Learning Partially Observable Action Models

Eyal Amir
Computer Science Department
University of Illinois, Urbana-Champaign
Urbana, IL 61801, USA
eyal@cs.uiuc.edu

Abstract. In this paper we present tractable algorithms for learning a logical model of actions' effects and preconditions in deterministic partially observable domains. These algorithms update a representation of the set of possible action models after every observation and action execution. We show that when actions are known to have no conditional effects, then the set of possible action models can be represented compactly indefinitely. We also show that certain desirable properties hold for actions that have conditional effects, and that sometimes those can be learned efficiently as well. Our approach takes time and space that are polynomial in the number of domain features, and it is the first exact solution that is tractable for a wide class of problems. It does so by representing the set of possible action models using propositional logic, while avoiding general-purpose logical inference. Learning in partially observable domains is difficult and intractable in general, but our results show that it can be solved exactly in large domains in which one can assume some structure for actions' effects and preconditions. These results are relevant for more general settings, such as learning HMMs, reinforcement learning, and learning in partially observable stochastic domains.

1 Introduction

Agents that act in complex domains usually have limited prior knowledge of their actions' preconditions and effects (the *transition model* of the world). Such agents need to learn about these action to act effectively, and they also need to track the state of the world, when their sensory information is limited. For example, a robot moving from room to room in a building can observe only its immediate environment. Upon discovering a switch in the wall, it may not know the consequences of flipping this switch. After flipping it, the agent may observe those effects that occur in its immediate environment, but not those outside the room. When it leaves the room and discovers some change in the world, it may want to ascribe this change to flipping the switch.

Learning transition models in partially observable domains is hard. In stochastic domains, learning transition models is central to learning Hidden Markov Models (HMMs) [17] and to reinforcement learning [8], both of which afford only solutions that are not guaranteed to approximate the optimal. In HMMs the transition model is learned using the Baum-Welch algorithm, which is a special case of EM. It is a hill-climbing algorithm which is only guaranteed to reach a local optima, and there is no time guarantee for convergence on this local optima. *Reinforcement learning* in partially observable domains [7] can be solved (approximately) by interleaving learning the POMDP with solving it (the learning and solving are both approximate because finite memory or finite granularity is always assumed) [3, 12, 13]. It is important to notice that this problem is harder than solving POMDPs. In some cases, one can solve the POMDP with some guarantee for relatively fast convergence and approximation, if one knows the underlying transition model [9, 14]. Also, in deterministic cases, computing the optimal undiscounted infinite horizon policy in (known) POMDPs is PSPACE-hard (NP-complete if a polynomial horizon) in the number of states [11], but reinforcement learning has no similar solution known to us.

In this paper we present a formal, exact, many times *tractable*, solution to the problem of *simultaneously learning and filtering (SLAF) preconditions and effects of actions* from experiences in partially observable domains. We put emphasis on the solution being tractable as a function of the *number of state features* rather than the (exponentially larger) number of states.

First, we present a formal system that captures this problem precisely for possibly nondeterministic actions. It maintains a set of pairs $\langle$*state,transition-relation*$\rangle$ that are consistent with the actions and observations collected so far (the *transition belief state*). Then, we present a generic algorithm that uses logical deduction and learns transition models in deterministic partially observable domains.

We present more tractable algorithms for special cases of SLAF. We examine actions that are (1) *always executable* or *sometimes inexecutable* (depending on deterministic preconditions), and (2) *conditional* or *nonconditional* (whenever executable, have the same effect). For the case of STRIPS actions (always executable, nonconditional) we show that our algorithm runs in time linear in the number of propositional domain features and the space taken to represent our transition belief state. We can maintain this transition belief state in polynomial space (in the number of features and actions available in our domain) under very relaxed conditions. We present a more general algorithm (than our STRIPS one) that treats other cases with a polynomial time per time step, when actions are known to act as 1:1 mapping on states, and they provide an approximation otherwise.

Our algorithms are the first to learn exact action models in partially observable domains. They are also first to find an action model at the same time that they determine the agent's knowledge about the state of the world. They draw on intuitions and results of [1] for known (nondeterministic) action models. If we assume that our transition model is fully known, then our results reduce to those of [1] for deterministic actions.

A wide range of virtual domains satisfy our assumptions of determinism and structured actions, and we are in the process of testing our algorithms in large domains, including over 1000 features (see [6] for current progress).

Previous work on learning action's effects and preconditions focused on fully observable domains. [5, 19] learn STRIPS actions with parameters by finding the most general and most specific in a version space of STRIPS operators. [15] uses a general-purpose classification system (in their case, MSDD) to learn the effects and preconditions of actions, identifying irrelevant variables. [2] presents an approach that is based on inductive logic programming. Most recently, [16] showed how to learn stochastic actions with no conditional effects (i.e., the same stochastic change occurs at every state in which the action is executable). The common theme among these approaches is their assumption that the state of the world is fully observed at any point in time. [18] is the only work that considers partial observability, and it does so by assuming that the world is fully observable, giving approximate computation in relevant domains.

2 Filtering Transition Relations

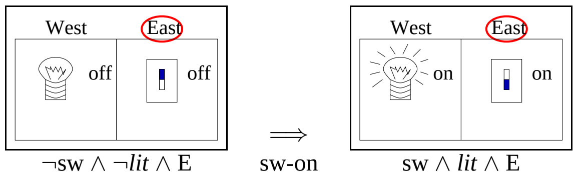
West
East
off
off
$\neg$sw $\wedge$ $\neg$*lit* $\wedge$ E
$\Longrightarrow$
sw-on
West
East
on
on
sw $\wedge$ *lit* $\wedge$ E

Figure 1. Two rooms and flipping the light switch

Consider a simple world with two rooms, one with a switch, and the other with a light bulb whose state can be observed only when the agent is in that room (see Figure 1). Assume that our agent initially knows nothing about the three actions go-E (go to the Eastern room), go-W (go to the Western room), and sw-on (flip the switch to *on*). Our agent's problem is to determine the effects of these actions (to the extent that it can, theoretically), while also tracking the world.

We describe the combined problem of filtering (updating the agent's belief state) and learning the transition model formally. A *transition system* is a tuple $\langle \mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$, where

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

Here, a *world state*, $s \in \mathcal{S}$, is a subset of $\mathcal{P}$ that contains propositions true in this state, and $R(s, a, s')$ means that state s' is a possible result of action a in state s.

A *transition belief state* is a set of tuples $\langle s, R \rangle$ where s is a state and R a transition relation. Let $\mathfrak{R} = Pow(\mathcal{S} \times \mathcal{A} \times \mathcal{S})$ be the set of all possible transition relations on $\mathcal{S}, \mathcal{A}$. Let $\mathfrak{S} = \mathcal{S} \times \mathfrak{R}$. Every $\rho \subseteq \mathfrak{S}$ is a *transition belief state*. When we hold a a transition belief state ρ we consider every tuple $\langle s, R \rangle \in \rho$ possible. With this formal system we assume that observations are given to us (if at all) as logical sentences after performing an action. They are either *state formulae* (propositional combinations of fluent names) or OK or $\neg OK$ (observing the action is possible or impossible to execute). We denote the former kind of observation with o, and the latter with OK,$\neg OK$, respectively.

Definition 1 (Transition Filtering Semantics) *Let $\rho \subseteq \mathfrak{S}$ be a transition belief state. The* filtering *of ρ with actions and observations $\langle a_1, o_1, \ldots, a_t, o_t \rangle$ is*
1. $Filter[\epsilon](\rho) = \rho$;
2. $Filter[a, OK](\rho) =$
$\{\langle s', R \rangle \mid \langle s, a, s' \rangle \in R,\ \langle s, R \rangle \in \rho\}$;
3. $Filter[a, \neg OK](\rho) =$
$\{\langle s, R \rangle \mid \langle s, R \rangle \in \rho,\ \forall s' \in \mathcal{S}\ \langle s, a, s' \rangle \notin R\}$;
4. $Filter[o](\rho) = \{\langle s, R \rangle \in \rho \mid o$ *is true in* $s\}$;
5. $Filter[\langle a_i, o_i, \ldots, a_t, o_t \rangle](\rho) =$
$Filter[\langle a_{i+1}, o_{i+1}, \ldots, a_t, o_t \rangle]$
$(Filter[o_i](Filter[a_i](\rho)))$.
We call Step 2 progression with a, *Step 3* disqualifying a, *and Step 4* filtering with o.

The intuition behind this definition is that every transition relation, R, and initial state, s, produce a set of state-relation pairs $\{\langle s_i, R \rangle\}_{i \in I}$ in the result of an action. If an observation discards some state s_i, the pair $\langle s_i, R \rangle$ is removed from this set. We conclude that R is not possible when all pairs including it are removed from the set.

A nondeterministic domain description D is a finite set of *transition rules* of the form "a **causes** F **if** G" which describe the effects of actions, for F and G propositional state formulae. We say that F is the *head* and G is the *tail* of those rules. When $G \equiv TRUE$ we write "a **causes** F".

The semantics of a domain description that we choose is compatible with the *standard semantics* belief update operator of [20]. We define it below by first *completing* the description and then mapping the completed description to a transition relation.

For domain description D we define a transition system with $\mathcal{P}_D$ and $\mathcal{A}_D$ the sets of propositional fluents and actions mentioned in D, respectively. For action a and fluent f, let

$$G_D(a, f) = \bigvee \{G \mid \text{“}a \textbf{ causes } F \textbf{ if } G\text{”} \in D,\ f \in L(F)\},$$

a disjunction of the preconditions of rules that possibly affect fluent f (an empty disjunction is equivalent to FALSE). We use "a **keeps** f **if** G" as a shorthand for the rules "a **causes** f **if** $f \wedge G$" and "a **causes** $\neg f$ **if** $\neg f \wedge G$". It designates the *non-effects* of action a. Define $Comp(D)$, the *completion of* D

$$Comp(D) = D \cup \{\text{“}a \textbf{ keeps } f \textbf{ if } \neg G_D(a, f)\text{”} \mid \\ a \in \mathcal{A}, f \in \mathcal{P},\ G_D(a, f) \not\equiv TRUE\}.$$

This definition is well behaved, in the sense that $Comp(D) = Comp(D \cup D')$, if $D' \subseteq Comp(D)$.

Let $F_D(a, s) = \{F \mid \text{“}a \textbf{ causes } F \textbf{ if } G\text{”} \in D,\ s \models G\}$, the set of effects of a in s, according to D. D defines a transition relation R_D as follows

$$R_D = \{\langle s, a, s' \rangle \mid s, s' \in \mathcal{S},\ a \in \mathcal{A},\ s' \models F_D(a, s)\} \quad (1)$$

When there is no confusion, we write R for R_D. We say that two domain descriptions D_1, D_2 are equivalent ($D_1 \equiv D_2$), if $R_{D_1} = R_{D_2}$. D is a *complete domain description*, if $R_D = R_{Comp(D)}$. In that case we say that R is *completely defined by* D.

Time step	1		2		3		4		5		6		7
Action		go-W		sw-on		go-E		sw-on		go-W		go-E	
Location	E		$\neg E$		$\neg E$		E		E		$\neg E$		E
Bulb	?		$\neg lit$		$\neg lit$		?		?		lit		?
Switch	$\neg sw$		?		?		$\neg sw$		sw		?		sw
Possible		OK		$\neg OK$		OK		OK		OK		OK	

Figure 2. An action-observation sequence (table entries are observations). Legend: E: east; $\neg E$: west; lit: light is on; $\neg lit$: light is off; sw: switch is on; $\neg sw$: switch is off; OK: action executable; $\neg OK$: action not executable.

Example 2 *Consider the scenario of Figure 2 and assume that actions are deterministic, unconditional, and always executable (assuming no action was performed at step 2). Then, every action affects every fluent either negatively, positively, or not at all. Consequently, every transition relation R is completely defined by some D such that (viewing a tuple as a set of its elements)*

$$D \in \prod_{a \in \left\{ \begin{array}{c} \textit{go-W} \\ \textit{go-E} \\ \textit{sw-on} \end{array} \right\}} \left\{ \begin{array}{l} a \textbf{ causes } E, \\ a \textbf{ causes } \neg E \\ a \textbf{ keeps } E \end{array} \right\} \times \left\{ \begin{array}{l} a \textbf{ causes } sw, \\ a \textbf{ causes } \neg sw \\ a \textbf{ keeps } sw \end{array} \right\} \times \left\{ \begin{array}{l} a \textbf{ causes } \textit{lit}, \\ a \textbf{ causes } \neg \textit{lit} \\ a \textbf{ keeps } \textit{lit} \end{array} \right\}$$

Say that initially we know the effects of go-E, go-W, but do not know what sw-on does. Then, transition filtering starts with the product set of $\mathcal{R}$ (of 27 possible relations) and all possible 2^3 states. Also, at time step 4 we know that the world state is exactly $\{E, \neg \textit{lit}, \neg sw\}$. We try sw-on and get that $Filter[\textit{sw-on}](\rho_4)$ includes the same set of transition relations but with each of those transition relations projecting the state $\{E, \neg \textit{lit}, \neg sw\}$ to an appropriate choice from $\mathcal{S}$. When we receive the observations $o_5 = \neg E \wedge \neg sw$ of time step 5, $\rho_5 = Filter[o_5](Filter[\textit{sw-on}](\rho_4))$ removes from the transition belief state all the relations that gave rise to $\neg E$ or to $\neg sw$. We are left with transition relations satisfying one of the tuples in

$$\left\{ \begin{array}{l} \textit{sw-on} \textbf{ causes } E, \\ \textit{sw-on} \textbf{ keeps } E \end{array} \right\} \times \left\{ \textit{sw-on} \textbf{ causes } sw \right\} \times \left\{ \begin{array}{l} \textit{sw-on} \textbf{ causes } \textit{lit} \\ \textit{sw-on} \textbf{ causes } \neg \textit{lit} \\ \textit{sw-on} \textbf{ keeps } \textit{lit} \end{array} \right\}$$

Finally, when we perform action go-W, again we update the set of states associated with every transition relation in the set of pairs ρ_5. When we receive the observations of time step 6, we conclude $\rho_6 = Filter[o_6](Filter[\textit{go-W}](\rho_5)) =$

$$\left\{ \left\langle \left\{ \begin{array}{l} \neg E \\ \textit{lit} \\ sw \end{array} \right\}, \left\{ \begin{array}{l} \textit{sw-on} \textbf{ causes } E, \\ \textit{sw-on} \textbf{ causes } sw, \\ \textit{sw-on} \textbf{ causes } \textit{lit}, \\ \textit{go-E}\ldots \end{array} \right\} \right\rangle, \left\langle \left\{ \begin{array}{l} \neg E \\ \textit{lit} \\ sw \end{array} \right\}, \left\{ \begin{array}{l} \textit{sw-on} \textbf{ keeps } E, \\ \textit{sw-on} \textbf{ causes } sw, \\ \textit{sw-on} \textbf{ causes } \textit{lit}, \\ \textit{go-E}\ldots \end{array} \right\} \right\rangle \right\} \quad (2)$$

SLAF reduces to filtering (updating the agent's belief state) [1, 20, 10] when the transition model is fully specified.

Theorem 3 *Let $\rho = \sigma \times \{R\}$, where $\sigma \subseteq \mathcal{S}$ and $R \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, and let $\langle a_i, o_i \rangle_{i \leq t}$ be a sequence of actions and observations. If $Filter_R[\langle a_i, o_i \rangle_{i \leq t}](\sigma)$ is the belief-state filtering*[1] *of σ with $\langle a_i, o_i \rangle_{i \leq t}$, then $Filter[\langle a_i, o_i \rangle_{i \leq t}](\rho) = Filter_R[\langle a_i, o_i \rangle_{i \leq t}](\sigma) \times \{R\}$.*

3 Logical Filtering of Transition Models

The example in the previous section illustrates how the explicit representation of transition belief states may be doubly exponential in the number of domain features and the number of actions. In this section we follow the intuition that propositional logic can serve to represents ρ more compactly. *From here forth we assume that our actions are deterministic.*

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

3.1 Representing Transition Belief States

We define a propositional logical language that allows us to represent sets of domain descriptions (thus, sets of transition relations). Let $\mathbb{P}_1, \mathbb{P}_2 \subseteq \mathcal{L}(\mathcal{P})$ be sets of state formulae such that $\mathbb{P}_1$ includes only literals or $FALSE$, $\mathbb{P}_2$ includes only terms (conjunctions of literals) that are not equivalent to FALSE, and for all $\varphi, \psi \in \mathbb{P}_1 \cup \mathbb{P}_2$, if $\varphi \equiv \psi$, then $\varphi = \psi$ or $\varphi \equiv TRUE$. We define a propositional vocabulary

$$L(\mathbb{P}_1, \mathbb{P}_2) = \{a_G^F \mid a \in \mathcal{A},\ F \in \mathbb{P}_1,\ G \in \mathbb{P}_2\}.$$

Theorem 4 *For every rule $r =$"a **causes** F **if** G", for F, G state formulae, there is a set of transition rules $TR = \{$"a **causes** l_i **if** t_i"$\}_{i \in I}$, with a set of indices I, terms t_i, and literals l_i, such that $r \equiv TR$ (i.e., we can exchange r for TR, and get an equivalent domain description).*

In the rest of this paper we implicitly assume that $\mathbb{P}_1, \mathbb{P}_2 \subseteq \mathcal{L}(\mathcal{P})$ are sets of state formulae as above. Also, D is a *complete* domain description with effects in $\mathbb{P}_1$ and preconditions in $\mathbb{P}_2$. We also assume that if $\neg \exists s' R_D(s, a, s')$ for some $s \in \mathcal{S}, a \in \mathcal{A}$, then there is a rule "$a$ **causes** FALSE **if** G" such that $s \models G$.

For set of formulae $\mathbb{P}_2$ we define $Bottom(\mathbb{P}_2) = \{G \in \mathbb{P}_2 \mid G \not\equiv FALSE,\ \ \forall G' \in \mathbb{P}_2\ [(G' \models G) \wedge (G' \not\equiv FALSE)] \Rightarrow G = G'\}$, i.e., $Bottom(\mathbb{P}_2)$ is the set of strongest preconditions in $\mathbb{P}_2$.

Definition 5 *We define the theory*

$$T_D^L = \textit{rules}_D \wedge \textit{implied-weaker-rules} \wedge \\ \textit{implied-stronger-rules} \wedge \textit{exec-preconds}_D \quad (3)$$

[1] Filtering semantics as defined in [1].

$$\textit{rules}_D = \{a_G^F \in L \mid \text{“}a \textbf{ causes } F \textbf{ if } G\text{”} \in D\}$$
$$\textit{implied-weaker-rules} = \bigwedge_{\substack{F \in \mathbb{P}_1 \\ G, G' \in \mathbb{P}_2 \\ a \in \mathcal{A}}} (a_{G'}^F \wedge (G \Rightarrow G') \Rightarrow a_G^F)$$
$$\textit{implied-stronger-rules} = \bigwedge_{\substack{F \in \mathbb{P}_1 \\ G, G', G'' \in \mathbb{P}_2 \\ G'' \equiv G \vee G' \\ a \in \mathcal{A}}} (a_G^F \wedge a_{G'}^F \Rightarrow a_{G''}^F) \wedge$$
$$\bigwedge_{\substack{F, F', F'' \in \mathbb{P}_1 \\ F'' \equiv F \wedge F' \\ G \in \mathbb{P}_2 \\ a \in \mathcal{A}}} (a_G^F \wedge a_G^{F'} \Rightarrow a_G^{F''})$$
$$\textit{exec-preconds}_D = \{\neg a_G^{FALSE} \in L \mid G \in Bottom(\mathbb{P}_2), \forall G' \in \mathbb{P}_2\ ((G \models G') \Rightarrow \text{“}a \textbf{ causes } \textit{FALSE} \textbf{ if } G'\text{”} \notin D)\}.$$

The intention is that $a_G^F \in L$ is true in T_D^L exactly when “a **causes** F **if** G” is in D or there is a stronger transition rule “a **causes** F' **if** G'” in D.

The following theorem shows how we can represent deterministic transition relations (with conditional effects) using only the positive causality statements[2].

Theorem 6 (Representing Deterministic Actions) *1. T_D is a complete theory*

2. If $T_D \models a_G^F$, then for every s, s', if $R_D(s, a, s')$ and $s \models G$, then $s' \models F$.

3. If $T_D \models \neg a_G^F$, then there are s, s' such that $R_D(s, a, s')$ and $s \models G$, $s' \models \neg F$.

Consequently, for every pair of complete domain descriptions D_1, D_2, $D_1 \equiv D_2$ iff $T_{D_1} \equiv T_{D_2}$. Thus, every transition relation R has a unique theory T_D and every theory defines a unique transition relation. (We write T_R for the theory representing transition relation R.)

Corollary 7 (Always-Executable, Deterministic) *If in D all actions are always executable, then[3] exec-preconds $= \{\neg a_G^{FALSE} \mid a \in \mathcal{A},\ G \in Bottom(\mathbb{P}_2)\}$ and*

$$T_D \equiv \textit{rules}_D \ \wedge\ \textit{exec-preconds} \wedge$$
$$\bigwedge_{\substack{F \in \mathbb{P}_1 \\ G, G' \in \mathbb{P}_2 \\ a \in \mathcal{A}}} (a_{G'}^F \wedge (G \Rightarrow G') \Rightarrow a_G^F) \ \wedge$$
$$\bigwedge_{\substack{F \in \mathbb{P}_1 \\ G \in \mathbb{P}_2 \\ a \in \mathcal{A}}} \neg(a_G^F \wedge a_G^{\neg F}) \ \wedge \bigwedge_{\substack{F \in \mathbb{P}_1 \\ G, G', G'' \in \mathbb{P}_2 \\ G'' \equiv G \vee G' \\ a \in \mathcal{A}}} (a_G^F \wedge a_{G'}^F \Rightarrow a_{G''}^F)$$

Define $a^{f\circ} = a_f^f \wedge a_{\neg f}^{\neg f}$.

Corollary 8 (Unconditional, Always-Exec., Deterministic) *Let $\mathbb{P}_2 = \{TRUE\}$, and assume that D possibly includes sentences of the form “a **keeps** F”, and no sentences of the form “$a\ causes\ FALSE$”. Then,*

$$T_D \equiv rules_D \ \wedge \bigwedge_{f \in \mathcal{P}, a \in \mathcal{A}} (a^f \bar{\vee} a^{\neg f} \bar{\vee} a^{f\circ}).$$

We encode sets of domain descriptions as follows: For a set $\mathcal{R} \subset \mathbb{P}(\mathcal{S} \times \mathcal{A} \times \mathcal{S})$ let[4] $T_\mathcal{R} = \bigvee_{R \in \mathcal{R}} T_R$. For a tuple $\langle s, R \rangle$, $s \in \mathcal{S}$, we define $T_{\langle s,R \rangle} = T_R \wedge s$. Finally, for a transition belief state, ρ, we define $T_\rho = \bigvee_{\langle s,R \rangle \in \rho} T_{\langle s,R \rangle}$.

[2] At present it is not clear to the author how one can observe non-causality in nondeterministic settings.

[3] D is omitted as a subscript because it is not relevant.

[4] We assume that the set of fluents $\mathcal{P}$ is finite.

Example 9 *Consider ρ_6 from Example 2 (equation 2). There, we considered only deterministic, same-effect, always-executable actions. We take $\mathbb{P}_1$ to include only unit clauses, and $\mathbb{P}_2 = \{TRUE\}$. We can write ρ_6 using a logical formula that is satisfied only by the tuples in ρ_6:*

$$T_{\rho_6} \equiv \begin{pmatrix} \neg E \wedge \\ sw \wedge \\ lit \end{pmatrix} \bigwedge \begin{pmatrix} \textit{go-W}^{\neg E} \wedge \\ \textit{go-W}^{sw\circ} \wedge \\ \textit{go-W}^{lit\circ} \end{pmatrix} \bigwedge \begin{pmatrix} \textit{go-E}^{E} \wedge \\ \textit{go-E}^{sw\circ} \wedge \\ \textit{go-E}^{lit\circ} \end{pmatrix} \bigwedge$$
$$\left\{ \begin{matrix} (\textit{sw-on}^E \vee \textit{sw-on}^{E\circ}) \wedge \\ \textit{sw-on}^{sw} \wedge \\ \textit{sw-on}^{lit} \end{matrix} \right) \wedge \bigwedge\nolimits_{f \in \mathcal{P}, a \in \mathcal{A}} (a^f \bar{\vee} a^{\neg f} \bar{\vee} a^{f\circ})$$

Notice that, e.g., $\neg$go-E$^{\neg E}$ and $\neg$go-E$^{E\circ}$ are logical consequences of T_{ρ_6}.

The same way allows us to represent a much larger set of transition relations. For example, in the previous section we avoided listing the contents of the set of tuples ρ_0 because it was too large (27 possible relations cross-combined with all 2^3 world states). Now we can write it simply as follows:

$$\begin{pmatrix} \textit{go-W}^{\neg E} \\ \textit{go-W}^{sw\circ} \\ \textit{go-W}^{lit\circ} \end{pmatrix} \wedge \begin{pmatrix} \textit{go-E}^{E} \wedge \\ \textit{go-E}^{sw\circ} \wedge \\ \textit{go-E}^{lit\circ} \end{pmatrix} \wedge$$
$$\begin{pmatrix} (\textit{sw-on}^E \vee \textit{sw-on}^{\neg E} \vee \textit{sw-on}^{E\circ}) \wedge \\ (\textit{sw-on}^{sw} \vee \textit{sw-on}^{\neg sw} \vee \textit{sw-on}^{sw\circ}) \wedge \\ (\textit{sw-on}^{lit} \vee \textit{sw-on}^{\neg lit} \vee \textit{sw-on}^{lit\circ}) \end{pmatrix} \wedge \bigwedge\nolimits_{f \in \mathcal{P}, a \in \mathcal{A}} (a^f \bar{\vee} a^{\neg f} \bar{\vee} a^{f\circ})$$

Thus, we can represent a transition belief state, ρ, with a logical formula, φ, over the propositional state fluents and the propositional symbols for effect sentences. φ represents the set of tuples $\langle s, R \rangle$ that satisfy it. We call this logical representation a *transition belief formula*. It follows that $\varphi \models$ *base*, if we take *base* to be the subformula of T_D that is not *rules*$_D$ in the appropriate case of Corollaries 7, 8 because *base* is independent of D in those cases.

3.2 Filtering Logical Action Models

For a deterministic (possibly conditional) action, a, define the effect model of a for time t to be

$$T_{\textit{eff}}(a, t) = \bigwedge\nolimits_{l \in \mathbb{P}_1, G \in \mathbb{P}_2} ((a_t \wedge a_G^l \wedge G_t) \Rightarrow l_{t+1}) \ \wedge$$
$$\bigwedge\nolimits_{l \in \mathbb{P}_1} (l_{t+1} \wedge a_t \Rightarrow (\bigvee\nolimits_{G \in \mathbb{P}_2} (a_G^l \wedge G_t))) \tag{4}$$

where a_t is a propositional symbol asserting that action a occurred at time t, and we use the convention that $\varphi_t = \varphi_{[\mathcal{P}/\mathcal{P}_t]}$, i.e., φ_t is the result of replacing every propositional symbol of φ with the same propositional symbol that now has an added subscript, t. The first part of the conjunction is the assertion that if a executes at time t, and it causes l, if G holds, and G holds at time t, then l holds at time $t+1$. The second part of the conjunction says that l is true at time $t+1$ after a's execution only if a has an effect l conditional on some G, and this G is true at time t.

For an always-executable, non-conditional action, a, we get a simpler formula

$$T_{\textit{eff}}(a, t) \equiv \ \bigwedge\nolimits_{l \in \mathbb{P}_1} ((a_t \wedge (a^l \vee (a_l^l \wedge l_t)) \Rightarrow l_{t+1}) \ \wedge$$
$$\bigwedge\nolimits_{l \in \mathbb{P}_1} (l_{t+1} \wedge a_t \Rightarrow (a^l \vee (a_l^l \wedge l_t)))$$

Definition 10 (Logical Transition Filtering)
Progression: $Filter[a](\varphi) = Cn^{L_{t+1}}(\varphi_t \wedge a_t \wedge T_{\textit{eff}}(a, t))$
Filtering: $Filter[o](\varphi) = \varphi \wedge o$

Thus, $Filter[a](\varphi)$ is the set of consequences of φ_t in the vocabulary $L_{t+1} = \mathcal{P}_{t+1} \cup L$, the vocabulary that includes only fluents of time $t+1$ and effect propositions from L. The following theorem shows that filtering a transition belief formula is equivalent to filtering a transition belief state.

Theorem 11 *For φ transition belief formula, a action,*

$$Filter[a](\{\langle s,R\rangle \in \mathfrak{S} \mid \langle s,R\rangle \text{ satisfies } \varphi\}) = \{\langle s,R\rangle \in \mathfrak{S} \mid \langle s,R\rangle \text{ satisfies } Filter[a](\varphi)\}$$

3.3 Distribution Properties

Several distribution properties always hold for filtering of transition belief states (or formulae). The first one follows from set theoretical considerations. 11.

Corollary 12 *For φ, ψ transition belief formulae, a action,*
1. $Filter[a](\varphi \vee \psi) \equiv Filter[a](\varphi) \vee Filter[a](\psi)$
2. $\models Filter[a](\varphi \wedge \psi) \Rightarrow Filter[a](\varphi) \wedge Filter[a](\psi)$

Stronger properties hold if filtering an action is a 1:1 mapping between state-transition-relation pairs.

Corollary 13 *Let a be an action, and φ, ψ be transition belief formulae. Then, $Filter[a](\varphi \wedge \psi) \equiv Filter[a](\varphi) \wedge Filter[a](\psi)$, if*
1. *For every transition relation R possible with $\varphi \vee \psi$, a maps states in $\{s \mid \langle s,R\rangle \models \varphi\}$ 1:1 to states in $\mathcal{S}$, or*
2. *Whenever $\langle s_1,R\rangle \models \varphi \vee \psi$, $\langle s_2,R\rangle \models \varphi \vee \psi$, then $s_1 = s_2$.*

Corollary 14 *For action a, state $s \in \mathcal{S}$, and φ, ψ transition belief formulae,*

$$Filter[a](s \wedge \varphi \wedge \psi) \equiv Filter[a](s \wedge \varphi) \wedge Filter[a](s \wedge \psi)$$

This last corollary explains the relationship between learning in fully observable and partially observable worlds. Our algorithms for learning world models will be more tractable when our agent observes more of the environment. We see in Section 4 that polynomial-time algorithms exist for SLAF when filtering distributes over conjunctions.

Finally, when $T_{\mathit{eff}}(a,t) \equiv T^1 \wedge T^2$ and $\varphi \equiv \varphi^1 \wedge \varphi^2$, such that $L(T^1) \cap L(T^2) = \emptyset$ and $L(\varphi^i) \subseteq L(T^i)$, for $i \in \{1,2\}$, then the filtering factors into filtering of φ^1, φ^2 separately. More generally, the following holds.

Theorem 15 *Let a be an action, let $s \in \mathcal{S}$ be a state, let $\mathbb{P}_1$ include literals in $\mathcal{P}$ and FALSE, let $\mathbb{P}_2^i$ ($i \in \{1,2\}$) include clauses in $\mathcal{P}^i$ such that $L(\mathcal{P}) = L(\mathcal{P}^1)\dot{\cup}L(\mathcal{P}^2)$, and let $\varphi^i \in \mathcal{L}(L(\mathbb{P}_1, \mathbb{P}_2^i) \cup \mathcal{P})$ ($i \in \{1,2\}$) be transition belief formulae. Then,*

$$Filter[a](\varphi \wedge \psi) \equiv Filter[a](\varphi) \wedge Filter[a](\psi)$$

4 Factored Learning and Filtering

Learning world models is easier when filtering distributes over logical connectives. The computation becomes tractable, with the bottleneck being the time to filter each part separately. Figure 3 presents an algorithm for SLAF using this observation. Filtering of a single fluent (done in function *Fluent-SLAF*) and more efficient solutions are the focus of the rest of this section.

```
PROCEDURE Factored-SLAF($\langle a_i, o_i\rangle_{0<i\leq t}$,$\varphi$)
$\forall i$, $a_i$ action, $o_i$ observation, $\varphi$ transition belief formula.
1. For $i$ from 1 to $t$ do,
   (a) Set $\varphi \leftarrow$ Step-SLAF($o_i$,$a_i$,$\varphi$).
   (b) Eliminate subsumed clauses in $\varphi$.
2. Return $\varphi$.
```

```
PROCEDURE Step-SLAF($o$,$a$,$\varphi$)
$o$ an observation sentence (conjunction of literals), $a$ an action, $\varphi$ a transition belief formula.
1. If $\varphi$ is a literal, then return $o \wedge$Fluent-SLAF($o$,$a$,$\varphi$).
2. If $\varphi = \varphi_1 \wedge \varphi_2$, return Step-SLAF($o$,$a$,$\varphi_1$)$\wedge$Step-SLAF($o$,$a$,$\varphi_2$).
3. If $\varphi = \varphi_1 \vee \varphi_2$, return Step-SLAF($o$,$a$,$\varphi_1$)$\vee$Step-SLAF($o$,$a$,$\varphi_2$).
```

```
PROCEDURE Fluent-SLAF($o$,$a$,$\varphi$)
$o$ an observation sentence (conjunction of literals), $a$ an action, $\varphi$ a fluent.
1. Return $Cn^{L_{t+1}}(\varphi_t \wedge a_t \wedge T_{eff}(a,t))$.
```

Figure 3. SLAF using distribution over $\wedge, \vee$

4.1 Always-Executable STRIPS Actions

STRIPS actions [4] are deterministic and unconditional (but sometimes not executable). In this section we examine them with the assumption that our they always executable. We return to inexecutability in Section 4.2.

Let $L_f = \{f\} \cup \{a^f, a^{\neg f}, a^{f\circ} \mid a \in \mathcal{A}\}$ be the propositional vocabulary including only the propositional fluent symbol f and effect propositions mentioning f. We say that φ is a *fluent-factored transition belief formula*, if $\varphi = base \wedge \bigwedge_{f\in\mathcal{P}} \varphi_f$, with $L(\varphi_f) \subseteq L_f$. When a transition belief formula φ is fluent-factored, then the result of filtering is also a fluent-factored formula.

Theorem 16 *Let $\varphi = base \wedge \bigwedge_{f\in\mathcal{P}} \varphi_f$ be a fluent-factored transition belief formula, with $L(\varphi_f) \subseteq L_f$. Then,*

$$Filter[a](\varphi) \equiv base \wedge \bigwedge_{f\in\mathcal{P}} Filter[a](\varphi_f)$$

and $L(Filter[a](\varphi_f)) \subseteq L_f$. Also, if o is a conjunction of literals, then $Filter[o](\varphi)$ is fluent-factored.

We are left with the problem of filtering each φ_f with a and o. Let $L_f^0 = L_f \setminus \{f\}$.

Theorem 17 *Let φ_f be a transition belief formula with $L(\varphi_f) \subseteq L_f$. Then,*

$$\begin{aligned} Filter[a](\varphi_f) \equiv\ & (f \Rightarrow (a^f \vee ((\varphi_f \Rightarrow f) \wedge a^{f\circ})))\wedge \\ & (\neg f \Rightarrow (a^{\neg f} \vee ((\varphi_f \Rightarrow \neg f) \wedge a^{f\circ})))\wedge \\ & Cn(\varphi_f) \cap \mathcal{L}(L_f^0) \end{aligned}$$

We can compute $Cn(\varphi_f) \cap \mathcal{L}(L_f^0)$ without general-purpose automated deduction, if we keep φ_f in a the following form

$$(\neg f \vee expl_f) \wedge (f \vee expl_{\neg f}) \wedge \xi_f$$

where $expl_f$, $expl_{\neg f}$, and ξ_f are in $\mathcal{L}(L_f^0)$. Every formula in $\mathcal{L}(L_f)$ is logically equivalent to a formula in this form,

```
PROCEDURE AE-STRIPS-SLAF($\langle a_i, o_i\rangle_{0<i\leq t}$,$\varphi$)
$\forall i$, $a_i$ an action, $o_i$ an observation, $\varphi = \bigwedge_{f\in\mathcal{P}} \varphi_f$ a fluent-factored transition belief formula.
1. For $i$ from 1 to $t$ do,
   (a) Set $\varphi \leftarrow \bigwedge_{f\in\mathcal{P}}$ AE-STRIPS-Fluent-SLAF($o_i$,$a_i$,$\varphi_f$).
   (b) Eliminate subsumed clauses in $\varphi$.
2. Return $\varphi$.
```

```
PROCEDURE AE-STRIPS-Fluent-SLAF($o$,$a$,$\varphi$)
$o$ conjunction of literals, $a$ action, $\varphi = (\neg f \vee expl_f) \wedge (f \vee expl_{\neg f}) \wedge \xi_f$ in $f$-free form.
1. Set $expl'_f = a^f \vee (a^{f\circ} \wedge expl_f))$.
2. Set $expl'_{\neg f} = a\neg f \vee (a^{f\circ} \wedge expl_{\neg f}))$.
3. If $f$ does not appear (positively or negatively) in $o$, then
   set $\xi'_f = \xi_f$.
4. Else, if $o \models f$ (we observed $f$), then
   (a) Set $\xi'_f \leftarrow \xi_f \wedge expl'_f$.
   (b) Set $expl'_f \leftarrow TRUE$ and $expl'_{\neg f} \leftarrow FALSE$.
5. Else (we observed $\neg f$),
   (a) Set $\xi'_f \leftarrow \xi_f \wedge expl'_{\neg f}$.
   (b) Set $expl'_f = FALSE$ and $expl'_{\neg f} = TRUE$.
6. Return $(\neg f \vee expl'_f) \wedge (f \vee expl'_{\neg f}) \wedge \xi'_f$
```

Figure 4. SLAF with always-executable STRIPS.

which we call *f-free form*. Figure 4 presents a complete algorithm for SLAF using this observation and form.

Now, we examine the size of the formula that results from filtering. A transition belief formula φ in CNF is in *f-k-CNF* if every clause mentioning f or $\neg f$ has at most k literals. For example, $f \vee a^f$ is in f-2-CNF, but $a_1^f \vee a_2^{\neg f}$ is in f-0-CNF. We also say that *a, o determine f in φ* if $\varphi \models a^f$ or $\varphi \models a^{\neg f}$ or $o \models f$ or $o \models \neg f$.

Corollary 18 *Let $\varphi = \bigwedge_{f\in\mathcal{P}} \varphi_f$ be a fluent-factored transition belief formula, and o a conjunction of literals. Then, Procedure AE-STRIPS-SLAF($\langle a, o\rangle$, φ) returns a fluent-factored transition belief formula $\varphi' \equiv Filter[o](Filter[a](\varphi))$ in time $O(|\varphi|)$. Further, if φ is in f-k-CNF and φ, a, o determine f, then φ' is in f-1-CNF. Otherwise, φ' is in f-$(k+1)$-CNF.*

Thus, our transition belief formula remains compact, if we know the effect of our action on a in φ, or we observe f frequently enough. For example, if we observe every fluent every 4 actions, then our transition belief state is always in 5-CNF, meaning that it is of size at most $O(n \cdot m^5)$ for n fluents and m actions (this is much better than the worst case which can be doubly-exponential in n, m).

4.2 STRIPS Actions

Assume that we allow actions to fail but we always observe such success and inexecutability. In both executable/inexecutable cases we learn something about the executability of the action under consideration. Unfortunately, this prevents factoring for the general case of actions, unless one of the conditions of section 3.3 holds. In the rest of this section we assume that either one of those conditions holds, or we accept the approximation offered by Corollary 12. Define

$$a_e^l \equiv \bigwedge_{G\in Bottom(\mathbb{P}_2)} (\neg a_G^{FALSE} \Rightarrow a_G^l)$$
$$a_e^{l\circ} \equiv \bigwedge_{G\in Bottom(\mathbb{P}_2)} (\neg a_G^{FALSE} \Rightarrow a_G^{l\circ})$$

Let $\mathcal{B}(a) \equiv \bigwedge_{l\in\mathbb{P}_1} (a_e^l \Rightarrow l_{t+1}) \wedge base$.

Corollary 19 (STRIPS-SLAF of a literal) *Let l be a literal in $L(\mathbb{P}_1, \mathbb{P}_2)$ and a an action. If $l \in \mathbb{P}_1$, then*

$$Filter[a, OK](l) \equiv (l_{t+1} \Leftrightarrow (a_e^l \vee a_e^{l\circ})) \wedge \neg a_l^{FALSE} \wedge \mathcal{B}(a)$$
$$Filter[a, \neg OK](l) \equiv l_{t+1} \wedge a_l^{failed} \wedge base$$

If $l \notin \mathbb{P}_1$ (i.e., l is an effect literal), then

$$Filter[a, OK](l) \equiv l \wedge \neg a_{TRUE}^{FALSE} \wedge \mathcal{B}(a)$$
$$Filter[a, \neg OK](l) \equiv l \wedge a_{TRUE}^{failed} \wedge base$$

Now we replace Procedure Fluent-SLAF in Figure 3 with Procedure STRIPS-Fluent-SLAF of Figure 5.

```
PROCEDURE STRIPS-Fluent-SLAF($o$,$a$,$\varphi$)
$o$ conjunction of literals, $a$ action, $\varphi$ fluent.
1. If $l \in \mathbb{P}_1$, then
   (a) If $o \models OK$, then return
       $(l_{t+1} \iff (a_e^l \vee a_e^{l\circ})) \wedge \neg a_l^{FALSE} \wedge \mathcal{B}(a)$.
   (b) ($o \models \neg OK$) Return $l_{t+1} \wedge a_l^{failed} \wedge base$.
2. ($l \in \mathbb{P}_1$) If $o \models OK$, then return $l \wedge \neg a_{TRUE}^{FALSE} \wedge \mathcal{B}(a)$.
3. Return $l \wedge a_{TRUE}^{failed} \wedge base$.
```

Figure 5. SLAF with STRIPS actions, observing success/failure.

4.3 Conditional Effects

A similar formula to the one above holds for the general case of deterministic actions (possibly conditional). We assume that a has preconditions using the propositions in $\{l^1, ..., l^k\}$.

Theorem 20 *Filtering for a literal $l \in \mathbb{P}_1$ satisfies*

$$Filter[a, OK](l) \equiv (l_{t+1}^e \Rightarrow \bigvee_{\substack{G \in Bottom(\mathbb{P}_2, \{l^1, ..., l^k\}) \\ G \models l^p}} (a_G^{l_e} \wedge \bigwedge_{j\leq k} ((a_G^{l^j} \Rightarrow l_{t+1}^j) \wedge (a_G^{\neg l^j} \Rightarrow \neg l_{t+1}^j))))$$

5 Conclusions

We presented general principles and algorithms for learning and filtering in partially observable domains. Some of our results guarantee polynomial-time filtering of transition belief states indefinitely. In particular, STRIPS domains in which actions are always executable (or when the preconditions for those actions are known) can be learned in polynomial time, if fluents are observed frequently enough.

We expect our algorithms to generalize to action schemas, where actions are parametrized in various ways (e.g., objects on which they operate, and numbers that modify the extent of the action). We plan to explore this direction in the future, as well as extending this work to agents that have a prior distribution, knowledge, or preference over the possible worlds or the actions' effects.

REFERENCES

[1] Eyal Amir and Stuart Russell, ‘Logical filtering’, in *Proc. Eighteenth International Joint Conference on Artificial Intelligence (IJCAI ’03)*, pp. 75–82. Morgan Kaufmann, (2003).

[2] Scott Benson, ‘Inductive learning of reactive action models’, in *Proceedings of the 12th International Conference on Machine Learning (ICML-94)*, (1995).

[3] Lonnie Chrisman, ‘Abstract probabilistic modeling of action’, in *Proc. National Conference on Artificial Intelligence (AAAI ’92)*. AAAI Press, (1992).

[4] Richard Fikes, Peter Hart, and Nils Nilsson, ‘Learning and executing generalized robot plans’, *Artificial Intelligence*, **3**, 251–288, (1972).

[5] Yolanda Gil, ‘Learning by experimentation: Incremental refinement of incomplete planning domains’, in *Proceedings of the 11th International Conference on Machine Learning (ICML-94)*, pp. 10–13, (1994).

[6] Brian Hlubocky and Eyal Amir, ‘Knowledge-gathering agents in adventure games’, in *AAAI-04 Workshop on Challenges in Game AI*. AAAI Press, (2004).

[7] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra, ‘Planning and acting in partially observable stochastic domains’, *Artificial Intelligence*, **101**, 99–134, (1998).

[8] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, ‘Reinforcement learning: a survey’, *Journal of Artificial Intelligence Research*, **4**, 237–285, (1996).

[9] Michael Kearns, Yishay Mansour, and Andrew Y. Ng, ‘Approximate planning in large pomdps via reusable trajectories’, in *Proceedings of the 12th Conference on Neural Information Processing Systems (NIPS’99)*, pp. 1001–1007. MIT Press, (2000).

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

[11] Michael L. Littman, *Algorithms for sequential decision making*, Ph.D. dissertation, Department of Computer Science, Brown University, 1996. Technical report CS-96-09.

[12] R. Andrew McCallum, ‘Instance-based utile distinctions for reinforcement learning with hidden state’, in *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*. Morgan Kaufmann, (1995).

[13] Nicolas Meuleau, Leonid Peshkin, Kee-Eung Kim, and Leslie Pack Kaelbling, ‘Learning finite-state controllers for partially observable environments’, in *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI ’99)*. Morgan Kaufmann, (1999).

[14] Andrew Y. Ng and Michael Jordan, ‘Pegasus: A policy search method for large mdps and pomdps’, in *Proc. Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI ’00)*, pp. 406–415. Morgan Kaufmann, (2000).

[15] Tim Oates and Paul R. Cohen, ‘Searching for planning operators with context-dependent and probabilistic effects’, in *Proc. National Conference on Artificial Intelligence (AAAI ’96)*, pp. 863–868. AAAI Press, (1996).

[16] Hanna M. Pasula, Luke S. Zettlemoyer, and Leslie Pack Kaelbling, ‘Learning probabilistic relational planning rules’. AAAI Press, (2004).

[17] L. R. Rabiner, ‘A tutorial on hidden Markov models and selected applications in speech recognition’, *Proceedings of the IEEE*, **77**(2), 257–285, (February 1989).

[18] Matthew D. Schmill, Tim Oates, and Paul R. Cohen, ‘Learning planning operators in real-world, partially observable environments’, in *Proceedings of the 5th Int’l Conf. on AI Planning and Scheduling (AIPS’00)*, pp. 246–253. AAAI Press, (2000).

[19] Xuemei Wang, ‘Learning by observation and practice: an incremental approach for planning operator acquisition’, in *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pp. 549–557. Morgan Kaufmann, (1995).

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