, but it is incomplete as the converse relation does not always hold. The translation scheme accommodates 'reasoning by cases' by means of a 'split-protect-and-merge' strategy; namely, atoms L/X_i that represent conditional beliefs 'if X_i then L' are introduced in the classical encoding that are then combined by suitable actions to yield the literal L when the disjunction $X_1 \vee \cdots \vee X_n$ holds and certain invariants in the plan are verified.

By accounting for this type of simple disjunctive reasoning in a translation scheme that has a clear semantics, we will see that many other patterns of inference fall into place. For example, if in the robot domain above 'push' actions that move objects from one cell to the next are added (for each one of the possible directions), and at the same time, the pick up actions are restricted to particular cells (like corners or centers), then the classical encoding would produce valid conformant plans where enough pushes are done so that all objects are forced into such cells *regardless of their original location*, from which they can be collected. While several effective but incomplete formulations of conformant planning have been formulated before (some of which handle sensing as well; see (Baral & Son 1997; Petrick & Bacchus 2002)), none, as far as we know, can represent these types of plans.

In this paper, we will look first at some proposals that tradeoff expressivity for efficiency, present then the proposed translation scheme, look at some empirical results, and discuss the current limitations.

Taming Complexity

The problem of conformant planning can be formulated as a deterministic search problem in belief space, where a sequence of actions that map a given initial belief state bel_0 into a target set of beliefs is sought. A belief state bel represents the set of states s that are deemed possible, and actions a, whether deterministic or not, deterministically map

one belief state bel into another, denoted as bel_a (Bonet & Geffner 2000). Since the number of belief states is exponential in the number of states, it is clear that the search for conformant plans takes place in a space that is exponentially larger than the search for classical plans. Indeed, while under polynomial length restrictions, classical plan existence is NP-Complete (Bylander 1994), under the same conditions conformant plan existence is harder, at least Σ_2^P (Turner 2002). This is because plan verification is 'easy' in the classical setting but 'hard' in the conformant one, as the verification requires to evaluate the plan for every possible initial state and transition.

A way to trade off completeness for efficiency in conformant planning results from approximating belief states or transitions. For example, the 0-approximation introduced in (Baral & Son 1997) represents belief states bel by means of two sets: the set of literals that are true in bel, and the set of literals that are false in bel. Variables which do not appear in either set are unknown. In this representation, checking whether an action a is applicable in bel, computing the next belief state bel_a, and verifying polynomial length plans are all polynomial time operations. Roughly, a fluent literal L makes it into bel_a iff a) action a has some conditional effect $C \rightarrow L$ such that all literals in C are in bel, or b) L is in bel and for all conditional effects $C' \rightarrow \neg L$ of action a, the complement of some literal $L' \in C'$ is in bel.

Conformant planning under the 0-approximation is thus no more complex, theoretically, than classical planning. The problem however is that the 0-approximation is strongly incomplete, as it does not capture any non-trivial form of disjunctive inference. For example, given a disjunction $p \vee q$ and an action a that maps either p or q into r, the semantics will not validate a as a conformant plan for r. Indeed, disjunctions that are not tautologies are thrown away. The 0-semantics does capture, on the other hand, situations in which the information that is missing is not relevant. For example, if there are actions that can make a variable p true or false, then uncertainty in the initial state of p would not hurt. Classical planners, on the other hand, cannot handle such situations.

Another sound but incomplete approach to planning with incomplete information is advanced in (Petrick & Bacchus 2002) where belief states bel are represented by more complex formulas which may include disjunctions. Yet in order to make belief updates efficient several approximations are introduced, and in particular, while existing disjunctions can be carried from one belief state to the next and can be simplified, no new disjunctions are added. This too imposes a serious limitation in the type of problems that can be handled.

Expressivity, however, is not the only problem; efficiency or control is the other. Indeed, it is not enough to introduce restrictions that under polynomial length constraints bring the complexity of conformant planning to that of classical planning or SAT; the control knowledge needed for solving the resulting problem must be made available as well. The approach in (Petrick & Bacchus 2002) leave this problem largely unaddressed relying on a blind search over compact belief representations and efficient update rules. Recent elaborations of the 0-approximation in (Son, Tu, & M. Gelfond 2005) rely in turn on a fixed heuristic function that counts the number of goals achieved, which applies well to some problems but not to others. In this work, we aim to address both problems, expressivity and control, by accounting for certain forms of disjunctive reasoning in a translation scheme that maps conformant problems into classical problems that can then be handled efficiently by an off-the-shelf classical planner.

Basic Translation

The proposed translation maps a conformant planning problems P into a classical planning problems $K(P)$ and is described in three parts, starting with the basic core $K_0(P)$.

We assume that P is given by a tuple of the form $P = \langle F, O, I, G\rangle$ where F stands for the fluent symbols in the problem, O stands for a set of actions a, I is a set of clauses over F defining the initial situation, and G is a set of literals over F defining the goal. In addition, 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. We assume that actions are all *deterministic* and hence that all uncertainty lies in the initial situation.

We will usually refer to the conditional effects $C \rightarrow L$ of an action a as the *rules* associated with a, and sometimes write them as $a : C \rightarrow L$. Also, we use the expression $C \wedge X \rightarrow L$ to refer to rules with literal X in their bodies. In both cases, C may be empty. Last, when L is a literal, we take $\neg L$ to denote the complement of L.

Definition 1 (Core Translation) *The* core translation *maps the conformant problem P into the classical problem $K_0(P) = \langle F', O', I', G'\rangle$ where*

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

For any literal L in P, KL denotes its 'epistemic' counterpart in $K_0(P)$ whose meaning is that L is known. We write KC for $C = L_1 \wedge L_2 \ldots$ as an abbreviation for $KL_1 \wedge KL_2 \ldots$, and $\neg K\neg C$ for $\neg K\neg L_1 \wedge \neg K\neg L_2 \ldots$.[1] $L \in I$ ($L \notin I$) means that literal L is (not) a unit clause in I.

The intuition behind the translation is simple: first, complementary literals L and $\neg L$ whose status is not known in the initial situation in P are 'negated', by mapping them into the negated K-literals $\neg KL$ and $\neg K\neg L$ that are jointly consistent and can only appear in the body of conditions leading to other negated K-literals only. This mapping removes all uncertainty from $K_0(P)$. In addition, to ensure soundness, each conditional effect $a : C \rightarrow L$ in P maps, not only

[1] Taking $\neg K\neg C$ as an abbreviation for $\neg K\neg L_1 \wedge \neg K\neg L_2$ when $C = L_1 \wedge L_2$ means that we take C to be known as false only when one of the literals in C is known to be false. In modal logics, this is correct but not required; C may be known to be false even when no literal in C is; see (Fagin *et al.* 1995).

into the 'supporting' rule $a : KC \rightarrow KL$ but also into the 'cancellation' rule $a : \neg K\neg C \rightarrow \neg K\neg L$ that guarantees that literal $K\neg L$ is deleted (prevented to persist) when action a is applied except when C is known to be false. The soundness of the translation can be then expressed as:

Theorem 2 (Soundness $K_0(P)$) *If π is a plan that solves the classical planning problem $K_0(P)$, then π is a plan that solves the conformant planning problem P.*

This can be proved through the following lemma that formally captures the meaning of K-literals:

Lemma 3 (Meaning K-literals) *If π is a plan that yields the literal KL in $K_0(P)$, then π is a plan that yields the literal L with certainty in P.*

A plan π here is an applicable action sequence, and π yields a formula if the formula is necessarily true upon completion of the plan. The reader can verify that this translation is in agreement with the 0-approximation semantics (Baral & Son 1997):

Theorem 4 (Equivalence $K_0(P)$ and 0-Approximation) *π solves the classical planning problem $K_0(P)$ iff it solves the conformant planning problem P according to the 0-Approximation.*

This correspondence is not surprising as both formulations throw away the disjunctive information and restrict the valid plans to those that render the missing information irrelevant. Also, the states s_0^k, s_1^k, ... generated by the action sequence $\pi = a_0, a_1, \ldots$ over the classical encoding $K_0(P)$ encode precisely the literals that are known according to the 0-approximation; namely, L is known at time t according to the 0-approximation iff the literal KL is true in s_t^k.

As an illustration, given a conformant problem P with $I = \{p, r\}$ (i.e., nothing else is known; there is no CWA), and actions a and b with effects $a : p \rightarrow q$, $a : r \rightarrow \neg s$, and $b : q \rightarrow s$, the plan $\{a, b\}$ is valid for achieving q and s according to both $K_0(P)$ and the 0-approximation, while the singleton sequence $\{a\}$ is not valid according to either. At the same time, if the initial situation is changed to $I = \{p \vee q\}$, neither approach would sanction that plan $\{a\}$ for q, even if it is a valid conformant plan. For this, some ability to reason with disjunctions is needed.

Case Analysis over Single Actions

We will make the formulation stronger by accounting for certain disjunctive inferences in the translation. This will result into more actions and conditional effects added to $K(P)$ which is initially set to $K_0(P)$.

Consider an action a that in a given context C' can force a literal L to make the transition from false to true, while preventing the opposite transition. In such a context C', even if L is unknown, a can be used to make L true. This type of inference is captured in the translation as follows:

Rule 2 (Action Compilation) *If P contains a rule $a : C \wedge \neg L \rightarrow L$, and the rules for the same action a with $\neg L$ in the head are $C_i \rightarrow \neg L$, $i = 1, \ldots, n$ for $n \geq 0$, then add to $K(P)$ the rules $KC \wedge K\neg L_1 \wedge \cdots \wedge K\neg L_n \rightarrow KL$ where L_i is a literal in C_i.*

This is a modular translation rule in which the context C' above is the formula $C \wedge \neg L_1 \wedge \cdots \wedge \neg L_n$, for any combination of literals L_i chosen as to preempt the rules $C_i \rightarrow \neg L$ associated with the same action a that can clobber L. All the literals in C' are preceded by K's as they refer to literals in $K(P)$ that ensure that the condition holds with certainty. This translation remains polynomial as long as the number of rules $a : C_i \rightarrow L$ associated with the same action a and the number of literals in the conditions C_i remain bounded, which is normally the case (in the existing benchmarks both numbers are pretty small, one or two at most).

It is not difficult to show that this translation rule preserves soundness. A key characteristic of the rule and others to be introduced below is that they make use of the conditional effects $a : C \wedge X \rightarrow L$ in the problem P for deriving L with certainty when the body of the rule $C \wedge X$ is not fully known.

In an example like 'empty room', where a robot moves in an empty square grid and literals X_i are used to represent the column location of the robot, this translation ensures that literal $K\neg X_1$ is obtained right after a single 'move right' action (namely, that the robot cannot be in the leftmost column then), and similarly, that $K\neg X_2$ is obtained after two consecutive right moves, etc. If the grid is nxn, the resulting classical theory yields $K\neg X_i$ for all $i < n$ after $n-1$ steps, although it does not yield KX_n (being in the rightmost column). For this, the disjunction expressing the possible column positions, namely $X_1 \vee X_2 \vee \cdots \vee X_n$, needs to be taken into account as well. We address this next.

Case Analysis over Action Sequences

We extend the translation further so that the disjunctions in P are taken into account in a form that is similar to the Disjunction Elimination inference rule used in Logic (Barwise & Etchemendy 1991):

$$\text{If } X_1 \vee \cdots \vee X_n,\ X_1 \supset L, \ldots, \text{and } X_n \supset L \text{ then } L \quad (1)$$

For this, we create new atoms in $K(P)$, written L/X_i, that aim to capture the conditional beliefs $X_i \supset L$. Then, the resulting classical encoding will be such that when these atoms are 'achieved' for each $i = 1, \ldots, n$, and they are suitably 'protected', the literal L will be rendered 'achievable' by means of an extra 'dummy' action with conditional effect similar to (1).

As already mentioned, the atoms L/X_i will stand for the conditional belief 'if X_i then L'. In principle, any rule $a : C \wedge X_i \rightarrow L$ in P with X_i uncertain can be used to produce a rule $a : KC \rightarrow L/X_i$ in $K(P)$, meaning that if KC is known and a is applied, then if X_i was true, L will become true. However, we want L/X_i to mean exactly that '*right after the action* a, if X_i is true, then L is true', and for this, some additional care is needed. Indeed, if a contains also rules $a : C_k \rightarrow X_i$ that can make X_i true, it may be possible that L and X_i are false at time t when a is applied, and that L remains false but X_i becomes true, and then that 'if X_i at t, then L at $t+1$' is true, but 'if X_i at $t+1$, then L at $t+1$' is false. In order to rule out this situation we define the corresponding translation rule as follows:

Rule 3 (Split) *For each rule $a : C \wedge X_i \rightarrow L$ in P where X_i is a literal that appears in a disjunction $X : X_1 \vee X_2 \vee \cdots \vee X_n$, if $a : C_k \rightarrow X_i$, $k = 1, \ldots, m$ for $m \geq 0$ are the rules in P for the same action a with X_i in the head, then add to $K(P)$ the atoms L/X_j, $j = 1, \ldots, n$, all initialized to **false**, and the rules $a : KC \wedge K\neg L_1 \wedge \cdots \wedge K\neg L_m \rightarrow L/X_i$ where L_k is a literal in C_k.*

The combination of the conditional beliefs represented by the atoms L/X_i is achieved by means of extra actions added to the classical encoding $K(P)$ that generalize (1) slightly, allowing some of the cases X_i to be disproved:[2]

Rule 4 (Merge) *For each disjunction $X : X_1 \vee \cdots X_n$ and atom L in P such that L/X_i is an atom in $K(P)$, add to $K(P)$ a new action $a_{X,L}$ with conditional effect*

$$(L/X_1 \vee K\neg X_1) \wedge \cdots \wedge (L/X_n \vee K\neg X_n) \wedge FLAG_{X,L} \rightarrow KL$$

*where $FLAG_{X,L}$ is a boolean initialized to **true**. If $L = X_i$ for some $i \in [1, n]$, remove the conjunct $(L/X_i \vee K\neg X_i)$ from the rule body.*

A key distinction from Logic is that the disjunction $X_1 \vee \cdots \vee X_n$ and the conditional beliefs 'if X_i then L' represented by the atoms L/X_i need all be **preserved** until they are combined together to yield L. This is the purpose of the boolean $FLAG_{X,L}$ that is initially set to true, but which is deleted when an action is done in a context where it is not possible to prove that 1) L is preserved (if true), 2) the disjunction $X \vee L$ is preserved (the disjunction X is initially true but it is actually sufficient to preserve the weaker disjunction $X \vee L$), and 3) the conditional beliefs represented by the atoms L/X_i achieved are preserved. This is accomplished by extending $K(P)$ with the following cancellation rules:

Rule 5 (Protect) *If there is a boolean flag $FLAG_{X,L}$ in $K(P)$ for $X : X_1 \vee \cdots \vee X_n$, then for each action a: 1) if $a : C \rightarrow \neg L$ in P, add to $K(P)$ the rule $a : \neg K\neg C \rightarrow \neg FLAG_{X,L}$, 2) if $a : C \rightarrow \neg X_i$ in P and neither $a : C \rightarrow X_k$ nor $a : C \rightarrow L$ in P for X_i and X_k in X, add to $K(P)$ the rule $a : \neg K\neg C \rightarrow \neg FLAG_{X,L}$, and 3) if $a : C \rightarrow X_k$ for X_k in X, then add to $K(P)$ the rule $a : \neg K\neg C \wedge L/X_k \rightarrow \neg FLAG_{X,L}$.*

These rules, as we will see, yield expressivity without sacrificing efficiency, as they manage to *accommodate nontrivial forms of disjunctive inference in a classical theory without having to carry disjunctive information explicitly in the belief state:* disjunctive information is represented implicitly in $K(P)$ in terms of the the conditional atoms L/X_i, the 'merge' actions, and the *invariants* enforced by the 'flags'.

Theorem 5 (Soundness $K(P)$) *Any plan that achieves the literal KL in $K(P)$ is a plan that achieves L in the conformant problem P.*[3]

[2]When using the classical plans obtained from $K(P)$ as conformant plans for P, such 'dummy' actions must be removed.

[3]For this result to hold, we assume that for every pair of conflicting rules $a : C \rightarrow \alpha$ and $a : C' \rightarrow \neg\alpha$ associated with the same action a in P, the bodies C and C' are such that they contain a mutex pair L, L'. This mutex relation is enforced on the corresponding K-literals by adding to every effect $C'' \rightarrow KL$ associated with the 'dummy' merge action $a_{X,L}$ in $K(P)$, the effects $C'' \rightarrow K\neg L'$ and $C'' \rightarrow \neg KL'$.

The key element in the proof is the following lemma that captures the meaning of the L/X_i atoms:

Lemma 6 (L/X_i Atoms) *Any plan that yields L/X_i while preserving $FLAG_{X,L}$ in $K(P)$ is a plan that achieves the conditional $X_i \supset L$ in P.*

A proof sketch goes as follows. Let us assume that L/X_i, which is initially false, is made true at time t by an action a in the plan. We need to prove that if $FLAG_{X,L}$ remains true in $K(P)$ until time $t' \geq t$, then the conditional $X_i \supset L$ remains true until t' in P, which we write as $X_i(t') \supset L(t')$. From the argument above, if L/X_i became true in $K(P)$ at time t, so does the conditional $X_i(t) \supset L(t)$ in P. From this, $X_i(t') \supset L(t')$ follows if we can show both $X_i(t') \supset X_i(t)$ and $L(t) \supset L(t')$. The latter is true because the rules in $K(P)$ ensure that if a rule $a' : C' \rightarrow \neg L$ gets triggered by the plan in P, the rule $a' : \neg K\neg C' \rightarrow \neg FLAG_{X,L}$ will be triggered by the plan in $K(P)$. Similarly, the former is true because the rules in $K(P)$ ensure that if a rule $a' : C' \rightarrow X_i$ is triggered by the plan in P when L/X_i is true in $K(P)$, then the rule $a' : \neg K\neg C' \wedge L/X_i \rightarrow FLAG_{X,L}$ will be triggered in $K(P)$. In either case, $FLAG_{X,L}$ would be deleted, so if it is not, $X_i(t') \supset X_i(t)$ and $L(t) \supset L(t')$ must hold, and since $X_i(t) \supset L(t)$ holds, so must $X_i(t') \supset L(t')$.

As an illustration, given an object O to be collected from an unknown location in a grid with two cells A and B using the actions $pick(X)$, $push(X, Y)$, and $go(X, Y)$, where X and Y are cells and the three actions have as a precondition that the agent is at X, it follows that if the agent is initially at A, the plan $\pi_1 = \{pick(A), go(A, B), pick(B)\}$ achieves $hold(O)$ in $K(P)$ and so does $\pi_2 = \{push(A, B), go(A, B), pick(B)\}$, but $\pi_3 = \{pick(A), go(A, B), push(B, A)\}$ does not. If $at(O, A) \vee at(O, B)$ is the disjunction X in P and L is $hold(O)$, then π_1 achieves $hold(O)/at(O, A)$ and $hold(O)/at(O, B)$, π_2 achieves $K\neg at(O, A)$ and $hold(B)/at(O, B)$, while π_3 achieves both $hold(O)/at(O, A)$ and $K\neg at(O, B)$ but clobbers $FLAG_{X,L}$, preempting the merge action $a_{X,L}$ from achieving $hold(O)$. This happens because the rule $push(B, A) : at(O, B) \rightarrow at(O, A)$ in P yields the rule $push(B, A) : \neg K\neg at(O, B) \wedge L/at(O, A) \rightarrow \neg FLAG_{X,L}$ which gets triggered in $K(P)$ by the action sequence π_3.

Experimental Results

We have implemented the translation scheme into a program `cf2cs` that takes a conformant planning problem P as input an outputs a classical problem $K(P)$. In the experiments below, this problem is fed into the FF v2.3 classical planner (Hoffmann & Nebel 2001). We refer to the resulting conformant planner as `cf2cs(ff)`. The experiments were tested on a Intel/Linux machine running at 2.80GHz with 2Gb.

Problem	P			Translation time	$K(P)$		
	#Actions	#Atoms	#Effects		#Actions	#Atoms	#Effects
Bomb-100-60	6060	320	24120	1.35	6260	1041	79560
Cube-11-Ctr	6	33	120	0.036	72	226	1152
Cube-75-Ctr	6	225	888	1.08	456	1789	8448
Sqr-64-Ctr	4	128	504	0.31	260	893	4796
Sqr-240-Ctr	4	480	1912	6.11	964	3833	18172
Grid-4-5	174	155	444	5.65	183	351	1244
Safe-100	100	101	100	0.11	101	304	804
Logistics-4-10-10	3320	610	6640	3.52	3370	1321	13880

Table 1: Data concerning the translation of some conformant problems P into classical encodings $K(P)$. The sizes refer to the grounded versions, and all times are in seconds and they include grounding time.

We report results on two classes of instances: existing benchmarks and some domains of our own. In both cases, we compare the results (times and plan lengths) with those obtained by running Conformant FF, an state-of-the-art conformant planner (Brafman & Hoffmann 2004). We could have used other recent classical and conformant planners such as (Cimatti, Roveri, & Bertoli 2004) and (Bryce & Kambhampati 2004), but as a reference, this should do. We want to show that our approach solves a wide variety of non-trivial problems without any 'help' in the encoding or control, scaling up like a classical planner. For the existing benchmarks we use the actual encodings from the Conformant FF repository, the other encodings will be made available from us. None of these encoding can be solved by either the basic $K_0(P)$ translation or the 0-approximation.

Table 1 shows data concerning the translation of a number of problems from various sources, used and explained in (Brafman & Hoffmann 2004). Bomb-x-y refers to the Bomb-in-the-toilet problem with x packages, y toilets, and clogging. Cube-n-Ctr refers to the problem of reaching the center of a cube of size n^3 from a completely unknown location. Square-n-Ctr is similar but involves only n^2 possible locations. Logistics-i-j-k, Grid-n and Safe-n are from (Brafman & Hoffmann 2004).

Table 2 shows the plan times and lengths obtained by `cf2cs(ff)` vs. Conformant FF over various benchmarks, where it can be seen that `cf2cs(ff)` scales up much better, solving problems like Sqr-240-Ctr and Cube-75-Ctr that are well beyond the reach of current complete or incomplete conformant planners with the exception of (Cimatti, Roveri, & Bertoli 2004).

Among the existing benchmarks, not included in the table, there are three domains, Sorting-Nets, (Incomplete) Blocks, and Ring, which in their standard encodings, cannot be handled by the proposed translation scheme; namely, in none of these encodings the planner finds a classical plan in $K(P)$ even though P has conformant solutions. We will say more about the incompleteness of the translation below.

Finally, Table 3 shows plan times and lengths for a family of grid problems that we devised: Retrieve is about picking up objects whose locations are unknown; Dispose is about retrieving such objects and placing them in a trash can at a given, known location; Push is a variation of Retrieve when there is also a push action that can move objects; and Push-To is a further variation where the pick-up actions

Problem	`cf2cs(ff)`		CFF	
	Time	Length	Time	Length
Bomb-50-50	2.13	50	0.2	50
Bomb-100-1	0.84	199	96.2	199
Bomb-100-60	9.64	140	23.53	140
Cube-7-Ctr	0.02	24	38.2	39
Cube-9-Ctr	0.05	33	—-	—-
Cube-11-Ctr	0.09	42	—-	—-
Cube-75-Ctr	484.0	330	—-	—-
Sqr-8-Ctr	0.03	22	140.5	50
Sqr-12-Ctr	0.04	32	—-	—-
Sqr-64-Ctr	9.66	188	—-	—-
Sqr-120-Ctr	59.4	356	—-	—-
Sqr-240-Ctr	858.0	716	—-	—-
Grid-4-4	0.06	25	0.11	25
Grid-4-5	0.05	30	0.14	30
Safe-30	0.01	30	6.6	30
Safe-70	0.08	70	561.8	70
Safe-100	0.28	100	—-	—-
Logistics-3-10-10	4.42	109	11.15	108
Logistics-4-10-10	5.91	125	11.74	121

Table 2: Plan times and lengths obtained by a classical planner (FF) over $K(P)$ translation (`cf2cs(ff)`) in relation to Conformant FF. Times in seconds. The symbol '—-' means cutoff exceeded (30 mins or 800Mb)

are applicable only at certain locations, and therefore, objects need to be pushed into those locations first. Problem P-n-m stands for problem P over grid of size n and m objects. Once again the Table shows a different scaling behavior between `cf2cs(ff)` and Conformant FF. The exception is the Push-To domain where the resulting classical encoding $K(P)$ has many dead-ends that are not detected by the heuristic used in FF. This an interesting problem that could be solved in principle by refining the FF heuristic, the translation, or both. We plan to look further into this.

Discussion

We have introduced a translation scheme that enables a wide class of conformant problems to be solved by an off-the-shelf classical planner. The translation accounts for a limited form of 'reasoning by cases' by means of an 'split-protect-and-merge' strategy; namely, atoms L/X_i that represent conditional beliefs 'if X_i then L' are introduced for

Problem	cf2cs(ff) Time	cf2cs(ff) Length	CFF Time	CFF Length
Retrieve-4-1	0.02	41	0.04	33
Retrieve-4-2	0.08	75	0.23	49
Retrieve-4-3	0.17	91	0.8	65
Retrieve-8-1	1.18	220	204.68	210
Retrieve-8-2	3.13	291	—-	—-
Retrieve-8-3	132.49	415	—-	—-
Retrieve-12-1	—-	—-	—-	—-
Dispose-4-1	0.02	37	0.12	39
Dispose-4-2	0.05	54	0.47	56
Dispose-4-3	0.09	71	1.49	73
Dispose-8-1	1.83	265	361	227
Dispose-8-2	2.87	280	—-	—-
Dispose-8-3	6.87	367	—-	—-
Dispose-12-1	—-	—-	—-	—-
Push-4-1	0.07	41	0.09	33
Push-4-2	0.24	75	0.41	49
Push-4-3	0.53	91	1.23	65
Push-8-1	3.29	220	—-	—-
Push-8-2	12.89	291	—-	—-
Push-8-3	—-	—-	—-	—-
Push-to-3-1	0.32	21	0.03	29
Push-to-4-1	—-	—-	0.48	46

Table 3: Plan times and lengths obtained by a classical planner (FF) over $K(P)$ translation (`cf2cs(ff)`) in relation to Conformant FF. Times in seconds. The symbol '—-' means cutoff exceeded (30 mins or 800Mb).

disjunctions $X_1 \vee \cdots \vee X_n$ in the problem, and when certain invariants are verified in the plan, they are combined to yield the literal L. Empirical results over a variety of problems illustrate the power of this approach.

The simplicity of the translation and the semantics captured by the theorems not only enable us to prove the soundness of the approach, but as importantly, to delimit its scope. In relation to natural deduction systems in the style of Ficht (Barwise & Etchemendy 1991), the type of disjunctive reasoning accounted for in the translation is limited in two ways. First, while disjunctions $X_1 \vee \cdots \vee X_n$ in P are used to create sub-derivations by making assumptions of the form X_i, these sub-derivations are not nested, and therefore, disjunctions are not combined. This implies, for example, that four action rules like $a_{i,j} : X_i \wedge Y_j \rightarrow L$ for $i = 1, 2$ and $j = 1, 2$ cannot be used to produce a plan for L given the disjunctions $X_1 \vee X_2$ and $Y_1 \vee Y_2$ in the initial situation. Second, the sub-derivations that arise when making the assumptions X_i are very limited, and in particular the atoms L/X_i can only be used for proving L but no other literal. Thus, as a result, four action rules like $a_1 : X_1 \rightarrow L_1$, $a_2 : X_2 \rightarrow L_2$, $b_1 : L_1 \rightarrow L$, and $b_2 : L_2 \rightarrow L$ cannot be used to generate a plan that achieves L given the single disjunction $X_1 \vee X_2$. These are the two sources of incompleteness in the translation that is aimed at capturing conformant plans that can be verified easily, by reasoning with 'one disjunction' at a time. Verification that involves reasoning that combines all disjunctions is intractable, yet verifications that use a bounded number of disjunctions N at a time are tractable and could be accommodated in a polynomial translation scheme as the one proposed (which may not be effective though for large N). We plan to work on such extensions in the future. Also pending is the treatment of actions with non-deterministic effects.

Acknowledgements

We thank J. Hoffmann and M. Helmert for help with their planners, B. Bonet and R. Givan for their IPC5 conformant plan verifier, and A. Frangi and A. Sanz for the use of the Hermes Computing Resource at U. of Zaragoza. H. Geffner is partially supported by grant TIN2005-09312-C03-03 from MEC/Spain.

References

Baral, C., and Son, T. C. 1997. Approximate reasoning about actions in presence of sensing and incomplete information. In *Proc. ILPS 1997*, 387–401.

Baral, C.; Kreinovich, V.; and Trejo, R. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence* 122(1-2):241–267.

Barwise, J., and Etchemendy, J. 1991. *The Language of First-Order Logic*. CSLI, Stanford.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS-2000*, 52–61. AAAI Press.

Brafman, R., and Hoffmann, J. 2004. Conformant planning via heuristic forward search: A new approach. In *Proc. ICAPS-04*.

Bryce, D., and Kambhampati, S. 2004. Heuristic guidance measures for conformant planning. In *Proc. ICAPS-04*, 365–375.

Bylander, T. 1994. The computational complexity of STRIPS planning. *Artificial Intelligence* 69:165–204.

Cimatti, A.; Roveri, M.; and Bertoli, P. 2004. Conformant planning via symbolic model checking and heuristic search. *Artificial Intelligence* 159:127–206.

Fagin, R.; Halpern, J.; Moses, Y.; and Vardi, M. 1995. *Reasoning about Knowledge*. MIT Press.

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

Haslum, P., and Jonsson, P. 1999. Some results on the complexity of planning with incomplete information. In *Proc. ECP-99, Lect. Notes in AI Vol 1809*, 308–318. Springer.

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

Petrick, R., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. AIPS'02*, 212–221.

Rintanten, J. 2004. Complexity of planning with partial observability. In *Proc. ICAPS-2004*, 345–354.

Smith, D., and Weld, D. 1998. Conformant graphplan. In *Proceedings AAAI-98*, 889–896.

Son, T. C.; Tu, P. H.; and M. Gelfond, A. M. 2005. Conformant planning for domains with constraints-a new approach. In *Proc. AAAI-05*, 1211–1216.

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.