Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Logical Filtering and Smoothing: State Estimation in Partially Observable Domains

Brent Mombourquette,[†] Christian Muise,[*] Sheila A. McIlraith[†]
[†]Department of Computer Science, University of Toronto
[*]CSAIL, Massachusetts Institute of Technology
[†]{bgmomb,sheila}@cs.toronto.edu, [*]cjmuise@mit.edu

Abstract

State estimation is the task of estimating the state of a partially observable dynamical system given a sequence of executed actions and observations. In logical settings, state estimation can be realized via logical filtering, which is exact but can be intractable. We propose logical smoothing, a form of backwards reasoning that works in concert with approximated logical filtering to refine past beliefs in light of new observations. We characterize the notion of logical smoothing together with an algorithm for backwards-forwards state estimation. We also present an approximation of our smoothing algorithm that is space efficient. We prove properties of our algorithms, and experimentally demonstrate their behaviour, contrasting them with state estimation methods for planning. Smoothing and backwards-forwards reasoning are important techniques for reasoning about partially observable dynamical systems, introducing the logical analogue of effective techniques from control theory and dynamic programming.

Introduction

Many applications of artificial intelligence from automated planning and diagnosis to activity recognition require reasoning about dynamical systems that are only partially observable. A necessary component of such systems is *state estimation* – the task of estimating the state of the systems given a sequence of executed actions and observations. With stochastic transition systems, state estimation is commonly realized via filtering, of which Kalman filtering (Kalman 1960) is a well-known example. In logical settings, an analogous form of *logical filtering* was proposed by Amir and Russell [2003] in which an agent's belief state – the set of possible world states – can be compactly represented as a formula, and filtering is a form of belief update. While logical filtering is intractable in the general case (Eiter and Gottlob 1992), there are tractable subclasses often involving restricted transition systems or compact encodings of the belief state (e.g., (Shahaf and Amir 2007; Shirazi and Amir 2011)). Unfortunately, typical belief state representations often require further inference to ascertain beliefs about individual fluents – a frequent and time critical component of many decision-making systems.

Our concern is with logical state estimation in service of tasks such as planning, execution monitoring, diagnosis, and activity recognition. We are particularly concerned with systems that include a rich characterization of how the actions of an agent indirectly affect their environment. These are typically captured by causal or ramification constraints (e.g., *a causal constraint might say that if the battery and radio are ok and the radio is on then sound is emitted.*). We assume that such constraints are compiled into the transition system as additional effects of actions following, e.g., (Pinto 1999; Strass and Thielscher 2013; Baier, Mombourquette, and McIlraith 2014). In planning such constraints tend to create problems with large conformant width (Palacios and Geffner 2009).

We exploit the observation that only a subset of the state is necessary to track. Planning systems need to know when actions are applicable, and when the goal is reached (Bonet and Geffner 2014). Execution monitoring systems need only track the conditions under which a plan remains valid (e.g., (Fikes, Hart, and Nilsson 1972; Fritz and McIlraith 2007)). Diagnosis systems track the confirmation and refutation of candidate diagnoses. These observations motivate the development of state estimation techniques tailored to the task of tracking the truth of (conjunctions of) fluent literals. In Section we formalize state estimation as semantic logical filtering and propose a sound under-approximation that is computationally appealing. Motivated by the technique of *smoothing* for stochastic systems (e.g., (Einicke 2012)), in Section , we introduce the notion of *logical smoothing*, which allows for the updating of beliefs about the past in light of observations about the present. In Section 7, we propose an algorithm for *backwards-forwards reasoning* that combines smoothing and filtering in order to perform state estimation. The application of (approximate) logical smoothing mitigates for the incompleteness of approximate logical filtering, while preserving many of its computational advantages. We evaluate our approach and discuss related work.

The Problem: State Estimation

State estimation is a core task in reasoning about dynamical systems with partial observability. Informally, the state estimation task we address is:

Given a dynamical system, a belief state, and a sequence of executed actions and observations, infer the

resulting belief state of the system.

For logical theories, state estimation is captured by logical filtering (Amir and Russell 2003).

Example: Consider the simplified action-observation sequence in support of diagnosing a car. For simplicity of exposition, we assume that actions have no preconditions – they are always executable – and only $\neg sound$ known in the initial state. The action $turn\_ignition$, results in $ignition\_turned$. If $ignition\_turned$, $battery\_ok$ and gas_ok hold, then so will $car\_started$. You $turn\_ignition$ and observe ($\neg car\_started$), and so, under the assumption that the characterization of the vehicle functioning is complete, you can infer $\neg battery\_ok \vee \neg gas\_ok$. You $turn\_on\_radio$, causing $radio\_on$ as well as $sound$ if $battery\_ok \wedge radio\_ok$. You observe ($sound$). Under completeness and frame assumptions, you are now able to infer $radio\_ok$, $battery\_ok$ and $\neg gas\_ok$. So following the action-observation sequence ($turn\_ignition$,$\neg car\_started$,$turn\_on\_radio$,$sound$), your estimated belief state comprises just one state here represented by the set of fluents $\{ignition\_turned$, $\neg car\_started$, $radio\_on$, $battery\_ok$, $radio\_ok$, $\neg gas\_ok$, $sound\}$.

We appeal to standard finite domain planning language syntax. A dynamical system is a tuple $\Sigma = \langle \mathcal{F}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$, where $\mathcal{F}$ is a finite set of propositional fluent symbols such that if $p \in \mathcal{F}$, then p and $\neg p$ are *fluent literals*, $\mathcal{S} = Pow(\mathcal{F})$ is the set of *possible world states*, $\mathcal{A}$ is a set of actions including sensing actions, $R \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition relation (equivalently we use the notation $(s, a, s') \in R$ or $s' = R(s, a)$), and $\mathcal{I}$ is a set of clauses over $\mathcal{F}$ that defines a set of possible initial states, collectively – the initial belief state. As noted in the introduction, we assume that causal constraints are compiled into our transition system as extra effects of actions (e.g., following (Pinto 1999)). For the purposes of this paper, non-sensing actions $a \in \mathcal{A}$ are assumed to be deterministic and are defined by a precondition $prec(a)$, which is a conjunction of fluent literals, and $\mathit{eff}(a)$, a set of conditional effects of the form $C \rightarrow L$, where C is a conjunction of fluent literals and L is a fluent literal. We write the unconditional effect $true \rightarrow L$ as simply L, and use $true$ to denote an empty precondition. Each sensing action is defined solely by its precondition $prec(a)$, which is a conjunction of fluent literals, and $obs(a)$, which is the fluent literal that is observed by the sensing action, following in the spirit of sensing actions in the situation calculus (Reiter 2001). Note that our account of logical filtering and smoothing supports both the use of explicit sensing actions, and the scenario where observations are automatically returned following an action to create an action-observation sequence. Finally, we assume there are no exogenous actions.

Throughout this paper we take the viewpoint that the state of Σ represents the belief state of the agent. The semantics of logical filtering is defined by considering the *belief state* to be a set of possible world states $\rho \subseteq S$. Since there are $2^{2^{|\mathcal{F}|}}$ belief states, algorithms for logical filtering typically represent the belief state (henceforth simply *state*), ρ, compactly as a logical formula ϕ, called a *belief-state formula* or *state formula*. Later we will be restricting ϕ to a conjunction of fluent literals, sometimes denoted as a set of fluent literals and referred to as a *conjunctive state formula*. Following Amir and Russell (2003):

Definition 1 (Logical Filtering Semantics (Amir, Russell)) *Given belief state $\rho \subseteq S$ of dynamical system Σ, the filtering of a sequence of actions and observations $\langle a_1, o_1, \ldots, a_t, o_t \rangle$ with respect to ρ is defined as:*

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

We call step 2 progression with a and step 3 filtering with o.

When action a is filtered, every state $s \in \rho$ is updated via transition system $R(s, a)$. When an observation o is filtered, every state inconsistent with the observation is eliminated. This results in an interleaving of action progression and observation filtering (e.g., (Vassos and Levesque 2013)).

Logical filtering can cause fundamental inference operations to be intractable, such as inferring beliefs about individual fluents (Shahaf and Amir 2007). A core component of automated planning is determining the applicability of actions during the search for a plan, as well as determining whether the goal condition has been achieved. Similarly dynamical diagnosis requires determination of the refutation or confirmation of candidate diagnoses as the result of (treatment) actions and sensing (e.g., (McIlraith and Reiter 1992)). Diagnoses, action preconditions, and planning goals are typically represented as conjunctions of fluents, motivating us to propose *approximate logical filtering* which describes the belief state in terms of the subset of the fluent literals entailed by the belief state:

Definition 2 (Approximate Logical Filtering) *Given belief-state formula ϕ of dynamical system Σ, the approximate filtering of a sequence of actions and observations $\langle a_1, o_1, \ldots, a_t, o_t \rangle$ with respect to ϕ is defined as:*

1. $\mathsf{Filter_a}[\langle\rangle](\phi) = \phi$;
2. $\mathsf{Filter_a}[\mathsf{a}](\phi) = \bigwedge\{L \mid (C \rightarrow L) \in eff(a) \wedge \phi \models C\}$
   $\bigwedge\{L \mid \phi \models L \wedge\ \forall (C \rightarrow \neg L) \in eff(a),\ \phi \models \neg C\}$;
3. $\mathsf{Filter_a}[\mathsf{o}](\phi) = \phi \wedge o$;
4. $\mathsf{Filter_a}[\langle \mathsf{a_i}, \mathsf{o_i}, \ldots, \mathsf{a_t}, \mathsf{o_t} \rangle](\phi) =$
   $\mathsf{Filter_a}[\langle \mathsf{a_{i+1}}, \mathsf{o_{i+1}}, \ldots, \mathsf{a_t}, \mathsf{o_t} \rangle](\mathsf{Filter_a}[\mathsf{o_i}](\mathsf{Filter_a}[\mathsf{a_i}](\phi)))$

One can see from line 2 that progressing a state ϕ through an action a produces a state consisting of what was known to be *caused* (directly or indirectly) by a and what was known to *persist* through a.

Example: Returning to our car example, we see how approximate logical filtering provides a weak approximation. After the previously noted sequence of actions and observing ($\neg car\_started$) and ($sound$) the resulting approximated belief state is $\{ignition\_turned$, $radio\_on$, $\neg car\_started$,

$sound\}$ missing $radio\_ok$, $battery\_ok$ and $\neg gas\_ok$ which full logical filtering would produce.

Theorem 1 (Sound Under Approximation) *Given dynamical system, Σ, belief-state formula ϕ representing possible belief state $\rho \subseteq S$, and action-observation sequence $\langle a_1, o_1, \ldots, a_t, o_t\rangle$,* $\mathsf{Filter}[\langle \mathsf{a_1}, \mathsf{o_1}, \ldots, \mathsf{a_t}, \mathsf{o_t}\rangle](\rho) \models \mathsf{Filter_a}[\langle \mathsf{a_1}, \mathsf{o_1}, \ldots, \mathsf{a_t}, \mathsf{o_t}\rangle](\phi)$.

This follows from Definitions 1 and 2. Approximate Logical Filtering is not complete with respect to logical filtering semantics.

Proposition 1 (Conjunctive State Formula Preservation) *Given dynamical system, Σ, conjunctive state formula ϕ, and action-observation sequence $\langle a_1, o_1, \ldots, a_t, o_t\rangle$, where each o_i is a conjunctive formula,* $\mathsf{Filter_a}[\langle \mathsf{a_1}, \mathsf{o_1}, \ldots, \mathsf{a_t}, \mathsf{o_t}\rangle](\phi)$ *is a conjunctive formula.*

The above proposition follows naturally from lines 2 and 3 of Definition 2 and is key to our complexity results and tractable approximate representations.

Theorem 2 (Complexity) *Given dynamical system, Σ, conjunctive state formula ϕ, and action-observation sequence $\langle a_1, o_1, \ldots, a_t, o_t\rangle$, where each o_i is a conjunctive formula,* $\mathsf{Filter_a}[\langle \mathsf{a_1}, \mathsf{o_1}, \ldots, \mathsf{a_t}, \mathsf{o_t}\rangle](\phi)$ *is computable in time $O(t \cdot c \cdot |\mathcal{F}|)$ where c is the maximum number of conditional effects over actions in the sequence.*

This follows from fluent entailment of a conjunctive formula and the limit of $|\mathcal{F}|$ fluents per consistent conditional effect. For the many tasks, including planning, where observations are conjunctive formulae, approximate logical filtering enables state estimation to reduce to set operations. While computationally appealing, the approximation is weak and thus of limited use on its own for AI planning, dynamical diagnosis, and similar tasks. In the next section, we combine approximate logical filtering with a new approach for reasoning over the past – logical smoothing.

Logical Smoothing

Filtering with a stochastic transition system involves estimating the marginal posterior probability distribution of the system conditioned on data that was received *prior* to the current state. There is an analogous concept for estimating not the current state but instead a *previous* state. This is called *smoothing* – refining a previous state estimation (e.g., (Einicke 2012)). For stochastic models, this amounts to a re-computation of the marginal posterior probability distribution. We can carry this idea to the logical setting and show how observations can be used to refine estimates of past states. The notion of logical smoothing is only of interest if the belief state is approximated in some way. For example, in later sections, we show how logical smoothing can be used to improve weak approximations of logical filtering and thus produce a refined estimate of the current state.

We begin our treatment of logical smoothing by defining the semantics of logical smoothing with respect to a belief state. Given a representation of the sequence of actions, observations, and intermediate state estimates, *logical smoothing* recursively refines previous state estimates through a backwards update procedure. To this end, we store previous state estimates, coupled with the actions executed to construct successor states in a so-called *history*. Note that while we use a set of possible worlds to represent a belief state, histories can be defined with any sort of state representation (logical formulae, sets of fluents understood as a logical conjunction, etc.), which we exploit later in this section.

Definition 3 (Belief State History) *Given dynamical system, Σ, a belief state history over Σ is a sequence of tuples (ρ_0, a_0),(ρ_1, a_1),$\ldots$ (ρ_n, a_n) such that each ρ_i is a belief state, a set of possible world states $\rho_i \subseteq S$ and each a_i is an action of Σ.*

The intent of a history is to capture the evolution of the system starting from some designated initial belief state ρ_0. Tuple (ρ, a) records the belief state, ρ, prior to the execution of action a. The observations are not modeled as separate entities. Rather, they reside in the intermediate belief states, presumably as part of an original filtering process.

Definition 4 (Logical Smoothing Semantics) *Given belief state $\rho \subseteq S$ of dynamical system Σ, the smoothing of a belief state history (ρ_0, a_0),(ρ_1, a_1),$\ldots$ (ρ_n, a_n) with respect to ρ is defined as:*

1. $\mathsf{Smooth}[\langle\rangle](\rho) = \rho$
2. $\mathsf{Smooth}[\mathsf{o}](\rho) = \mathsf{Filter}[\mathsf{o}](\rho)$
   $= \{s \mid s \in \rho$ *and* o *is true in* $s\}$
3. $\mathsf{Smooth}[(\rho', \mathsf{a}')](\rho) =$
   $\rho' \cap \{s \mid s \in \mathsf{PreImage}(s', a'), s' \in \rho\}$
4. $\mathsf{Smooth}[\langle(\rho_0, \mathsf{a_0}), (\rho_1, \mathsf{a_1}), \ldots (\rho_\mathsf{n}, \mathsf{a_n})\rangle](\rho) =$
   $\mathsf{Smooth}[\langle(\rho_0, \mathsf{a_0}), \ldots (\rho_{\mathsf{n}-1}, \mathsf{a_{n-1}})\rangle]$
   $(\mathsf{Smooth}[(\rho_\mathsf{n}, \mathsf{a_n})](\rho))$

   where $\mathsf{PreImage}(s', a) = \{s \mid s' = R(s, a)\}$

Logical smoothing works by propagating acquired information (typically an observation or the resultant filtered state-action pair) back through a given history and updating its constituent state estimates. Note that this is more akin to *belief updating* than to *belief revision*. Each smoothing step refines previous state estimates in that the smoothed state's set of possible world states are always a (non-strict) subset of the state's original set of possible world states.

With our semantics in hand, Definition 6 provides a formal characterization of logical smoothing with respect to a compact belief-state formula representation of the history.

Definition 5 (History) *Given dynamical system, Σ, a history H over Σ is a sequence of tuples $(\sigma_0, a_0), (\sigma_1, a_1), \ldots (\sigma_n, a_n)$ such that each σ_i is a belief-state formula over $\mathcal{F}$ and each a_i is an action of Σ.*

When there are no active sensing actions and the system instead just returns observations, a dummy sensing action,

obs, can be added to the history tuple and correspondingly defined in the domain to have no preconditions or effects.

Definition 6 employs *regression* (e.g., (Reiter 2001)) to propagate updates back through the history. Denoted $\mathcal{R}[\phi, a]$, regression takes a logical formula ϕ and action a and rewrites the formula in terms of the weakest conditions necessary for ϕ to hold after executing a. I.e., if $s' = R(s, a)$, the result of applying a in s, then

$$s' \models \phi \text{ if and only if } s \models \mathcal{R}[\phi, a]$$

We appeal to Reiter's definition of regression in the situation calculus with syntactic notation suitably modified, and readers are referred to (Reiter 2001) for further details. The one modification we make is to include an action's precondition in the regression: i.e., $\mathcal{R}[\phi, a] \models prec(a)$. We do so because we are regressing solely over actions we know to have been executed, so it follows that their preconditions hold in the state in which they were executed.

When the history comprises a single state-action pair, ϕ is regressed through the action and conjoined to the state formula. Otherwise, if the history is a sequence of state-action pairs, then ϕ is regressed step by step through the history, and new information garnered from the regression, ϕ_{NEW}, is conjoined to the associated state formula. $PI(\phi)$ refers to the prime implicates of formula ϕ.

Definition 6 (Logical Smoothing) *Given dynamical system Σ, history $\mathsf{H} = (\sigma_0, a_0), \ldots (\sigma_n, a_n)$ and formula ϕ, the logical smoothing of H with respect to ϕ is defined as:*

1. $\mathsf{Smooth}[(\sigma, \mathsf{a})](\phi) = (\sigma \wedge \mathcal{R}[\phi, a], a)$
2. $\mathsf{Smooth}[(\sigma_0, \mathsf{a}_0), \ldots, (\sigma_\mathsf{n}, \mathsf{a}_\mathsf{n})](\phi) =$
   $\mathsf{Smooth}[(\sigma_0, \mathsf{a}_0), \ldots, (\sigma_{\mathsf{n}-1}, \mathsf{a}_{\mathsf{n}-1})](\phi_{\mathsf{NEW}}),$
   $\mathsf{Smooth}[(\sigma_\mathsf{n}, \mathsf{a}_\mathsf{n})](\phi)$

where $\phi_{NEW} = \bigwedge\{\varphi \in PI(\mathcal{R}[\phi, a_{n-1}]) \mid \sigma_{n-1} \not\models \varphi\}$

The soundness and completeness of Logical Smoothing (Definition 6) relative to the semantic account in Definition 4 follow straightforwardly from the correspondence between Prelmage and regression.

Example: We return to our car example, this time with explicit sensing actions included to help with the exposition. Approximate logical filtering the initial state, σ_0, with action $turn\_ignition$ yields state σ_1 as $\sigma_0 \wedge ignition\_turned$. $car\_started$ is unknown in σ_1 as it's outcome is predicated on fluents whose truth values are unknown. Action $obs\_car\_started$ yields observation ($\neg car\_started$). The smoothed history is $\mathsf{Smooth}[\mathsf{h}_0, \mathsf{h}_1](\neg\mathsf{car\_started})$ where h_0=$(\sigma_0, turn\_ignition)$ and h_1=$(\sigma_1, obs\_car\_started)$. Following Definition 6, this is equal to $\mathsf{Smooth}[(\mathsf{h}_0)](\phi_{\mathsf{NEW}}), \mathsf{Smooth}[(\mathsf{h}_1)](\neg\mathsf{car\_started})$ (line 2). σ_1 of h_1 is smoothed to $\sigma_1 \wedge \neg car\_started$ (line 1). Action $obs\_car\_started$ has no effects, so observation ($\neg car\_started$) must hold in the previous state. ϕ_{NEW} hinges on $\mathcal{R}[\neg car\_started, turn\_ignition]$. By the regression rewriting and frame axioms, this gives $\phi = \neg car\_started \wedge (\neg battery\_ok \vee \neg gas\_ok)$. Since $\sigma_0 \models \neg car\_started$, $\phi_{NEW} = \neg battery\_ok \vee \neg gas\_ok$ after restricting to prime implicates, the intuitive refinement of the initial state given the observation.

Algorithm 1: $\mathsf{LSmooth}(\mathsf{H}, \phi, i)$ Perform logical smoothing on history H given that ϕ holds at σ_i in H. Returns updated H and the index of termination.

```
1 σ_i = σ_i ∧ φ
2 if i > 0 then
3 |   ψ = R[φ, a_{i-1}]
4 |   ψ_NEW = ⋀{φ ∈ PI(ψ) | σ_{i-1} ⊭ φ}
5 |   if ψ_NEW is not empty then
6 |   |   return LSmooth(H, ψ_NEW, i − 1)
7 return (H, i)
```

Algorithm 1 realizes Logical Smoothing together with an optimization to support early termination of the regression. Included in its input is a state index parameter, identifying the location within history H where ϕ is to be integrated. Logical smoothing can thus acquire information about a past state and smooth the preceding state estimates in light of it, as well as smoothing from the most recent state. The heart of the smoothing procedure lies within $\mathcal{R}[\phi, a_{i-1}]$ (line 3) as explained above. Line 4 identifies those aspects of ψ that are new to state σ_{i-1} by identifying prime implicates of ψ not entailed by σ_{i-1}. This is an optimization as it allows for early termination (line 5) when further smoothing would be unnecessary and it minimizes the subsequent formula to be regressed in the next iteration. The process then repeats on the newly computed ψ_{NEW} formula at index $i - 1$. Finally, $\mathsf{LSmooth}$ returns a tuple of the refined history and the index of termination i. The purpose of returning i will become apparent in Section 7 when we leverage logical smoothing in a state estimation algorithm.

Proposition 2 (Early Termination Completeness) *Given a dynamical system Σ and a history H over Σ with at least n state-action tuples and a formula ϕ over $\mathcal{F}$, the updated history returned by $\mathsf{LSmooth}(\mathsf{H}, \phi, n)$ is the updated history returned by $\mathsf{LSmooth}(\mathsf{H}, \phi, n)$ with the early termination condition of line 5 removed.*

It follows straightforwardly from the close correspondence between this observation and Definition 6 and Algorithm 1, and Proposition 2 that the history computed in Algorithm 1 is equivalent to the specification in Definition 6.

We say that $\mathsf{H}' = (\sigma'_0, a'_0), \ldots, \sigma'_n, a'_n)$ is a *sound and complete refinement* of $\mathsf{H} = (\sigma_0, a_0), \ldots, (\sigma_n, a_n)$ with respect to formula ϕ that holds at index i of H if for all $0 \leq j \leq n$, (1) $a'_j = a_j$; (2) $\sigma'_j \models \sigma_j$; and (2) if $j < i$, $\mathsf{Filter}[\langle \mathsf{a}'_\mathsf{j}, \sigma'_{\mathsf{j}+1}, ..., \mathsf{a}'_\mathsf{i}, \sigma'_\mathsf{i}\rangle](\sigma'_\mathsf{j}) \models \phi$. In other words, a sound and complete refinement captures all of what must be known based on the existing history and necessary conditions for ϕ. Note that we substitute observations with belief states to ensure all knowledge from observation actions are accounted for. With this definition in hand we have,

Theorem 3 (Soundness and Completeness) *Given a dynamical system Σ, history H over Σ, and formula ϕ over $\mathcal{F}$ that must be true at σ_i of H,* $\mathsf{LSmooth}(\mathsf{H}, \phi, i)$ *produces a sound and complete refinement of* H.

Logical smoothing provides a sound, complete, and principled approach to smoothing previous state estimations in a logical system in light of additional observations or information which must hold in the associated state. As this is merely the general algorithmic structure of logical smoothing, further optimizations are possible but omitted for clarity of exposition of the core concepts.

Approximate Logical Smoothing

Unfortunately, as with *unapproximated* logical filtering, the querying of belief states resulting from logical smoothing may not be tractable. Further, while a single regression step results in a linear increase in formula size with respect to the input, recursive applications of regression result in an exponential blow up (Fritz 2009). To remedy these issues, we analogously define a procedure for *approximate logical smoothing*, $\mathsf{LSmooth_a}$. $\mathsf{LSmooth_a}$ is $\mathsf{LSmooth}$ outlined in Algorithm 1, with the following modification:

$$\text{Line 4: } \psi_{NEW} = \bigwedge\{f \mid \psi \wedge \sigma_{i-1} \models f \text{ and } \sigma_{i-1} \not\models f\} \quad (1)$$

where f is restricted to fluent literals. This approximation limits the updating of the history to fluent literals entailed by these regressed additions, and not already present in the history. Note that this is but one way to deal with the formula size increase from repeated regression. Alternatives that preserve completeness include adding new fluents in place of certain sub-formulae of the regressed formula (van Ditmarsch, Herzig, and Lima 2007) or representing formulae as circuits (Rintanen 2008).

Example: In the previous example, smoothing with respect to the observation ($\neg car\_started$) refined the initial state with $\neg battery\_ok \vee \neg gas\_ok$. In contrast, if we perform approximate smoothing of ($\neg car\_started$), captured by the $\mathsf{LSmooth_a}$ algorithm, there is no refinement as disjunctive information is lost. Continuing with the sequence of actions, $turn\_on\_radio$ is executed and ($sound$) is observed. Since the previous state entails $\neg sound$, the regression as part of the approximate smoothing would produce the inference $battery\_ok \wedge radio\_ok$. Therefore, even with the approximation, we refine the estimate of the prior state, and all states back to the initial state, with $battery\_ok \wedge radio\_ok$.

Proposition 3 (Soundness) *Let Σ be a dynamical system, H be a history over Σ of n state-action tuples, and ϕ be a formula over $\mathcal{F}$ that must be true at the state at index i of H. If $\mathsf{LSmooth}(\mathsf{H}, \phi, i) = (\mathsf{H}', j)$ and $\mathsf{LSmooth_a}(\mathsf{H}, \phi, i) = (\mathsf{H}'', k)$ then for all $0 \leq i \leq n$, if σ'_i is the i-th state of H' and σ''_i is the i-th state of H'' then $\sigma'_i \models \sigma''_i$.*

By Proposition 3, $\mathsf{LSmooth_a}$ is an under-approximation of $\mathsf{LSmooth}$. It amounts to a version that smooths only with respect to conjunctive formulae. Moreover, analogously to approximate logical filtering (Proposition 1), approximate logical smoothing is conjunctive state formula preserving.

Note that $\mathsf{LSmooth_a}$ maintains the restriction to conjunctive formulae avoiding the potential blowup of formulae, resulting from repeated regression rewriting.

Theorem 4 (Complexity) *Given a history H over a dynamical system Σ of n state-action tuples such that all states are conjunctive formulae, a conjunctive formula ϕ over $\mathcal{F}$, and an index i of H, $\mathsf{LSmooth_a}(\mathsf{H}, \phi, i)$ can be computed in time $O(n \cdot 2^{|\mathcal{F}|})$ with propositional entailment or $O(n \cdot |\mathcal{F}|^2)$ with unit propagation entailment.*

While the worst case complexity is exponential, in practice this is not the case. Actions typically trigger few indirect effects (ramifications) relative to the size of the domain. This results in a compact regressed formula with unit entailments computable through unit propagation in most cases. Since we place no restrictions on the syntactic form of regressed formulae (e.g., Horn clauses) unit propagation may not produce all entailments, resulting in a sound under-approximation.

Backwards-Forwards Algorithm

Logical smoothing refines previous state estimates by reasoning *backwards* (regressing) over the history of actions and states, with respect to some acquired information, removing some uncertainty about the past, particularly in dynamical systems where actions have causal ramifications. This information can then be propagated *forwards* (progressed) through the state-action history to potentially produce further refinements.

Example: As we last left the car example from Section 7, each state in the history, including the initial state, was smoothed with $battery\_ok \wedge radio\_ok$ after observing ($sound$). While it is obvious that this should be propagated forward to the current state, as it stands it is not so obvious how and why, in general, this should be done. Consider the case where the action $turn\_ignition$ actually had an *additional* effect $battery\_charging$ if $battery\_ok$. Propagating $battery\_ok$ forward from the initial state given the actions and intermediate states in the history further refines the post-$turn\_ignition$ state estimates (including the current state) with $battery\_charging$.

We can realize the forward phase outlined above via filtering. Note that when operating on a history, simulating the forwards reasoning phase by filtering works on the corresponding sequence of actions with observation formulae being the state formula. For notational convenience, we define $\text{subseq}(\mathsf{H}, i)$ of a history $\mathsf{H} = (\sigma_0, a_0), (\sigma_1, a_1), \ldots, (\sigma_n, a_n)$ and index $0 \leq i \leq n$ as the sub-sequence $\langle a_i, \sigma_{i+1}, a_{i+1}, ..., \sigma_n, a_n \rangle$.

Algorithm 2 outlines our *backwards-forwards state estimation* algorithm, BF, which uses *approximate* logical smoothing and *approximate* logical filtering. BF maintains conjunctive state formulae while computing sound state estimates. Although it may appear excessive to filter with re-

Algorithm 2: $\mathsf{BF}(\mathsf{H}, \phi, i)$ Perform backwards-forwards state estimation on history H given that formula ϕ holds at σ_i in H.

$(\mathsf{H}', \mathit{term\text{-}idx}) = \mathsf{LSmooth}_{\mathsf{a}}(\mathsf{H}, \phi, i)$
return $\mathsf{Filter}_{\mathsf{a}}[(\mathsf{subseq}(\mathsf{H}', \mathit{term\text{-}idx})](\sigma_{\mathit{term\text{-}idx}})$

spect to state formulae, it is efficiently realized because of the restriction to conjunctive formulae. Logical smoothing allows us to encode complex information about any state of the system into the history, keeping each individual state estimate in a compact and computationally manageable form, maintained through a simple logical filtering procedure. Towards tractability, BF leverages approximate logical smoothing with unit propagation entailment. When combined, the result is an intuitive reasoning mechanism for dynamical states with partial observability.

Proposition 4 (Complexity) *Given a history* H *over a dynamical system* Σ *of* n *state-action tuples such that all states are conjunctive formulae, a conjunctive formula* ϕ *over* $\mathcal{F}$*, and an index* i *of* H*,* $\mathsf{BF}(\mathsf{H}, \phi, i)$ *can be computed in time* $O(n \cdot c \cdot 2^{|\mathcal{F}|})$ *with propositional entailment or* $O(n \cdot c \cdot |\mathcal{F}|^2)$ *with unit propagation entailment, where* c *is the maximum number of conditional effects over actions of* Σ*.*

This follows from the previous complexity results.

Space Optimization for Smoothing

State Relevance Minimization. The smoothing process only relies on a subset of the state. Given a history H with tuple (σ, a), it is sufficient for σ to only include the fluents f that are involved in the conditional effects of a. This is due to the regression formula being purely over these fluents plus fluents of ϕ which have *no* positive or negative effects with respect to a. Such an optimization of the state fluents would greatly reduce the memory overhead as actions typically involve and effect a small fraction of the domain. The downside is that the ψ_{NEW} may contain old inferences and thus the early termination becomes less robust.

Sliding Window History. A second optimization is to smooth over a fixed window size of the History tuples instead of always smoothing back to the initial state. This is called *fixed-lag smoothing* (Einicke 2012) in stochastic systems. Such windowed smoothing is routinely used and greatly improves the memory footprint, at the potential expense of the quality of the state estimation.

Experimental Evaluation

We investigate three questions: (1) how effective is our approach in capturing the necessary fluent literals to determine action preconditions and the goal; (2) how does our state relevance minimization impact the system; and (3) how does our approach perform in light of different manifestations of a domain's dynamics. Unfortunately, logical filtering code was unavailable for comparison. We also forgo direct empirical comparison with the belief state estimation of Bonet and Geffner (2014), as their implementation is domain specific (see further discussion below regarding the complementary nature of our methods). Instead, we study algorithm behaviour using existing and new benchmark domains for automated planning with partial observability and sensing.

We ran the BF algorithm on valid plans produced by the offline contingent planner POPRP (Muise, Belle, and McIlraith 2014) and present statistics on the proportion of action traces (plan branches) for which all action preconditions and goal conditions were correctly inferred by BF. That is, for each action a in the trace, running BF from the initial state through all actions preceding a and iteratively building the history H results in a final belief state formula of H that entails $prec(a)$ at the point where a is to be executed. Furthermore, after the last action in the trace, the final belief state formula of H must entail the goal. We compare with statistics for BF with those when we only use Approximate Logical Filtering (ALF). Average history state sizes over the action traces are reported for the BF algorithm and with the state relevance minimization optimization (BF+SRM) from Section 2. Problem statistics and running times are also reported.

Table 1 shows the results of evaluating the BF algorithm on plans for each of the benchmarks (See (Muise, Belle, and McIlraith 2014) for descriptions of the domains Wumpus, Doors, Colored Balls, and CTP (Canadian Traveler's Problem).). First, consider the Precondition / Goal Coverage section. Even with a judicious under-approximation, the BF algorithm is capable of inferring every relevant fluent for every action trace of all plans for the above problems. This is due in part to the fact that these problems belong to the class of width-1 simple contingent problems, which has the property that once a fluent becomes known it stays known (Bonet and Geffner 2011). For problems like Colored Balls, ALF is capable of solving a significant portion of the action traces. This is to be expected; the domain has very little dynamics. Domains like Wumpus and CTP, require heavy reasoning, causing ALF to fail in most cases.

Table 1 also reports the average history state size as a percentage of the total number of fluents in the problem. The large reduction in state size from SRM is due to the fact that only sensing actions have relevant information: no action affects observable fluents.

Lastly, Table 1 reports how much time it takes to solve every action trace of the plan for the associated problem. There are two main take-aways here. First, the SRM optimization creates a space vs time trade off. It significantly reduces the memory footprint of the history but has an impact on computation time. Note that the SRM implementation was itself not optimized for efficiency. One might anticipate that a smaller history could result in reduced time to solve. We expect that a well-engineered SRM implementation would better reflect this. Second, as one would expect, the time to solve a given instance correlates with the number of branches (action traces) in the plan as well as their average length in terms of the number of actions. This is why large problems like Wumpus05 can be solved much quicker than a smaller problem like Balls4-2.

For further evaluation, we introduce two new domains that specifically involve system dynamics and hidden state information that must be inferred – the types of problems

Problem	Problem Statistics				Precondition / Goal Coverage (% of runs)		Avg History State Size (% of total fluents)		Time to Solve (seconds)	
	$\|\mathcal{F}\|$	#-BR	max-BR	avg-BR	BF	ALF	BF	BF+SRM	BF	BF+SRM
Wumpus05	802	35	43	32.9	100	0	17.4	0.02	0.91	1.69
Doors05	100	25	26	16.0	100	64.0	43.5	0.3	0.01	0.17
Doors07	196	343	60	33.2	100	62.9	43.2	0.1	9.02	16.52
CTP05	76	32	10	10.0	100	3.1	43.4	0.6	0.06	0.10
CTP07	134	128	13	14.0	100	0.7	34.3	0.3	0.57	1.01
CTP09	205	512	18	18.0	100	0.2	28.3	0.2	4.34	7.73
Balls4-1	374	48	46	26.1	100	81.2	21.3	0.1	0.83	1.54
Balls4-2	396	2304	90	51.4	100	66.0	25.1	0.09	166.85	324.69

Table 1: BF Algorithm - Planning Benchmarks Performance. (**#-BR**) number of branches; (**max-BR**) max branch depth; (**avg-BR**) average branch depth; (**Coverage**) percentage of runs where all preconditions and goal fluents are captured; (**History State Size**) average percentage of all fluents that must be tracked; (**Time to Solve**) time for each technique to process and verify each branch of the plan.

that motivated this work.

Password. n switches may be flipped by the agent. Each has an unobservable correct position. There are $n+1$ lights such that the i-th light signifies that exactly i switches are in the wrong position. The goal is for the 0-th light to be on and for the agent to know the correct position of each switch.

Lights. n lights are connected in series and each may potentially be broken. A light may be off if it or a light downstream is broken. The agent must fix lights that are known to be broken and reach a state where all lights are no longer broken. This domain has cascading effects as fixing a single light may change the "lit" status of all lights upstream.

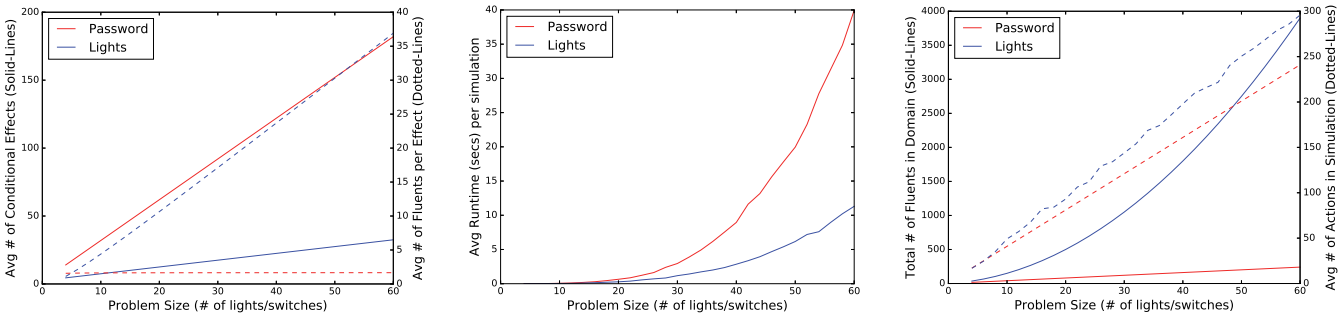
Note that the contingent width (as defined by (Bonet and Geffner 2014)) for these problems is precisely n.

These problems represent two orthogonal classes of dynamic domains with their differences best summarized by Figure 1a. As the problem size grows, the average number of conditional effects in the Password domain grows dramatically compared to Lights. Conversely, the Lights domain sees a dramatic increase in *effect size* as the problem size grows compared to Password. By the parametrization of the Password domain, each conditional effect of flipping a switch depends only on the correctness of the switch and the light that is currently on. For the Lights domain, as more lights are added, fixing any single light has a longer chain of potential ramifications given the status of the lights both upstream and downstream. Therefore, these problems allow us to compare how BF scales with respect to these two important domain characteristics. Many examples of real-world problems whose models include causal constraints or ramifications – typical of electro-mechanical systems that are controlled by a discrete event controller – manifest in one of these two ways.

Finally, we evaluate using simulations that create a randomized hidden state and produce a sequence of actions resulting in a goal state. Figure 1b shows how the BF algorithm performs (averaged over 100 simulations per problem) as the problem size increases, and Figure 1c shows the growth in problem size. As with the standard benchmarks, BF correctly deduces all action preconditions and goal states. The main point of comparison here is not pure performance, but instead performance of problem type as per the preceding discussion of Figure 1a. The Password domain scales slightly better; partially due to the general domain growth with respect to problem size being lower than the Lights domain as per Figure 1c. As Figures 1a and 1c show, an increase in a simple notion of problem size can have a large impact on multiple facets of the problem representation. Figure 1b shows that the BF algorithm scales similarly; regardless of how the dynamics of the system manifest when compiled into conditional effects.

Related Works and Discussion

In this paper we propose an approach to logical state estimation for dynamical systems that is tailored to address the tracking of individual fluents in a computationally efficient manner. Similar to Amir and Russell's original work on logical filtering, we elected not to perform our analysis in the situation calculus, but rather to use a dynamical system model that is employed by those who develop efficient algorithms for AI automated planning and diagnosis. Nevertheless, it is important to note that various work on belief update and progression, much of it done in the situation calculus, is related to logical filtering and thus to state estimation and to his work. Lin and Reiter's (1997) study of progression arguably culminated in Vassos and Levesque's (2013) treatment of first-order progression, which appears to subsume Shirazi and Amir's (2011) first-order logical filtering. Related to approximate filtering, and in small degree smoothing, is the work by Liu and Levesque (2005) that studies progression in dynamical systems with respect to so-called Proper Knowledge Bases. This work shares motivation with our conjunctive formulae restriction in attempting to avoid disjunction in favor of tractability. The authors also present a limited integration of sensing via regression to determine the context of actions to be performed, building on a similar idea for projection by De Giacomo and Levesque (1999). Finally, Ewin, Pearce, and Vassos (2014) study the problem of query projection for long-living agents by combining regression and progression in a manner similar in spirit to the work presented here.



(a) Differences in Password and Lights (b) Runtime Performance of BF (c) Password and Lights domain growth

Figure 1: Comparison of the Password and Lights domains: size, performance of BF, and instance growth.

One of the first works on the approximation of logical belief states in dynamical systems was the 0-approximation semantics proposed by Baral and Son (1997). It also approximates states to conjunctions of fluents, but does no backward-forward reasoning. The ALF results in Table 1 provide a sense of how 0-approximation might compare with our BF approach. The field of automated planning has spawned numerous systems which necessarily have a sub-component to perform state estimation. For example, the systems CNF_{ct}, DNF_{ct}, and PI_{ct} (To, Son, and Pontelli 2011; To, Pontelli, and Son 2011) explicitly maintain sets of the possible belief states but in a minimal and partial fashion given the sequence of actions. This facilitates exact estimation of the state over multiple branches of possibilities in compact forms for most problem structures but in the worst case has exponential space complexity. Palacios and Geffner realize a form of approximate state estimation via a compilation process that introduces additional fluents and actions corresponding to possible initial worlds (2009). This approach provides efficient state querying but the number of additional fluents required for completeness grows exponentially in problem's contingent width.

The SDR Planner (Brafman and Shani 2012) maintains a history of actions similar to our history of state-action pairs but for a slightly different purpose. The planner samples a possible complete initial state then assumes it is correct and plans appropriately. When information is gained through sensing that refutes the correctness of the initial state sample, re-planning is performed and a new state sampled. To ensure action preconditions are respected, precondition fluents are regressed through the history to the initial state to ensure satisfiability. This aspect of the algorithm is similar, at a high level, to the fundamental idea of logical smoothing - that understanding of the evolution of the past can produce new information about the present. In 2014, Brafman and Shani also exploit regression for effective state estimation (2014). While we use newly discovered information about past states to reduce the uncertainty of more recent states and approximate for efficiency, they only perform full regression on the history of actions and observations.

As noted previously, Bonet and Geffner (2014) recently developed algorithms for belief tracking for so-called simple planning problems. As previously noted, we were unable to perform an experimental comparison with their work because their implementation is domain-specific. Nevertheless, it is interesting to consider when one approach works well and the other does not. As width-n problems, both the Password and Lights domains would cause an exponential blowup for their technique. Conversely, there are width-1 problems where our approximation does not capture simple entailments, such as conformant-like conditions where reasoning by cases plays a role. The complementary nature of the two approaches makes their combination an obvious step for future research.

Finally, although this approach is described in the context of deterministic actions, it extends to non-deterministic actions quite easily by employing the modeling trick that deterministic actions can be modeled as deterministic actions whose outcomes are predicated on an unobservable condition. E.g., if foo is true then action $flip$ will result in $heads$, if false then the outcome is $tails$.

Concluding Remarks

We examine the task of approximated logical belief state estimation of partially observable dynamical systems with causal constraints that yield indirect actions effects. In planning, such constraints tend to create problems with large conformant width. Approximations to logical belief state estimation are often necessary for tractability of state estimation and fluent querying, and are adequate for many applications. Inspired by stochastic smoothing techniques, we propose logical smoothing and backwards-forward reasoning which work in concert with approximated logical filtering to produce sound and sometimes complete estimates of fluents relative to full logical filtering. Experiments demonstrate flawless recall on state estimation for the preconditions and goal conditions in plans for existing partially observable planning problems, and a dramatic reduction in the number of fluent literals that must be monitored. Our approach complements existing techniques for logical belief state estimation, unifying the notions of logical smoothing and logical filtering and capturing the logical analogue of commonly used techniques in control theory and other dynamic programming settings.

Acknowledgements: The authors gratefully acknowledge

funding from the Natural Sciences and Engineering Research Council of Canada (NSERC). We also thank the reviewers for their insightful comments.

References

Amir, E., and Russell, S. J. 2003. Logical filtering. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 75–82.

Baier, J. A.; Mombourquette, B.; and McIlraith, S. A. 2014. Diagnostic problem solving via planning with ontic and epistemic goals. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference (KR 2014)*.

Baral, C., and Son, T. C. 1997. Approximate reasoning about actions in presence of sensing and incomplete information. In *Proceedngs of the 1997 International Symposium on Logic Programming*, 387–401.

Bonet, B., and Geffner, H. 2011. Planning under partial observability by classical replanning: Theory and experiments. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 1936–1941.

Bonet, B., and Geffner, H. 2014. Belief tracking for planning with sensing: Width, complexity and approximations. *Journal of Artificial Intelligence Research* 923–970.

Brafman, R. I., and Shani, G. 2012. Replanning in domains with partial information and sensing actions. *Journal of Artificial Intelligence. (JAIR)* 45:565–600.

Brafman, R. I., and Shani, G. 2014. On the properties of belief tracking for online contingent planning using regression. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*, 147–152.

De Giacomo, G., and Levesque, H. J. 1999. Projection using regression and sensors. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, 160–165.

Einicke, G. A. 2012. *Smoothing, Filtering and Prediction - Estimating The Past, Present and Future*. InTech.

Eiter, T., and Gottlob, G. 1992. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence* 57(2-3):227–270.

Ewin, C. J.; Pearce, A. R.; and Vassos, S. 2014. Transforming situation calculus action theories for optimised reasoning. In *Proceedings of the 14th International Conference on the Principles of Knowledge Representation and Reasoning (KR)*.

Fikes, R.; Hart, P.; and Nilsson, N. 1972. Learning and executing generalized robot plans. *Artificial Intelligence* 3:251–288.

Fritz, C., and McIlraith, S. A. 2007. Monitoring plan optimality during execution. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*, 144–151.

Fritz, C. 2009. *Monitoring the Generation and Execution of Optimal Plans*. Ph.D. Dissertation, University of Toronto.

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

Lin, F., and Reiter, R. 1997. How to progress a database. *Artificial Intelligence* 92:131–167.

Liu, Y., and Levesque, H. J. 2005. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 522–527.

McIlraith, S. A., and Reiter, R. 1992. On tests for hypothetical reasoning. In Hamschers, W.; de Kleer, J.; and Console, L., eds., *Readings in Model-Based Diagnosis*. Morgan Kaufmann. 89–95.

Muise, C.; Belle, V.; and McIlraith, S. A. 2014. Computing contingent plans via fully observable non-deterministic planning. In *The 28th AAAI Conference on Artificial Intelligence*.

Palacios, H., and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research* 35:623–675.

Pinto, J. 1999. Compiling ramification constraints into effect axioms. *Computational Intelligence* 15:280–307.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. Cambridge, MA: MIT Press.

Rintanen, J. 2008. Regression for classical and nondeterministic planning. In *Proceedings of the 18th European Conference on Artificial Intelligence*, 568–572.

Shahaf, D., and Amir, E. 2007. Logical circuit filtering. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2611–2618.

Shirazi, A., and Amir, E. 2011. First-order logical filtering. *Artificial Intelligence* 175(1):193–219.

Strass, H., and Thielscher, M. 2013. A general first-order solution to the ramification problem with cycles. *Journal of Applied Logic* 11(3):289–308.

To, S. T.; Pontelli, E.; and Son, T. C. 2011. On the effectiveness of CNF and DNF representations in contingent planning. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2033–2038.

To, S. T.; Son, T. C.; and Pontelli, E. 2011. Contingent planning as AND/OR forward search with disjunctive representation. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*. AAAI.

van Ditmarsch, H. P.; Herzig, A.; and Lima, T. D. 2007. Optimal regression for reasoning about knowledge and actions. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, 1070–1076.

Vassos, S., and Levesque, H. J. 2013. How to progress a database III. *Artificial Intelligence* 195:203–221.