Belief State Estimation for Planning via Approximate Logical Filtering and Smoothing

Brent Mombourquette† and **Christian Muise**∗ and **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. Unfortunately such filtering, though exact, can be intractable. To this end, we propose *logical smoothing*, a form of backwards reasoning that works in concert with 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 prove properties of our algorithms, and experimentally demonstrate their behaviour. Smoothing together with 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.

1 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. State estimation is well-studied in control systems where transition systems are stochastic and the belief state is typically represented as a probability distribution. 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 *et al.*, 2014]. In planning such constraints tend to create problems with large conformant width [Palacios and Geffner, 2009].

We exploit the observation that for planning, only a subset of the state is necessary to track. Planning systems need to know (1) when actions are applicable, and (2) 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 *et al.*, 1972]). 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 2 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 3, 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 4, 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 the behaviour of our approach. This is followed by a discussion of related work and concluding remarks.

2 The Problem: State Estimation

State estimation is a core task in reasoning about dynamical systems with partial observability. Consider the simplified action-observation sequence in support of diagnosing a car. You turn the key in the ignition, $turn\_ignition$, resulting in $ignition\_turned$. If $ignition\_turned$, $battery\_ok$ and gas_ok hold, then so will $car\_started$. You observe that the car did not start ($car\_started = False$), and so, under the assumption that the characterization of the vehicle functioning is complete, you can you infer $\neg battery\_ok \vee \neg gas\_ok$. You turn on the radio, $turn\_on\_radio$, causing $radio\_on$ as well as $sound$ if $battery\_ok \wedge radio\_ok$. You observe sound ($sound = True$). 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 $\{ignition\_turned$, $radio\_on$, $\neg car\_started$, $battery\_ok$, $radio\_ok$, $\neg gas\_ok$, $sound\}$.

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].

To relate our work to planning, execution monitoring and diagnosis, 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 $eff(a)$, 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, on the other hand, is defined 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. We assume no exogenous actions and that the transition relation is complete, characterizing all and only the conditions under which the truth value of a fluent changes.

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 $\rho \subseteq S$. For our purposes , we will often represent the belief state (henceforth also the state of the system) as a *state formula* ϕ. 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)
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. $\mathsf{Filter}[\mathsf{o}](\rho) = \{s \mid s \in \rho \text{ and } \mathsf{o} \text{ is true in } s\}$
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 with respect to the transition system $R(s, a)$. When an observation o is filtered, every state inconsistent with the observation is eliminated. Logical filtering results in an interleaving of action progression and observation filtering (e.g.,[Vassos and Levesque, 2013]). Logical filtering naïvely is hard: there are $2^{2^{|\mathcal{F}|}}$ belief states. As such, algorithms for logical filtering typically represent the belief state ρ compactly as a logical formula ϕ, called a *belief-state formula* or *state formula*.

While logical filtering realizes state estimation, the resulting belief state representation 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 the following definition of *approximate logical filtering*:

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. $\mathsf{Filter_a}[\langle \mathsf{a_i}, \mathsf{o_i}, \ldots, \mathsf{a_t}, \mathsf{o_t} \rangle](\phi) =$ $\mathsf{Filter_a}[\langle \mathsf{a_{i+1}}, \mathsf{o_{i+1}}, \ldots, \mathsf{a_t}, \mathsf{o_t} \rangle](\mathsf{Filter_a}[\mathsf{o_i}](\mathsf{Filter_a}[\mathsf{a_i}](\phi)))$

Approximate logical filtering describes the belief state only in terms of a subset of the fluent literals entailed by the belief state. 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.

Returning to our car example, we can see how this is a weak approximation. After the observations $car\_started = False$ and $sound = True$, no further inferences could be made, yielding the belief state $\{ignition\_turned$, $radio\_on$, $\neg car\_started$, $sound\}$

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 the 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 automated planning and similar problems where the general assumption is that 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 show how to combine approximate logical filtering with a new approach for reasoning over the past – logical smoothing.

3 Logical Smoothing

Filtering with a stochastic transition system involves estimating the marginal posterior probability distribution of the system conditioned on some data that was received *prior* to the current system 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 into the 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),... (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. 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),... (ρ_n, a_n) with respect to the ρ is defined as:*

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

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 the semantic account of logical smoothing in hand, we now define the notion of logical smoothing with respect to a more compact representation of the set of belief state in terms of a belief state formula.

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.*

Definition 6 provides a formal characterization of logical smoothing with respect to a belief-state formula representation of the history. It utilizes regression rewriting [Reiter, 2001] to propagate updates back through the history. Regression, denoted as $\mathcal{R}[\phi, a]$, takes a logical formula ϕ and action a and rewrites the formula in terms of the weakest conditions necessary for ϕ to hold following the execution of a. We appeal to Reiter's definition of regression in the situation calculus [Reiter, 2001] with syntactic notation suitably modified.

When the history comprises a single state-action pair, formula ϕ is simply conjoined to the state. Otherwise, the formula is regressed step by step through the history, and any new information garnered from the regression, ϕ_{NEW}, is conjoined to the associated state. $PI(\phi)$ 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. $\mathsf{Smooth}[(\sigma, \mathsf{a})](\phi) = (\sigma \wedge \phi, a)$
2. $\mathsf{Smooth}[(\sigma_0, \mathsf{a_0}), \ldots, (\sigma_\mathsf{n}, \mathsf{a_n})](\phi) =$
$\mathsf{Smooth}[(\sigma_0, \mathsf{a_0}), \ldots, (\sigma_{\mathsf{n-1}}, \mathsf{a_{n-1}})](\phi_{\mathsf{NEW}}),$
$\mathsf{Smooth}[(\sigma_\mathsf{n}, \mathsf{a_n})](\phi)$

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 $\mathsf{PreImage}$ and regression.

Returning to our car example, consider logical smoothing with respect to the observation $car\_started = False$. For ease of presentation, assume approximate logical filtering is used for action progression. Progressing the initial state, σ_0, through the action $turn\_ignition$ results in state σ_1, which is equivalent to σ_0 since the effect of $turn\_ignition$ is predicated on fluents whose truth value is unknown. When the observation action $obs(car\_started)$ is subsequently performed and $car\_started = False$ observed, the smoothed history is given by $\mathsf{Smooth}[\mathsf{h_0}, \mathsf{h_1}](\neg\mathsf{car\_started})$ where $h_0 = (\sigma_0, turn\_ignition)$ $h_1 = (\sigma_1, obs(car\_started))$. Following point 2 of Definition 6, this is equivalent to $\mathsf{Smooth}[(\mathsf{h_0})](\phi_{\mathsf{NEW}}), \mathsf{Smooth}[(\mathsf{h_1}))](\neg\mathsf{car\_started})$. Rule 1 of the logical smoothing semantics smooths σ_1 of h_1 as $\sigma_1 \wedge \neg car\_started$. The action $obs(car\_started)$ has no effects and therefore the observation $car\_started = False$ must hold in the prior state. Returning to ϕ_{NEW}, this formula hinges on $\mathcal{R}[\neg car\_started, turn\_ignition]$. By the regression re-writing and frame axioms, this gives $\phi = \neg car\_started \wedge (\neg battery\_ok \vee \neg gas\_ok)$. As $\sigma_0 \models \neg car\_started$, we are left with $\phi_{NEW} = \neg battery\_ok \vee \neg gas\_okay$ after restricting to prime implicates, giving 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 $\sigma_i = \sigma_i \wedge \phi$
2 if $i > 0$ then
3     $\psi = \mathcal{R}[\phi, a_{i-1}]$
4     $\psi_{NEW} = \bigwedge\{\varphi \in PI(\psi) \mid \sigma_{i-1} \not\models \varphi\}$
5     if $\psi_{NEW}$ is empty then
6         return
7     LSmooth(H, $\psi_{NEW}$, $i-1$)
8 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 $term\text{-}idx$. The purpose of returning $term\text{-}idx$ will become apparent in Section 4 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 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. 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.

3.1 Approximate Logical Smoothing

Unfortunately, like logical filtering, the querying of belief states resulting from logical smoothing may not be tractable. To make matters worse, 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 define, as we did with logical filtering, an analogous procedure of *approximate logical smoothing*. We do so by one minor adjustment to the logical smoothing outlined in Algorithm 1:

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

Where f is restricted to fluent literals. This approximation limits the updating of the history to fluent literals entailed by these regressed additions. Again we consider only the entailments not already captured by a state in the history. We denote the resulting algorithm obtained by modifying $\mathsf{LSmooth}$ as per (1) as $\mathsf{LSmooth_a}$. It should be stressed that this is but one way to deal with the formula size increase due to repeated applications of regression. Alternatives which do not sacrifice completeness include adding new fluents in place of certain sub-formulae of the regressed formula [van Ditmarsch *et al.*, 2007] or representing the formula as a circuit [Rintanen, 2008].

Following Definition 6, logical smoothing produced the correct refinement of the initial state with $\neg battery\_ok \vee \neg gas\_ok$. However, with approximate logical smoothing ψ_{NEW} is limited to fluent literals, resulting in an empty formula and therefore no refinement. Some information is lost but other inferences can still be made. Continuing with the sequence of actions in our example, after $obs(car\_started)$ the actions $turn\_on\_radio$ and $obs(sound)$ are executed, obtaining $sound = True$. By smoothing with respect to this observation the regression $\mathcal{R}[sound, turn\_on\_radio]$ gives $\psi = sound \vee (battery\_ok \wedge radio\_ok)$. Following the approximation rule (1) we get $\psi_{NEW} = battery\_ok \wedge radio\_ok$ due to the previous state in the history entailing $\neg sound$. Even with the approximation, we refine the estimate of the prior state with the knowledge $battery\_ok \wedge radio\_ok$. Continuing this example, approximate logical smoothing would smooth each state in the history, including the initial state, to include $battery\_ok \wedge radio\_ok$.

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 the language of $\mathcal{F}$ 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.

We also sidestep the exponential size formula blowup from repeated regression operations by always regressing conjunctive formulae except for, potentially, in the first step.

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 the language of $\mathcal{F}$, 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 system effects (ramifications) in comparison to the size of the propositional domain. This results in a compact regressed formula where all unit entailments are computable through unit propagation in most cases. Note that since we place no restrictions on the syntactic form of regressed formulae, such as restricting to Horn clauses, unit propagation may not produce all entailments, resulting in a sound under-approximation.

4 Backwards-Forwards Algorithm

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

As we last left the car example from Section 3.1, each state in the history, including the initial state, was smoothed with $battery\_ok \wedge radio\_ok$ after observing $sound = True$. 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 $\mathsf{subseq}(\mathsf{H}, i)$ 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.

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.

```
1 $(\mathsf{H}', term\text{-}idx) = \mathsf{LSmooth_a}(\mathsf{H}, \phi, i)$
2 return $\mathsf{Filter_a}[(\mathsf{subseq}(\mathsf{H}', term\text{-}idx)](\sigma_{term\text{-}idx})$
```

The BF algorithm uses *approximate* logical smoothing and *approximate* logical filtering. BF maintains conjunctive state formulae while computing sound state estimates. 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 that is maintained through a simple logical filtering procedure. Towards this goal of 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.

Although it may appear excessive to filter with respect to state formulae, the states are always conjunctive formulae and approximate filtering operates by conjoining the observation formula with the state formula resulting from progressing the previous action. Since all refinements are sound, this reduces to simple set operations over the fluents.

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 the* *language of $\mathcal{F}$, 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.

4.1 Space Optimization

Here we outline two optimizations to alleviate the space requirements of logical 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 called *fixed-lag smoothing* [Einicke, 2012] in stochastic systems. This greatly improves the memory footprint, at the potential expense of quality of the estimation.

5 Experimental Evaluation

We investigate three key 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. Ideally, we would compare our approach empirically with logical filtering. However, the original authors confirmed in private correspondence that the code is unavailable. Instead, we look to the field of automated planning that deals with partial observability and sensing. We use a set of standard benchmark problems and two newly created domains.

We ran the BF algorithm on valid plans, and here we present statistics on the proportion of action traces (plan branches) for which all action preconditions and goal condition are known to hold. For comparison, these are also reported for Approximate Logical Filtering (ALF). Average history state sizes over the action traces are reported for the BF algorithm and with the state relevance minimization (BF+SRM) as outlined in Section 4.1. Problem statistics and running times are also reported.

The plans were produced by the planner POPRP [Muise *et al.*, 2014], which leverages a compilation of partially observable planning problems that exhibit particular properties making them easier to solve (i.e., "simple" contingent problems [Bonet and Geffner, 2011]). For a description of the benchmark domains Wumpus, Doors, Colored Balls, and CTP (Canadian Traveler's Problem), see [Muise *et al.*, 2014]. Full details and source code used for evaluation will be made available online.

Table 1 shows the results of evaluating the BF algorithm on plans for each of the benchmarks. First, consider the Precondition / Goal Coverage section. Even with a judicious underapproximation, the BF algorithm is capable of tracking 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 the 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. In the case of four colors with one ball, the only unsolvable traces result from the situations when the ball must be in the last location not observed and the plan is able to infer this without directly observing it. Other domains, like Wumpus and CTP, require heavy reasoning, causing ALF to fail in most, if not all, 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. This is a byproduct of the problem structures that the planning community has chosen to focus on and highlights the orthogonality of this work to what is currently being researched.

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. 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.

To further expand the evaluation, we introduce two new domains in the class of width-n non-simple contingent problems that specifically involve system dynamics and hidden state information that must be inferred.

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 is broken 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 1. First, consider the Password domain. As the problem size grows, the average number of conditional effects grows dramatically. Any time a switch is flipped, all possible cases for the $n + 1$ lights changing must be covered. With the Lights domain, the actions of fixing a single light have more repercussions as lights are added but the chain is

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.

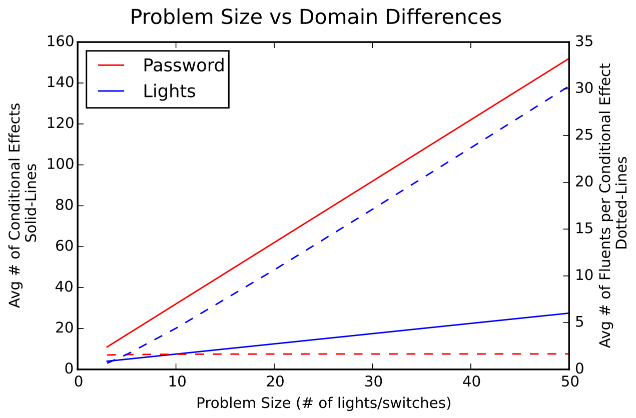

Figure 1: Differences in Password and Lights domains

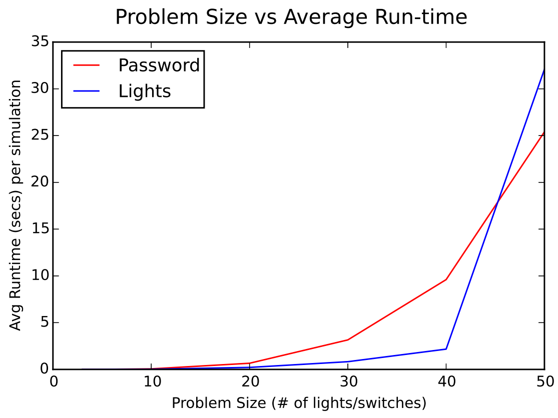

Figure 2: Performance of BF on Password and Lights domains

only increased by one per new light. If we look at how the average number of conditions (fluents) of the conditional effects grow with respect to problem size we see the reverse - the Lights domain grows much faster than 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.

We evaluate via a simulation that creates a randomized true state and produces a sequence of actions that should result in a state where the goal holds. Figure 2 shows how the BF algorithm performs as the problem size increase, averaged over 100 random action traces. As with the standard benchmarks, the BF algorithm correctly deduced all action preconditions and goal states over all generated action traces. The main point of comparison here is not pure performance but instead performance of problem type as per the preceding discussion of Figure 1. 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 3. As Figures 1 and 3 show, an increase in a simple notion of problem size can have a large impact on multiple facets of the problem representation. Regardless of how the dynamics of the system manifest when compiled into conditional effects of actions, Figure 2 shows that the BF algorithm scales equivalently.

6 Related Works

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, there is a large body of work in the knowledge representation literature, much of it in the situation calculus, that is related to logical filtering; particularly work on belief update and on progression. Among these, Lin and Reiter (1997) provided a broad study of progression spawning a number of other advances and culminating in Vassos and Levesque's (2013) treatment of first-order progression, which appears to subsume Shirazi and Amir's (2011) work on first-order logical filtering. Also relevant 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 some motivation with our conjunctive formulae restriction in attempting to avoid disjunction in favor of tractability. Further,

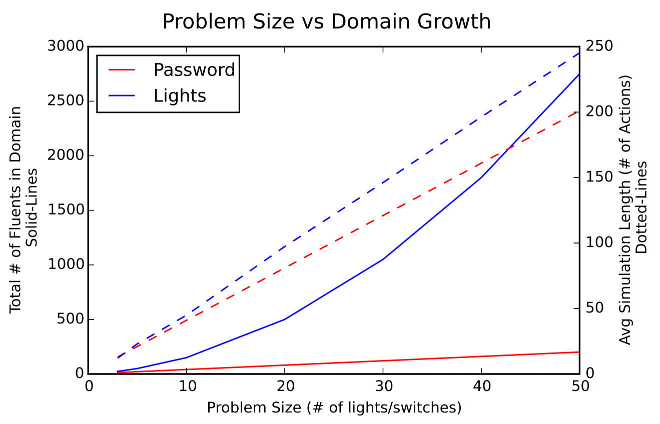

Figure 3: Password and Lights domain growth

the authors discuss 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 *et al.* (2014) study the problem of query projection for long-living agents by combining regression and progression in a manner that is similar in spirit to the work presented here.

Our work is inspired by work on smoothing and backwards-forwards reasoning in stochastic systems, building on previous work on logical filtering [Amir and Russell, 2003], the adaptation of the classical filtering to a logical setting. Various works on (database) progression (e.g., [Vassos and Levesque, 2013]) are also closely related to this exact logical state estimation task. In 2007, Shahaf and Amir developed a version of logical filtering that represents the underlying belief state as a circuit. By solving the update problem completely, logical circuit filtering is able to compute and represent the belief state, exactly, in polynomial time. Unfortunately, querying of the underlying data structure used in logical circuit filtering requires use of a SAT solver, making it untenable for tasks such as AI planning which could require excessive SAT calls to evaluate action executability during plan construction.

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). Like our approximation techniques, they represent beliefs in terms of conjunctions of fluents, but do not exploit a notion of backwards-forwards reasoning to do state estimation.

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 *et al.*, 2011b; 2011a]) explicitly maintain sets of the possible belief states but in the worst case have 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). While this translation based approach provides efficient state querying, 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 actions that disprove the correctness of the initial state sample, re-planning is performed and a new state is sampled. To ensure action preconditions are correctly followed, precondition fluents are regressed through the history to the initial state to ensure satisfiability. This portion of the algorithm is similar, at a high level, to the fundamental ideas of logical smoothing - that an understanding of the evolution of the past can produce new information about the present. More recently Brafman and Shani (2014) also exploit regression for effective state estimation. A key difference, however, is that we use newly discovered information about past states to reduce the uncertainty of more recent states. In contrast, they only perform a backwards pass on the history of actions and observations, using full regression, while we approximate for efficiency.

Most recently Bonet and Geffner (2014) developed algorithms for belief tracking for so-called simple planning problems, as noted previously. 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 case-based reasoning plays a role. The complementary nature of the two approaches makes their combination an obvious step for future research.

7 Concluding Remarks

Logical smoothing and backwards-forwards reasoning are important techniques in service of reasoning about partially observable dynamical systems. They characterize the logical analogue of commonly used techniques in control theory and other dynamic programming settings. This paper provides the formal foundations for a number of interesting theoretical and algorithmic tools that are of practical significance for automated planning, execution monitoring, diagnosis and beyond. We demonstrated the effectiveness of our approach, which has flawless recall on state estimation for the preconditions and goal conditions in the plans for existing partially observable planning problems. Further, we witnessed a dramatic reduction in the number of fluent literals that must be monitored. In future work we plan to integrate our algorithms with a contingent planning system.While our work was presented in the context of deterministic actions, the account and algorithms extend trivially to non-deterministic actions, and can be extended to exogenous actions. We plan to elaborate on such cases and to further explore variants of logical smoothing and our BF algorithm as they relate to non-deterministic and probabilistic transition systems. Finally we wish to further explore the theoretical relastionship between our work and progression over Proper KBs, and relationship to the various approximations defined by Palacios and Geffner in [Palacios and Geffner, 2009] and by Bonet and Geffner in [Bonet and Geffner, 2014].

Acknowledgements: The authors gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

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

[Baier *et al.*, 2014] Jorge A. Baier, Brent Mombourquette, and Sheila A. McIlraith. 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)*, 2014.

[Baral and Son, 1997] Chitta Baral and Tran Cao Son. Approximate reasoning about actions in presence of sensing and incomplete information. In *Proceedngs of the 1997 International Symposium on Logic Programming*, pages 387–401, 1997.

[Bonet and Geffner, 2011] Blai Bonet and Hector Geffner. 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*, pages 1936–1941, 2011.

[Bonet and Geffner, 2014] Blai Bonet and Hector Geffner. Belief tracking for planning with sensing: Width, complexity and approximations. *Journal of Artificial Intelligence Research*, pages 923–970, 2014.

[Brafman and Shani, 2012] Ronen I. Brafman and Guy Shani. Replanning in domains with partial information and sensing actions. *J. Artif. Intell. Res. (JAIR)*, 45:565–600, 2012.

[Brafman and Shani, 2014] Ronen I. Brafman and Guy Shani. On the properties of belief tracking for online contingent planning using regression. In *Proc. of the 21st European Conference on Artificial Intelligence (ECAI)*, pages 147–152, 2014.

[De Giacomo and Levesque, 1999] Giuseppe De Giacomo and Hector J. Levesque. Projection using regression and sensors. In *Proc. of the 16th Int'l Joint Conference on Artificial Intelligence (IJCAI)*, pages 160–165, 1999.

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

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

[Ewin *et al.*, 2014] Christopher James Ewin, Adrian R. Pearce, and Stavros Vassos. Transforming situation calculus action theories for optimised reasoning. In *Proc. of the 14th Int'l Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 2014.

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

[Fritz, 2009] Christian Fritz. *Monitoring the Generation and Execution of Optimal Plans*. PhD thesis, University of Toronto, April 2009.

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

[Lin and Reiter, 1997] Fangzhen Lin and Raymond Reiter. How to progress a database. *Artificial Intelligence*, 92:131–167, 1997.

[Liu and Levesque, 2005] Yongmei Liu and Hector J. Levesque. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proc. of the 19th Int'l Joint Conference on Artificial Intelligence (IJCAI)*, pages 522–527, 2005.

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

[Muise *et al.*, 2014] Christian Muise, Vaishak Belle, and Sheila A. McIlraith. Computing contingent plans via fully observable nondeterministic planning. In *The 28th AAAI Conference on Artificial Intelligence*, 2014.

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

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

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

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

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

[Shirazi and Amir, 2011] Afsaneh Shirazi and Eyal Amir. First-order logical filtering. *Artificial Intelligence*, 175(1):193–219, 2011.

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

[To *et al.*, 2011a] Son Thanh To, Enrico Pontelli, and Tran Cao Son. On the effectiveness of CNF and DNF representations in contingent planning. In *Proc. of the 22nd Int'l Joint Conference on Artificial Intelligence (IJCAI)*, pages 2033–2038, 2011.

[To *et al.*, 2011b] Son Thanh To, Tran Cao Son, and Enrico Pontelli. Contingent planning as AND/OR forward search with disjunctive representation. In Fahiem Bacchus, Carmel Domshlak, Stefan Edelkamp, and Malte Helmert, editors, *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*. AAAI, 2011.

[van Ditmarsch *et al.*, 2007] Hans P. van Ditmarsch, Andreas Herzig, and Tiago De Lima. Optimal regression for reasoning about knowledge and actions. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 1070–1076, 2007.

[Vassos and Levesque, 2013] Stavros Vassos and Hector J. Levesque. How to progress a database III. *Artificial Intelligence*, 195:203–221, 2013.