On the Use of Prime Implicates in Conformant Planning*

Son Thanh To
New Mexico State University
Department of Computer Science
sto@cs.nmsu.edu

Tran Cao Son
New Mexico State University
Department of Computer Science
tson@cs.nmsu.edu

Enrico Pontelli
New Mexico State University
Department of Computer Science
epontell@cs.nmsu.edu

Abstract

The paper presents an investigation of the use of two alternative forms of CNF formulae—prime implicates and minimal CNF—to compactly represent belief states in the context of conformant planning. For each representation, we define a transition function for computing the successor belief state resulting from the execution of an action in a belief state; results concerning soundness and completeness are provided. The paper describes a system (PIP) which dynamically selects either of these two forms to represent belief states, and an experimental evaluation of PIP against state-of-the-art conformant planners. The results show that PIP has the potential of scaling up better than other planners in problems rich in disjunctive information about the initial state.

Introduction and Motivation

Conformant planning (Smith and Weld 1998) is the problem of planning in presence of incomplete information about the initial state. One of the most important questions in conformant planning is how to represent the information about the initial situation, which is often referred as the *initial belief state*. In a domain with n propositions, the size of the initial belief state can be 2^n. In the literature, the description of the initial belief state is often given as a CNF formula with some additional constructs (as discussed later).

The representation method used to encode belief states affects the performance of a conformant planner in several ways. It can quickly increase the memory usage of the planner, leading to undesirable out-of-memory situations, if the size of the belief state is large. It also directly affects the time complexity in computing the successor belief states, since this task often requires the planner to test for the satisfaction of a conjunction of literals in a belief state, which is a NP-hard problem.

In the past, several representations have been developed. An indirect representation of belief states is used in CFF (Brafman and Hoffmann 2004), while ordered binary decision diagram (OBDD) is employed in POND (Bryce, Kambhampati, and Smith 2006); approximation states has been introduced in CPA (Tran et al. 2009); the work presented in DNF (To, Pontelli, and Son 2009) relies on explicit disjunctive formulae. The investigation presented in (To, Pontelli, and Son 2009) also discusses in more details the advantages and disadvantages of each of the aforementioned representations. It is worth mentioning that, regardless of the representation of belief states, *all planners* have difficulties scaling up when the size of the initial belief state is large. For example, all planners fail to find a solution of a modified version of the `coins-21` problem instance from the IPC-2006 competition, either because they run out of memory or the plan computation exceeds reasonable time limit. The initial belief state of this problem contains 10^{16} states, compared to the "easy instances" in the same domain—e.g., the `coins-20` instance has an initial belief state containing less than 10^6 possible states, and can be solved by all planners in less than two minutes.

The above issues motivated us to investigate alternative representations of belief states in conformant planning. Inspired by recent developments in other areas (e.g., the d-DNNF representation of SAT (Darwiche 2001)) and aware of the difficulties involved in using OBDD in computing the successor state, our goal is to identify a belief state representation with two desirable properties. First, the size of the representation should be minimal (as defined later). Second, the representation should facilitate a simple and efficient way for determining the satisfaction of a set of literals given a belief state.

In this paper, we investigate two different CNF-based belief state representations. The first representation employs *prime implicates*, called *pi-formula*. In this representation, a formula is replaced by its set of prime implicates or its *pi-form*. This representation exhibits a number of desirable properties. The complexity of checking tautological in a pi-form of a formula is linear for a literal (and polynomial for a clause) in the number of the propositions. Second, the representation is unique among the set of equivalent CNF formulae, making tractable the problem of checking for repetitions of belief states in a search tree. Third, this representation is, in many cases, very compact. Finally, the computation of the successor belief states under this representation can be done efficiently. The main disadvantage of this representation is that the size of the pi-form of a formula is sometimes very large. To this end, we investigate an alternative representa-

*Partial supported by NSF grants IIS-0812267, CBET-0754525, and CREST-0420407.

tion, called *minimal CNF*, whose size is often significantly smaller than its equivalent pi-form and can compensate for the pi-form representation in many cases.

In this paper, we provide a formal description of the two representations and design a best-first search conformant planner (PIP) that adopts them. We provide an experimental comparison of PIP against other planners. Our experiments show that PIP is comparable in speed with state-of-the-art planners and scales up better in domains rich in disjunctive information about the initial belief state.

Background: Conformant Planning

A *planning problem* is a tuple $P = \langle F, O, I, G\rangle$, where F is a set of propositions, O is a set of actions, I describes the initial state of the world, and G describes the goal. A *literal* is either a proposition $p \in F$ or its negation $\neg p$. $\bar{\ell}$ denotes the complement of a literal ℓ. For a set of literals L, $\overline{L} = \{\bar{\ell} \mid \ell \in L\}$. A conjunction of literals is often represented as the set of its literals.

A set of literals X is *consistent* if there is no $p \in F$ such that $\{p, \neg p\} \subseteq X$, and *complete* if, for each $p \in F$, either $p \in X$ or $\neg p \in F$. A *state* s is a consistent and complete set of literals. A *belief state* is a set of states. We will often use lowercase (resp. uppercase) letter, possibly with indices, to represent a state (resp. a belief state).

Each action a in O is associated with a precondition ϕ (denoted by $pre(a)$) and a set of conditional effects of the form $\psi \rightarrow \ell$ (also denoted by $a : \psi \rightarrow \ell$), where ϕ and ψ are sets of literals and ℓ is a literal.

A state s satisfies a literal ℓ, denoted by $s \models \ell$, if $\ell \in s$. s satisfies a conjunction of literals X, denoted by $s \models X$, if it satisfies every literal belonging to X. The satisfaction of a formula in a state is defined in the usual way. Likewise, a belief state S satisfies a literal ℓ, denoted by $S \models \ell$, if $s \models \ell$ for every $s \in S$.

Given a state s, an action a is *executable* in s if $s \models pre(a)$. The effects of executing a in s is

$$e(a, s) = \{\ell \mid \exists (a : \psi \rightarrow \ell).\ s \models \psi\}$$

The transition function, denoted by Φ, in the planning domain of P is defined by

$$\Phi(a, s) = \begin{cases} s \setminus \overline{e(a,s)} \cup e(a,s) & s \models pre(a) \\ \bot & \text{otherwise} \end{cases} \quad (1)$$

where $\bot$ denotes a failed state.

We can extend the function Φ to define $\widehat{\Phi}$, a transition function which maps sequences of actions and belief states to belief states. $\widehat{\Phi}$ is used to reason about the effects of plans. Let S be a belief state. We say that an action a is executable in a belief state S if it is executable in every state belonging to S. Let $[a_1, \ldots, a_n]$ be a sequence of actions:

- If $n = 0$ then $\widehat{\Phi}([\,], S) = S$;
- If $n > 0$ then
  - if $\widehat{\Phi}([a_1, \ldots, a_{n-1}], S) = \bot$ or if a_n is not executable in $\widehat{\Phi}([a_1, \ldots, a_{n-1}], S)$, then $\widehat{\Phi}([a_1, \ldots, a_n], S) = \bot$;
  - if $\widehat{\Phi}([a_1, \ldots, a_{n-1}], S) \neq \bot$ and a_n is executable in $\widehat{\Phi}([a_1, \ldots, a_{n-1}], S)$ then $\widehat{\Phi}([a_1, \ldots, a_n], S) = \{\Phi(a_n, s') \mid s' \in \widehat{\Phi}([a_1, \ldots, a_{n-1}], S)\}$.

The initial state of the world (I) is a belief state and is represented by a formula. In all benchmarks, I consists of a conjunction of a set of literals, a set of `oneof`-statements—representing an exclusive-or of its components—and a set of `or`-statements—representing the logical or of its components. By S_I we denote the set of all states satisfying I. The goal description G can contain literals and `or`-clauses.

A sequence of actions $[a_1, \ldots, a_n]$ is a solution of P if $\widehat{\Phi}([a_1, \ldots, a_n], S_I)$ satisfies the goal G. In this paper, for an action a, we will denote with C_a the set of conditional effects of a.

The function $\widehat{\Phi}$ can be used in the implementation of a best-first search based conformant planner. As we have discussed earlier, one of the important factors to this effort lies in the representation of belief states.

CNF Representations for Belief States

In this section, we explore the use of prime implicates in representing belief states for the development of a conformant planner. Observe that the development of a new representation of belief states needs to come with a set of operations such as checking for the satisfaction of a condition and/or updating a belief state with a set of effects.

A *clause* α is a set of fluent literals. α is *tautological* if $\{f, \neg f\} \subseteq \alpha$ for some $f \in F$ and it is a *unit clause* if $|\alpha| = 1$. A *CNF formula* is a set of clauses. A literal l is in a CNF formula φ, denoted by $l \in \varphi$, if there exists $\alpha \in \varphi$ such that $l \in \alpha$. By φ_l (resp. $\varphi_{\bar{l}}$) we denote the set of clauses in φ which contain l (resp. $\bar{l}$).

A clause α *subsumes* a clause β (or β is *subsumed by* α) if $\alpha \subset \beta$. Given a CNF formula φ, a clause α in φ is said to be *trivially redundant for* φ if it is tautological or it is subsumed by another clause in φ. The technique of simplifying a CNF formula by removing subsumed clause(s) from that formula is called *subsumption*.

A clause α is said to be *resolvable with* another clause β if there exists a literal ℓ such that $\ell \in \alpha$, $\bar{\ell} \in \beta$, and their *resolvent* $\alpha|\beta$, defined by $\alpha|\beta = (\alpha \setminus \{\ell\}) \cup (\beta \setminus \{\bar{\ell}\})$, is a non-tautological clause. In this case, we say that α is resolvable with β on ℓ. Observe that, if α and β are two clauses in a CNF formula φ and there exists a clause in φ which is subsumed by $\alpha|\beta$ then φ can be simplified to an equivalent smaller formula by replacing all the clauses subsumed by $\alpha|\beta$ in φ with this resolvent. This technique is referred to as subsumable resolution and $\alpha|\beta$ is called a subsumable resolvent. For example, applying subsumable resolution to the set $\{\{f, g\}, \{f, h, \neg g\}\}$ results in $\{\{f, g\}, \{f, h\}\}$, whereas applying this technique to $\{\{f, g\}, \{f, \neg g\}, \{f, h\}\}$ returns $\{\{f\}\}$. For two sets of clauses φ and ψ, we denote with $\varphi|\psi = \{\alpha|\beta \mid \alpha$ *is resolvable with* $\beta \wedge \alpha \in \varphi \wedge \beta \in \psi\}$.

For two CNF formulae $\varphi = \{\alpha_1, \ldots, \alpha_n\}$ and $\psi = \{\beta_1, \ldots, \beta_m\}$, the *cross-product* of φ and ψ, denoted by $\varphi \times \psi$, is the CNF-formula defined by $\{\alpha_i \cup \beta_j \mid \alpha_i \in \varphi, \beta_j \in \psi\}$. If either φ or ψ is empty then $\varphi \times \psi = \emptyset$. The *reduced-cross-product* of φ and ψ, denoted by $\varphi \otimes \psi$, is the CNF-formula obtained from $\varphi \otimes \psi$ by removing all trivially redundant clauses from $\varphi \times \psi$.

For a set of CNF formulae $\Psi = \{\varphi_1, \ldots, \varphi_n\}$, $\times[\Psi]$

(resp. $\otimes[\Psi]$) denotes $\varphi_1 \times \varphi_2 \times \ldots \times \varphi_n$ (resp. $\varphi_1 \otimes \varphi_2 \otimes \ldots \otimes \varphi_n$). It is easy to see that both $\times[\Psi]$ and $\otimes[\Psi]$ are a CNF-formula equivalent to $\bigvee_{i=1}^{n} \varphi_i$.

A clause α is said to be an *implicate* of a formula φ if $\varphi \models \alpha$. It is a *prime implicate* of φ if there is no other implicate β of φ such that β subsumes α. Obviously, if a unit clause is an implicate of a formula then it is also a prime implicate of that formula. We denote the set of prime implicates of a formula φ by $PI(\varphi)$. Clearly, a CNF formula φ is *in prime implicate form* (*pi-formula*, for short) if $\varphi = PI(\varphi)$. One can prove the following proposition

Proposition 1. *The reduced-cross-product of a set of pi-formulae is a pi-formula.*

Thus, the pi-formula of the disjunction of a small set of pi-formulae can be computed in polynomial time.

Prime Implicate Representation

Definition 1. *A* PI-state *is a pi-formula. A set of PI-states is called a* PI-belief state.

It is easy to see that if φ is a pi-formula, checking whether a literal ℓ is satisfied by φ can be done in linear time in the size of F (the set of propositions).

We will now specify how the function Φ can be computed given that a belief state is represented by a PI-state.

Definition 2. *Let φ be a PI-state and ℓ be a literal. The update of φ by ℓ, denoted by $upd_{pi}(\varphi, \ell)$, is defined as follows:*

$$upd_{pi}(\varphi, \ell) = (\varphi \setminus (\varphi_{\ell} \cup \varphi_{\bar{\ell}})) \wedge \ell$$

Intuitively, $upd_{pi}(\varphi, \ell)$ encodes the PI-state after execution of an action, that causes ℓ to be true, in φ. For example,

- $upd_{pi}(\{\{f\}, \{\bar{\ell}\}\}, \ell) = \{\{f\}, \{\ell\}\}$
- $upd_{pi}(\{\{f\}, \{h, \ell\}\}, \ell) = \{\{f\}, \{\ell\}\}$
- $upd_{pi}(\{\{g, h\}, \{g, \bar{\ell}\}, \{h, \ell\}\}, \ell) = \{\{g, h\}, \{\ell\}\}$

It is easy to see that the following proposition holds.

Proposition 2. *If φ is a PI-state and ℓ_1, ℓ_2 are literals such that $\ell_1 \neq \ell_2$ and $\ell_1 \neq \bar{\ell}_2$*

- *$upd_{pi}(\varphi, \ell_1)$ is a PI-state; and*
- *$upd_{pi}(upd_{pi}(\varphi, \ell_1), \ell_2) = upd_{pi}(upd_{pi}(\varphi, \ell_2), \ell_1)$.*

The above proposition shows that the result of updating a PI-state φ using a consistent set of literals L is independent from the order in which the various literals of L are introduced. For a consistent set of literals L, we define $upd_{pi}(\varphi, L) = upd_{pi}(upd_{pi}(\varphi, \ell), L \setminus \{\ell\})$ for any $\ell \in L$ if $L \neq \emptyset$ and $upd_{pi}(\varphi, \emptyset) = \varphi$.

Let us now define the transition function Φ_{PI} for PI-states. Given an action a with the precondition $pre(a)$, its set of conditional effect C_a, and a PI-state φ, we need to define the successor PI-state $\Phi_{PI}(a, \varphi)$. Note that, when computing $\Phi_{PI}(a, \varphi)$, for each $\psi \rightarrow \ell$ in C_a there are three cases that need to be considered:

- $\varphi \models \psi$
- $\varphi \models \neg\psi$
- $\varphi \not\models \psi$ and $\varphi \not\models \neg\psi$

As an example, if $\varphi = p \wedge q$ and $\psi = r$, then we have $\varphi \not\models \psi$ and $\varphi \not\models \neg\psi$. In order to define Φ_{PI}, we need the following definition.

Definition 3. *Let φ be a PI-state and γ a consistent set of literals. The* enabling form of φ w.r.t. γ, *denoted by $\varphi \oplus \gamma$, is a PI-belief state defined by*

$$\varphi \oplus \gamma = \begin{cases} \{\varphi\} & \text{if } \varphi \models \gamma \text{ or } \varphi \models \neg\gamma \\ \{PI(\varphi \wedge \gamma), PI(\varphi \wedge \neg\gamma)\} & \text{otherwise} \end{cases}$$

where $\neg\gamma$ is the clause $\{\bar{\ell} \mid \ell \in \gamma\}$.

It is easy to see that $\varphi \oplus \gamma$ is a set of (at most two) PI-states such that for every $\delta \in \varphi \oplus \gamma$, $\delta \models \gamma$ or $\delta \models \neg\gamma$. We can prove the following:

Proposition 3. *Let φ be a PI-state and let γ be a consistent set of literals. Let σ be the union set of all literals in unit clauses in φ;*

- *$\varphi \oplus \gamma = \varphi \oplus (\gamma \setminus \sigma)$;*
- *If the number of literals in γ is bounded by a constant, then $\varphi \oplus \gamma$ can be computed in polynomial time; and*
- *If the number of literals in $\gamma \setminus \sigma$ is bounded by a constant, then $\varphi \oplus \gamma$ can be computed in polynomial time.*

For a PI-belief state Ψ, let $\Psi + \gamma = \bigcup_{\varphi \in \Psi} (\varphi \oplus \gamma)$.

Proposition 4. *Let φ (resp. Ψ) be a PI-state (resp. PI-belief state). If γ is a consistent set of literals, then $\varphi \oplus \gamma$ (resp. $\Psi \oplus \gamma$) is a PI-belief state equivalent to φ (resp. Ψ). If γ_1 and γ_2 are two consistent sets of literals, then*

$$(\varphi \oplus \gamma_1) \oplus \gamma_2 = (\varphi \oplus \gamma_2) \oplus \gamma_1.$$

Definition 4. *Let a be an action with the set of conditional effects C_a. A PI-state φ is said to be* enabling *for a if for every conditional effect $\psi \rightarrow \ell$ in C_a, either $\varphi \models \psi$ or $\varphi \models \neg\psi$. A PI-belief state Ψ is* enabling *for a if every PI-state in Ψ is enabling for a.*

For an action a and a PI-state φ, let $enb_{pi}(a, \varphi) = ((\varphi \oplus \psi_1) \oplus \ldots) \oplus \psi_k$ where $C_a = \{\psi_1 \rightarrow \ell_1, \ldots, \psi_k \rightarrow \ell_k\}$.

Proposition 5. *For every PI-state φ and action a, $enb_{pi}(a, \varphi)$ is a PI-belief state which is equivalent to φ and enabling for a.*

For an action a and a PI-state φ, the effect of a in φ, denoted $e(a, \varphi)$, is defined as follows:

$$e(a, \varphi) = \{\ell \mid \psi \rightarrow \ell \in C_a, \varphi \models \psi\}.$$

We are now ready to define the function Φ_{PI}.

Definition 5. *Let φ be a PI-state and let a be an action. $\Phi_{PI}(a, \varphi)$ denotes the transition function for PI-states:*

$$\Phi_{PI}(a, \varphi) = \begin{cases} \otimes[\{upd_{pi}(\phi, e(a, \phi)) \mid \phi \in enb_{pi}(a, \varphi)\}] & \\ \qquad \text{if } \varphi \models pre(a) & \\ \bot \qquad \text{otherwise} & \end{cases} \tag{2}$$

The computation of Φ_{PI} can be done efficiently under reasonable assumption, as stated next.

Proposition 6. *Let φ be a PI-state and let a be an action whose conditional effects are $C_a = \{\psi_i \rightarrow \ell_i \mid i = 1, \ldots, k\}$. The computation of $\Phi_{PI}(a, \varphi)$ can be done in polynomial time in the size of φ if k and $|\psi_i|$, $\forall i = 1, \ldots, k$, are bounded by a constant.*

Observe that, since k and the size of every ψ_i is usually small, the assumption can be acceptable and computing

$\Phi_{PI}(a, \varphi)$ largely depends on the size of φ. Φ_{PI} can be extended to define $\widehat{\Phi}_{PI}$, which allows us to reason about the effects of sequences of actions, in the same manner $\widehat{\Phi}$ is defined. The next theorem shows that $\widehat{\Phi}_{PI}$ is equivalent to the complete semantics defined by $\widehat{\Phi}$. Thus, any planner using $\widehat{\Phi}_{PI}$ in its search for solutions will be sound and complete.

Theorem 1. *Let φ be a PI-state and $[a_1, \ldots, a_n]$ be an action sequence. Then, $\widehat{\Phi}_{PI}([a_1, \ldots, a_n], \varphi) \equiv \widehat{\Phi}([a_1, \ldots, a_n], BS(\varphi))$ where $BS(\varphi)$ denotes the set of states satisfying φ.*

The prime implicate representation has nice properties, e.g., checking entailment of a set of literals or a clause can be done efficiently and the reduced-cross-product of a set of PI-states will result in a PI-state. Moreover, since the PI-state equivalent to a given CNF formula is unique, it is easy to ensure that no PI-state is explored more than once in an implementation.

The critical problem with this representation lies in the fact that the complexity of the function Φ_{PI} depends on the size of the PI-state which, unfortunately, can be much larger than the size of an equivalent CNF formula. For example, for $\varphi = \{\{f, \neg g\}, \{g, \neg h\}, \{\neg f, h\}\}$, we have that $PI(\varphi) = \{\{f, \neg g\}, \{g, \neg h\}, \{\neg f, h\}, \{f, \neg h\}, \{\neg f, g\}, \{\neg g, h\}\}$ which is twice as big as φ. For this reason, we investigate the second CNF-representation.

Minimal CNF Representation

Definition 6. *A CNF formula φ is* minimal *if*

- *φ does not contain a trivially redundant clause; and*
- *φ does not contain two clauses γ and δ such that γ is resolvable with δ and $\gamma|\delta$ subsumes a clause in φ.*

A CNF-state *is a minimal CNF formula. A set of CNF-states is called a* CNF-belief state.

Intuitively, a CNF-state φ is minimal in the sense that it does not contain trivially redundant clauses and it cannot be simplified by subsumable resolution. Observe that a PI-state is also a CNF-state, but the converse is not necessarily true. Furthermore, for one CNF formula φ, the PI-state equivalent to φ is unique but there can be more than one CNF-state equivalent to φ. In the following, we write $min(\cdot)$ to denote an idempotent function that converts an arbitrary CNF formula φ to an equivalent CNF-state. For CNF-states, updating a state with a literal is defined as follows.

Definition 7. *Let φ be a CNF-state and l be a literal. The update of φ by l, denoted by $upd(\varphi, l)$, is defined as follows:*

$$upd(\varphi, l) = min(\ (\varphi \setminus (\varphi_l \cup \varphi_{\bar{l}})) \wedge l \wedge \varphi_l|\varphi_{\bar{l}}\ )$$

Similar to Proposition 2, we can extend upd to define the updating of a CNF-state by a consistent set of literals by $upd(\varphi, L) = upd(upd(\varphi, l), L \setminus \{l\})$ for some $l \in L$. Using this definition, most of the operations on CNF-states and CNF-belief states can be defined similarly to the operations on PI-states and PI-belief states (Definitions 3-5). Due to lack of space, we omit their precise formulations which are similar to those defined for reduced-CNF representation (To, Son, and Pontelli 2010), a preliminary version of minimal CNF form. Let us note that this allows us to define a function, $\widehat{\Phi}_{CNF}$, for computing the result of a sequence of actions applied on a CNF-state. Furthermore, this function is equivalent to $\widehat{\Phi}_{PI}$.

PIP—a Conformant Planner with Dynamic Representation Selection

In this section, we describe a conformant planner, called PIP, that employs both the PI-state and the CNF-state in its repesentation of belief states. PIP is a heuristic best-first search, progression-based planner. We develop PIP from the source code of DNF (To, Pontelli, and Son 2009).

Since preprocessing a CNF formula to a compact CNF form makes the computation of its prime implicates more efficient, first we start computing the minimal CNF formula encoding I using a fixed-point algorithm of subsumption, subsumable resolution, and unit propagation (Piette et al. 2008) whose running time is polynomial in the size of the formula. The resulting CNF-state is fed to a test phase which decides whether the CNF-state representation or the PI-state representation should be used as follows.

- The minimal CNF representation will be selected if the computation of the initial PI-state takes too long. More precisely, if it is greater than $|F|^3 * T_I$, where F is the set of propositions and T_I is the time spent for computing CNF-state from the CNF-formula encoding I.
- The minimal CNF representation will be used if, within the exploration of a small number of belief states (N) (in our experiment, we set $N = min(10, |F|)$), a pilot search using PI representation takes longer time than that using CNF representation. In addition, if the total size of the PI-states generated by the pilot search using PI representation is not smaller than twice of the total size of the CNF-states generated by the other pilot search then the minimal CNF representation is used.
- Otherwise, the PI-state representation will be chosen.

Algorithm 1 Search(F,O,I,G)

```
1: Input: Problem ⟨F, O, I, G⟩
2: Output: A plan if exists
3: Compute CNF-state φ_I from I
4: Create an empty queue Q and let PI be a boolean variable and P be a plan variable
5: Set (P, Q, PI) = TestPhase(F, O, φ_I, G)
6: if P ≠ NULL then
7:     return P
8: end if
9: if PI = true then
10:    return Search_PI(F, O, Q, G)
11: else
12:    return Search_CNF(F, O, Q, G)
13: end if
```

PIP will commit to a representation based on the result of the test phase, then will use the corresponding transition function to search for a plan. The overall algorithm is given in Algorithm 1 where $Search_{PI}(F, O, Q, G)$ and $Search_{CNF}(F, O, Q, G)$ implement the best-first search engine using the PI-state and the CNF-state representations.

For computing the PI-states, we use an incremental algorithm for computing prime implicates, called IPIA (de Kleer 1992).

To reduce the cost of computing the PI-state from a disjunction of PI-states, we use the following proposition

Proposition 7. *Given two PI-states φ and ψ and two clauses c_1 and c_2 such that $c_1 \in \varphi$ and $c_2 \in \psi$, it holds that:*

- *If $c_1 \subseteq c_2$ then c_2 is a prime implicate of $\varphi \otimes \psi$.*
- *If c_1 is a unit clause, $c_1 \cup c_2$ is not tautological, and c_2 does not belong to the previous case then $c_1 \cup c_2$ is a prime implicate of $\varphi \otimes \psi$.*

Proposition 7 allows us to reduce subsumption checking and avoid the creation of redundant clauses.

Problem	PIP	DNF	CPA	T0	CFF	POND
block-1	0.58/7*	0.67/7	0.68/4	0.08/5	0.02/6	**0.01**/6
block-2	0.83/18*	0.72/38	0.76/14	0.2/23	TO	**0.06**/34
block-3	TO*	216.1/331	OM	48/80	TO	**3.9**/80
bomb-50-10	1.24/90	1.28/90	21/58	F	**1.16**/90	OM
bomb-100-10	**2.48**/190	**2.69**/190	110/110	F	34/190	OM
bomb-100-20	**4.65**/180	5.15/180	244/118	F	28/180	OM
cc-40-20	**1.11/175**	7.79/535	F	87/918	TO	TO
cc-64-32	**1.53/283**	21.24/872	F	F	TO	TO
cc-87-43	**2.19/387**	40.4/1249	F	F	TO	TO
coins-15	1.29/97*	1.07/67	7/362	**0.12/79**	2.6/89	10/124
coins-20	2.46/146*	1.44/99	17/105	**0.15/107**	16/143	105/153
coins-21	OM*	OM	OM	F	TO	TO
ds-6-5	26.8/464	**13**/570	235/1971	98.5/347	TO	TO
ds-8-5	211/1110	**65/878**	2152/541	F	TO	OM
ds-10-3	509/2102	**193/680**	4694/648	2388/1360	TO	OM
1d-4-2	9.8/110*	**2.18/64**	2.24/52	12.44/72	TO	OM
1d-5-2	81.6/312*	**5.85/122**	11.24/88	188/126	TO	OM
1d-6-2	434/492*	**15.4/186**	33.7/124	F	TO	OM
1d-6-3	OM*	506/186	OM	F	TO	OM
gripper-50	3.9/298	8.71/234	106.7/106	52.1/198	**3.2**/294	TO
gripper-60	**5.28**/358	13.7/278	189.3/286	91.6/238	6.4/354	TO
gripper-70	**7.17**/418	16.51/316	278.6/580	151/278	11/414	TO
gripper-80	**10.36**/478	22.58/350	432/166	234/318	18/474	TO
lng-4-3-3	**2.73/4**	3.53/4	4.48/4	F	TO	OM
lng-5-3-3	**10.9**/14	13.9/6	20/6	F	TO	OM
lng-6-3-3	**46/6**	52/6	82/6	F	TO	OM
lng-7-3-3	**171**/52	178/14	OM	F	TO	OM
push-5-6	**23.6/312**	24.1/783	TO	F	TO	TO
push-6-5	**51/216**	58/1346	OM	F	TO	TO
push-7-5	163/956	**130**/1767	OM	F	TO	TO
push-7-7	**391/774**	494/2565	OM	F	TO	TO
raok-2	1.65/27*	0.57/26	1.1/32	**0.04**/21	0.07/34	F
raok-3	2.7/177*	1.68/153	3.8/152	**0.25**/66	12/102	F
raok-4	TO*	TO	TO	F	TO	F
sortnet-5	1.2/15*	0.94/15	0.94/12	0.26/15	NA	**0/12**
sortnet-10	22.1/55*	1.85/54	3.18/39	OM	NA	**0.03/38**
sortnet-15	255/119*	35.9/118	244/65	F	NA	**0.14/65**
sortnum-5	1.58/10*	2.57/10	OM	1.92/10	2.9/10	**0.49**/10
sortnum-6	22.9/15*	397/15	OM	**18.1**/15	TO	19.4/15
sortnum-7	537/21*	OM	OM	**91**/21	TO	1077/21
sortnum-8	6970/28*	OM	OM	F	TO	TO
uts-c-3	4.22/3*	0.54/3	1.19/3	**0.15**/3	NA	F
uts-c-4	TO*	0.56/6	18.3/6	**0.47**/7	NA	F
uts-c-5	TO*	**0.73**/10	OM	1.8/10	NA	F

Table 1: Benchmarks from Literature

Heuristic

PIP uses two search heuristics: the number of satisfied subgoals and the size of the CNF-state, i.e., the sum of the size of all clauses in the CNF formula. This heuristic may seem naive, nevertheless it is very close to that of DNF, a planner similar to PIP except for the belief state representation. This choice allows us to use DNF in our comparisons. The comparison of PIP with other planners is for proving that pi-formulae can be used to build a competitive conformant planner.

Experimental Evaluation

We compare PIP with the following conformant planners: DNF (To, Pontelli, and Son 2009), CPA (Tran et al. 2009), CFF (Brafman and Hoffmann 2004), POND (Bryce, Kambhampati, and Smith 2006), and `t0` (Palacios and Geffner 2007) using conformant planning benchmarks from literature. To the best of our knowledge, these planners currently yield the best performance in these domains. We also use a set of new benchmarks modified from those in literature by replacing oneof-clause(s) with or-clause(s) in the initial state description, to generate cases that are rich in disjunctive information. All the experiments have been carried out using a dedicated Intel Core 2 Dual 9400 2.66GHz 4GB Linux workstation. The time-out limit is set to 2 hours. The experimental results are reported in tables 1 and 2. We report the time and plan length for each planner. 'OM', 'TO', and 'F' denote out-of-memory, time-out, and abnormal termination of the planner. Due to space limitation, we only report the results of our experiments in a few large instances of each benchmark. In the following, we discuss each table and evaluate the strengths and weaknesses of PIP against other planners.

Benchmarks From Literature

Table 1 contains the results obtained from experiments on the domains *block*, *dispose* (ds-n-m), *raokeys* (raok-n), and *uts-cycle* (uts-c-n) used in IPC-2008, *coins* and *sortnet* are from IPC-2006, and *bomb* and *gripper* are from the authors of CFF. The remaining problems, including *corner-cube* (cc-n-m), *1dispose* (1d-n-m), *look-and-grab* (lng-n-m-k), *push* (push-n-m), and *sort-number* (sortnum-n) are included in the package of `t0`.

The results show that PIP is the best in five (*bomb, corner-cube, gripper, look-and-grab,* and *push*) out of thirteen domains. DNF outperforms the others in three domains (*dispose, 1dispose,* and *uts-cycle*). The overall performance of PIP is comparable to that of DNF for the benchmarks. Note that, DNF and CPA take advantage of the *oneof-combination* technique, aimed at reducing the disjunctive form of the initial belief state. Without this technique, these planners would have trouble dealing with several benchmarks, such as *coins*, *dispose*, *1dispose*, *look-and-grab*, and *push*. Although this technique does not harm the soundness and completeness of the planners, the initial data is not equivalent to the original. Thus, it is somewhat "unfair" to compare the effectiveness of the belief state representations under these conditions.

Our experiments reveal that the search trees of PIP and DNF for the instance problems of *bomb* and the first three instance problems of *look-and-grab* are the same but PIP performs better. Hence, we can conclude that the prime implicate representation is better than the DNF representation for these cases. For *sort-number*, `t0` is the best on most instances but only PIP is able to deal with the largest instance. `t0` is also the best on *coins* and *raokey*. POND outperforms the others on *block* and *sortnet* but its performance is poor on the other domains. Note that, on the PIP column, "*" indicates that the minimal CNF representation has been selected . Observe that most of the best solutions are found by PIP using prime implicate representation.

Challenging Problems

In this subsection, we introduce a new set of problems aimed at demonstrating situations where PIP can show its full potential. They are variants of problems in Table 1, obtained by replacing `oneof` statements with `or` statements of the same set of literals. This modification is carried out only for the problems whose descriptions remain consistent. These variants are renamed by adding the prefix "or-" to their original name and they are shown in table 2. Due to the large size of the disjunctive normal form formulae representing belief states in these problems, a DNF belief state representation based planner, e.g., DNF and CPA, provide much poorer performance compared to that on the original problems. Moreover, in these problems, the oneof-combination technique is not applicable, making the performance of DNF and CPA even worse. On the contrary, due to the capability of maintaining a compact size of CNF formulae representing belief states, PIP outperforms impressively not only DNF belief state representation based planners, but also all other competitive state-of-the-art conformant planners. It is worth mentioning that there is no significant difference between the performance of any other planner on these problems in comparison with that on the original problems.

Problem	PIP	DNF	CPA	T0	CFF	POND
or-coins-15	0.91/81*	OM	OM	**0.112/79**	2.36/89	10.6/124
or-coins-21	**1478**/7321*	OM	OM	F	TO	TO
or-coins-23	**407.9**/3185*	OM	OM	F	TO	TO
or-coins-26	**448.2**/7321*	OM	OM	F	TO	TO
or-coins-29	**999.1**/5368*	OM	OM	F	TO	TO
or-coins-30	**7026**/11213*	OM	OM	F	TO	TO
or-ds-6-5	**4.83**/358	OM	OM	208.6/347	TO	OM
or-ds-8-3	**27.85/392**	OM	OM	278/761	TO	OM
or-ds-8-5	**30.84/537**	OM	OM	F	TO	OM
or-ds-10-5	**171.1/2486**	OM	OM	F	TO	OM
or-ds-10-9	**198/2485**	OM	OM	F	TO	OM
or-1d-5-2	**4.56**/344	34.1/94	OM	F	TO	OM
or-1d-5-4	**14.45**/312*	OM	OM	F	TO	OM
or-1d-6-2	**10.41**/492*	190.1/150	OM	F	TO	OM
or-1d-6-4	**33.24**/492*	OM	OM	F	TO	OM
or-1d-10-4	**667.12/3806***	OM	OM	F	TO	OM
or-push-4-4	1.23/132	OM	OM	15.45/210	TO	TO
or-push-5-6	**3.42/312**	OM	OM	TO	TO	TO
or-push-6-5	**6.16/216**	OM	OM	TO	TO	TO
or-push-10-5	**203.7**/841	OM	OM	TO	TO	TO
or-push-10-8	**255.3**/1218	OM	OM	TO	TO	TO

Table 2: Challenging Domains

Conclusion and Future Work

In the paper, we showed that prime implicate formulae have the potential of offering highly desirable properties in the representation of belief states for conformant planning. We developed an effective complete transition function for computing successor belief states using the prime implicate representation. We also proposed an alternative compact CNF form, minimal CNF, for representing belief states in the case the prime implicate form is not suitable for them. Another complete transition function for this representation has also been provided. A planner that uses alternative representations, like PIP, appears to be a valid approach which combines strengths of different representations and reduces their disadvantages. This has been validated by the experimental results. We also identified a set of problems where a planner based on conjunctive representations of belief states offers better results. The theoretical results shown in Propositions 1 and 7 may also be very useful in solving other problems which require computation of prime implicates.

Using different types of formulae to represent belief states in a conformant planner and identifying a suitable representation for an arbitrary problem are a promising research direction. Finally, an improved heuristic is also desirable.

References

Brafman, R., and Hoffmann, J. 2004. Conformant planning via heuristic forward search: A new approach. In Koenig, S.; Zilberstein, S.; and Koehler, J., eds., *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 355–364. Whistler, Canada: Morgan Kaufmann.

Bryce, D.; Kambhampati, S.; and Smith, D. 2006. Planning Graph Heuristics for Belief Space Search. *Journal of Artificial Intelligence Research* 26:35–99.

Darwiche, A. 2001. Decomposable negation normal form. *Journal of the ACM* 48(4):608–647.

de Kleer, J. 1992. An improved incremental algorithm for computing prime implicates. In *AAAI*, 780–785.

Palacios, H., and Geffner, H. 2007. From Conformant into Classical Planning: Efficient Translations that may be Complete Too. In *ICAPS*.

Piette, C.; Hamadi, Y.; ; and Saiuml, L. 2008. Vivifying Propositional Clausal Formulae. In *ECAI*, 525–529.

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

To, S. T.; Pontelli, E.; and Son, T. C. 2009. A Conformant Planner with Explicit Disjunctive Representation of Belief States. In *ICAPS*.

To, S. T.; and Son, T. C.; Pontelli, E. 2010. A New Approach to Conformant Planning using CNF. To Appear In *ICAPS*.

Tran, D.-V.; Nguyen, H.-K.; Pontelli, E.; and Son, T. C. 2009. Improving performance of conformant planners: Static analysis of declarative planning domain specifications. In *PADL*, LNCS 5418, 239–253. Springer.