Planning in the presence of incomplete information [Goldman and Boddy, 1996; Smith and Weld, 1998] is the problem of generating a plan that can achieve a given goal regardless of what the actual truth value of unknown information about the initial world is. This paper focuses on conformant planning—a special class of planning problems in the presence of incomplete information—that deals with incomplete information about the initial state and with deterministic and non-sensing actions.

One of the challenges in conformant planning is the problem of reasoning about action effects in the presence of incomplete information. Specifically, consider an action \( a \) with a set of conditional effects of the form \( (a : \psi \rightarrow \eta) \), where \( \psi \) and \( \eta \) are a set of literals. Each effect \( (a : \psi \rightarrow \eta) \) denotes that the set of literals \( \eta \) will be true in the resulting state after the execution of \( a \) in the current state if the e-condition \( \psi \) is true in the current state. Due to incomplete information, the current state and, hence, the truth value of \( \psi \) in it can be unknown. As a consequence, the fact that \( \eta \) must be true in the successor state can be unknown, making the computation of the successor state particularly challenging. This agrees with the results in [Baral et al., 2000; Haslum and Jonsson, 1999], that conformant planning in domains with conditional effects is at a higher complexity level than that in domains without conditional effects. Since its introduction, conformant planning has attracted the attention of several researchers—leading to the development of several state-of-the-art conformant planners. It is important to observe that most efficient conformant planners are best-first search and progression-based planners, whose development starts with the selection of a representation language for incomplete (partially known) states and the definition of a transition (progression) function that, given a state \( S \) and an action, computes the next state \( S' \) of the world, where \( S \) and \( S' \) are generally incomplete and encoded in the representation.

To deal with incomplete information about the world, the notion of a belief state—defined as the set of possible states—has been introduced and is widely used in planning in presence of incomplete information [Bonet and Geffner, 2000; Smith and Weld, 1998]. An advantage of this notion lies in its simplicity in representing and reasoning about the effects of actions in this planning problem. Indeed, any formalism for Reasoning about Action and Change (RAC), that assumes completeness of the state, can be straightforwardly generalized to deal with incomplete states, by dealing with each individual possible state in the belief state separately, and by collecting the set of resulting states as the successor belief state. More precisely, let \( S \) be a state and \( a \) be an action executable in \( S \) (\( S \) satisfies the precondition of \( a \)). The effect of executing \( a \) in \( S \) is the set of literals that become true in the successor state, defined as

\[
e(a, S) = \{ \ell | \exists (a : \psi \rightarrow \eta). \psi \subseteq S \land \ell \in \eta \}
\]  

The transition function \( \Phi(a, S) \) that returns the result of the execution of \( a \) in \( S \) is defined by

\[
\Phi(a, S) = S \setminus e(a, S) \cup e(a, S)
\]  

where \( e(a, S) \) denotes the set of negations of literals in \( e(a, S) \). Then the function \( \Phi(a, S) = \{ \Phi(a, S) | S \in S \} \) characterizes the transition between belief states and can be used for reasoning with incomplete information.

The downside of the direct use of the belief state lies in its size, which is exponential in the number of unknown fluents. This presents two challenges for planners, that employ this representation. First, it can quickly increase the memory usage of the planner, that needs to maintain the set of generated belief states to avoid repeated search. This can often lead to the undesirable ‘Out of memory’ situation before the planner finds a solution. Second, it directly influences the time complexity in computing the successor belief states.

To address the problem of exponential size of belief states, different approaches have been proposed in state-of-the-art planners, resulting in significant improvement in their performance, both planning time and scalability on the problem size.

The approach adopted in the planner KACMBP
represents belief states as OBDDs [Bryant, 1992] and uses the model checking techniques for expanding the search space. The OBDD formula was also used in the planner POND [Bryce et al., 2006] to represent literals and actions in the planning graph for computing heuristics. The advantage of this BDD-based approach is that it does not require any extra BDD manipulation operations and it can be applied in both directions: progression and regression. However, the transition function defined for OBDDs may require a huge memory consumption in the manipulation of the OBDDs. This is a potential reason why planners employing this formulation, like POND or KACMBP, do not scale well, as confirmed in the experimental evaluations [To, Son, and Pontelli, 2015].

Hoffmann and Brafman (2006) proposed an opposite extreme to the direct use of belief states by implicitly encoding belief states with the sequence of actions that leads to the belief state from the initial state. To determine whether an unknown proposition holds in a belief state, the resulting conformant planner CFF has to reason about a CNF formula that captures the semantic of the initial belief state and the entire action sequence from it to the current belief state. This representation requires very little memory and, hence, allows CFF to scale up well on a number of problems. The trade-off is that it incurs excessive repeated computations and checking whether two belief states are equivalent under this representation is challenging and expensive. Furthermore, checking whether a proposition holds after the execution of even one single action in the presence of incomplete information is co-NP complete. We believe those are the main reasons why CFF has difficulties in finding a solution for even small instances of harder problems, where the structure of the actions is complex or there are unknown propositions in the e-conditions [Palacios and Geffner, 2009; Tran et al., 2009; To, Pontelli, and Son, 2009].

The approach employed in the planners T0 [Palacios and Geffner, 2007; Palacios and Geffner, 2009] and T1 [Albore, Ramirez, and Geffner, 2011] compiles incomplete information away by translating a conformant problem into a classical problem and then using the efficient classical planner FF [Hoffmann and Nebel, 2001] to solve the resulting problem. This approach demonstrates great performance improvements, e.g., T0 and T1 can solve many hard problems of large size. However, the complete translation is exponential in the conformant width of the problem, which is related to the number of relevant unknown literals, and the number of literals in the resulting problem can be exponential in that of the original problem, making the state space extremely large and preventing the planners to scale up. To reduce the complexity of the translation, T0 sacrifices the completeness of the translation in certain problems. This is the reason why T0 is unable to return a solution in several problems that have solutions or the translation fails for large instances of several domains, as shown in our experiments.

GC[LAMA] [Nguyen et al., 2012] is another conformant planner that uses a classical planner. This approach is based on the observation that every solution for a conformant problem is also a solution for the classical problem obtained by replacing the initial belief state with a state in it. A tentative plan α for such a subproblem then can be corrected, by inserting an action sequence to produce precondition for the action the plan α violates at that place. If α cannot be fixed to become a plan for the conformant problem. GC[LAMA] continues with another plan of the same or different subproblem. GC[LAMA] is efficient on problems where the action effects are monotonic, i.e., useful actions create useful literals without destroying other useful literals, and the belief state is not large. Otherwise, it is not as efficient, as reported in the experimental evaluation. This approach is complete only if the underlying classical planner is complete.

The introduction of CPA [Son et al., 2005] brought a different perspective to deal with incomplete information in conformant planning. Instead of using the complete transition function in the search, CPA uses an approximation technique, first proposed in [Son and Baral, 2001], which approximates a belief state by the intersection of the states in it. The advantage of this approach lies in the low-complexity of the approximation: the successor (approximated) belief state can be computed in polynomial time. The approximation is, however, incomplete, and so are planners which employ this approximation. To address this issue, a complete condition for the approximation has been identified along with corresponding techniques developed in CPA to make the planner complete [Son and Tu, 2006; Tu et al., 2011]. These techniques require the system to deal with sets of approximated states, which—in the worst case—are the same as the actual belief states being represented. The advantage of this approach is that the computation of successor belief states is very simple—i.e., can be performed in the same manner as that for belief states—since the approximated formulae are in disjunctive normal form and they contain sufficient information needed for the state computation. The drawback of this approach is that the approximated formula explodes in many problems, as observed in the experiments, preventing the planner to effectively scale. To address this issue, the authors of CPA [Tran et al., 2009] developed preprocessing techniques which help reduce the size of the initial disjunctive formula in a certain classes of problems. These techniques enabled the planner to perform very well in several benchmarks and to win the conformant planning category in the IPC-08.

In this paper, we propose a generic approach to planning with incomplete information. The key idea underly-
ing our approach is the observation that belief states are propositional formulae. Thus, reasoning about effects of actions on belief states can be expressed using suitable operations on formulae, which is usually (possibly exponentially) more compact than the belief states they represent.

As an example, let us consider a domain \( F = \{ f, g, h \} \), an action \( a \) with three conditional effects \( C_a = \{ f \rightarrow \neg g, f \land \neg g \rightarrow \neg f, \neg f \rightarrow f \} \), and a belief state \( S \) that contains all possible states of the world (i.e., all possible interpretations of propositions in \( F \)) except for \( \{ \neg f, \neg g, \neg h \} \). Figure 1 shows the belief state \( S = \{ s_1, \ldots, s_7 \} \) and the computation of the successor belief state \( S' \) after executing \( a \) in \( S \) using the function \( \Phi \) (Equation 2).

Let \( \varphi \) be a formula and \( S \) be a belief state. We say that \( \varphi \) represents \( S \) or is the belief state of \( \varphi \), denoted by \( BS(\varphi) \), if \( S \) is the set of states that satisfy \( \varphi \). In the above example, we observe that \( S \) and \( S' \) can be represented by the more compact CNF-formulae \( \varphi = f \lor g \lor h \) and \( \varphi' = f \lor \neg g \), respectively. Since each formula encodes a unique belief state, we want to use their compact encoding \( \varphi \) and \( \varphi' \) instead of the belief states \( S \) and \( S' \) of exponential size. To this end, we need to construct a transition function (e.g., \( \Phi_{\mu,CNF} \)) such that (1) it returns a CNF-formula representing the successor belief state given a CNF-formula and an action (e.g., \( \Phi_{\mu,CNF}(a, \varphi) = \varphi' \)) and (2) its computation involves operations over a small set of formulae instead of enumerating each individual in belief states.

To compute the result \( s' \) of executing an action \( a \) in a state \( s \), we need to compute the effect \( e(a, s) \) of executing \( a \) in \( s \), the set of literals that must be true in the successor state \( s' \). Observe from Figure 1 that, the effects of executing \( a \) in \( s_1, s_3 \), and \( s_5 \) are all different. However, \( e(a, s_1) = e(a, s_2) = \{ \neg g \} \), \( e(a, s_3) = e(a, s_4) = \{ \neg f, \neg g \} \), and \( e(a, s_5) = e(a, s_6) = e(a, s_7) = \{ f \} \). Hence, if we divide \( BS(\varphi) \) into three sets of states \( S_1 = \{ s_1, s_2 \}, S_2 = \{ s_3, s_4 \} \), and \( S_3 = \{ s_5, s_6, s_7 \} \) (Figure 2) then for each \( i \in \{ 1, 2, 3 \} \) and \( u, v \in S_i \), we have that \( e(a, u) = e(a, v) \). This is because the truth value of each \( e \)-condition of \( a \) in all the states in each subset \( S_i \) is the same and it can differ in states from different subsets (e.g., the \( e \)-condition \( f \) in \( a : \{ f \} \rightarrow \{ \neg g \} \) is true in all states \( s_1 \) and \( s_2 \) in \( S_1 \) and it is false in all states \( s_5, s_6 \), and \( s_7 \) in \( S_3 \)). Observe also that \( S_1, S_2, \) and \( S_3 \) can be represented by the formulae \( \varphi_1 = f \land g, \varphi_2 = f \land \neg g, \) and \( \varphi_3 = \neg f \land (g \lor h) \) respectively. Since the effects of \( a \) in all states in \( BS(\varphi_i) \), for each \( \varphi_i \), are the same, we define \( e(a, \varphi_i) = e(a, s) \) for \( s \in BS(\varphi_i) \) and \( i \in \{ 1, 2, 3 \} \). Let \( s'_j = \Phi(a, s_j) \) (for \( j = 1, \ldots, 7 \)) and \( S'_i = \Phi(a, S_i) \). Let \( \varphi'_i \) be the formula obtained by updating \( \varphi_i \) with the minimum change such that the literals in \( e(a, \varphi_i) \) become true in the new formula \( \varphi'_i \). Intuitively, \( BS(\varphi'_i) = S'_i \) and, hence, \( BS(\varphi'_1) \cup BS(\varphi'_2) \cup BS(\varphi'_3) = BS(\varphi) (= S') \). Indeed, it is easy to see that \( \varphi'_1 = f \land \neg g, \varphi'_2 = \neg f \land \neg g, \) \( \varphi'_3 = f \land (g \lor h) \) and the disjunction \( \varphi'_1 \lor \varphi'_2 \lor \varphi'_3 \) can be simplified to the formula \( \varphi' \). On the other hand, \( S = S_1 \cup S_2 \cup S_3 \) and \( \varphi \equiv \varphi_1 \lor \varphi_2 \lor \varphi_3 \). In summary, the computation of \( \varphi' \) includes three major steps as described in Figure 2.

As shown in our analysis and experiments [To, Son, and Pontelli, 2010a; To, Son, and Pontelli, 2010b; To, Son, and Pontelli, 2011a; To, Son, and Pontelli, 2013], the size of formulae representing belief states and, hence, the performance of a corresponding planner depend substantially on both the type of the formulae being used.
and the specific problem. For this reason, we propose a generic approach based on the abstract notion representation, instead of limiting to some specific types of formulae, aimed at the use of any type of propositional formulae to represent belief states.

Specifically, a collection \( R \) of formulae is a representation of belief states (or representation, for short) if for every belief state \( S \) there exists a formula \( \varphi \in R \) that represents \( S \) (\( BS(\varphi) = S \)). A formula \( \varphi \) in \( R \) is called an \( R \)-state. A function \( \Phi_{R} \) that maps pairs composed of an action \( a \) and \( R \)-state \( \varphi \) into \( R \)-states is a transition function for \( R \) if \( \Phi_{R}(a, \varphi) \) represents the belief state \( \Phi(a, BS(\varphi)), i.e., BS(\Phi_{R}(a, \varphi)) = \Phi(a, BS(\varphi)) \).

As shown in the above example (Figure 2), a transition function \( \Phi_{R} \) for \( R \) will be developed based on three computation steps as follows:

- **Step 1:** Compute the set of \( R \)-states \( \{ \varphi_{i} \mid i = 1, \ldots, n \} \) such that \( BS(\varphi_{1}) \cup \ldots \cup BS(\varphi_{n}) = BS(\varphi) \) and \( \forall i, \forall s, s' \in BS(\varphi_{i}), e(a, s) = e(a, s') \). Compute \( e(a, \varphi_{i}) \) for \( i = 1, \ldots, n \);

- **Step 2:** For each \( R \)-state \( \varphi_{i} \), compute \( \varphi'_{i} \) by updating \( \varphi_{i} \) such that \( e(a, \varphi_{i}) \) is true in \( \varphi'_{i} \) and nothing else changes;

- **Step 3:** Convert the disjunction \( \varphi'_{1} \lor \ldots \lor \varphi'_{n} \) into an equivalent \( R \)-state \( \varphi' \), i.e., \( BS(\varphi') = BS(\varphi'_{1} \lor \ldots \lor \varphi'_{n}) \).

To realize the above computation steps, we then specify a set of abstract operations on a representation \( R \) and use these operations in defining the abstract transition function \( \Phi_{R} \). We devise a set of abstract algorithms for computing the transition function that can be instantiated in each concrete representation and used in the implementation of conformant planners.

We illustrate the generic framework by instantiating it in three concrete representations of belief states:

- **minimal-DNF**, a compact form of disjunctive normal form formulae,
- **minimal-CNF**, a compact form of conjunctive normal form formulae, and
- **prime implicates**, a well-known, special form of minimal-CNF.

For each representation, we define its set of operations, provide a concrete implementation of these operations, and instantiate the abstract algorithms to create a conformant planner. This results in three conformant planning systems, called DNF, CNF, and PIP, that employ the minimal-DNF, minimal-CNF, and prime implicates as the belief state representation, respectively.

We evaluate these systems using benchmarks available in the literature as well as new benchmarks that appear to be particularly challenging for other planners. The experimental evaluation shows that the three systems are very competitive with other planners. In addition, the two systems using conjunctive representations (CNF and PIP) are shown to be particularly effective on the new problems.

**Acknowledgements**

Thanks to Knexus Research Corporation for sponsoring this article.
References


