Logical Filtering

Eyal Amir and Stuart Russell
Computer Science Division, University of California at Berkeley
{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. The properties of domains that we identify can be used to launch further investigation into efficient projection, execution monitoring, planning, diagnosis, real-time control and reinforcement learning in partially observable domains.

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., [Levesque *et al.*, 1997; 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 with filtering 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 hard, because there are exponentially many possible states. We identify several classes of models that allow efficient filtering. 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 with certain effects. 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; Baral *et al.*, 2000; Liberatore, 1997; Amir, 2002]. As we show in Section 3.3, in general it is not possible to maintain a compact-sized belief state representation even when we start with an compact-sized representation of an initial belief state.

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; Doucet *et al.*, 2000].

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 last 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]. This operator is simple and allows us to examine computational properties easily. It can represent any 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. The following semantics describes the way a state changes after an action:

- If, before the execution of an action a, the state formula G is true, and the domain description contains a rule "a **causes** F **if** G", then this rule is *activated*, and after the execution of action a, F becomes true.
- If for some fluent f no activated effect rule includes the fluent f in its head, this means that the execution of action a does not influence the truth value of this fluent. Therefore, f is true in the resulting state if and only if it was true in the old state.
- If an action a has a set of rules with a combined inconsistent effect F (e.g., $F = FALSE$) and that set of rules is activated in s, then there is no state that is the result of a in s (we take this to mean that a is not executable in s).
- The result of applying an action a for which no rule is activated in s is the state s (we consider the action as possible in this state, but having no impact).

The last two principles ensure that the conditions of the rules act as conditions for the action's effects (if no conditions are met, then there are no effects), and an action is not executable iff it leads to contradictory effects (e.g., if we include a rule saying that "a **causes** FALSE **if** G"). In the latter case for s and a, there is no transition tuple $\langle s, a, s' \rangle$ in $\mathcal{R}$.

Formally, 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 $\mathcal{R}_D$.

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 (e.g., $holdA \vee holdB$ is a possible observation).

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.

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]. The difference is that now we wish to do so in the context of nondeterministic actions and observations.

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. *Eff*(a) is the set of fluents possibly affected by a, and $\overline{Eff}(a)$ is $\mathcal{P} \setminus Eff(a)$. 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:

$$\begin{array}{l} \mathcal{C} = \{\text{"}a \textbf{ causes } p \textbf{ if } p\text{"}, \text{"}a \textbf{ causes } \neg p \textbf{ if } \neg p\text{"} \mid p \notin Eff(a)\} \\ \quad \cup \{\text{"}a \textbf{ causes } p \textbf{ if } p \wedge \bar{G}\text{"} \mid p \in Eff(a)\} \\ \quad \cup \{\text{"}a \textbf{ causes } \neg p \textbf{ if } \neg p \wedge \bar{G}\text{"} \mid p \in Eff(a)\} \end{array}$$

where $\bar{G} = \neg G_1 \wedge ... \wedge \neg G_l$, 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. Finally, $Cn(\Psi)$ is the set of logical consequences of Ψ (i.e., formulae ψ such that $\Psi \models \psi$), and $Cn^{\mathcal{L}}(\Psi)$ is $Cn(\Psi) \cap \mathcal{L}$, the set of logical consequences of Ψ in the language $\mathcal{L}$. For a set of symbols L we sometimes write $Cn^{L}(\Psi)$ for $C^{\mathcal{L}(L)}(\Psi)$.

We filter a belief-state formulae as follows. (We reuse $Filter[\cdot](\cdot)$ for filtering a belief-state formula.)

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

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 uses Theorem 3.1 to compute $Filter[\langle a_1, o_1, ..., a_t, o_t \rangle](\varphi)$. It iterates 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, such as those based on ordered resolution (e.g., [Simon and del Val, 2001]).

One way to make the belief state representation of the resulting state more succinct is to represent the new belief state using a conjunction of prime implicates of $Filter[a](\varphi)$ instead of all the consequences in that language. This can be done by known consequence finders, such as [Baumgartner *et al.*, 1997; Iwanuma *et al.*, 2000; Simon and del Val, 2001].

From here forth, 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. for filtering.

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. For such action a, for every state s there is *at most one* s' such that $\mathcal{R}(s, a, s')$. We call actions that satisfy this requirement *permuting actions*, and the domain is a *permutation domain*.

For example, $pickUp(A, B)$ is an action that picks up A from B. It is executable only when A is clear, is on B, and the hand is empty. If successful, it has the effect that A is not clear and not on B, and the hand is holding A. It is one-to-one when it is possible because we can find a single previous state for every resulting state. The same holds for $putDown(A, C)$. Other natural examples include *turning a row* in a Rubik's cube, *flipping a light switch,* and *purchasing coffee*. Notice that we allow different actions to map different states to the same state (e.g., accelerating by 5*MPH* when driving 40*MPH* results in the same state as when decelerating by 5*MPH* when driving 50*MPH*).

Theorem 3.3 (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. $Filter[a](\neg\varphi) \equiv \neg Filter[a](\varphi) \wedge Filter[a](\mathit{TRUE})$

3.3 Limitations for Compact Representation

Every belief state can be represented using a DNF formula of size at most 2^n symbols, if n is the number of propositional fluents in a state. Also, A simple combinatorial argument shows that there are belief states of n fluents that cannot be described using formulae of size $o(2^n)$, i.e., strictly smaller than $O(2^n)$. Nevertheless, some suggest that it may be possible to create representations with which filtering requires only polynomial space, if we limit ourselves to initial belief states that are represented compactly. In Theorem 3.4 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 3.4 *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 Theorem 3.4 uses the following.

Theorem 3.5 ([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 Filtering NNF Belief States

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.

$$Filter[a](\varphi) \equiv \bigwedge_{i_1,\ldots,i_u \leq m,\ \varphi \models G_{i_1} \vee \ldots \vee G_{i_u}} (F_{i_1} \vee \ldots \vee F_{i_u}). \quad (1)$$

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. Since $m > 2n$ (recall, m is the number of rules, including the completion rules), 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 $r_1, ..., r_t$ ($t \leq l$) satisfy $L(F_i) \cap L(G_i) = \emptyset$, and $r_{t+1}, ..., r_l$ satisfy $L(F_i) \cap L(G_i) \neq \emptyset$. Furthermore, let r_{m+1} be the additional rule "a **causes** $\bar{G}$ **if** $\bar{G}$". We define $\mathcal{B}$ to be

$$\mathcal{B} = \bigwedge_{i \leq t} (\neg G_i \vee F_i) \wedge \bigwedge_{i_1,\ldots,i_u \in \{t+1,\ldots,l,m+1\}} (\widetilde{G}_{i_1,\ldots,i_u} \vee \bigvee_{f \leq u} F_{i_f}) \quad (2)$$

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

$\mathcal{B}$ 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. Even when we know nothing before performing a, we will know that either the effect occurred or the precondition did not hold and some fraction of it still does not hold (this fraction may be empty).

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 main theorem of this Section. It holds for all domains expressed using our action language.

Theorem 4.1 *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*

$$Filter[a](\varphi) \equiv \bigwedge_{\substack{i_1,\ldots,i_u \in \{1,\ldots l,\, m+1\} \cup \mathcal{C}(L(\varphi)),\\ \varphi \models G_{i_1} \vee \ldots \vee G_{i_u}}} (F_{i_1} \vee \ldots \vee F_{i_u}) \bigwedge \mathcal{B} \quad (3)$$

The intuition for equation (3) 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($\langle a_i, o_i \rangle_{0<i\leq t}$,$\varphi$)
$\forall i$, 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 (2) for a_t but in NNF form.
3. Return $o_t \wedge \mathcal{B} \wedge$ NNF-ProgressStep(a_t, NNF-Filter($\langle a_i, o_i \rangle_{0<i\leq(t-1)}$,$\varphi$)).

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 4.1, 3.2, and 3.3. 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 $t = |\bigcup_{i\leq l} L(G_i)|$.

Corollary 4.2 (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 $O(h \cdot 2^l \cdot 2^t)$. If a is permuting, then this computation is exact.*

As an example, consider a blocks-world belief state represented by *clear*$(A) \wedge$ *handEmpty* $\wedge$ $(on(A, B) \vee on(A, C))$, and consider action *pickUp*(A, B) with $G_1 =$ *handEmpty* $\wedge$ *clear*$(A) \wedge on(A, B)$, and $F_1 = \neg$*clear*$(A) \wedge \neg on(A, B) \wedge \neg$*handEmpty* $\wedge$ *holding*(A). NNF-Filter filters each of the conjuncts separately with a and conjoins the results (similarly for disjunction). In step 1 of NNF-ProgressStep(a,*clear*(A)) it finds that *clear*$(A) \models G_1 \vee (\neg G \wedge$ *clear*$(A))$, and includes $F_1 \vee$ *clear*(A) in the result. Similarly, $Filter[a]$(*handEmpty*$) \models F_1 \vee$ *handEmpty*, and $Filter[a](on(A, B) \vee on(A, C)) \models F_1 \vee on(A, B) \vee on(A, C)$. The conjunction of these sentences implies $F_1 \vee$ (*clear*$(A) \wedge$ *handEmpty* $\wedge (on(A, B) \vee on(A, C)))$, which implies $F_1 \vee$ (*clear*$(A) \wedge$ *handEmpty* $\wedge on(A, C))$, which represents the new belief state. Note that $\mathcal{B} \equiv$ *TRUE* because a's only effect rule satisfies $L(G_1) \subset L(F_1)$.

4.1 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 4.3 (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 $O(h \cdot 2^l \cdot 2^t)$ 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 $D \wedge a \models \neg\bar{G}$ and a is *deterministic* (every rule's effect is a conjunction of literals) with a single effect rule, then the number of disjuncts in the formula does not grow as the filtering progresses.

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

Theorem 4.4 (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 if $f \models \bigvee_{i\in I} G_i$ 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 $O(2^l \cdot 2^t)$.*

Note that $\mathcal{B}$ is in k-CNF for an action a when $|\bigcup_{i\leq l} L(G_i) \cup$ *Eff*$(a)| \leq k$ or $|\bigcup_{i\leq l} L(G_i) \setminus$ *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: if A is on top, A is moved. If B is on top, then B is moved, etc. For literal f, if $f \models$ *top*$(A) \vee$ *top*(B), then the disjunction is implied by $D \wedge a$ or a single precondition follows from f.

Corollary 4.5 (Iterating CNF Filtering) *If φ is in k-CNF, then the assumptions of Theorem 4.4 imply that $Filter[a](\varphi)$ is approximated safely in time $O(|\varphi| \cdot 2^l \cdot 2^t)$, with a result in k-CNF. If a is permuting, then this computation is exact.*

4.2 Prime-Implicate Belief States

It turns out that not only *permutation domains* allow filtering with distribution over conjunctions. Surprisingly, if our belief state is represented as the conjunction of all of its *prime implicates* (formulae we call *prime implicate belief states* (PI-CNF)), then we can distribute the computation to the conjuncts and conjoin the result of filtering small subgroups of them separately. More precisely,

$$Filter[a](\varphi) \equiv \bigwedge_{j_1,...,j_z\leq s} Filter[a](\bigwedge_{g\leq z} C_{j_g}) \qquad (4)$$

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

Theorem 4.6 (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 each G_i of at most c clauses. Then, equation (4) holds for $z = c^l \cdot (d^c + 1)$, and PI-Filter (Figure 2) computes*

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 $Pre(a) \setminus$ *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 $\bigvee_{h\leq j}(F_{i_h} \vee (C_{i_h} \cap \overline{\textit{Eff}(a)}))$.
3. Return res.

Figure 2: Filtering a Prime-Implicate CNF formula.

$Filter[a](\varphi)$ exactly in time $O(2^{l\cdot|Pre(a)\cup \textit{Eff}(a)|} \cdot (s^z + z))$. If $D \wedge a \models \neg\bar{G}$, then $z = c^l$, with $O(2^l \cdot (s^z + z))$ time.

Corollary 4.7 (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.

Unfortunately, a simple counter example shows that k-PI-CNF cannot be maintained when filtering with an action, unless we make very restricting assumptions about the form of the preconditions of effect rules (e.g., they use only one literal): Consider the clauses $(p_1 \vee q_1)$ and $(p_2 \vee q_2)$ and the sole rule precondition $G_1 = q_1 \wedge q_2$. We can prove $(p_1 \vee p_2 \vee (q_1 \wedge q_2))$ from the two clauses, but nothing that does not include both p_1, p_2 and still includes G_1. This forces us to have the consequence $p_1 \vee p_2 \vee F_1$ in the filtered formula, a formula that breaks into the conjunction of clauses longer (three literals) than the original two clauses.

Also, for $k > 2$ there is a counter example for the filtering being in k-PI-CNF. Consider the clauses $(r_1 \vee p_1 \vee q_1)$, $(r_2 \vee p_2 \vee q_2)$, and two rules with preconditions $G_1 = q_1$, $G_2 = q_2$, and effects $F_1 = q_1 \wedge q_2$, $F_2 = \neg q_1 \wedge \neg q_2$. The filtering does not include the clause $(r_1 \vee r_2 \vee p_1 \vee p_2)$, which is implied by the filtered formulae $(r_1 \vee p_1 \vee q_2)$ and $(r_2 \vee p_2 \vee \neg q_2)$. Thus, the filtering of the conjunction of those clauses can be in k-CNF only if we give up some of the prime implicates in the representation (which are implied by the rest of the clauses).

4.3 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

$$Filter[a](l_1 \vee ... \vee l_k) \equiv \begin{cases} T_a & \exists i \leq k\ l_i \in \mathcal{L}(\textit{Eff}(a)) \\ T_a \wedge \bigvee_{i\leq k} l_i & l_1, ..., l_k \notin \mathcal{L}(\textit{Eff}(a)) \end{cases} \quad (5)$$

Theorem 4.8 (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.*

As a consequence of this corollary we get that we can maintain a compact representation for STRIPS domains that satisfy the conditions of the corollary. These include a wide variety of domains. Practically all STRIPS domains used in planning today exhibit these properties, i.e., actions that have limited effects and preconditions, relatively speaking. In particular, when every action has no more than k fluents in its preconditions, and every effect has no more than k nondeterministically chosen effects (e.g., all traditional STRIPS domains are deterministic, thus satisfying this requirement), then the belief state can be kept in k-CNF, thus having no more than $(2n)^k$ clauses (in fact, no more than $\binom{n}{k} \cdot 2^k$ clauses). If k is small for a certain domain (e.g., up to 4), then this is an important guarantee on the computational feasibility of performing filtering for this domain.

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

Theorem 4.9 (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 $a = \textit{pickUp}(A, B)$ that has $G_1 = on(A, B) \wedge \textit{clear}(A)$, and $F_1 = \textit{inHand}(A) \wedge \neg on(A, B) \wedge \neg\textit{clear}(A) \wedge \textit{clear}(B)$, if we know $(\textit{clear}(B) \vee \textit{clear}(C)) \wedge (\textit{clear}(C) \vee \textit{clear}(D))$ before applying a, then after it we know $F_1 \wedge (\textit{clear}(C) \vee \textit{clear}(D))$.

Corollary 4.10 (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.*

[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.

This means that we can filter in practice any prime implicate belief state in any nondeterministic STRIPS domain, regardless of whether the domain is permuting or not. 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. An interesting special case of the last corollary is when the belief state is represented as a formula in 2-CNF. Such belief states can be filtered in appropriate STRIPS domains (not necessarily permuting) with every action, producing a 2-PI-CNF belief state.

We have implemented and tested our STRIPS-filter algorithm in blocks-world domains specified in pddl. The results are shown in Figure 4. They show that the time taken per step converges after about 200–2000 steps, regardless of the number of propositional fluents in the domain. The growth of the time taken per step is roughly quadratic, a figure which is explained by our relatively inefficient implementation (e.g., no indexing).

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 (5), 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.

4.4 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,

$$Filter[o](\varphi) = \varphi \wedge o \wedge \mathcal{O}. \quad (6)$$

This allows all the results that we enumerated above to apply with this model as well. In particular, it is always very simple to filter the belief state with an observation (by conjoining the observation and observation model). When we want to maintain a particular property of the belief state, we need to perform some more work. For example, this is the case when we want to maintain a belief state in k-CNF for some predetermined k, or when we want the belief state to be represented using all its prime implicates. In those cases we need to find all the new prime implicates and remove subsumed clauses from the resulting PI-CNF.

The following connects the results of Sections 3,4 to filtering with observations.

Corollary 4.11 *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.

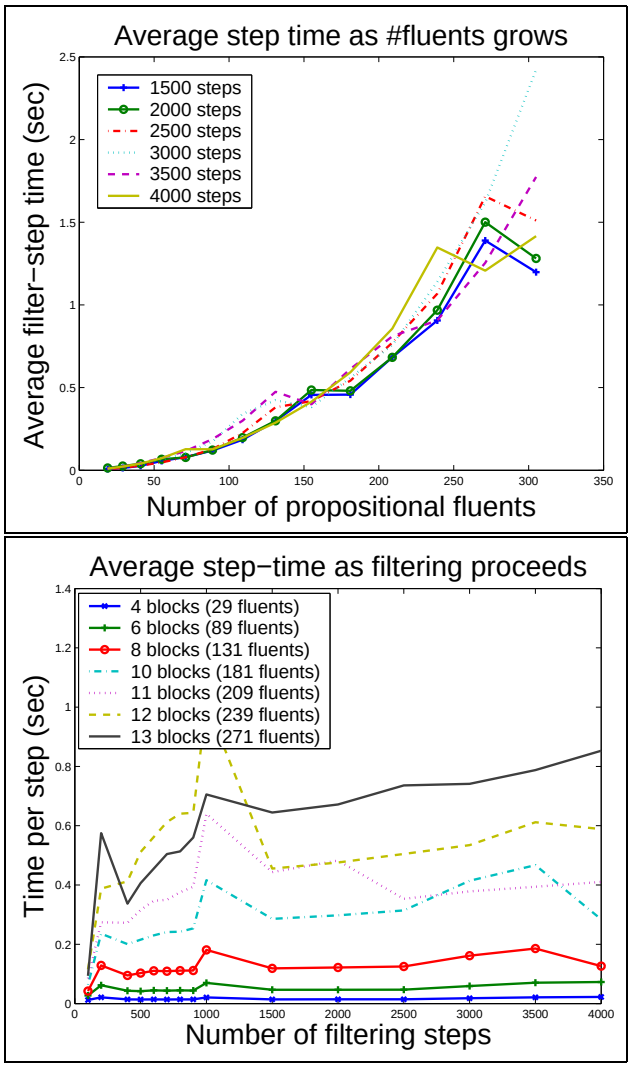

Figure 4: STRIPS-Filter in the blocks world.

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 4.12 (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 their world, such as mobile robots, adventure-game players, Internet crawlers and spacecrafts.

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 ($D \wedge a \models \neg \tilde{G}$) 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.

6 Acknowledgments

This research was supported by funds from ONR MURI Fund N00014-01-1-0890, ONR MURI Fund N00014-00-1-0637, and NSF grant ECS-9873474. The first author also wishes to thank Xuanlong Nguyen for a stimulating discussion on Theorem 3.3.

References

[Amir, 2002] Eyal Amir. Planning with nondeterministic actions and sensing. Technical report, AAAI'02 workshop on Cognitive Robotics, 2002. http://www.cs.berkeley.edu/~eyal/.

[Baral *et al.*, 2000] Chitta Baral, Vladik Kreinovich, and Raul Trejo. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122(1-2):241–267, 2000.

[Baumgartner *et al.*, 1997] Peter Baumgartner, U. Furbach, and F. Stolzenburg. Computing answers with model elimination. *Artificial Intelligence*, 90(1-2):135–176, 1997.

[Boppana and Sipser, 1990] Ravi B. Boppana and Michael Sipser. The complexity of finite functions. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 1: Algorithms and Complexity, pages 757–804. Elsevier/MIT Press, 1990.

[Cimatti and Roveri, 2000] Alessandro Cimatti and Marco Roveri. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research*, 13:305–338, 2000.

[Doherty *et al.*, 1998] Patrick Doherty, Witold Lukaszewicz, and Ewa Madalinska-Bugaj. The PMA and relativizing change for action update. In *Principles of Knowledge Representation and Reasoning: Proc. Sixth Int'l Conference (KR '98)*, pages 258–269. Morgan Kaufmann, 1998.

[Doucet *et al.*, 2000] Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proc. Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI '00)*, pages 176–183. Morgan Kaufmann, 2000.

[Ferraris and Giunchiglia, 2000] Paolo Ferraris and Enrico Giunchiglia. Planning as satisfiability in nondeterministic domains. In *Proc. National Conference on Artificial Intelligence (AAAI '00)*, pages 748–753. AAAI Press, 2000.

[Fikes *et al.*, 1972] Richard Fikes, Peter Hart, and Nils Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Iwanuma *et al.*, 2000] Koji Iwanuma, Katsumi Inoue, and Ken Satoh. Completeness of pruning methods for consequence finding procedure sol. In *Proceedings of the Third International Workshop on First-Order Theorem Proving (FTP'2000)*, pages 89–100, 2000.

[Kalman, 1960] Emil Kalman, Rudolph. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[Kautz *et al.*, 1996] Henry Kautz, David McAllester, and Bart Selman. Encoding plans in propositional logic. In J. Doyle, editor, *Proceedings of KR'96*, pages 374–384, Cambridge, Massachusetts, November 1996. KR, Morgan Kaufmann.

[Levesque *et al.*, 1997] H.J. Levesque, R. Reiter, Y. Lesprance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.

[Liberatore, 1997] Paolo Liberatore. The complexity of the language A. *Electronic Transactions on Artificial Intelligence (http://www.etaij.org)*, 1(1-3):13–38, 1997. http://www.ep.liu.se/ej/etai/1997/002/.

[Lin and Reiter, 1997] Fangzhen Lin and Ray Reiter. How to Progress a Database. *Artificial Intelligence*, 92(1-2):131–167, 1997.

[Reiter, 2001] Raymod Reiter. *Knowledge In Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press, 2001.

[Sandewall, 1994] Erik Sandewall. *Features and Fluents*. Oxford University Press, 1994.

[Simon and del Val, 2001] Laurent Simon and Alvaro del Val. Efficient consequence-finding. In *Proc. Seventeenth International Joint Conference on Artificial Intelligence (IJCAI '01)*, pages 359–365. Morgan Kaufmann, 2001.

[Son and Baral, 2001] Tran Cao Son and Chitta Baral. Formalizing sensing actions a transition function based approach. *Artificial Intelligence*, 125(1–2):19–91, 2001.

[Winslett, 1990] Mary-Anne Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.