Temporal Logical Filtering – Preliminary Results

****** DRAFT October 16, 2002 ******

Eyal Amir
Computer Science Division
University of California at Berkeley
eyal@cs.berkeley.edu

Stuart Russell
Computer Science Division
University of California at Berkeley
russell@cs.berkeley.edu

Abstract

In this paper we provide algorithms for determining the *belief state* of an agent, i.e., its knowledge about the state of the world. We describe our domains using a logical action language that allows nondeterministic actions and partial observability. Our algorithms update an initial belief state with every execution of an action and when collecting observations. This iterative updating process is called *logical filtering*.

Our algorithms are computationally superior to current methods that are used in nondeterministic planning. Several classes of dynamic systems that we identify allow particularly efficient filtering. In some cases our algorithms compute the filtering of action/observation sequences of arbitrary length efficiently while maintaining compact representation of belief states over time. Other cases allow efficient approximation of the belief state under reasonable conditions. Some of the properties of domains that we identify can be used to launch further investigation into efficient projection, execution monitoring, planning and diagnosis.

1 Introduction

An agent acting in the world must find answers to questions about the state of the world after it performed some actions and made some observations. For example, a block-stacking robot needs to determine the state of the world after picking up block A. It can use such knowledge in planning its actions, determining if an action succeeded, and choosing among corrective actions if it did not.

There are several ways to answer questions about the state of the robot after it did some actions and made some observations. One approach is to use automated reasoning techniques directly on a representation of the sequence of actions (e.g., [Kautz, McAllester, & Selman, 1996]). Another approach is to use regression of the question to a query on the initial state (e.g., [Levesque *et al.*, 1997]). Finally, we can maintain a representation of the robot's *belief state* (its knowledge about the state of the world) and update this representation with actions and observations. This latter approach is called *filtering*, and it is particularly attractive with long sequences of actions and observations (e.g., [Doucet *et al.*, 2000]).

The two computational difficulties involved with filtering are the time needed to update the belief state and the space required to represent it. Many of the early filtering approaches (e.g., Wiener [Wiener, 1949] and even Gauss) have tackled this problem in the context of stochastic processes, culminating in the the *Kalman filter* [Kalman, 1960]. There, the belief state is represented as a Gaussian, some variables can be observed, and world progress (the transition relation) is expressed using linear equations and Gaussian noise. The Kalman filter has limited applicability but has been studied and used extensively and successfully in the control literature. Its two attractive properties are that it is efficient to compute and its belief state representation is compact. It is the only tool known to science today that has those properties.

In this paper we investigate an approach to *logical filtering*, which assumes a representation of the transition between situations in the world in a logical (or logic-like) language. We provide a formal treatment and algorithms for propositional logical filtering for domains that have nondeterministic actions or that have an incomplete description of the initial state. We represent a belief state using a logical formula over the fluents that define world states. Our algorithms update a belief state by creating a new formula that describes the result of performing an action and making observations. This framework allows us to also model domains in which there are no actions known to us, but only events (with some known (nondeterministic) transition model) and observations (as in HMMs).

We identify several classes of nondeterministic dynamic systems that allow efficient filtering. We show that filtering can always be distributed over logical disjunction in the belief state formula, and can also be distributed over logical conjunction and negation if some conditions hold about the dynamic system. We point out one class of such dynamic systems that we call *permutation domains*. In those domains, actions serve as one-to-one mappings between states, for those states in which they can be applied. This gives rise to more efficient algorithms for some representations of belief state formulae, such as NNF, DNF, CNF, and CNF of prime implicates. Our algorithms approximate the belief state in those domains that are not permutation domains.

We show that if our action theory has some natural properties then filtering of k-CNF formulae (CNF with clauses of size at most k, when k is given beforehand) can be done so that the result is a k-CNF formula. Consequently, we maintain a compact representation of the belief state in those domains where this can be done and k is small.

We show that filtering in many nondeterministic STRIPS domains can be done efficiently while keeping the belief-state representation compact. For example, in permutation domains, if our belief state formula and actions' effects are in k-CNF, and every action has a precondition that includes at most k symbols not affected by this action, then our belief state is in k-CNF after applying any action (we allow making an observation in k-CNF). We get a similar result for all nondeterministic STRIPS domains (not necessarily permuting) that satisfies the conditions above about actions' effects and precondition, if our belief state is in k-CNF that includes all its prime implicates.

Our algorithms are computationally superior to earlier methods used in nondeterministic planning, and in some cases they enable efficient filtering with action/observation sequences of arbitrary length while maintaining small representation size (see Section 1.1). Our filtering operator is only the second example (first is *Kalman filters*) known to science today where a natural compact representation of a belief state can be maintained efficiently over an arbitrarily long sequence of transitions.

1.1 Related Work

The computation of filtering is relatively simple when the initial state of the system is fully known and the actions and events in this system are fully known and are deterministic (e.g., [Fikes, Hart, & Nilsson, 1981; Lin & Reiter, 1997]). The problem of performing logical filtering in nondeterministic domains is computationally more difficult. In particular, the related problem of logical temporal projection is known to be coNP-hard when the state of the world is not fully known, or when actions have nondeterministic effects [Liberatore, 1997; Baral, Kreinovich, & Trejo, 2000; Amir, 2002].

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 (e.g., [Ferraris & Giunchiglia, 2000; Cimatti & Roveri, 2000; Bertoli *et al.*, 2001]), 2) List the sequence of actions and prove a query by regression or by using the theory of action directly to prove a query for the sequence of actions and observations in question (e.g., [Reiter, 2001; Lifschitz, 2000]), or 3) approximate the belief state representation (e.g., [Son & 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 1000 actions). However, many dynamic systems have a large number of fluents (especially when they are propositionalized) and we would like to track them in an efficient manner over very long sequences of actions ($> 100,000$). Examples include robot localization, tracking of objects and their relationships, and data mining. In many domains our approach is efficient yet general enough to capture and fulfill such demands. The last approach is not always usable, it requires an approximation that fits the given problem, is computationally slow still, and it gives rise to many mistakes that are sometimes dangerous. For example, few people will fly an aircraft that uses an approximate controller for landing.

2 Logical Filtering

In this section we define the transition model and action description language that we use to describe nondeterministic dynamic domains. Our model supports sequences of actions, and is not intended for concurrent actions or ramifications of actions (although it may be extended to such). It is kept simple so that we can examine the computational properties of the system easily. We take some elements from action language $\mathcal{AR}$ [Giunchiglia, Kartha, & Lifschitz, 1997].

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. $Cn(\Psi)$ is the set of logical consequences of Ψ (i.e., those formulae that are valid consequences of Ψ in propositional logic), 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 $C^{\mathcal{L}(L)}(\Psi)$.

2.1 Nondeterministic Transition Model

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 (these may be actions that change the state of the world, sensing actions, or a combination of both) and $\mathcal{R}(\langle s, a, s' \rangle)$ means that state s' is a possible result of action a in state s.

A *belief state* is a set of world states $\sigma \subseteq \mathcal{S}$. The belief state in the result of performing an action, given a belief state σ, is the belief state comprising of all the world states that may result from that action and a world state in σ.

Notice that 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. We return to this subject in Section 2.3 and again in Section 6 when we introduce observation models.

2.2 Action Description Language

The transition model and belief state update function given above can represent a rich set of dynamic systems. In this section we describe a specification language that defines some of those dynamic systems and that we will use as our basic language in the rest of this paper. In this language we describe actions whose effects may be nondeterministic and declarations describing our knowledge about the initial state of the system.

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 propositions* 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)

$$\mathcal{R}_D = \left\{ \langle s, a, s'\rangle \;\middle|\; \begin{array}{l} (s' \cap I(a, s)) = (s \cap I(a, s)) \\ \text{and } F(a, s) \text{ is true in } s' \end{array} \right\} \tag{1}$$

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

We explain this definition. First, inertia is applied to all fluents that do not appear in an activated rule (regardless of whether they are positive or negative in the current state). Then, our knowledge about those fluents that are possibly affected is determined by the head of the activated effect axioms after releasing those fluents from persistence (they are not assumed to stay the same even if they might).

EXAMPLE Let $F = holdA \vee holdB$, and assume that the rule "$pickUpBlock$ **causes** F **if** $TRUE$" is the only rule activated in state s for action $a = pickUpBlock$. Then, all of $\langle s, a, s_1\rangle, \langle s, a, s_2\rangle, \langle s, a, s_3\rangle$, are in our transition relation $\mathcal{R}_D$, for s_1, s_2, s_3 such that $holdA \in s_1$, $holdB \in s_2$ and $holdA, holdB \in s_3$ (i.e., we allow nondeterminism to not only choose between the effects $holdA, holdB$ but also possibly affect both $holdA, holdB$). ■

This example may seem unintuitive at first because if $holdA, holdB$ are both true in s, then one of our resulting states is s_2 in which $holdA$ is not true. This is sanctioned by our effect rule for a, which explicitly allow this effect. If we want to state inertia so that this does not happen (e.g., that this state is possible only if we started from a state in which $holdA$ was already FALSE), then we need to provide explicit rules that say so. We regard this as the responsibility of the knowledge engineer or an automated process that may generate those rules for us (as in [Lin & Reiter, 1994] and others).

Our choice of this semantics for nondeterminism is mainly for its simplicity (thus, also computational properties) and its natural properties. It resembles the specification of a situation calculus theory after a solution to the frame problem has already been applied [Lin, 1996; Reiter, 2001]. There are other action languages and semantics that are used for specifying nondeterministic dynamic systems [Gelfond & Lifschitz, 1998; Reiter, 2001; Levesque *et al.*, 1997; Thielscher, 2000]. Mostly, they can be translated into our system without much technical effort, so we avoid further discussion of those here.

2.3 Filtering

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. When there is no transition in $\mathcal{R}$ for s, a, we end up with an empty belief state (no state is possible). 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 step with a *and Step 3* filtering step with o.

One approach to computing the belief state after a sequence of actions and observations is to represent a belief state as a set of states. A more sophisticated approach is to represent the belief state more compactly, but still compute the resulting belief state for every original world state and represent the set of resulting states in a more compact form. This approach is taken (implicitly) in works on planning as model checking (e.g., [Cimatti & Roveri, 2000; Bertoli *et al.*, 2001; Rintanen, 2002]) and planning as satisfiability (e.g., [Ferraris & Giunchiglia, 2000]), where it has proved to be computationally expensive in domains with more than a few state features, even if BDDs are used for the update operations.

An alternative approach is to perform logical progression in a form similar to the one described by [Lin & Reiter, 1997]. The difference is that now we wish to do so in the context of nondeterministic actions and observations. The computationally difficult part is going to be finding the result of applying the action to the belief state. This is the topic of the next section.

3 Basic Algorithm for Logical Filtering

An actual procedure for logical filtering may perform excessive work if it represents a belief state explicitly as a set of states. In this section we present a simple algorithm for filtering that avoids representing or enumerating all the states possible in a belief state.

3.1 Basic Filtering Algorithm

Recall that $I(a, s)$ was defined in Section 2.2 to be the set of literals that are not affected by action a in state s. In this section we make the simplifying assumption that for all states s, s' in which a has some effect (i.e., $I(a, s), I(a, s') \neq \mathcal{P}$), $I(a, s) = I(a, s')$. This assumption means that we always release from the frame assumption a constant set of fluents, $Eff(a)$, for a given action a, as long as at least one rule for a is applicable in s. This has the effect that it is simple to specify those fluents that remain constant. $\overline{Eff(a)}$ is the set of fluents that are not in $Eff(a)$. In what follows, most of our notation is developed for an implicit action, and we add that action explicitly only when confusion may arise if it is omitted.

We represent a belief state σ as a logical formula φ such that a state is in σ iff it satisfies φ. The filtering algorithm we present applies two steps, a progression step and a filtering step, to a belief state formula φ in order to produce the next belief state. Before we define these steps we detail some formal tools.

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 $F_i' = F_{i[f_1,\ldots,f_n/f_1',\ldots,f_n']}$, for $\{f_1, \ldots, f_n\} = \mathcal{F}$ the set of fluents in our domain, $\mathcal{F}' = \{f_1', \ldots, f_n'\}$ a set of new symbols for fluents, and $[x/y]$ in the subscript means that we replace all instances of the symbol x in the formula by instances of the symbol y (for sequences of symbols we replace the symbols in respective locations). The intuition is that the fluents in $\mathcal{F}$ are taken with respect to some time t, and those in $\mathcal{F}'$ are the same fluents taken with respect to time $t + 1$.

We add the following set of rules for action a:

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

for $\bar{G} = \neg G_1 \wedge ... \wedge \neg G_l$, 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, recall that $Cn^{\mathcal{F}'}$ means the classical consequence relation restricted to the language that includes only $\mathcal{F}'$.

We define filtering of belief-state formulae as follows. (We reuse the symbol $Filter[.](.)$ for filtering a belief-state formula; There is no confusion about this reuse because a belief state formula is never a set of world states and a set of world states is never a belief state formula.)

1. $Filter[a](\varphi) =$ $(Cn^{\mathcal{F}'}(\varphi \wedge \bigwedge_{i\leq m}((\varphi \Rightarrow G_i) \Rightarrow F_i')))_{[f_1',\ldots,f_n'/f_1,\ldots,f_n]}$;
2. $Filter[o](\varphi) = \varphi \wedge o$.

Thus, a simple algorithm for computing $Filter[\langle a_1, o_1, ..., a_t, o_t\rangle](\varphi)$ is to apply these rules recursively, setting $\varphi_0 = \varphi$ and $\varphi_i = Filter[o_i](Filter[a_i](\varphi_{i-1})$ for $i > 0$ using the two equalities defined above. This algorithm is correct, as the following theorem shows.

Theorem 3.1 *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", and $r_{l+1}, ..., r_m$ are the completion rules for a as added above, then*

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

PROOF See Section A.1 ■

This theorem says that our algorithm takes a formula that represents a set of states and produces a formula that represents the set of states sanctioned by Definition 2.1.

The algorithm implied by Theorem 3.1 (iterative application of filtering of a belief-state formula with an action and observation) can be implemented using a consequence finder in a limited language, such as those based on ordered resolution (e.g., [del Val, 1999]).

Also, one way to make the belief state representation of the resulting state more succinct is to take the representation of the new belief state to be the conjunction of prime implicates of $Filter[a](\varphi)$ instead of all the consequences in that language. This can be done by several consequence finders, such as [Baumgartner, Furbach, & Stolzenburg., 1997; Iwanuma, Inoue, & Satoh, 2000; del Val, 1999].

In addition, if our belief-state formula is a conjunction of all of its prime implicates, then there are specialized methods that we can use to speed up the computation of the progression and filtering steps. This topic is outside the scope of this paper for lack of space, and we leave it for a different report.

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

3.2 Distribution Properties

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

Corollary 3.2 (Distribution over Connectives) *If φ, ψ are formulae in the language $\mathcal{L}(\mathcal{F})$ 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", and $r_{l+1}, ..., r_m$ are completion rules as added above, 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. $\models Filter[a](\neg\varphi) \Leftarrow \neg Filter[a](\varphi) \wedge Filter[a](TRUE)$

PROOF See Section A.2 ■

The intuition coming out of this theorem is the following. Filtering $\varphi \vee \psi$ can be done by filtering φ and ψ separately and then combining the results. Also, filtering $\varphi \wedge \psi$ can be approximated by filtering φ and ψ separately and then combining the result. The formula that is the conjunction of the separate filtering of φ and ψ is a weaker formula that the filtering of $\varphi \wedge \psi$. Thus, everything that follows from that combination is also true in every state that is the result of the original filtering. Finally, filtering of $\neg\varphi$ can be approximated in the other direction. The formula that is the negation of filtering φ is a stronger formula than the filtering of $\neg\varphi$. Thus, everything that follows from the filtering of $\neg\varphi$ necessarily holds in the negation of the filtering of φ and that of TRUE.

While the last two are only approximations (according to Corollary 3.2), they can be used in a safe way. If we only distribute filtering over disjunction and conjunction (but no negation), then our approximation is safe. For example, we will not land a plane if it is not possible to do so, but there may be times where it will be OK to land the plane but we will not know it.

3.3 Permutation Domains

For an interesting class of dynamic systems we can say something stronger. In this class we include systems in which every action acts as a *permutation* on the states in $\mathcal{S}$. In other words, every action acts as a one-to-one mapping from states to states, i.e., $\mathcal{R}_D(s, a, s')$ is a one-to-one mapping of a state and an action to a resulting state. We call domains that satisfy this requirement *permutation domains*.

Example permutation actions include: *Turning a row* in a Rubik's cube; *flipping a light switch* turns the light on if it was off and off if it was on; *Purchasing* coffee has the effect of increasing the amount of coffee we have and decreasing the amount of money we have. 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*).

Corollary 3.3 (Distribution for Permutation Domains) *Let D be a permutation domain. If φ, ψ are formulae in the language $\mathcal{L}(\mathcal{F})$ 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", and $r_{l+1}, ..., r_m$ are completion rules as added above, 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. $Filter[a](\neg\varphi) \equiv \neg Filter[a](\varphi) \wedge Filter[a](TRUE)$

PROOF See Section A.3 ■

From this corollary we get that the decomposition strategy offered above is an exact computation for permutation domains and not only an approximation. It can be applied to CNF formulae and any other propositional formula.

A somewhat weaker requirement that still allows us to use distribution over conjunction and negation connectives is asking that: For every state s and action a, there is *at most one* s' such that $\mathcal{R}_D(s, a, s')$. Following the proof-line of Corollary 3.3 shows that even with this weaker requirement the distribution holds. For example, STRIPS domains can be *weakly permuting* in the sense that, if a condition holds for an action, then it changes the state of the world in a one-to-one way, but if it does not hold, then the action is not executable. 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. It is one-to-one when it is possible because every two resulting states that are identical can be reverse-engineered to see what the previous state was. Similar situation holds with $putOn(A, B)$.

4 Filtering NNF Belief States

Computation of filtering can become burdensome in large domains, even if we use decomposition of the kind suggested in the last section. Particularly, we still have to perform general-purpose logical deduction for every formula that we filter.

A formula is in negation normal form (NNF) if all negations are in front of atoms (e.g., CNF formulae and DNF formulae are in NNF). In this section we describe more efficient algorithms that are available when we make the general assumption that our belief state formula φ is in NNF.

4.1 Negation Normal Form (NNF) Filtering

A simple computation that is in the spirit of Corollary 3.2 but is *not* equivalent to our definition for filtering of an action is $Filter[a](\varphi) \overset{?}{=} \bigwedge_{i \leq l,\ \varphi \Rightarrow G_i} F_i$. It says that we conclude the results of those rules that are activated for a, i.e., if G_1 can be proved from our current belief state, then we should believe F_1 in the result of the action a. This simple conjunction describes some of the knowledge that we hold about the state of the world after action a, but it does not hold all of it. For example, if φ implies $G_i \vee G_j$, then our effect axioms will imply that $F_i \vee F_j$ holds in the result of a, but neither of F_i, F_j holds by itself.

We can enhance the last formula and find an equivalent definition for filtering that under some conditions has better computational properties.

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

This formula has the benefit that the only inference that needs to be done is that of checking if $\varphi \Rightarrow G_{i_1} \vee \ldots \vee G_{i_u}$. 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 show how to avoid using most of the completion rules when filtering a CNF formula. This allows us to avoid the main difficulty with using equation (2). The main intuition is that we may omit most of those rules if we know that φ includes only a small subset of $\mathcal{F}$, the fluent symbols of our domain. This is not be the case, in general, 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 will include only a small subset of $\mathcal{F}$, and filtering each of the parts separately will become easy.

We need a few definitions before we can make use of this intuition. First, 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** $\bar{G}$". Then, define $\mathcal{B}$ to be

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

for $\widetilde{G}_{i_1,...,i_u} \equiv Cn^{\overline{Eff(a)}}(\bigwedge_{f \leq u} \neg G_{i_f})$, the consequences of $\bigwedge_{f \leq u} \neg G_{i_f}$ in the language that does not include fluents from $Eff(a)$. We explain this below.

$\mathcal{B}$ is a term that is always implied by $Filter[a](\mathit{TRUE})$, i.e., the progression of zero knowledge with the action a. It is not equivalent to $Filter[a](\mathit{TRUE})$, but it is required to complete the definition of the general case. 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 or may not be empty).

Define $\mathcal{C}(L)$ to be the set of completion rules of a for fluents in L, i.e.,

$$\mathcal{C}(L) = \{i > l \mid \text{the head of } r_i \in \mathcal{C} \text{ is in } L\}$$

Theorem 4.1 below shows that the following equivalence holds:

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

The intuition for this formula 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 summarizes in $\mathcal{B}$, which is independent of φ.

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 equivalence (4) holds.*

PROOF See Section A.4. ■

This formula together with the decomposition property of Corollaries 3.2,3.3 suggests much faster algorithms for computing the filtering of an NNF. We describe this result in the following theorem and corollaries. The complete algorithm is in Figure 1.

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

PROCEDURE NNF-ProgressStep($a$,$\varphi$)
$a$ is an action. $\varphi$ is a belief-state formula. $r_1, ..., r_l, r_{m+1}$ as in Section 4 for $a$.
  1. If $\varphi$ is a literal, then
     (a) Set $\varphi' = TRUE$.
     (b) While there is a set $i_1, ..., i_u \in \{1, ..., l, m+1\} \cup \mathcal{C}(\varphi)$
         that has not been tried already or subsumed by a kept combination, do
         • If $\varphi \models \bigvee_{f\leq u} G_{i_f}$, then set $\varphi' \leftarrow \varphi' \wedge F$ where $F$
           is an NNF form of $\bigvee_{f\leq u} F_{i_f}$.
     (c) Return $\varphi'$.
  2. If $\varphi = \varphi_1 \vee \varphi_2$, then return
     NNF-ProgressStep($a$, $\varphi_1$) $\vee$ NNF-ProgressStep($a$, $\varphi_2$).
  3. It must be that $\varphi = \varphi_1 \wedge \varphi_2$. Return
     NNF-ProgressStep($a$, $\varphi_1$) $\wedge$ NNF-ProgressStep($a$, $\varphi_2$).

PROCEDURE NNF-FilterStep($o$,$\varphi$)
$o$ is an observation. $\varphi$ is a belief-state formula.
  1. Set $o'$ to be an NNF form of $o$. Return $\varphi \wedge o'$.
```

Figure 1: Filtering an NNF formula.

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 whose preconditions mention t symbols cumulatively. Then, filtering of φ with action a can be approximated safely in time $O(h \cdot 2^l \cdot 2^t)$. If D is a permutation domain, then this computation is exact.*

PROOF See Section A.5 ■

4.2 DNF Belief States

A weakness of the NNF filtering above is that it is only an approximation (albeit a safe one) for domains that are not permuting. Another weakness of the result above is that we do not have a bound on the size of the resulting formula. It may as well be that the resulting formula has exponential size in the size of the original belief-state formula. For these reasons we turn to special cases of NNF, namely, DNF and CNF.

In this section we focus on the case in which φ is given to us in DNF (a disjunction of conjunctions), i.e., $\varphi = D_1 \vee \ldots \vee D_s$. In this case the computation can be simplified as follows: For each D_i compute the belief state $Filter[a](D_i)$ separately. Then, compute

$$Filter[a](\varphi) = \bigvee_{i \leq s} Filter[a](D_i)$$

This is analogous to case splitting (e.g., in Natural Deduction [Gentzen, 1969]) and is a valid computation, as stated in Corollary 3.2.

An action a has an *exhaustive set of rules* if the preconditions of the rules for a cover all the cases (i.e., for every state s there is an effect rule for a that is activated in this state).

Corollary 4.3 (Iterating DNF Filtering) *Let $\varphi = D_1 \vee \ldots \vee D_s$ be a belief state formula in DNF and a an action with l effect rules. Assume that every effect F_i is in DNF and that the total number of disjuncts in $F_1, ..., F_l$ is d. Further assume that every precondition G_i is in CNF with $c_i \leq c$ conjuncts and that there is a total of t symbols appearing in the G_i's. Then, computing the filtering of φ with a can be done exactly in time $O(h \cdot 2^l \cdot 2^t)$, producing a DNF formula with no more than $2s \cdot \binom{d}{d/2} \cdot c^l$ disjuncts.*

If a has an exhaustive set of rules, then the resulting DNF formula has no more than $s \cdot \binom{d}{d/2} \cdot c^l$ disjuncts.

PROOF See Section A.6. ■

As a conclusion from the last corollary we note that when the set of rules is exhaustive and there is only a single action rule (e.g., the action is always executable with the same effect) and the action is deterministic, then the number of disjuncts in the formula does not grow as the filtering progresses.

Another consequence is that despite results on the hardness of projection in nondeterministic or partially known domains [Liberatore, 1997; Baral, Kreinovich, & Trejo, 2001; Amir, 2002], simple queries in domains with few actions rules per action and few disjuncts in the effects of action rules can be answered efficiently by first performing filtering without observations (progression) and then using the DNF form of the result to answer the query, if the initial DNF form is compact.

In some cases, we can say more about the number of disjuncts and their form in the result of filtering a DNF formula. Consider the case when $\mathcal{B}$ is a tautology. An example when this can occur is when every precondition G_i of a rule r_i for action a is equal to some F_j (e.g., this can happen when our domain is a permutation domain). Our assumption that all effect rules state an effect for all of $Eff(a)$ serves to guarantee that $L(G_i) \cap L(F_i) \neq \emptyset$.

When $\mathcal{B}$ is a tautology, the result of the filtering is captured by the first part of formula (4). If we also guarantee that $\neg\bar{G}$ is always provable from φ (e.g., when our action rules for a is exhaustive), then Lemma A.2 guarantees that the DNF form of the filtering of a disjunct D_i with $L(D_i) \subseteq Eff(a)$ includes only disjuncts that are the result of conjoining effects of a (the F_i's). We can extend this to any D_i by saying that any disjunct of the resulting DNF from filtering D_i includes only disjuncts that conjoin effects of a with each other and the part of D_i that is not in $Eff(a)$.

From this, if we have only one action (e.g., when we have a fixed transition model for the system, such as in HMMs), then filtering φ of s disjuncts with an arbitrary number of executions of a does not put the number of disjuncts in the resulting DNF over $s \cdot \binom{d}{d/2}$ when d is as in Corollary 4.3. In fact, a similar analysis shows that when we do not require that $\mathcal{B}$ is not a tautology or that the effect rules of action a are exhaustive, then the number of disjuncts in the filtering of an arbitrary number of executions of a is at most $s \cdot \binom{d+c}{(d+c/2)}$, when d and c are as in Lemma A.2.

4.3 CNF Belief States

The following corollary describes conditions under which filtering can maintain a formula that is in k-CNF, thus keeping the representation compact.

Corollary 4.4 (Iterating CNF Filtering) *Let $\varphi = C_1 \wedge ... \wedge C_s$ be a k-CNF formula, and let action a have l effect rules with effects expressed in DNF and a total of t symbols appearing in the preconditions. Assume that the total number of disjuncts in all the effects of rules of a is $f \leq k$, and the preconditions of rules of a have CNF form with each rule r_i having a number of conjuncts $c_i \leq k - f$. If every clause C_i satisfies one of*

1. *$L(C_i) \subseteq Eff(a)$ (all literals are possibly affected) or*
2. *$L(C_i) \cap Eff(a) = \emptyset$ (no literal is modified) or*
3. *$|L(C_i) \setminus Eff(a)| \leq k - f - c_i$ (few literals are not affected),*

and one of

1. *$C_i \models G_1 \vee ... \vee G_l$ or*
2. *$C_i \models \bar{G}$ (all completion rules are used for C_i),*

then filtering of φ with action a can be approximated safely in time $O(s \cdot 2^l \cdot 2^t)$, producing a k-CNF formula as the result. If D is a permutation domain, then this computation is exact.

PROOF See Section A.7. ■

In the last corollary, if we have an *exhaustive set of rules* (i.e., the preconditions of the rules for a cover all the cases), then we can drop the condition on the preconditions of rules of a (i.e., we no longer require that $c_i \leq k - f$ or that they are in CNF). In this case it is always true that $G_1 \vee ... \vee G_l$ is a tautology, so the second condition on C_i holds immediately.

4.4 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* (we call such belief state formulae *prime implicate belief states*), then we can distribute the computation to each of the conjuncts and conjoin the result of filtering each small subgroup of them separately. This is a direct result of equivalence (2).

Theorem 4.5 (Filtering Prime Implicates: DNF Precond.) *Let $\varphi = C_1 \wedge ... \wedge C_s$ be a k-CNF formula that is a prime implicate belief state. Let action a have l effect rules with effects expressed in k-CNF and preconditions expressed in t-DNF with at most d disjuncts. Then filtering of φ with action a can be computed exactly in time $O(2^{l\cdot|Pre(a)\cup Eff(a)|} \cdot (s^z + z))$ for $z = t^{l\cdot d} \cdot (l \cdot d + 1)$. If $\models \neg\bar{G}$, then the computation takes $O(2^l \cdot (s^z + z))$ time, with $z = t^{l\cdot d}$.*

PROOF See Section A.8 ■

When $\bar{G}$ cannot happen (i.e., $\models \neg\bar{G}$), then we have only $t^{l\cdot d}$ clauses. Also, we can use G_i's in d-CNF with at most t clauses in each and get that each $G_{i_1} \vee ... \vee G_{i_u}$ has a representation as a conjunction of $t^l \cdot (d^t + 1)$ clauses, or t^l if $\models G_1 \vee ... \vee G_l$.

Corollary 4.6 (Filtering Prime Implicates: CNF Precond.) *Let $\varphi = C_1 \wedge ... \wedge C_s$ be a k-CNF formula that is a prime implicate belief state. Let action a have l effect rules with effects expressed in k-CNF and preconditions expressed in d-CNF with at most t clauses. Then filtering of φ with action a can be computed exactly in time $O(2^{l\cdot|Pre(a)\cup Eff(a)|} \cdot (s^z + z))$ for $z = t^l \cdot (d^t + 1)$. If $\models \neg\bar{G}$, then the computation takes $O(2^l \cdot (s^z + z))$ time, with $z = t^l$.*

5 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. In this section we look at such domains but relax some of the original STRIPS assumptions. For example, we allow nondeterministic effects, we represent both negative and positive information in the belief state (i.e., we use an *open world STRIPS*), and allow the belief state to be any CNF formula 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, action a can be executed only when G_1 holds. Consequently, when we filter with action a then we implicitly get the assertion that the preconditions of a 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 dramatic effects on the filtering of the belief state. For a set of literals $l_1, ..., l_k$ we get that

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

Consequently, we get the following theorem.

Theorem 5.1 (Iterating STRIPS Filtering: CNF) *Let D be a STRIPS domain, φ be a k-CNF formula with s clauses and a an action with effect rule "a* **causes** *F_1* **if** *G_1". If F_1 is in CNF with at most k literals in each clause (or DNF with at most k disjuncts) 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 D is a permutation domain, then this computation is exact.*

PROOF See section A.9. ■

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.

Theorem 5.2 (Factoring STRIPS filtering: Prime Implicat.) *Let D be a STRIPS domain and let $\varphi = C_1 \wedge ... \wedge C_s$ be a k-CNF formula that is a prime implicate belief state. Let a an action. Then,*

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

PROOF See Section A.10. ■

Corollary 5.3 (Iterating STRIPS filtering: Prime Implicat.) *Let D be a STRIPS domain and let $\varphi = C_1 \wedge ... \wedge C_s$ be a k-CNF prime implicate belief state. Let a an action with effect rule "a* **causes** *F_1* **if** *G_1". If F_1 is a CNF prime implicate formula with at most k literals in each clause (or DNF with at most k disjuncts), and $|L(G_1) \setminus L(F_1)| \leq t$, for some $t \leq k$, then $Filter[a](\varphi)$ can be computed exactly in time $O(s \cdot k + 2^t)$, yielding a k-CNF prime implicate belief state.*

PROOF See Section A.11. ■

This means that we can filter 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 of the largest clause depending only on the prime implicate representation of the effects and the number of propositional symbols that are in the preconditions of an action but not in its effects.

An interesting special case of the last corollary is when the belief state is represented as a formula in 2-CNF. Every prime implicate of a formula in 2-CNF is a clause with at most two literals. Thus, Such knowledge states can be filtered in appropriate STRIPS domains (not necessarily permuting) with every action, producing a 2-CNF belief state.

6 Observation Model

7 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 space crafts.

We made use of several assumptions in this paper in different contexts and with different consequences. We presented permutation domains and exhaustive action-rule sets 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.

A Proofs

This section will include all the proof in the submitted version (currently they are in the body of the paper).

A.1 Proof of Theorem 3.1

PROOF We show that the two sets of world states have the same elements. We show first that the left-hand side of the equality is contained in the right-hand side.

Take $s' \in Filter[a](\{s \in \mathcal{S} \mid s \text{ satisfies } \varphi\})$. We show that s' satisfies $Filter[a](\varphi)$. From Definition 2.1 there is $s \in \mathcal{S}$ such that $s \in \{s \in \mathcal{S} \mid s \text{ satisfies } \varphi\}$ such that $\langle s, a, s'\rangle \in \mathcal{R}$. In other words, there is $s \in \mathcal{S}$ such that s satisfies φ and $\langle s, a, s'\rangle \in \mathcal{R}$.

To prove that s' satisfies $Filter[a](\varphi)$ we need to show that $\varphi \wedge \bigwedge_{i\leq m}(\varphi \Rightarrow G_i) \Rightarrow F'_i))$. together with the truth assignment s' to $\mathcal{F}'$ is satisfiable. We show that the truth assignment s to $\mathcal{F}$ satisfies this formula together with the truth assignment s' to $\mathcal{F}'$. It is not satisfying this formula only if one of the conjuncts $(\varphi \Rightarrow G_i) \Rightarrow F'_i$ or φ is falsified. This cannot be the case for φ by our choice of s.

Assume by contradiction that this is the case for some i. Then, the truth assignments of s, s' to $\mathcal{F}, \mathcal{F}'$ sanction that $\varphi \Rightarrow G_i$ holds but F'_i does not. From the way we defined $\mathcal{R}$ (i.e., $\mathcal{R}_D$ in equation (1)) we can conclude that $F(a, s)$ is true in s' and that $s' \cap I(a, s) = s \cap I(a, s)$. However, $F(a, s)$ is the conjunction of heads of activated rules and $I(a, s)$ is the set of unaffected fluents. If $i \leq l$ (i.e., r_i is an original rule), then $\varphi \wedge (\varphi \Rightarrow G_i)$ implies that G_i holds in s and the rule r_i is activated. Thus, $F(a, s)$ includes F_i, and F_i is true in s'. This contradicts our assumption that F'_i does not hold with the truth assignment s' to $\mathcal{F}'$. Thus, there is no such conjunct in $\bigwedge_{i\leq m}(\varphi \Rightarrow G_i) \Rightarrow F'_i))$ and the truth assignment s, s' to $\mathcal{F}, \mathcal{F}'$, respectively, satisfies this formula. From the definition of $Filter[a](\varphi)$ and Craig's interpolation theorem for propositional logic (See Theorem A.1) we get that s' satisfies $Filter[a](\varphi)$.

For the opposite direction (showing the right-hand side is contained in the left-hand side), take $s' \in \mathcal{S}$ that satisfies $Filter[a](\varphi)$. We show that $s' \in Filter[a](\{s \in \mathcal{S} \mid s \text{ satisfies } \varphi\})$. From Craig's interpolation theorem for propositional logic we get that there is a truth assignment s for $\mathcal{F}$ such that the truth assignment s, s' to $\mathcal{F}, \mathcal{F}'$, respectively, together satisfy $\varphi \wedge \bigwedge_{i\leq m}(\varphi \Rightarrow G_i) \Rightarrow F'_i))$ (otherwise, there is no such truth assignment, and $Filter[a](\varphi)$ is not satisfiable; in particular, s' does not satisfy it). In a manner similar to the first part of this proof (observing the way $\mathcal{R}$ is defined) we can show that $\mathcal{R}(s, a, s')$ and the second part is done. ■

A.2 Proof of Corollary 3.2

PROOF We show this theorem for the set-of-states representation of belief states, and it will follow for the formula-based representation.

1. Take a state s' that satisfies $Filter[a](\varphi \vee \psi)$. Then, there is a state s that satisfies $\varphi \vee \psi$ such that $R_D(s, a, s')$. Thus, s satisfies one of φ or ψ because s is a complete setting of the fluents. Thus, s' is in one of $Filter[a](\varphi)$, $Filter[a](\psi)$. From Theorem 3.1 it follows that $Filter[a](\varphi \vee \psi) \Rightarrow Filter[a](\varphi) \vee Filter[a](\psi)$.

For the other direction, take s' that satisfies $Filter[a](\varphi) \vee Filter[a](\psi)$. Then, it satisfies one of $Filter[a](\varphi)$, $Filter[a](\psi)$. Thus, there is a state s such that $R_D(s, a, s')$ and s satisfies one of φ,ψ. Thus, s satisfies $\varphi \vee \psi$ and s' satisfies $Filter[a](\varphi \vee \psi)$. From Theorem 3.1 it follows that $Filter[a](\varphi \vee \psi) \Leftarrow Filter[a](\varphi) \vee Filter[a](\psi)$.

2. Take a state s' that satisfies $Filter[a](\varphi \wedge \psi)$. Then, there is a state s that satisfies $\varphi \wedge \psi$ such that $R_D(s, a, s')$. Thus, s satisfies both of φ and ψ. Thus, s' is in both of $Filter[a](\varphi)$, $Filter[a](\psi)$. We conclude that every s' that satisfies $Filter[a](\varphi \wedge \psi)$ also satisfies $Filter[a](\varphi) \wedge Filter[a](\psi)$. From Theorem 3.1 it follows that $Filter[a](\varphi \wedge \psi) \Rightarrow Filter[a](\varphi) \wedge Filter[a](\psi)$.

3. Take s' that satisfies $\neg Filter[a](\varphi) \wedge Filter[a](TRUE)$. Then, there is no state s such that $R_D(s, a, s')$ and s satisfies φ. Thus, for every state s such that $R_D(s, a, s')$ s satisfies $\neg\varphi$. Since s' satisfies $Filter[a](TRUE)$ there is a state s such that $R_D(s, a, s')$. Thus, this s satisfies $\neg\varphi$ and s' satis-

fies $Filter[a](\neg\varphi)$. From Theorem 3.1 it follows that $Filter[a](\neg\varphi) \Leftarrow \neg Filter[a](\varphi) \wedge Filter[a](TRUE)$. ■

A.3 Proof of Corollary 3.3

PROOF Corollary 3.2 supplies the proof of 1, the "⇒" direction of 2, and the "⇐" direction of 3. Thus, we are left to prove the "⇐" direction of 2 and the "⇒" direction of 3.

For "⇐" 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 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 $\models Filter[a](\varphi \wedge \psi) \Leftarrow Filter[a](\varphi) \wedge Filter[a](\psi)$.

For "⇒" of 3, let s' be a world state that satisfies $Filter[a](\neg\varphi)$. Then, there is a state s that satisfies $\neg\varphi$ such that $R_D(s,a,s')$. Thus, s does not satisfies φ. Since a acts as a one-to-one mapping from $\mathcal{S}$ to $\mathcal{S}$, there is only one state that maps to s' after a. Thus, there is no state s_1 that satisfies φ and for which $R_D(s_1,a,s')$. Thus s' does not satisfy $Filter[a](\varphi)$ meaning that it satisfies $\neg Filter[a](\varphi)$. Clearly, s' also satisfies $Filter[a](TRUE)$. We get that s' satisfies $\neg Filter[a](\varphi) \wedge Filter[a](TRUE)$. From Theorem 3.1 it follows that $\models Filter[a](\neg\varphi) \Rightarrow \neg Filter[a](\varphi) \wedge Filter[a](TRUE)$. ■

A.4 Proof of Theorem 4.1

PROOF Let Ψ be the formula on the right-hand side of (4).

Showing that $Filter[a](\varphi) \models \Psi$ **:** Take $F_{i_1} \vee ... \vee F_{i_u}$ that is implied by formula (4). Then, $\varphi \models G_{i_1} \vee ... \vee G_{i_u}$, by the same formula. When we look at the definition of filtering of a formula (Theorem 3.1) we notice that every G_{i_f}, $f \leq u$, appears in this definition or $G_{i_f} = \bar{G}$. The latter belongs to r_{m+1}, which is the only rule that we use in (4) that is not a completion rule or an original rule. However, r_{m+1} follows from the completion rules in $\mathcal{C}$. Thus, $F_{i_1} \vee ... \vee F_{i_u}$ follows from the definition of filtering.

We show that $\mathcal{B}$ follows from $Filter[a](\varphi)$. For a rule r_i with $i \leq t$, we know that $L(G_i) \cap Eff(a) = \emptyset$. Thus, the completion rules of a ensure that $\neg G_i$ holds after executing a if it holds before executing a. However, $\neg G_i \vee G_i$ is a tautology (thus, in particular, it follows from φ), so we get that $\neg G_i \vee F_i$ holds in the result of executing a.

Similarly, for a disjunction of rule preconditions, $\bigvee_{f\leq u} G_{i_f}$ the disjunction $\neg(\bigvee_{f\leq u} G_{i_f}) \vee \bigvee_{f\leq u} G_{i_f}$ is a tautology. From case analysis and Theorem 3.1 we get that in the consequence of executing a we know $\widetilde{G}_{i_1,\ldots,i_u} \vee \bigvee_{f\leq u} F_{i_f}$.

Showing that $\Psi \models Filter[a](\varphi)$ **:** We use the equivalence stated in formula (2) by showing that every disjunction $\bigvee f \leq uF_{i_f}$ present in (2) is implied by Ψ.

Take $\bigvee_{f\leq u} F_{i_f}$ to be a conjunct in (2), and assume that it is minimal (i.e., no other conjunct includes a strict subset of F_{i_f}'s). Then, $\varphi \models \bigvee_{f\leq u} G_{i_f}$. If all r_{i_f}, $f \leq u$, are original rules of a (not completion rules) or completion rules for literals in $L(\varphi)$, then they all appear in (4), and therefore $\bigvee_{f\leq u} F_{i_f}$ appears in (4).

W.l.o.g. assume that we ordered i_f such that

- r_{i_f}, $f \leq t$, are original rules for a,
- r_{i_f}, $t < f \leq v$, are completion rules literals in $L(\varphi)$,
- r_{i_f}, $v < f \leq w$, are completion rules for literals in $Eff(a) \setminus L(\varphi)$, and
- r_{i_f}, $w < f \leq u$, are completion rules for literals in $\overline{Eff(a)} \setminus L(\varphi)$.

Denote the literals that are the heads of the completion rules l_{i_f}, $t < f \leq u$, respectively. If $v = u$ (there are no literals of the second and third sort), then we are done, by the previous paragraphs. Thus, assume that $v < u$.

We show that $\bigvee_{f\leq u} F_{i_f}$ is implied by Ψ.

$$\varphi \models \bigvee_{f\leq v} G_{i_f} \vee \bigvee_{v<f\leq w} (l_{i_f} \wedge \bar{G}) \vee \bigvee_{w<f\leq u} l_{i_f}$$

by the way we sorted F_{i_f}, and the fact that the disjunction $\bigvee_{f\leq u} F_{i_f}$ is one of the conjuncts in (2).

Let $\psi = \bigvee_{f\leq v} G_{i_f}$. Then, $\varphi \models \psi \vee (\bar{G} \wedge \bigvee_{v<f\leq w} l_{i_f}) \vee \bigvee_{w<f\leq u} l_{i_f}$. From this we get $\varphi \models \psi \vee \bigvee_{v<f\leq u} l_{i_f}$.

We make use of Craig's interpolation Theorem:

Theorem A.1 ([Craig, 1957]) *Let α, β be sentences such that $\alpha \vdash \beta$. Then there is a formula γ involving only non-logical symbols common to both α and β, such that $\alpha \vdash \gamma$ and $\gamma \vdash \beta$.*

We know that $L(\bigvee_{v<f\leq u} l_{i_f}) \cap L(\varphi) = \emptyset$. Craig's interpolation theorem implies that $\varphi \models \psi$ or $\psi \vee \bigvee_{v<f\leq u} l_{i_f}$ is a tautology. We assumed minimality of $\bigvee_{f\leq u} F_{i_f}$ in (2), so the first case cannot be.

Thus, it must be that $\psi \vee \bigvee_{v<f\leq u} l_{i_f}$ is a tautology. Placing the meaning of ψ in this formula we get that $\bigvee_{f\leq t} G_{i_f} \vee \bigvee_{t<f\leq u} l_{i_f}$ is a tautology. This means that $\neg(\bigvee_{f\leq t} G_{i_f}) \models \bigvee_{t<f\leq u} l_{i_f}$.

We know that $\neg(\bigvee_{f\leq t} G_{i_f}) \vee \bigvee_{f\leq t} G_{i_f}$ is a tautology because $a \vee \neg a$ is a tautology for every sentence a. We look at two cases:

Case 1: $L(\bigvee_{f\leq t} G_{i_f}) \cap Eff(a) = \emptyset$ **:** From this assumption, for every $f \leq t$ we have $\neg G_{i_f} \vee F_{i_f}$ as a conjunct in $\mathcal{B}$. For each $j \leq t$ we get the implied sentence $\neg G_{i_j} \vee \bigvee_{f\leq t} F_{i_f}$. The conjunction of those sentences for $j \leq t$ implies $\bigwedge_{j\leq t}(\neg G_{i_j} \vee \bigvee_{f\leq t} F_{i_f})$ which is equivalent to $\neg(\bigvee_{f\leq t} G_{i_f}) \vee \bigvee_{f\leq t} F_{i_f}$.

We already concluded above that $\neg(\bigvee_{f\leq t} G_{i_f}) \models \bigvee_{t<f\leq u} l_{i_f}$, so we get that $\bigvee_{t<f\leq u} l_{i_f} \vee \bigvee_{f\leq t} F_{i_f}$ is logically entailed by $\mathcal{B}$ and we are done (this last formula is exactly $\bigvee_{f\leq u} F_{i_f}$ with some replacement of positions of disjuncts).

Case 2: $L(\bigvee_{f\le t} G_{i_f}) \cap Eff(a) \neq \emptyset$ **:** Our earlier conclusion $\varphi \models \bigvee_{f\le v} G_{i_f} \vee \bigvee_{v<f\le w}(l_{i_f} \wedge \bar{G}) \vee \bigvee_{w<f\le u} l_{i_f}$ implies that $\varphi \models \bigvee_{f\le v} G_{i_f} \vee \bar{G} \vee \bigvee_{w<f\le u} l_{i_f}$ or $\varphi \models \bigvee_{f\le v} G_{i_f} \vee \bigvee_{w<f\le u} l_{i_f}$, depending on whether $v < w$ or $v = w$, respectively. For the rest of this proof we write θ for $\bar{G}$ or *FALSE*, according to whether $\bar{G}$ appears in this disjunction or not, respectively.

Craig's interpolation theorem implies that either $\varphi \models \bigvee_{f\le v} G_{i_f} \vee \theta$ or $\bigvee_{f\le v} G_{i_f} \vee \theta \vee \bigvee_{w<f\le u} l_{i_f}$ is a tautology. This is because l_{i_f} is not in $L(\varphi)$. We look at these two cases separately.

We assumed minimality of $\bigvee_{f\le u} F_{i_f}$ in (2), so the first case can be only if $w = u$ (i.e., no literals of the third kind). Thus, in this case, $\varphi \models \bigvee_{f\le v} G_{i_f} \vee \theta$, which is a precondition for a conjunction in (4). Thus, (4) includes the sentence $\bigvee_{f\le v} F_{i_f} \vee \theta$ in its conjunction.

Now, we know that $\bar{G} \models \neg(\bigvee_{f\le t} G_{i_f})$ because $\{G_{i_f}\}_{f\le t} \subseteq \{G_i\}_{i\le l}$. Also, we already concluded that $\neg(\bigvee_{f\le t} G_{i_f}) \models \bigvee_{t<f\le u} l_{i_f}$. We get that $\bigvee_{f\le v} F_{i_f} \vee \bar{G}$ logically entails $\bigvee_{f\le v} F_{i_f} \vee \bigvee_{t<f\le u} l_{i_f}$ which is equivalent to $\bigvee_{f\le u} F_{i_f}$. Thus, $\bigvee_{f\le v} F_{i_f} \vee \theta$ implies $\bigvee_{f\le u} F_{i_f}$, so (4) implies $\bigvee_{f\le u} F_{i_f}$ and we are done with this case.

Finally, the second case is that of $\bigvee_{f\le v} G_{i_f} \vee \theta \vee \bigvee_{w<f\le u} l_{i_f}$ is a tautology. (Notice that it is possible that we can conclude a stronger disjunction when $v = w$, but we ignore this case because a treatment of the weaker case implies a treatment for this one.)

W.l.o.g. assume that the literals l_{i_f} for $t < f \le v$ are ordered such that there is $v' \le v$ with $l_{i_f} \in \mathcal{L}(Eff(a))$ for all f such that $t < f \le v'$ and $l_{i_f} \in \mathcal{L}(\overline{Eff(a)})$ for all f such that $v' < f \le v$. Then, the formula $\bigvee_{f\le v} G_{i_f} \vee \theta \vee \bigvee_{w<f\le u} l_{i_f}$ is equal to $\bigvee_{f\le t} G_{i_f} \vee \bigvee_{t<f\le v'}(l_{i_f} \wedge \bar{G}) \vee \bigvee_{v'<f\le v} l_{i_f} \vee \theta \vee \bigvee_{w<f\le u} l_{i_f}$.

Again, we take θ' to be either $\bar{G}$ or *FALSE*, if $\bar{G}$ appears in this formula or not, respectively. This implies that $\bigvee_{f\le t} G_{i_f} \vee \bigvee_{v'<f\le v} l_{i_f} \vee \theta' \vee \bigvee_{w<f\le u} l_{i_f}$ is a tautology.

Consequently, $\neg(\bigvee_{f\le t} G_{i_f}) \wedge \neg\theta' \models \bigvee_{v'<f\le v} l_{i_f} \vee \bigvee_{w<f\le u} l_{i_f}$. W.l.o.g., assume that we ordered $\{G_{i_f}\}_{f\le t}$ such that there is $t' \le t$ such that $L(\{G_{i_f}\}_{f\le t'}) \cap Eff(a) = \emptyset$ and $\forall f\ (t' < f \le t \Rightarrow L(G_{i_f}) \cap Eff(a) \neq \emptyset$. Then, we rewrite the last formula into $\neg(\bigvee_{t'<f\le t} G_{i_f}) \wedge \neg\theta' \models \bigvee_{f\le t'} G_{i_f} \vee \bigvee_{v'<f\le v} l_{i_f} \vee \bigvee_{w<f\le u} l_{i_f}$.

Craig's interpolation theorem implies that there is a formula ξ such that $\xi \in \mathcal{L}(\overline{Eff(a)}) \cap \mathcal{L}(\bar{G})$ and $\neg(\bigvee_{t'<f\le t} G_{i_f}) \wedge \neg\theta' \models \xi$ and $\xi \models \bigvee_{f\le t'} G_{i_f} \vee \bigvee_{v'<f\le v} l_{i_f} \vee \bigvee_{w<f\le u} l_{i_f}$.

Let $i_{m+1} = m + 1$ if $\theta' = \bar{G}$ or $i_{m+1} = \emptyset$ otherwise. From the definition of $\widetilde{G}_{i_{t'+1},...,i_t,i_{m+1}}$ we get that $\widetilde{G}_{i_{t'+1},...,i_t,i_{m+1}} \models \xi$.

Thus, the formula $\widetilde{G}_{i_{t'+1},...,i_t,i_{m+1}} \vee \theta' \vee \bigvee_{t'<f\le t} F_{i_f}$ logically entails $\bigvee_{f\le t'} G_{i_f} \vee \bigvee_{v'<f\le v} l_{i_f} \vee \bigvee_{w<f\le u} l_{i_f} \vee \theta' \vee \bigvee_{f\le t} F_{i_f}$. Also, for every $f \le t'$ we have $\neg G_{i_f} \vee F_{i_f}$ in $\mathcal{B}$. We get that

$$(\bigvee_{f\le t'} G_{i_f} \vee \bigvee_{v'<f\le v} l_{i_f} \vee \bigvee_{w<f\le u} l_{i_f} \vee \theta' \vee \bigvee_{f\le t} F_{i_f}) \wedge \bigwedge_{f\le t'}(\neg G_{i_f} \vee F_{i_f})$$

entails $\bigvee_{v'<f\le v} l_{i_f} \vee \bigvee_{w<f\le u} l_{i_f} \vee \theta' \vee \bigvee_{f\le t} F_{i_f})$. If $\theta' =$ *FALSE*, then this formula subsumes $\bigvee_{f\le u} F_{i_f}$. On the other hand, if $\theta' = \bar{G}$, then, the same formula logically entails $\bigvee_{t<f\le u} l_{i_f} \vee \bigvee_{f\le t} F_{i_f})$ because $\neg(\bigvee_{f\le t} G_{i_f}) \models \bigvee_{t<f\le u} l_{i_f}$ and $\bar{G} \models \neg(\bigvee_{f\le t} G_{i_f})$.

Thus, we are done because $\mathcal{B}$ includes $\widetilde{G}_{i_{t'+1},...,i_t} \vee \theta' \vee \bigvee_{t'<f\le t} F_{i_f}$, so $\mathcal{B} \models \bigvee_{f\le u} F_{i_f}$ as needed. ∎

A.5 Proof of Corollary 4.2

PROOF We filter each of the literals separately and then combine the results. This is justified by Corollaries 3.2,3.3, where we apply distribution for conjunction and disjunction recursively until we get to single literals.

To filter a literal l_i in φ we need to find the strongest formulae of the form $G_{i_1} \vee ... \vee G_{i_u}$ for $i_1, ..., i_u \in \{1, ..., l\} \cup \mathcal{C}(l_i)$ that are implied by l_i. This can be done by enumerating all disjunctions of this form. There are 2^{l+2} such combinations because there are a total of $l + 2$ rules for every literal (there is one completion rule relevant for every literal, and we add r_{m+1}). For every such a disjunction, checking that l_i implies it can be done in time 2^t by exhaustive enumeration of all truth assignments on the t symbols comprising the preconditions of a. Finally, computing $\mathcal{B}$ can be done in time $O(2^l \cdot 2^t)$ as well. ∎

A.6 Proof of Corollary 4.3

PROOF

Lemma A.2 *For a formula φ, with $L(\varphi) \subseteq Eff(a)$ and $\varphi \models \neg\bar{G}$,*

$$Filter[a](\varphi) = \bigwedge_{\substack{i_1,...,i_u \in \{1,...l\},\\ \varphi \Rightarrow G_{i_1} \vee ... \vee G_{i_u}}} (F_{i_1} \vee ... \vee F_{i_u}) \bigwedge \mathcal{B}$$

We filter each of the literals separately and then combine the results using the recursive algorithm presented in Corollary 4.2. However, here we break only disjunctions and leave the conjunctions untouched.

Every disjunct D_i is a conjunction of literals $l_1 \wedge ... \wedge l_u$. For each such conjunction, asserting it and then testing whether $G_{i_1} \vee ... \vee G_{i_u}$ follows can be done in time 2^t. We can compute the DNF formula by distributing formula 4 over conjunctions.

We are left to prove that there are no more than $\binom{d+c}{(d+c)/2} \cdot s$ resulting disjuncts in this filtering. We do so by showing that the filtering of every D_i is a DNF with at most $\binom{d+c}{(d+c)/2}$ disjuncts. We divide the proof into three cases.

If $L(D_i) \cap Eff(a) = \emptyset$, then the only disjuncts coming as a result of filtering D_i are those coming from the conjunction of D_i and $\mathcal{B}$. The largest number of combinations of n elements that do not subsume each other (no combination is a subset of another) is $\binom{n}{n/2}$. Thus, the DNF form of $\mathcal{B}$

can have at most $\binom{d}{min(d/2,l)} \cdot c^l$ disjuncts because there are at most l choices from the F_i's in creating disjuncts from $\mathcal{B}$ and there are c disjuncts to choose from in each occurrence of a $\neg G_i$ (and we have a different G_i in each conjunct of $\mathcal{B}$). Thus, the total number of disjuncts in the filtering of D_i is at most $\binom{d}{min(d/2,l)} \cdot c^l$.

If $L(D_i) \subseteq Eff(a)$, then there are three cases according to the relationship between D_i and $\bar{G}$. If $D_i \models \bar{G}$, then the filtering is $D_i \wedge \mathcal{B}$, as above. Thus, in this case, the proof is done. If $D_i \models \neg\bar{G}$, then the filtering is in the form of equation (4) but with no completion rules involved. In this case, the second part of $\mathcal{B}$ is subsumed by one of the disjunctions in the first part of equation (4). Thus, we are left with a choice among the d disjunctions in the F_i's and then a choice of at most l elements among the disjuncts of the G_i's. This leaves us with at most $\binom{d}{d/2} \cdot c^l$ disjuncts in the filtering of D_i. Finally, if $D_i \not\models \bar{G}$ and $D_i \not\models \neg\bar{G}$, then we get that the filtering of D_i is the disjunction of the filtering of $D_i \wedge \bar{G}$, $D_i \wedge \neg\bar{G}$. This is $D_i \wedge \mathcal{B} \vee (4)$ for some formula of the form (4) that does not include completion rules. Using the previous cases we get that there are at most $2\binom{d}{d/2} \cdot c^l$ disjuncts in this case.

In the third case $L(D_i) \cap Eff(a), L(D_i) \setminus Eff(a) \neq \emptyset$. Here, the same argument as in the last paragraph shows that we have at most $2\binom{d}{d/2} \cdot c^l$ disjuncts in the filtering of D_i.

In the case of an exhaustive set of rules we always know that $D_i \models \neg\bar{G}$, so the factor 2 above never applies. ■

A.7 Proof of Corollary 4.4

PROOF We filter each of the literals separately and then combine the results using the recursive algorithm presented in Corollary 4.2.

First, we notice that under the conditions $f \leq k$ and $c_j \leq k - f$ (c_j is the number of conjuncts of rule r_j of a) $\mathcal{B}$ is in k-CNF.

Now, for a given clause C_i there are several possible filtering, depending on C_i's literals and on whether $C_i \models G_1 \vee ... \vee G_l$ or $C_i \models \bar{G}$.

If $L(C_i) \subseteq Eff(a)$, then the filtering is dependent on whether $C_i \models \bar{G}$ or $C_i \models \neg\bar{G}$ (one must be the case by the Corollary's assumption). If $C_i \models \bar{G}$, then the filtering is $C_i \wedge \mathcal{B}$, which is in k-CNF.

If $C_i \models \neg\bar{G}$, we look at the filtering of each of the literals of the clause separately (using Corollary 3.2). For a literal l in C_i, $l \models \neg\bar{G}$ because $l \models C_i$. Thus, we have a filtering of l which is a formula of the form of equation (4), but with no completion rules used ($p \wedge \bar{G}$ is not consistent with l). The disjunction of those formulae is the filtering of C_i, according to Corollary 3.2. On the other hand, for every two sub-clauses c_1, c_2 of C_i, the disjunctions of clauses from $Filter[a](c_1)$ and $Filter[a](c_2)$ is either subsumed by $\mathcal{B}$ (which appears in the filtering of C_i) or is a sub-clause of $F_1 \vee ... \vee F_l$. It follows that filtering of C_i is in k-CNF because $\mathcal{B}$ and $F_1 \vee ... \vee F_l$ are in k-CNF.

If $L(C_i) \cap Eff(a) = \emptyset$, then the only conclusion possible is $C_i \wedge \mathcal{B}$. This is in k-CNF because C_i and $\mathcal{B}$ are both k-CNF.

The same principle of the first two cases applies also in the case of $|L(C_i) \setminus Eff(a)| \leq k - f - c_i$. Take C_i^1, C_i^2 to be two subclauses of C_i such that $C_i \equiv C_i^1 \vee C_i^2$ and $L(C_i^1) \cap Eff(a) = \emptyset$ and $L(C_i^2) \subseteq Eff(a)$. In this case the filtering of C_i^1 is $C_i^1 \wedge \mathcal{B}$ and the filtering of C_i^2 is of the form of equation (4). It follows that the size of every clause in the filtering of C_i is at most $f + c_i + |L(C_i^1)| \leq k$ because $|L(C_i^1)| = |L(C_i) \setminus Eff(a)| \leq k - f - c_i$ and the size of each clause in the filtering of C_i^2 is at most $f + c_i$. We get that this filtering is in k-CNF. ■

A.8 Proof of Theorem 4.5

PROOF Every element of a disjunction $G_{i_1} \vee ... \vee G_{i_u}$ is either the precondition of an effect rule, a single literal (completion rule for literals not in $Eff(a)$) or a conjunction of a single literal with $\bar{G}$ (completion rule for literals in $Eff(a)$). Thus, each such disjunction is equivalent to a conjunction of (1) $\bar{G}$, (2) a disjunction of a set of literals and (3) a disjunction of preconditions of effect rules. The latter has at most $l \cdot d$ disjuncts each with at most t literals. Taking one literal from each of those disjunct leads to a representation of the latter part as a conjunction of $t^{l \cdot d}$ clauses. $\bar{G}$ has $l \cdot d$ conjuncts, each of at most t literals. Enumerating all the clauses that result from the combination of those parts yields a conjunction of at most $t^{l \cdot d} \cdot (l \cdot d + 1)$ clauses (we add 1 because if $\bar{G}$ participates, then $l \wedge \bar{G} \vee ...$ breaks into $(l \vee ...) \wedge (\bar{G} \vee ...)$).

Now, always if $\varphi \models A \wedge B$, then $\varphi \models A$ and $\varphi \models B$. Thus, if A and B are clauses, then $\varphi \models A \wedge B$ iff there are prime implicates of φ such that one subsumes A and the other subsumes B. More generally, if A_g, $i \leq z$, are clauses, then $\varphi \models \bigwedge_{g \leq z} A_g$ iff there are z prime implicates of φ, $C_{j_1}, ..., C_{j_z}$ such that $C_{i_g} \models A_g$, for all $g \leq z$.

We get that for z being the number of clauses that represent $G_{i_1} \vee ... \vee G_{i_u}$,

$$Filter[a](\varphi) = \bigwedge_{i_1,\ldots,i_u \leq m,\ \varphi \models G_{i_1} \vee \ldots \vee G_{i_u}} (F_{i_1} \vee \ldots \vee F_{i_u}) \equiv$$
$$\bigwedge_{i_1,\ldots,i_u \leq m,\ \bigwedge_{g \leq z} C_{j_g} \models \bigwedge_{g \leq z} A_g} (F_{i_1} \vee \ldots \vee F_{i_u}) \equiv$$
$$\bigwedge_{\exists j_1,\ldots,j_z; i_1,\ldots,i_u\ \ \bigwedge_{g \leq z} C_{j_g} \models \bigvee_{f \leq u} G_{i_f}} (F_{i_1} \vee \ldots \vee F_{i_u}) \equiv$$
$$\bigwedge_{j_1,\ldots,j_z \leq s} \bigwedge_{i_1,\ldots,i_u \leq m\ \ \bigwedge_{g \leq z} C_{j_g} \models \bigvee_{f \leq u} G_{i_f}} (F_{i_1} \vee \ldots \vee F_{i_u}) \equiv$$
$$\bigwedge_{j_1,\ldots,j_z \leq s} Filter[a](\bigwedge_{g \leq z} C_{j_g})$$

This shows that every conclusion of $Filter[a](\varphi)$ is a conclusion from some z clauses C_i for $i \leq s$. Thus, the conjunction of filtering the conjunction of every z clauses from φ results in a formula equivalent to the filtering of φ.

Now, we know that $Filter[a](\varphi)$ includes every clause C_i that satisfies $L(C_i) \cap Eff(a) = \emptyset$ (follows from (2)). It follows that we can eliminate from consideration in $\bigwedge_{j_1,\ldots,j_z \leq s} C_{j_g}$ those clauses C_i that satisfy $L(C_i) \cap (Eff(a) \cup Pre(a)) = \emptyset$, for $Pre(a) = \bigcup_{i \leq l} L(G_i)$, because every clause that they may imply is subsumed by the clause $G_{i_1} \vee ... \mathcal{G}_{i_u} = C_i$, for $G_{i_1}, ..., G_{i_u}$ being the set of preconditions of completion rules for literals that appear in C_i. Thus, these C_i's can be filtered separately from the rest. Call this set of indexes $\mathcal{I}_1$ (i.e., $\forall i \in \mathcal{I}_1\ L(C_i) \cap (Eff(a) \cup Pre(a)) = \emptyset$).

On the other hand, every prime implicate clause that we choose for the conjunction $\bigwedge_{j_1,\ldots,j_z \leq s} C_{j_g}$ determines a set of literals outside $Pre(a)$ that it can subsume, and that should be in the disjunction $G_{i_1} \vee ... \vee G_{i_u}$. We get that every conjunction of prime implicate clauses determines (uniquely) a disjunction $G_{i_1} \vee ... \vee G_{i_u}$ that it entails (it may entail a stronger formula, but the way we match clause sets with these disjunctions allows us the slack of knowing that if this formula is not the strongest we can entail, then there is another conjunction that will entail it for us). In particular, let $\mathcal{G}$ be the set of $g \leq z$ such that $L(C_{i_g}) \cap Pre(a) = \emptyset$. Then, $\bigwedge_{g \leq z} C_{j_g} \models G_{i_1} \vee ... \vee G_{i_u}$ implies that if $g \in \mathcal{G}$, then C_{i_g} subsumes $G_{i_1} \vee ... \vee G_{i_u}$. Thus, Filtering C_{i_g} implies something stronger than the one selected for $\bigwedge_{g \leq z} C_{j_g} \models G_{i_1} \vee ... \vee G_{i_u}$. Thus, we can filter all C_i's with $L(C_i) \cap Pre(a) = \emptyset$ separately from the rest.

We get that we can filter all clauses C_i such that $L(C_i) \cap (Pre(a) \cap Eff(a)) = \emptyset$ separately, and consider only clauses C_i with $L(C_i) \cap (Pre(a) \cap Eff(a)) \neq \emptyset$ in the second stage (when we choose z clauses and filter them together).

As above, for every disjunction $G_{i_1} \vee ... \vee G_{i_u}$ entailed by φ there is such a choice of z prime implicate clauses that entails this disjunction. The choice of those clauses is unique, as described above. Thus, when we choose a set of clauses it is enough to ignore the part of those clauses that is in $\overline{Pre(a)}$ and use the rest to find the effects that they generate in $Pre(a)$. In other words, it is enough to iterate over choices of disjunctions among $G_1, ..., G_l, \bar{G}$ and the literals in $Pre(a)$.

Here is the algorithm. For every choice from $G_1, ..., G_l, \bar{G}$ and the set of literals in $Pre(a) \setminus Eff(a)$ find the CNF representation of the disjunction of those G_i's (there are at most z clauses, as discussed above). For every clause C in this CNF select nondeterministically a prime implicate clauses C_i whose restriction to $Pre(a)$ subsumes C. The joint selection implies the disjunction of those G_i's and some literals from $\overline{Pre(a)}$. Add the proper disjunction $F_{i_1} \vee ... \vee F_{i_u}$ to the result of the filtering. Also, if $\bar{G}$ is in the set of original G_i's that we selected, then let A be the set of literals of $Eff(a)$ that appear in some C_i. Nondeterministically select a subset of A and let D be the disjunction of those literals. If the conjunction of our chosen clauses implies the disjunction of G_i, when we replace $\bar{G}$ with D, then add the disjunction of the original F_i and D to the filtering. (This takes care of the case that we prove the disjunction of some effect-rule preconditions and the preconditions of some $Eff(a)$-literals completion rules.)

This algorithm has $2^{l \cdot |Pre(a) \cup Eff(a)|}$ possible implications to check. Each implication involves generating the z-sized CNF, and every clause may match s subsuming clauses from φ. For each combination of such clauses from φ we generate their implied result in the filtered formula. Thus, the time to compute all the implied clauses from this filtering algorithm is $O(2^{l \cdot |Pre(a) \cup Eff(a)|} \cdot (s^z + z))$.

For the case of $\models \neg\bar{G}$ we know that there is no need to include $\bar{G}$ in the selection algorithm above. Also, we can omit the second part of the algorithm above. This means that in this case there are only 2^l possible implications to check. Consequently, the time of the resulting algorithm in this case is $O(2^l \cdot (s^z + z))$. ■

A.9 Proof of Theorem 5.1

PROOF First, we notice that the following hold in a STRIPS domain for every action a:

$$Filter[a](TRUE) \equiv F_1 \wedge Cn^{\overline{Eff(a)}}(G_1).$$

Let $T_a = Filter[a](TRUE)$. We get that for a literal l,

$$Filter[a](l) \equiv \begin{cases} T_a & l \in \mathcal{L}(Eff(a)) \\ T_a \wedge l & l \notin \mathcal{L}(Eff(a)) \end{cases}$$

Thus, for a set of literals $l_1, ..., l_k$ we get that equation (5) holds.

Thus, the main computation involved in filtering a k-CNF formula is computing T_a. In turn, the only computation needed there is finding the k-CNF form of $Cn^{\overline{Eff(a)}}(G_1)$. This can be done in time $O(2^t)$ and we need to do it only once per action (in fact, we can do that prior to the computation of filtering, and speed up the real-time filtering of the belief state).

Then, we use a simplified version of the algorithm described in Figure 1. We break the k-CNF formula into its clauses, and filter each of them separately using the equivalence above. This is done in time $O(k)$ per clause, thus yielding a total time of $O(s \cdot k)$.

To see that we maintain a k-CNF all is needed is to notice that each conjunction that results from filtering a clause in φ is a clause in F_1 (which has at most k literals) or a clause with literals in $L(\overline{Eff(a)}) \cap L(G_1)$ (which has at most $t \leq k$ literals). The only other clause that can be in the result of the filtering is a clause C_i with $L(C_i) \cap Eff(a) = \emptyset$. This C_i has k literals because φ is a k-CNF formula. ■

A.10 Proof of Theorem 5.2

PROOF From formula (2) and the proof of Theorem 4.5 we already know that clauses C_i with $L(C_i) \cap (Pre(a) \cup Eff(a)) = \emptyset$ can be filtered separately.

Looking at equivalence (2) and the proof of Theorem 4.5 we see that every disjunction $\bigwedge_{f \leq u} G_{i_f}$ is implied by some conjunction of prime implicate clauses from φ such that $L(C_i) \cap Pre(a) \neq \emptyset$.

However, G_1 always holds because this is a STRIPS domain. On the other hand, the only disjunctions that are not subsumed by G_1 are those that do not include it, i.e., those that include literals that do not appear in $Eff(a)$ (they belong to completion rules that do not include $\neg\bar{G}$ as a precondition, i.e., literals in $\overline{Eff(a)}$). Thus, each of these disjunctions is a single clause C. If $\varphi \models C$, then there is a clause in $\{C_i\}_{i \leq s}$ such that C_i subsumes C because all the prime implicates of φ are in $\{C_i\}_{i \leq s}$.

Thus, for some C_i, $Filter[a](C_i)$ will imply the matching clause sanctioned by (2). We get that every result of $Filter[a](\bigwedge_{i \leq s} C_i)$ is implied by $\bigwedge_{i \leq s} Filter[a](C_i)$, i.e., $Filter[a](\bigwedge_{i \leq s} C_i) \models \bigwedge_{i \leq s} Filter[a](C_i)$. Corollary 3.2 provides the opposite direction. ■

A.11 Proof of Corollary 5.3

PROOF Theorem 5.2 guarantees that we can filter each of the clauses separately and conjoint the result. However, from equivalence (5) we know that the filtering of each clause is equivalent to $Filter[a](TRUE)$ conjoined with either that clause or *TRUE*.

First, look at the clauses that are kept from φ and those prime implicates that are given by $Cn^{\overline{Eff(a)}}(G_1)$. For every original clause that is kept for the new belief state (after filtering with a), either a it is a prime implicate in the new belief state or it is subsumed by a prime implicate of $Cn^{\overline{Eff(a)}}(G_1)$. If it is not subsumed, then it is a prime implicate by definition. If it is subsumed, then the clause that subsumed it is a prime implicate by definition. A similar situation holds in the other direction (clauses of $Cn^{\overline{Eff(a)}}(G_1)$). In either case, those clauses that are left are in k-CNF. Since they are all in $\mathcal{L}(overlineEff(a))$, there is nothing else that can subsume them (the only other thing we add is F_1).

Finally, we look at the prime implicates of F_1. They are in the language $\mathcal{L}(Eff(a))$. For every original clause that intersects with this language, that clause is not transferred to the new belief state. Thus, there is no clause that can subsume any prime implicate of F_1. Since every prime implicate of F_1 has at most k literals, the resulting belief state is a prime implicate belief state that is in k-CNF. ■

References

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

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

[Baral, Kreinovich, & Trejo, 2001] Baral, C.; Kreinovich, V.; and Trejo, R. A. 2001. Computational complexity of planning with temporal goals. In *Proc. Seventeenth International Joint Conference on Artificial Intelligence (IJCAI '01)*, 509–514. Morgan Kaufmann.

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

[Bertoli *et al.*, 2001] Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proc. Seventeenth International Joint Conference on Artificial Intelligence (IJCAI '01)*, 473–478. Morgan Kaufmann.

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

[Craig, 1957] Craig, W. 1957. Linear reasoning. a new form of the herbrand-gentzen theorem. *Journal of Symbolic Logic* 22:250–268.

[del Val, 1999] del Val, A. 1999. A new method for consequence finding and compilation in restricted language. In *Proc. National Conference on Artificial Intelligence (AAAI '99)*, 259–264. AAAI Press/MIT Press.

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

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

[Fikes, Hart, & Nilsson, 1981] Fikes, R.; Hart, P.; and Nilsson, N. 1981. Learning and executing generalized robot plans. In Webber, B., and Nilsson, N., eds., *Readings in Artificial Intelligence*. Morgan Kaufmann. 231–249.

[Gelfond & Lifschitz, 1998] Gelfond, M., and Lifschitz, V. 1998. Action languages. *Electronic Transactions on Artificial Intelligence (http://www.etaij.org)* 3:nr 16.

[Gentzen, 1969] Gentzen, G. 1969. *The Collected Papers of Gerhard Gentzen, edited by M. E. Szabo*. Amsterdam: North Holland.

[Giunchiglia, Kartha, & Lifschitz, 1997] Giunchiglia, E.; Kartha, G. N.; and Lifschitz, V. 1997. Representing Action: Indeterminacy and Ramifications. *Artificial Intelligence* 95(2):409–438.

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

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

[Kautz, McAllester, & Selman, 1996] Kautz, H.; McAllester, D.; and Selman, B. 1996. Encoding plans in propositional logic. In Doyle, J., ed., *Proceedings of KR'96*, 374–384. Cambridge, Massachusetts: KR.

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

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

[Lifschitz, 2000] Lifschitz, V. 2000. Missionaries and cannibals in the causal calculator. In *Principles of Knowledge Representation and Reasoning: Proc. Seventh Int'l Conference (KR '2000)*, 85–96. Morgan Kaufmann.

[Lin & Reiter, 1994] Lin, F., and Reiter, R. 1994. State constraints revisited. *Journal of Logic and Computation* 4(5):655–678.

[Lin & Reiter, 1997] Lin, F., and Reiter, R. 1997. How to Progress a Database. *Artificial Intelligence*. to appear.

[Lin, 1996] Lin, F. 1996. Embracing causality in specifying the indeterminate effects of actions. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

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

[Rintanen, 2002] Rintanen, J. T. 2002. Backward plan construction for planning with partial observability. In *Proceedings of the 6th Int'l Conf. on AI Planning and Scheduling (AIPS'02)*. AAAI Press.

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

[Thielscher, 2000] Thielscher, M. 2000. Modeling actions with ramifications in nondeterministic, concurrent, and continuous domains—and a case study. In Kautz, H., and Porter, B., eds., *Proc. National Conference on Artificial Intelligence (AAAI '00)*, 497–502. Austin, TX: MIT Press.

[Wiener, 1949] Wiener, N. 1949. *Extrapolation, Interpolation and Smoothing of Stationary Time Series. With Engineering Applications*. MIT Press.