Appears in Proceedings of 18th Int'l Joint Conference on Artificial Intelligence (IJCAI '03).

Logical Filtering

Eyal Amir and **Stuart Russell**
Computer Science Division, University of California at Berkeley
Berkeley, CA 94720-1776, USA
{eyal,russell}@cs.berkeley.edu

Abstract

Filtering denotes any method whereby an agent updates its *belief state*—its knowledge of the state of the world—from a sequence of actions and observations. In *logical filtering*, the belief state is a logical formula describing possible world states and the agent has a (possibly nondeterministic) logical model of its environment and sensors. This paper presents efficient logical filtering algorithms that maintain a compact belief state representation indefinitely, for a broad range of environment classes including nondeterministic, partially observable STRIPS environments and environments in which actions *permute* the state space. Efficient filtering is also possible when the belief state is represented using prime implicates, or when it is approximated by a logically weaker formula.

1 Introduction

Any agent operating in a partially observable environment must perform computations that distinguish among the *a priori* possible current states of the world on the basis of past observations and actions. These computations may operate directly on a representation of the action–observation sequence (e.g., [Winslett, 1990; Kautz *et al.*, 1996]); they may reduce queries about the current state to queries about the initial state (e.g., [Reiter, 2001]); or, they may update the *belief state* (the agent's knowledge about the state of the world) after each action and observation. This latter approach, called *filtering* or *recursive state estimation* in the control theory literature, is particularly useful with unbounded sequences of actions and observations.

The main computational difficulties are 1) the time needed to update the belief state, and 2) the space required to represent it. These depend on the nature of the *transition model*, which describes how the environment evolves over time, the *observation model*, which describes the way in which the environment generates observations, and the family of representations used to denote belief states. Early work, beginning with Gauss, assumed *stochastic* models. For example, the *Kalman filter* [Kalman, 1960] is a ubiquitous device that maintains a multivariate Gaussian belief state over n variables, assuming linear–Gaussian transition and observation model. Crucially, the $O(n^3)$ update cost and the $O(n^2)$ space requirement *do not depend on the length of the observation sequence*; hence, a Kalman filter can run indefinitely. In this paper, we are interested in developing analogous results in the context of logical representations.

We adopt a simple logical language (Section 2) for describing the transition and observation models; the observations and the belief state itself are also logical formulae. The initial state may be only partially known; the transition model, which allows for actions by the agent itself, may be nondeterministic; and the observation model may be nondeterministic and *partial*, in the sense that the agent may not be able to observe the actual state.

Even when we restrict ourselves to propositional logic, it is clear that the general filtering problem is nontrivial (we prove it is computationally hard), because there are exponentially many possible states. We identify several classes of models that allow efficient filtering with respect to the belief-state representation size. Our primary method is based on decomposition theorems showing that 1) filtering distributes over disjunction in the belief state formula, and 2) filtering distributes over conjunction and negation if the actions are *permutations* of the state space. Such actions serve as one-to-one mappings between states, for those states in which they can be applied. We obtain efficient, exact algorithms for DNF belief states and for NNF (Negation Normal Form - all negations are in front of atoms) and CNF belief states with permuting actions. In other cases, we obtain efficient algorithms for *approximate filtering*.

In another class of dynamic systems, we can filter efficiently if the belief state is represented in CNF that includes all its prime implicates. Finally, we show that STRIPS models (possibly with nondeterministic effects of actions) also admit efficient filtering. The STRIPS assumption, that every action has no conditional effects and that an effect's preconditions are the preconditions for the action's execution, is key to this efficiency.

With respect to maintaining a compact representation, we show that properties similar to those mentioned above allow us to filter k-CNF formulae (CNF with clauses of at most k literals, when k is fixed) such that the result is represented in k-CNF (for the same fixed k). Thus, the belief state is maintained in $O(n^k)$ space indefinitely. In particular, we show mild conditions under which a compact belief state can be maintained in nondeterministic STRIPS domains and in permutation domains. Finally, we show that DNF belief states remain compact if the effects of actions are deterministic and guaranteed to hold. These results are the first analogues, in the logical arena, of the desirable properties possessed by Kalman filters for continuous variables.

Ours is by no means the first work on filtering in a logical context. Early on, it was pointed out that filtering is easy for deterministic systems with a known initial state [Fikes *et al.*, 1972; Lin and Reiter, 1997]. Filtering in nondeterministic domains is more difficult. In particular, the related problem of temporal projection is coNP-hard when the initial state is not fully known, or when actions have nondeterministic effects [Liberatore, 1997] (see also Section 3.3).

Traditionally, computational approaches for filtering take one of three approaches: 1) enumerate the world states possible in every belief state and update each of those states separately, together generating the updated belief state [Ferraris and Giunchiglia, 2000; Cimatti and Roveri, 2000], 2) list the sequence of actions and observations and prove queries on the updated belief state [Reiter, 2001; Sandewall, 1994], or 3) approximate the belief state representation [Son and Baral, 2001; Williams and Nayak, 1996].

The first two approaches cannot be used when there are too many possible worlds (e.g., when the domain includes more than a few dozens of fluents and there are more than 2^{40} possible states) or when the sequence of actions is long (e.g., more than 100 actions). Examples include robot localization, tracking of objects and their relationships, and data mining. The third approach gives rise to many mistakes that are sometimes dangerous, and requires an approximation that fits the given problem (if one exists). Many domains of 100 fluents or less are still computationally infeasible for it.

2 Logical Filtering

In this section we define logical filtering using a transition model and action semantics that are compatible with the *standard semantics* belief update operator of [Winslett, 1990]. (To avoid confusion, this is different from another operator presented in the same publication, *PMA*, that applies a *nonmonotonic* approach to formalize minimal change.) This operator is simple and allows us to examine computational properties easily. It can represent any logical transition system, and specifications in other action languages can be compiled into it [Winslett, 1990; Doherty *et al.*, 1998].

In what follows, for a set of propositional formulae, Ψ, $L(\Psi)$ is the signature of Ψ, i.e., the set of propositional symbols that appear in Ψ. $\mathcal{L}(\Psi)$ is the language of Ψ, i.e., the set of formulae built with $L(\Psi)$. Similarly, $\mathcal{L}(L)$ is the language of L, for a set of symbols L.

A transition system is a tuple $\langle \mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$, where

- $\mathcal{P}$ is a finite set of propositional fluents;
- $\mathcal{S} \subseteq Pow(\mathcal{P})$ is the set of world states;
- $\mathcal{A}$ is a finite set of actions;
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition relation.

The intuition for this transition system description is that $\mathcal{P}$ is the set of features that are available for us in the world, every element in $\mathcal{S}$ is a *world state* (i.e., a subset of $\mathcal{P}$, containing propositions that are true in this world state), $\mathcal{A}$ is the set of actions in the system and $\mathcal{R}(s, a, s')$ means that state s' is a possible result of action a in state s.

A *belief state* is a set of world states $\sigma \subseteq \mathcal{S}$. Performing an action a in a belief state σ results in a belief state that includes all the world states that may result from a in a world state in σ. We do not introduce *observations* in this transition model. Instead, we assume that observations are given to us (if at all) as logical sentences after performing an action.

A logical nondeterministic domain description D is a finite set of statements of the following kinds: *value propositions* of the form "**initially** F" describe the initial state and *effect rules* of the form "a **causes** F **if** G" describe the effects of actions, for F and G being *state formulae* (propositional combinations of fluent names). We say that F is the *head* and G is the *tail* of those rules.

For a domain description D we define $\mathcal{P}_D$, $\mathcal{A}_D$ to be the set of propositional fluents and actions mentioned in D, respectively. For a domain description D we define a transition relation $\mathcal{R}_D(s, a, s')$ as follows.

- A fluent $f \in \mathcal{P}_D$ is *possibly affected* by action a in state s, if there is a rule "a **causes** F **if** G" in D such that G is true in s and $f \in L(F)$.
- Let $I(a, s)$ denote the set of fluents in $\mathcal{P}_D$ that are *not* possibly affected by action a in state s.
- Let $F(a, s)$ be a set of all the heads of activated effect rules in s (i.e., if "a **causes** F **if** G" is activated in s, then $F \in F(a, s)$). We consider the case of $F(a, s) = \emptyset$ (no activated effect rules) as $F(a, s) \equiv TRUE$.
- Define (recalling that world states are sets of fluents)

$$\mathcal{R}_D = \left\{ \langle s, a, s' \rangle \;\middle|\; \begin{array}{l} (s' \cap I(a, s)) = (s \cap I(a, s)) \\ \text{and } F(a, s) \text{ is true in } s' \end{array} \right\}$$

When there is no confusion, we write $\mathcal{R}$ for $\mathcal{R}_D$.

If action a has an effect of *FALSE* in s, then it cannot execute.

In partially observable domains, we update our knowledge as a result of executing an action and collecting observations in the resulting state. The following definition of filtering assumes that σ is a set of world states. We use our transition operator $\mathcal{R}$ to define the resulting belief state from each action. An observation o is a formula in our language.

Definition 2.1 (Logical Filtering Semantics). *Let* $\sigma \subseteq \mathcal{S}$ *be a belief state. The* filtering *of a sequence of actions and observations* $\langle a_1, o_1, \ldots, a_t, o_t \rangle$ *is defined as follows:*

1. $Filter[\epsilon](\sigma) = \sigma$;
2. $Filter[a](\sigma) = \{s' \mid \langle s, a, s' \rangle \in \mathcal{R},\ s \in \sigma\}$;
3. $Filter[o](\sigma) = \{s \in \sigma \mid o \text{ is true in } s\}$;
4. $Filter[\langle a_i, o_i, \ldots, a_t, o_t \rangle](\sigma) = Filter[\langle a_{i+1}, o_{i+1}, \ldots, a_t, o_t \rangle](Filter[o_i](Filter[a_i](\sigma)))$.

We call Step 2 progression with a *and Step 3* filtering with o.

For example, consider a robot that is in charge of cleaning a room. It can execute an action $a = fetch(broom, closet)$ which has the single effect rule "a **causes** $has(broom) \wedge \neg in(broom, closet)$ **if** $in(broom, closet)$". Assume that the robot's belief state is $\sigma = Pow(\mathcal{P})$ (i.e., it considers all states possible). Then, $Filter[a](\sigma) = \{s \in$

$Pow(\mathcal{P}) \mid \neg in(\textit{broom}, \textit{closet})$ is true in $s\}$, i.e., after performing the action a we consider all worlds that satisfy $\neg in(\textit{broom}, \textit{closet})$ possible. This is because if we are initially in a state in which $in(\textit{broom}, \textit{closet})$, then the resulting state is one in which $\neg in(\textit{broom}, \textit{closet})$, and if we are initially in a state in which $\neg in(\textit{broom}, \textit{closet})$, then we stay in the same state (thus, still satisfying $\neg in(\textit{broom}, \textit{closet})$). Call this resulting belief state σ'. Now, if an observation $o = \textit{has}(\textit{broom})$ is received, then $Filter[o](\sigma')$ is exactly the set of worlds that satisfy $\neg in(\textit{broom}, \textit{closet})$ and $\textit{has}(\textit{broom})$.

3 Filtering Logical Formulae

Approaches to filtering actions and observations that at any stage enumerate the states in a belief state do not scale to large domains. An alternative approach is to perform logical progression in a form similar to the one described by [Lin and Reiter, 1997; McIlraith, 1998]. The difference is that now we wish to do so (efficiently) in the context of nondeterministic actions and observations.

In this section we present a straightforward algorithm that filters belief state formulae directly, but does so in worst-case exponential time. This algorithm serves as a starting point for Section 4, where we propose efficient algorithms. We also present distribution properties of filtering over the logical connectives $\wedge, \vee, \neg$, and examine the theoretical limitations of formula filtering. These will guide us in Section 4, and allow us to present classes of systems that are not subject to those limitations and can be tracked in polynomial time and a compact fashion indefinitely.

3.1 Zeroth-Order Filtering Algorithm

In the rest of the paper we assume that a fixed set of fluents persists for action a in all states in which it has an effect. $\textit{Eff}(a)$ is the set of fluents possibly affected by a, and $\overline{\textit{Eff}(a)}$ is $\mathcal{P} \setminus \textit{Eff}(a)$. Some of the following notation assumes an implicit action, when this causes no confusion.

We represent a belief state σ as a logical formula φ such that a state is in σ iff it satisfies φ. The filtering of a belief state formula with an action and an observation is a formula representing the consequences of our effect rules and observation on states that satisfy our initial formula. We specify this filtering process formally using the following notation.

For a set of effect rules $r_1, \ldots, r_l$ for action a, each of the form "a **causes** F_i **if** G_i", write $F_i' = F_{i[\mathcal{P}/\mathcal{P}']}$, for $\mathcal{P}' = \{f_1', \ldots, f_n'\}$ a set of new symbols for fluents, $[\mathcal{P}/\mathcal{P}']$ a shorthand for $[f_1/f_1', ..., f_n/f_n']$, and $[f/f']$ means that we replace all instances of symbol f in the formula by instances of symbol f'. Here, we view $\mathcal{P}$ as the set of fluents in some time t, and $\mathcal{P}'$ as the same fluents in time $t+1$. We add the following set of rules for action a:

$$\begin{aligned}\mathcal{C} = \{&\text{“}a \textbf{ causes } p \textbf{ if } p\text{”}, \text{“}a \textbf{ causes } \neg p \textbf{ if } \neg p\text{”} \mid p \notin \textit{Eff}(a)\} \\ \cup\{&\text{“}a \textbf{ causes } p \textbf{ if } p \wedge \bar{G}\text{”} \mid p \in \textit{Eff}(a)\} \\ \cup\{&\text{“}a \textbf{ causes } \neg p \textbf{ if } \neg p \wedge \bar{G}\text{”} \mid p \in \textit{Eff}(a)\}\end{aligned}$$

where

$$\bar{G} = \neg G_1 \wedge ... \wedge \neg G_l, \tag{1}$$

the assertion that no precondition of a holds. This has a similar effect to adding frame axioms to a set of effect axioms in an action language. We let $r_1, ..., r_m$ be the complete set of rules for a and call the new rules, $\mathcal{C} = \{r_{l+1}, ..., r_m\}$, *completion rules for* a.

We filter a belief-state formula as follows. (We reuse $Filter[\cdot](\cdot)$ for filtering a belief-state formula.) Let $Cn(\Psi)$ be the set of logical consequences of Ψ (i.e., formulae ψ such that $\Psi \models \psi$), and $Cn^{\mathcal{L}}(\Psi)$ be $Cn(\Psi) \cap \mathcal{L}$, the set of logical consequences of Ψ in the language $\mathcal{L}$. We write $Cn^{L}(\Psi)$, when L is a *set of symbols*, to mean $Cn^{\mathcal{L}(L)}(\Psi)$.

1. $Filter[a](\varphi) = $
$$(Cn^{\mathcal{P}'}(\varphi \wedge \textstyle\bigwedge_{i \leq m}((\varphi \Rightarrow G_i) \Rightarrow F_i')))_{[\mathcal{P}'/\mathcal{P}]};$$
2. $Filter[o](\varphi) = \varphi \wedge o$.

For example, consider action $a = \textit{fetch}(\textit{broom}, \textit{closet})$ which has the single effect rule "a **causes** $\textit{has}(\textit{broom}) \wedge \neg in(\textit{broom}, \textit{closet})$ **if** $in(\textit{broom}, \textit{closet})$", and consider a belief state formula $\varphi = \textit{TRUE}$ (i.e., we consider all states possible). Then, $Filter[a](\varphi) \equiv \neg in(\textit{broom}, \textit{closet})$, i.e., after performing the action a we consider all worlds that satisfy $\neg in(\textit{broom}, \textit{closet})$ possible (similar to our example in Section 2). Call this resulting belief state formula φ'. Now, if an observation $o = \textit{has}(\textit{broom})$ is received, then $Filter[o](\varphi') \equiv \neg in(\textit{broom}, \textit{closet}) \wedge \textit{has}(\textit{broom})$.

The following generalizes a theorem presented in [Doherty *et al.*, 1998] by allowing conditional and inconsistent effects.

Theorem 3.1. *If φ is a belief state formula, and a an action, then*

$$Filter[a](\{s \in \mathcal{S} \mid s \textit{ satisfies } \varphi\}) = \{s \in \mathcal{S} \mid s \textit{ satisfies } Filter[a](\varphi)\}$$

Our zeroth-order algorithm computes $Filter[\langle a_1, o_1, ..., a_t, o_t\rangle](\varphi)$ by iterating the application of filtering of a belief-state formula with an action and an observation. It sets $\varphi_0 = \varphi$ and $\varphi_i = Filter[o_i](Filter[a_i](\varphi_{i-1}))$ recursively for $i > 0$ using the two equalities defined above. This algorithm is correct, as shown by Theorem 3.1. It can be implemented using a consequence finder in a restricted language (e.g., [Simon and del Val, 2001]).

From here on, when we say *filtering* we refer to filtering of a belief-state formula, unless otherwise mentioned.

3.2 Distribution Properties and Permutation

We can decompose the filtering of a formula φ along logical connectives once we establish several distribution properties.

Theorem 3.2 (Distribution over Connectives). *Let a be an action, and let φ, ψ be formulae. Then,*

1. $Filter[a](\varphi \vee \psi) \equiv Filter[a](\varphi) \vee Filter[a](\psi)$
2. $\models Filter[a](\varphi \wedge \psi) \Rightarrow Filter[a](\varphi) \wedge Filter[a](\psi)$
3. $\models Filter[a](\neg\varphi) \Leftarrow \neg Filter[a](\varphi) \wedge Filter[a](\textit{TRUE})$

We can say something stronger for actions that act as *permutations* on the states in $\mathcal{S}$ in which they can be executed.

Definition 3.3 (Permuting Actions). *Action a is* permuting, *if for every state s' there is* at most one *s such that $\mathcal{R}(s, a, s')$.*

Domains that include only permuting actions are called *permutation domain*.

For example, consider $a = \mathit{fetch}(\mathit{broom}, \mathit{closet})$ from above, and assume that we add a second effect rule "a **causes** $FALSE$ if $\neg in(\mathit{broom}, \mathit{closet})$". Thus, a is not executable unless its first rule's precondition holds. Then, the action is a one-to-one mapping between states, when this mapping is defined (it is not defined when a state maps to no resulting state). If this second rule is not added, then the action is not one-to-one because it maps two different states (in the first we already have the broom and in the second the broom is in the closet) to the same state.

In the same spirit, an action $\mathit{pickUp}(A, B)$ that picks up block A from the top of block B is one-to-one when it is possible because we can find a single previous state for every resulting state. The same holds for $\mathit{putDown}(A, C)$. Other natural examples include *turning a row* in a Rubik's cube, *flipping a light switch*, and *buying a gallon of gas*. In contrast, turning on the light, setting a Rubik's cube to a particular configuration, and filling up the gas tank are *not permutation actions*. Notice that we allow *different* actions to map different states to the same state (e.g., accelerating by $5\mathit{MPH}$ when driving $40\mathit{MPH}$ results in the same state as when decelerating by $5\mathit{MPH}$ when driving $50\mathit{MPH}$).

Theorem 3.4 (Distribution for Permutation Domains). *Let a be a permuting action, and let φ, ψ be formulae. Then,*

1. $Filter[a](\varphi \vee \psi) \equiv Filter[a](\varphi) \vee Filter[a](\psi)$
2. $Filter[a](\varphi \wedge \psi) \equiv Filter[a](\varphi) \wedge Filter[a](\psi)$
3. $Filter[a](\neg\varphi) \equiv \neg Filter[a](\varphi) \wedge Filter[a](\mathit{TRUE})$

Proof Sketch. Space permits inclusion only of the most interesting part, i.e., that of "$\Leftarrow$" of 2.

Let s' be a world state that satisfies $Filter[a](\varphi) \wedge Filter[a](\psi)$. Then, it satisfies both of $Filter[a](\varphi)$, $Filter[a](\psi)$. For $Filter[a](\varphi)$ there is a state s such that $R_D(s, a, s')$ and s satisfies φ. Similarly, for $Filter[a](\psi)$ there is a state s_1 such that $R_D(s_1, a, s')$ and s_1 satisfies ψ. However, since a acts as a one-to-one mapping from $\mathcal{S}$ to $\mathcal{S}$, there is only one state in $\mathcal{S}$ that a maps to s'. Thus, $s = s_1$, and s satisfies ψ. Thus, s satisfies $\varphi \wedge \psi$ and s' satisfies $Filter[a](\varphi \wedge \psi)$. From Theorem 3.1 it follows that $\models Filter[a](\varphi \wedge \psi) \Leftarrow Filter[a](\varphi) \wedge Filter[a](\psi)$. □

3.3 Limitations for Compact Representation

It may be argued that filtering may require only polynomial space, if we limit ourselves to initial belief states that are represented compactly and to actions whose effects and preconditions are represented compactly. In Theorem 3.5 we show the contrary. That is, for every general-purpose representation of belief states there is a dynamic system, an initial belief state, and a sequence of actions after which our belief state representation is exponential in the representation of the initial belief state.

P/poly is a *nonstandard* computational complexity class that includes problems that can be answered in polynomial time, if we are given a polynomial-length *hint* that depends only on the length of the input problem. It is considered likely that NP $\cap$ co-NP $\not\subseteq$ P/poly. Assume so, and let $f(\sigma)$ ($\sigma \subseteq 2^{\mathcal{S}}$) encode belief states using n propositional symbols.

Theorem 3.5. *There is dynamic system D with n fluents, belief state σ_0, and action sequence $a_1, ..., a_n$ such that, for all $i \leq n$, $\sigma_i = Filter[a_i](\sigma_{i-1})$, and $|f(\sigma_n)| > poly(|f(\sigma_0)| \cdot |D| \cdot n)$. ($|D|$ is the representation size of D.)*

The proof of this theorem reduces the problem of representing the belief state after performing an action to that of representing a *Craig Interpolant*. It uses the following.

Theorem 3.6 ([Boppana and Sipser, 1990]). *Assume that for every propositional implication $A \models B$ there is a Craig interpolant C such that $A \models C$ and $C \models B$ and $|C| \leq poly(|A| + |B|)$. Then NP $\cap$ co-NP $\subseteq$ P/poly.*

4 Efficient and Indefinitely Compact Filtering

In this section we present the main contribution of this paper, namely, a polynomial-time algorithm that computes logical filtering exactly for a significant class of transition systems. For some special cases we present simpler algorithms that are even more efficient. For systems that do not fall within this class our algorithm gives an approximation to the filtering. Also, we show that we can keep the representation of the filtered belief state compact indefinitely for a class of dynamic systems. This class includes nondeterministic STRIPS systems and some systems whose actions are permuting.

The theorems that we consider most important are the following three. Let $|\varphi|$ be the number of literals in φ's representation, let $\#rules(a)$ be the number of effect rules defining a, and let $\bar{G}_a = \bar{G}$ be the precondition of a as defined in (1). Assume that the effect rules that define a given action have identical sets of affected propositional symbols (however, different actions may have different such sets).

Theorem 4.1 (Efficiency). *Given NNF formula φ, action a, and observation o, the filtering algorithm in Figure 1 returns the filtering of φ with a and o, if a is permuting. If a is not permuting, then the algorithm returns an logically weaker formula than the filtering. It does so in time $O(|\varphi| \cdot 2^{\#rules(a)+|L(\bar{G}_a)|})$.*

Theorem 4.2 (Compact Filtering). *Given k-CNF formula φ (fixed k),* deterministic *action a, and observation o in k-CNF, the filtering of φ with a and o is in k-CNF, if a is permuting, $Filter[a](TRUE)$ is in k-CNF, and for every literal ℓ in φ, $Filter[a](\ell) \equiv Filter[a](TRUE) \wedge T$, where T is a conjunction of literals.*

If a is not permuting, then the algorithm in Figure 1 still returns a formula in k-CNF.

The conditions on action a in the last theorem hold, e.g., for actions whose every defining rule has a precondition that is a single clause (e.g., a literal). It also holds for actions which are defined by at most two rules, and actions that affect all the literals that appear in their preconditions.

Finally, for nondeterministic STRIPS actions (actions that have no conditional effects) we get efficiency and compact representation. Let k-PI-CNF denote the class of k-CNF formulae that include all their prime implicates (i.e., φ in k-PI-CNF iff φ in k-CNF and for every clause C, if $\varphi \models C$, then there is a clause C' in φ such that C' subsumes C). Every formula can be represented in k-PI-CNF for some k.

Theorem 4.3 (STRIPS Actions). *Given k-PI-CNF formula φ (fixed k),* STRIPS *action a with effect in k-PI-CNF, and* *observation o in 2-CNF, the filtering of φ with a and o is computed in time $O(|\varphi| \cdot k + 2^{|L(\bar{G}_a)|})$. The result is in k-PI-CNF, if after observing o we know if a succeeded or failed.*

4.1 Closed-Form Solution and Factored Filtering

Our zeroth-order filtering algorithm uses consequence finding tools which do not scale to large domains. The following always holds and suggests a different reasoning procedure.

$$Filter[a](\varphi) \equiv \bigwedge_{i_1,\ldots,i_u \le m,\ \varphi \models G_{i_1} \vee \ldots \vee G_{i_u}} (F_{i_1} \vee \ldots \vee F_{i_u}). \quad (2)$$

We can compute $Filter[a](\varphi)$ by testing queries of the form $\varphi \models G_{i_1} \vee \ldots \vee G_{i_u}$ (instead of applying consequence finding). On the other hand, it requires an exponential number in m of such tests (recall, m is the number of rules, including the completion rules). Since $m > 2n$ (there are two completion rules for each of the n domain features), this is worse computationally than the method of enumerating all the states.

In what follows we use the intuition that we need only few rules if φ includes only a small subset of $\mathcal{P}$, the fluent symbols of our domain. In general, φ may include many fluent symbols because we may know many things about many different parts of our domain. Nevertheless, if we can decompose φ into small parts that can be filtered separately, then each of the parts includes only a small subset of $\mathcal{P}$, and filtering each of the parts separately becomes easy.

Assume that we order the rules of a such that $r_1, ..., r_t$ ($t \le l$) satisfy $L(F_i) \cap L(G_i) = \emptyset$ ($r_i =$ "a **causes** F_i **if** G_i", $i \le t$), and $r_{t+1}, ..., r_l$ satisfy $L(F_i) \cap L(G_i) \ne \emptyset$. Furthermore, let r_{m+1} be the additional rule "a **causes** $\bar{G}$ **if** $\bar{G}$" (recall, m is the number of rules of a, including completion rules). We define $\mathcal{B}$ to be

$$\mathcal{B} = \bigwedge_{i \le t} (\neg G_i \vee F_i) \wedge \bigwedge_{i_1,\ldots,i_u \in \{t+1,\ldots,l,m+1\}} (\widetilde{G}_{i_1,\ldots,i_u} \vee \bigvee_{f \le u} F_{i_f}) \quad (3)$$

where $\widetilde{G}_{i_1,\ldots,i_u} \equiv Cn^{\overline{\mathit{Eff}(a)}}(\bigwedge_{f \le u} \neg G_{i_f})$.

$\mathcal{B}$ is a term that is always implied by $Filter[a](\mathit{TRUE})$, i.e., the progression of zero knowledge with the action a. The first set of conjuncts of $\mathcal{B}$ is the result of applying a rule "a **causes** F_i **if** G_i" whose preconditions are not affected by executing a. Even when we know nothing before performing a, we will know that either the effect occurred or the precondition did not hold and still does not hold. The second set of conjuncts applies a similar intuition for the case of effect rules that may affect the truth value of their original preconditions.

Define $\mathcal{C}(L)$ to be the set of completion rules of a for fluents in L, i.e., $\mathcal{C}(L) = \{i > l \mid \text{the effect of } r_i \in \mathcal{C} \text{ is in } L\}$.

We can now state the fundamental theorem of this Section. It holds for all domains expressed using our action language.

Theorem 4.4 (Closed-Form Representation). *If φ is a belief state formula and $\{r_1, \ldots, r_l\}$ is the set of effect rules for action a, each of the form "a **causes** F_i **if** G_i", then*

$$Filter[a](\varphi) \equiv \bigwedge_{\substack{i_1, ..., i_u \in \{1, ...l, m+1\} \cup \mathcal{C}(L(\varphi)), \\ \varphi \models G_{i_1} \vee ... \vee G_{i_u}}} (F_{i_1} \vee ... \vee F_{i_u}) \bigwedge \mathcal{B} \quad (4)$$

The intuition for equation (4) is that progressing φ with an action a can be computed by looking at all the possible combination of preconditions of effect rules and completion rules for $L(\varphi)$. If we can prove that $G_1 \vee G_2$ holds from φ, then we can conclude that $F_1 \vee F_2$ holds in the result of executing a. The conclusions that are not accounted for with this intuition are the effects that we infer from the completion rules in $\mathcal{C}(L(G_1, ..., G_l))$ together with the effect rules for a. Those conclusions are summarized in $\mathcal{B}$, which is independent of φ.

PROCEDURE NNF-Filter($\langle a_i, o_i \rangle_{0<i\le t}$,$\varphi$)
$\forall i$, a_i an action, o_i an NNF observation, φ a belief-state formula.
1. If $t = 0$, return φ.
2. Set $\mathcal{B}$ as in equation (3) for a_t but in NNF form.
3. Return $o_t \wedge \mathcal{B} \wedge$ NNF-ProgressStep(a_t, NNF-Filter($\langle a_i, o_i \rangle_{0<i\le(t-1)}$,$\varphi$)).

PROCEDURE NNF-ProgressStep(a,φ)
a an action. φ a belief-state formula, $r_1, ..., r_l, r_{m+1}$ rules for a.
1. If φ is a literal, then return the NNF form of $\bigwedge\{\bigvee_{i\in I} F_i \mid I \subseteq \{1, ..., l, m+1\} \cup \mathcal{C}(L(\varphi)),\ \varphi \models \bigvee_{i \in I} G_i\}$.
2. If $\varphi = \varphi_1 \vee \varphi_2$, then return NNF-ProgressStep(a, φ_1) $\vee$ NNF-ProgressStep(a, φ_2).
3. It must be that $\varphi = \varphi_1 \wedge \varphi_2$. Return NNF-ProgressStep(a, φ_1) $\wedge$ NNF-ProgressStep(a, φ_2).

Figure 1: Filtering an NNF formula.

Our NNF filtering algorithm is presented in Figure 1. It is much faster than our zeroth-order algorithm, and it relies on Theorems 4.4, 3.2, and 3.4. In the following, if ψ is a logically weaker formula than our filtering, we say that it is a *safe approximation* for filtering. We denote effects by F_i and preconditions by G_i. For an action a, set $t = |\bigcup_{i \le l} L(G_i)|$.

Corollary 4.5 (NNF Filtering). *Let φ be a formula in NNF with h literals, and let a be an action with l effect rules. Then, NNF-Filter safely approximates $Filter[a](\varphi)$ in time $O(h \cdot 2^{l+t})$. If a is permuting, then this computation is exact.*

Extended Example

As an example, consider our room-cleaning robot from above, and the action $a = \mathit{fetch}(\mathit{broom}, \mathit{closet})$, with the slightly more elaborate effect rule "a **causes** $\mathit{has}(\mathit{broom}) \wedge \neg \mathit{in}(\mathit{broom}, \mathit{closet})$ **if** $\mathit{in}(\mathit{broom}, \mathit{closet}) \wedge \neg \mathit{locked}(\mathit{closet})$". Assume that our robot has its belief state represented by $\neg \mathit{locked}(\mathit{closet}) \wedge (\mathit{in}(\mathit{broom}, \mathit{closet}) \vee \mathit{in}(\mathit{broom}, \mathit{shed}))$.

NNF-Filter filters each of the literals separately with a and combines the results. For a given literal, filtering is mostly done is step 1 of NNF-ProgressStep. We describe how this is done for each of the literals in our robot's belief state.

NNF-ProgressStep(a,$\neg \mathit{locked}(\mathit{closet})$) tests in step 1 all possible entailments of the form $\neg \mathit{locked}(\mathit{closet}) \models \bigvee_{i \in I} G_i$, where I includes elements from the effect rule of a, the frame rule for $\bar{G}$ (rule r_{m+1}), and the completion rule for $\neg \mathit{locked}(\mathit{closet})$. For brevity, denote the effect of a by F_1, and the precondition for that rule by G_1. There is only one effect rule defining a, so in this case $\bar{G} = \neg G_1$.

In that step, one of our tests finds that

$$\neg \mathit{locked}(\mathit{closet}) \models G_1 \vee \bar{G}$$

(because $G_1 \vee \bar{G} = G_1 \vee \neg G_1 \equiv TRUE$). This makes us include $F_1 \vee \bar{G}$ in the filtering. In the same step, another test finds that $\neg \mathit{locked}(\mathit{closet}) \models \neg \mathit{locked}(\mathit{closet})$ (trivially true). This makes us include $\neg \mathit{locked}(\mathit{closet})$ in the filtering. The conjunction of the two is logically equivalent to $\neg in(\mathit{broom}, \mathit{closet}) \wedge \neg \mathit{locked}(\mathit{closet})$. This is the result of filtering $\neg \mathit{locked}(\mathit{closet})$ with a.

Similarly, for $in(\mathit{broom}, \mathit{closet})$, one of our tests in step 1 of NNF-ProgressStep finds that

$$in(\mathit{broom}, \mathit{closet}) \models G_1 \vee (\bar{G} \wedge in(\mathit{broom}, \mathit{closet}))$$

(because $G_1 \vee (\bar{G} \wedge in(\mathit{broom}, \mathit{closet})) \equiv G_1 \vee in(\mathit{broom}, \mathit{closet})$, and the latter follows trivially from $in(\mathit{broom}, \mathit{closet})$). As before, we also find that $in(\mathit{broom}, \mathit{closet}) \models G_1 \vee \bar{G}$. We find that the filtering of $in(\mathit{broom}, \mathit{closet})$ with a is $(in(\mathit{broom}, \mathit{closet}) \vee \mathit{has}(\mathit{broom})) \wedge (\neg in(\mathit{broom}, \mathit{closet}) \vee \mathit{locked}(\mathit{closet}))$. Notice that this belief state implies $\mathit{has}(\mathit{broom}) \vee \mathit{locked}(\mathit{closet})$.

For the filtering of the last literal, $in(\mathit{broom}, \mathit{shed})$, we find its filtering to be equivalent to $in(\mathit{broom}, \mathit{shed}) \wedge (\neg in(\mathit{broom}, \mathit{closet}) \vee \mathit{locked}(\mathit{closet}))$. (Notice, that we do not know that the broom is not in the closet because initially we did not know that the broom *cannot* be in more than one place; we made this choice to simplify this example.)

Now, we recurse back in the algorithm and combine the filtered formulae. We find that the filtering of $in(\mathit{broom}, \mathit{closet}) \vee in(\mathit{broom}, \mathit{shed})$ is equivalent to $(in(\mathit{broom}, \mathit{shed}) \vee in(\mathit{broom}, \mathit{closet}) \vee \mathit{has}(\mathit{broom})) \wedge (\neg in(\mathit{broom}, \mathit{closet}) \vee \mathit{locked}(\mathit{closet}))$. Notice that this belief state implies $in(\mathit{broom}, \mathit{shed}) \vee \mathit{has}(\mathit{broom}) \vee \mathit{locked}(\mathit{closet})$.

Finally, for the original belief state, $\neg \mathit{locked}(\mathit{closet}) \wedge (in(\mathit{broom}, \mathit{closet}) \vee in(\mathit{broom}, \mathit{shed}))$, we conjoin the last result with the filtering of $\neg \mathit{locked}(\mathit{closet})$ and get

$$(in(\mathit{broom}, \mathit{shed}) \vee \mathit{has}(\mathit{broom})) \wedge \neg in(\mathit{broom}, \mathit{closet}) \wedge \neg \mathit{locked}(\mathit{closet})$$

4.2 DNF and CNF Belief States

If φ is in DNF (a disjunction of conjunctions), then we can limit the size of the resulting formula. We write $D \wedge a \models \psi$ to say that action a has effect rules in D that logically imply ψ in a state in which a is executed (e.g., $\neg \bar{G}$ may be a tautology).

Corollary 4.6 (Iterating DNF Filtering). *Let φ be in DNF with h literals and s disjuncts, and a an action with l effect rules with $\{F_i\}_{i \leq l}$ in DNF with d disjuncts total, and $\{G_i\}_{i \leq l}$ in CNF, each with c conjuncts. Then, $Filter[a](\varphi)$ in DNF is computed exactly in time $O(h \cdot 2^{l+t})$ with at most $\max(2s, 1) \cdot \binom{d}{d/2} \cdot c^l$ disjuncts. If $D \wedge a \models \neg \bar{G}$, then it has at most $\max(s, 1) \cdot \binom{d}{d/2} \cdot c^l$ disjuncts.*

Thus, when a is a *deterministic* action (every rule's effect is a conjunction of literals) with a single effect rule that is always guaranteed to succeed, then the number of disjuncts in the formula does not grow as the filtering progresses.

For CNF formulae we can find a more significant class of actions that allow us to maintain compact representation. We show that under some conditions every k-CNF formula is filtered into a k-CNF formula (fixed k). This implies that the belief state representation is no larger than $(2n)^k$, which is manageable for small fixed k's.

The main observation that we use is that a clause of k literals may give rise to a larger clause after filtering, only if one of the following holds: (a) the filtering of $TRUE$ includes a clause of more than k literals; or (b) the filtering of a literal includes a clause of 2 or more literals that is not subsumed by $Filter[a](TRUE)$. The first case can occur when we do not know whether the action succeeded or not, and which rules applied if it did. In that case, we know that after the action one of the effects holds or no precondition holds (this yields a formula which may include many disjunctions). The second case can occur when the precondition of a rule includes a conjunction of literals. When we filter a single literal we may get a disjunction in the result (of the form *the effect holds, or the rest of the precondition does not*). When we filter a clause, this may cause the filtering to include a larger clause.

The following theorem describes sufficient conditions for filtering a k-CNF formula into k-CNF, thus keeping the representation compact (k is fixed).

Theorem 4.7 (Filtering a k-CNF Clause). *Let C be a k-CNF clause, and action a have l effect rules, all deterministic. Assume that $\mathcal{B}$ is in k-CNF, and that whenever $f \models \bigvee_{i \in I} G_i$ for literal f and $I \subset \{1, ..., l, m+1\}$, $|I| \geq 2$, then $D \wedge a \models \bigvee_{i \in I} G_i$ or $f \models G_i$ for some $i \in I$. Then, $Filter[a](C)$ is in k-CNF and can be computed in time $O(2^{l+t})$.*

Note that $\mathcal{B}$ is in k-CNF for an action a when $|\bigcup_{i \leq l} L(G_i) \cup \mathit{Eff}(a)| \leq k$ or $|\bigcup_{i \leq l} L(G_i) \setminus \mathit{Eff}(a)| + l \leq k$ or $l = 1$ and $|L(G_1)| - min(1, |L(F_1) \setminus L(G_1)|) \leq k$.

Consequently, filtering with actions that are permuting maintains a k-CNF if $\models \bigvee_{i \in I} G_i$ whenever $f \models \bigvee_{i \in I} G_i$ for some $|I| \geq 2$. An example of such an action is *flipping a switch* (if the light is on, it will get turned off, and vice versa). One of the preconditions always holds. Another example is moving a block (whichever it is) from the top of a stack. For literal f, if $f \models \mathit{top}(A) \vee \mathit{top}(B)$, then the disjunction is implied by $D \wedge a$ or a single precondition follows from f.

Corollary 4.8 (Iterating CNF Filtering). *If φ is in k-CNF, then the assumptions of Theorem 4.7 imply that $Filter[a](\varphi)$ is approximated safely in time $O(|\varphi| \cdot 2^{l+t})$, with a result in k-CNF. If a is permuting, then this computation is exact.*

4.3 Prime-Implicate Belief States

It turns out that a form of distribution over conjunction holds for all actions if the belief state is represented as the conjunction of all of its *prime implicates* (formulae we call *prime implicate belief states* (PI-CNF)). In this form we can distribute the computation to the conjuncts and conjoin the result of filtering small subgroups of them separately. More precisely,

$$Filter[a](\varphi) \equiv \bigwedge_{j_1, \ldots, j_z \leq s} Filter[a](\bigwedge_{g \leq z} C_{j_g}) \tag{5}$$

for z a number that depends on the representation of the preconditions of a and on the number of rules defining a.

Theorem 4.9 (Filtering Prime Implicates). *Let φ be in k-PI-CNF (PI-CNF and k-CNF), and let action a have l effect rules with effects in k-CNF and preconditions in d-CNF, with*

PROCEDURE PI-Filter($\langle a_i, o_i\rangle_{0<i\leq t}$,$\varphi$)
$\forall i$, a_i an action, o_i an observation, φ a belief-state formula.

1. If $t = 0$, return φ.
2. Return the PI-CNF form of
   $o_t \wedge$ PI-ProgressStep(a_t, PI-Filter($\langle a_i, o_i\rangle_{0<i\leq(t-1)}$,$\varphi$)).

PROCEDURE PI-ProgressStep(a,φ)
a an action, φ a belief-state formula. $r_1, ..., r_l, r_{m+1}$ rules for a.

1. Let $res \leftarrow$ *TRUE*.
2. For every $i_1, ..., i_j \in \{1, ..., l, m+1\}$, and $f_1, ..., f_u$ literals in $Pre(a) \setminus \mathit{Eff}(a)$, do
   (a) Let $\bigwedge_{g\leq z} \hat{C}_g$ be the CNF representation of $\bigvee_{h\leq j} G_{i_h}$.
   (b) For every $g \leq z$, nondeterministically choose a clause C_g in φ whose restriction to $Pre(a)$ subsumes $\hat{C}_g$.
   (c) If there is a clause for every $g \leq z$, then set $res \leftarrow res \wedge F$, where F is $\bigvee_{h\leq j}(F_{i_h} \vee (C_{i_h} \cap \overline{\mathit{Eff}(a)}))$.
3. Return res.

Figure 2: Filtering a Prime-Implicate CNF formula.

each G_i of at most c clauses. Then, equation (5) holds for $z = c^l \cdot (d^c + 1)$, and PI-Filter (Figure 2) computes $Filter[a](\varphi)$ exactly in time $O(2^{l\cdot|Pre(a)\cup \mathit{Eff}(a)|} \cdot (s^z + z))$. If $D \wedge a \models \neg\bar{G}$, then $z = c^l$, with $O(2^l \cdot (s^z + z))$ time.

Corollary 4.10 (Maintaining k-PI-CNF). *Let φ be a k-PI-CNF formula, and let action a have l effect rules, all deterministic. Assume that G_i is a disjunction of literals, for all $i \leq l$. Also, assume that $D \wedge a \models \neg\bar{G}$. Then, $Filter[a](\varphi) \equiv$ PI-ProgressStep(a,φ), and the latter is in k-CNF. If $k = 2$, then PI-Filter(a,φ) is in k-PI-CNF.*

The conclusion for $k = 2$ uses the fact that every prime implicate of a formula in 2-CNF is a clause with at most two literals. Simple counter examples (omitted) show that k-PI-CNF cannot be maintained without these conditions.

4.4 Nondeterministic STRIPS Domains

STRIPS domains present a special case of the results that we discussed above. In such domains every action has a single rule (no conditional effects) and actions can be executed only when their preconditions hold[1]. Unlike the original STRIPS, we allow nondeterministic effects, and allow belief states to be any CNF formulae in the fluents of the domain.

More precisely, every action a has exactly two effect rules, r_1, r_2. Their preconditions are such that $G_1 \equiv \neg G_2$. Also, $F_2 \equiv FALSE$. Thus, a can be executed only when G_1 holds. Consequently, when we filter with a we assert implicitly that its preconditions held in the last world state.

The assumption that there is only one rule that determines a's effects and otherwise the action is not executed has a dramatic effect. For $l_1, ..., l_k$ literals we get that

$$Filter[a](l_1 \vee ... \vee l_k) \equiv \begin{cases} T_a & \exists i \leq k\ l_i \in \mathcal{L}(\mathit{Eff}(a)) \\ T_a \wedge \bigvee_{i\leq k} l_i & l_1, ..., l_k \notin \mathcal{L}(\mathit{Eff}(a)) \end{cases} \quad (6)$$

[1]With the alternate assumption that actions have no effect unless their preconditions hold, but observations (or their absence) are guaranteed to distinguish actions' success from failure, the same results hold, except that now the compactness of representation remains only after filtering with observations.

Theorem 4.11 (Iterating STRIPS Filtering: CNF). *Let φ be in k-CNF with s clauses and a a STRIPS action. If F_1 is in k-CNF and $|L(G_1) \setminus L(F_1)| \leq t$, for some $t \leq k$, then $Filter[a](\varphi)$ can be approximated safely in time $O(s \cdot k + 2^t)$, yielding a k-CNF formula. If a is permuting, then this computation is exact.*

If our representation is PI-CNF, then we get an even stronger result, leading to the algorithm in Figure 3.

Theorem 4.12 (Factoring STRIPS filtering: PI-CNF). *Let $\bigwedge_{i\leq s} C_i$ be in PI-CNF, and let a be a STRIPS action. Then,*

$$Filter[a](\bigwedge_{i\leq s} C_i) \equiv \bigwedge_{i\leq s} Filter[a](C_i).$$

For example, for $a = \mathit{fetch}(\mathit{broom}, \mathit{closet})$ that has effect $F_1 = \mathit{has}(\mathit{broom}) \wedge \neg \mathit{in}(\mathit{broom}, \mathit{closet})$, if we know $(in(broom, closet) \vee \neg locked(closet)) \wedge (in(broom, shed) \vee locked(closet))$ before applying a, then after it we know $F_1 \wedge (in(broom, shed) \vee locked(closet))$.

Corollary 4.13 (Iterating STRIPS filtering: k-PI-CNF). *Let φ be in k-PI-CNF, and let a be a STRIPS action with F_1 in k-PI-CNF, $t = |L(G_1) \setminus L(F_1)|$, and $t \leq k$. Then, STRIPS-Filter(a,φ) computes $Filter[a](\varphi)$ exactly in time $O(|\varphi| \cdot k + 2^t)$, yielding a k-PI-CNF formula.*

This means that we can filter in practice any prime implicate belief state in any nondeterministic STRIPS domain. This filtering stays compact, with the size depending only on the PI-CNF representation of F_1 and the number of propositional symbols in G_1 but not F_1.

Finally, every STRIPS action can be filtered efficiently, even if we drop the assumption that either the action succeeds or we observe an error. This can be done by assuming that the action succeeds, finding the filtering of that action, and then disjoining the result with the initial belief state. Nonetheless, this scheme may cause representation space explosion as filtering proceeds over multiple steps, unless some care is taken.

PROCEDURE STRIPS-Filter($\langle a_i, o_i\rangle_{0<i\leq t}$,$\varphi$)
$\forall i$, a_i an action, o_i an observation, φ a belief-state formula.

1. For i from 1 to t do,
   (a) Set $\varphi' \leftarrow \bigwedge_{C\in\varphi} Filter[a_i](C)$, where $Filter[a_i](C)$ is computed using (6), and eliminate subsumed clauses.
   (b) Let φ be the prime implicates of $\varphi' \wedge o_i$.
2. Return φ.

Figure 3: Filtering a PI-CNF formula with STRIPS actions.

We tested our STRIPS-filter algorithm in partially observable blocks-world domains. The implementation in LISP includes a random action-sequence and observation-sequence generator, and both the generator and filtering algorithm receive a description of the domain, actions and observations specified in PDDL (a plan-problem specification language).

We ran the algorithm with blocks-world domains of increasing sizes (3 to 20 blocks), yielding domains that range from tens to over a thousand of propositional features. We collected the time taken per filtering step for each of the executions and the space taken overall at every iteration, starting with zero knowledge. The results are shown in Figure 4.

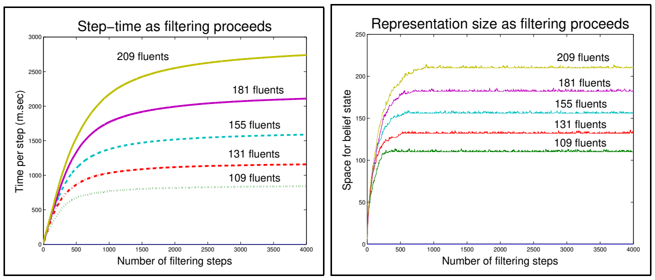

Figure 4: STRIPS-Filter in the blocks world.

4.5 Observation Model

An observation model is a theory $\mathcal{O}$ that includes *observation constraints*, axioms that describe the relationship between observed facts and other fluents. We allow $\mathcal{O}$ to include any axioms. Our model assumes that observations o are collected after an action is executed. o is a sentence made up with fluents, similar to the observation constraints. The conjunction, $\mathcal{O} \wedge o$ is then used to filter the resulting belief state. Formally,

$$Filter[o](\varphi) = \varphi \wedge o \wedge \mathcal{O}. \tag{7}$$

This allows all the results that we enumerated above to apply with this model as well. The following connects the results of Sections 3,4 to filtering with observations.

Corollary 4.14. *If $o \wedge \mathcal{O}$ is in k-CNF and $Filter[a](\varphi)$ is in k-CNF, then $Filter[a, o](\varphi)$ is in k-CNF.*

In particular, this means that our results for STRIPS domains and for NNF, CNF and DNF formulae still hold here. Thus, the representation remains compact in the cases that we indicated already, and computation remains easy in the same cases as well. Finally, for k-PI-CNF, we get the following.

Corollary 4.15 (Obs. and k-PI-CNF). *If $o \wedge \mathcal{O}$ is in 2-CNF, and φ is in k-PI-CNF, then $Filter[o](\varphi)$ is in k-PI-CNF.*

5 Conclusions

In this paper we presented the task of logical filtering and gave it a computational treatment. The results we obtained here have implications for monitoring and controlling dynamic systems. In many cases we present a closed-form computation of the filtering and in others show how to approximate this computation. In some cases we can guarantee that the size of the representation of the filtered formula can be bounded and kept small. In those cases, logical filtering can be used to control processes that run over very long periods of time. Examples of such systems are abundant and include robot motion control, natural language processing, and agents that explore partially observed worlds.

We made use of several assumptions in this paper in different contexts and with different consequences. We presented permutation domains and the certainty of existence of an effect ($D \wedge a \models \neg\bar{G}$) as characteristics of the domain that make filtering easier. We showed that the commonly used assumption that every action has a relatively small number of rules (at most polynomial in n), and that effects, preconditions and terms in the belief state typically use a small vocabulary, all have a drastic effect on the computational effort needed for filtering and on the size of the resulting belief state.

The need to track the state of the world is a basic one, and many works have appealed to it implicitly in the past. However, the computational treatment of such tracking has been avoided so far, partially due to the absence of a developed theory of nondeterministic domains, and partially due to negative results about the general cases of this task. Nonetheless, this problem and methods for its solution have received much attention in control theory. The results we obtained here promise to find their application in this domain and may be combined with stochastic filtering techniques.

Acknowledgments

This research was supported by ONR MURI Fund N00014-01-1-0890, ONR MURI Fund N00014-00-1-0637, and NSF grant ECS-9873474. The first author thanks Xuanlong Nguyen for a stimulating discussion on Theorem 3.4.

References

[Boppana and Sipser, 1990] R. Boppana and M. Sipser. The complexity of finite functions. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 757–804. Elsevier, 1990.

[Cimatti and Roveri, 2000] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. *JAIR*, 13:305–338, 2000.

[Doherty *et al.*, 1998] P. Doherty, W. Lukaszewicz, and E. Madalinska-Bugaj. The PMA and relativizing change for action update. In *Proc. KR '98*, pages 258–269, 1998.

[Ferraris and Giunchiglia, 2000] P. Ferraris and E. Giunchiglia. Planning as satisfiability in nondeterministic domains. In *Proc. AAAI '00*, pages 748–753, 2000.

[Fikes *et al.*, 1972] R. Fikes, P. Hart, and N. Nilsson. Learning and executing generalized robot plans. *AIJ*, 3:251–288, 1972.

[Kalman, 1960] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. of ASME – J. of Basic Engineering*, 82(Ser. D):35–45, 1960.

[Kautz *et al.*, 1996] H. Kautz, D. McAllester, and B. Selman. Encoding plans in propositional logic. In *Proc. KR '96*, 1996.

[Liberatore, 1997] P. Liberatore. The complexity of the language A. *ETAI*, 1:13–38, 1997.

[Lin and Reiter, 1997] F. Lin and R. Reiter. How to progress a database. *AIJ*, 92:131–167, 1997.

[McIlraith, 1998] S. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proc. KR '98*, pages 167–177, 1998.

[Reiter, 2001] R. Reiter. *Knowledge In Action*. MIT Press, 2001.

[Sandewall, 1994] E. Sandewall. *Features and Fluents*. Oxford, 1994.

[Simon and del Val, 2001] L. Simon and A. del Val. Efficient consequence-finding. In *IJCAI '01*, pages 359–365, 2001.

[Son and Baral, 2001] T. C. Son and C. Baral. Formalizing sensing actions: A transition function based approach. *AIJ*, 125, 2001.

[Williams and Nayak, 1996] B. Williams and P. Nayak. A model-based approach to reactive self-configuring systems. In *Proc. AAAI '96*, pages 971–978, 1996.

[Winslett, 1990] M. Winslett. *Updating Logical Databases*. Cambridge U. Press, 1990.