Learning Safe Action Models with Partial Observability

Brendan Juba
Washington University in St. Louis
bjuba@wustl.edu

Hai S. Le
Washington University in St. Louis
hsle@wustl.edu

Roni Stern
Ben Gurion University
sternron@post.bgu.ac.il

Abstract

A common approach for solving planning problems is to model them in a formal language such as the Planning Domain Definition Language (PDDL), and then use an appropriate PDDL planner. Several algorithms for learning PDDL models from observations have been proposed but plans created with these learned models may not be sound. We propose two algorithms for learning PDDL models that are guaranteed to be safe to use even when given observations that include partially observable states. We analyze these algorithms theoretically, characterizing the sample complexity each algorithm requires to guarantee probabilistic completeness. We also show experimentally that our algorithms are often better than FAMA, a state-of-the-art PDDL learning algorithm.

1 Introduction

Classical planning, i.e., planning in a discrete, deterministic, and fully observable environment, is a useful abstraction for solving many planning problems. In order to use these planners, however, one must first model the problem at hand in a formal language, such as the Planning Domain Definition Language (PDDL). This is not an easy task. Therefore, several approaches to learning a PDDL model from observations have been proposed [Aineto et al., 2019, Stern and Juba, 2017, Juba et al., 2021, Cresswell et al., 2013, Wu et al., 2007]. A prominent example is FAMA [Aineto et al., 2019], which is a state-of-the-art algorithm for learning a PDDL model from observations. A major advantage of FAMA is that it is able to learn a PDDL model even if the given observations are incomplete, in the sense that only a subset of the actions and state variables are observed. A major disadvantage of FAMA and most PDDL model learning algorithms is that they do not provide any guarantee on the performance of the learned model. Plans generated with the learned model may not be executable or may fail to achieve their intended goals. SAM Learning [Stern and Juba, 2017, Juba et al., 2021, Juba and Stern, 2022, Mordoch et al., 2022] is a recently introduced family of learning algorithms that provide *safety* guarantees over the learned PDDL model: any plan generated with the model they return is guaranteed to be executable and achieve the intended goals. SAM Learning, however, is limited to learning from fully observed trajectories.

In this paper, we propose two algorithms for learning safe PDDL models in partially observed domains. The first algorithm, PI-SAM, extends SAM [Juba et al., 2021] to support partially observable domains by only applying the SAM learning rules when a literal is observed in the states immediately before and after an action is applied. PI-SAM is easy to implement, has a polynomial running time, and outputs a classical planning PDDL model that provides the desired safety guarantee. The second algorithm, EPI-SAM, utilizes observations that PI-SAM ignores to learn a stronger formulation. EPI-

NeurIPS 2023 Workshop on Generalization in Planning (GenPlan 2023)

SAM compiles its knowledge and uncertainty about the underlying action model into a *conformant planning problem*, whose solution is also a safe solution to the underlying classical planning problem. We analyze the running time of EPI-SAM and prove that the conformant planning problem created by EPI-SAM is the strongest safe problem formulation.

In terms of sample complexity, we show that in general it is not possible to guarantee efficient learning of a safe action model when the observations are partially observable. Nevertheless, we introduce a form of *bounded concealment assumption*, adapted from prior work on learning from partial observations [Michael, 2010], under which both PI-SAM and EPI-SAM are guaranteed probabilistic completeness with a tractable sample complexity. Experimentally, we evaluated the performance of both algorithms and compared them with FAMA [Aineto et al., 2019] on common domains from the International Planning Competition (IPC) [McDermott, 2000]. Our results show that PI-SAM and EPI-SAM often outperform FAMA in terms of the number of samples they require to learn effective action models, while still preserving our safety guarantee.

2 Background and Problem Definition

A classical planning *domain* is defined by a tuple $\langle F, A\rangle$ where F is a set of Boolean state variables, also known as fluents, and A is a set of actions. A *state* is a complete assignment of values to all fluents, i.e., $s : F \to \{\textit{true}, \textit{false}\}$. A *partial state* is an assignment of values to some (possibly all) of the fluents. For a fluent f and a partial state p, we denote by $p[f]$ the value assigned to f according to p. A partial state p is consistent with a partial state p' if for every fluent f either $p[f] = p'[f]$, f is not assigned in p, or f is not assigned in p'. A *literal* in this context is either a fluent $f \in F$ or its negation $\neg f$. For a literal $\ell = \neg f$, we denote by $p[\ell] = \textit{true}$, and $p[\ell] = \textit{false}$ the fact that $p[f] = \textit{false}$ and $p[f] = \textit{true}$, respectively. We say that a literal ℓ is in a partial state p, denoted $\ell \in p$, if $p[\ell] = \textit{true}$. Similarly, if $p[\ell] = \textit{false}$ we say that ℓ is not in s, denoted $\ell \notin s$. An action a is defined by a tuple $\langle \textit{name}(a), \textit{pre}(a), \textit{eff}(a)\rangle$ where $\textit{name}(a)$ is a unique identifier of the action and $\textit{pre}(a)$ and $\textit{eff}(a)$ are partial states that specify the preconditions and effects of a, respectively. An *action model* of a planning domain is its set of actions including their names, preconditions, and effects. An action a is *applicable* in a state s if $\textit{pre}(a)$ is consistent with s. *Applying* a in s results in a state $a(s)$ where for every fluent $f \in F$: (1) if f is assigned in $\textit{eff}(a)$ then $\textit{eff}(a)[f] = a(s)[f]$, (2) otherwise, $s[f] = a(s)[f]$. A sequence of actions $\pi = (a_1, \dots a_n)$ is applicable in a state s if a_1 is applicable in s and for every $i = 2, \dots, n$, a_i is applicable in $a_{i-1}(\cdots a_1(s)\cdots)$. The result of applying such a sequence of actions in a state s, denoted $\pi(s)$, is the state $a_n(\cdots a_1(s)\cdots)$.

A classical planning *problem* is defined by a tuple $\langle F, A, I, G\rangle$ where $\langle F, A\rangle$ is a domain, I is the initial state, and G is a partial state representing the goal we aim to achieve. A state s is called a goal state if G is consistent with s. A *solution* to a planning problem is a *plan*, which is a sequence of actions π such that π is applicable in I and $\pi(I)$ results in a goal state. Classical planning domains and problems are often described in a *lifted* manner, where fluents and actions are parameterized over objects. For ease of presentation, we describe our work in a grounded manner, but our work fully supports a lifted domain representation directly following Juba et al. [2021]. A *trajectory* is an alternating sequence of states and actions. For a trajectory $T = (s_0, a_1, \dots, a_n, s_n)$, let $T.s_i = s_i$ and $T.a_i = a_i$. The last state and action in T are denoted by $T.s_{-1}$ and $T.a_{-1}$, respectively, and $T.s$ and $T.a$ denote the sequence of states and actions in T, respectively. An action model A is *consistent* with a trajectory T if according to A the sequence of actions $T.a$ is applicable in $T.s_0$ and $T.s_i = T.a_i(\cdots T.a_1(T.s_0)\cdots)$ for every $i \in \{1, \dots, |T|\}$.

Conformant planning [Bonet, 2010] and *contingent planning* [Majercik and Littman, 2003, Hoffmann and Brafman, 2005, Albore et al., 2009, Brafman and Shani, 2012] are previously studied types of planning under uncertainty that are directly related to our work. In both, the effects of some actions may be non-deterministic, and the initial state I is replaced by a formula φ_I over the set of fluents that defines a set of possible initial states. In conformant planning, the agent is assumed to be unable to collect observations during execution. As such, conformant planning algorithms output a *linear plan*, which is a sequence of actions, as in classical planning. A (strong) solution to a conformant planning problem is a linear plan that is guaranteed to achieve the goal regardless of the inherent uncertainty due to the initial state and non-deterministic effects. In contingent planning, some actions' effects may include observing the values of some fluents, and the agent is assumed to be able to collect these observations and adapt its behavior accordingly.

Many algorithms have been proposed for learning action models from a given set of trajectories [Cresswell et al., 2013, Yang et al., 2007, Aineto et al., 2019, Juba et al., 2021]. Algorithms from the LOCM family [Cresswell and Gregory, 2011, Cresswell et al., 2013] learn action models by analyzing observed action sequences and constructing finite state machines that capture how actions change the states of objects in the world. The FAMA algorithm [Aineto et al., 2019] translates the problem of learning an action model to a planning problem, where every solution to this planning problem is an action model consistent with the available observations. FAMA works even if the observations given to it are partially observable. Algorithms from the *SAM* learning family [Stern and Juba, 2017, Juba et al., 2021, Juba and Stern, 2022, Mordoch et al., 2022] are different from other action model learning algorithms in that they guarantee that the action model they return is *safe*, in the sense that plans consistent with it are also consistent with the real, unknown action model. Most algorithms from this family have a tractable running time and reasonable sample complexity to ensure a probabilistic form of completeness, but rely on perfect observability of the given observations.

The *partially observed trajectories* we consider are created by *masking* some fluent values in a trajectory, essentially changing some states into partial states. A literal ℓ is said to be *masked* in a partial state p, denoted by $p[\ell] = ?$ if the corresponding fluent is not assigned in p. We say that an action model A is consistent with a partially observable trajectory T if it is consistent with at least one trajectory created by assigning values to all masked literals in T.

Definition 2.1. A safe model-free planning problem is defined by a tuple $\langle \Pi, \mathcal{T} \rangle$ where $\Pi = \langle F, A, I, G \rangle$ is a classical planning problem, and $\mathcal{T}$ is a set of partially observable trajectories created by executing plans that solve other problems in the same domain, and masking some literals in the states of the resulting trajectories. A safe model-free planning algorithm accepts the tuple $\langle F, I, G, \mathcal{T} \rangle$ and outputs a plan π that is a solution to the underlying planning problem Π.

The key challenge in solving such problems is that the problem-solver is not given any prior knowledge about the action model or the values of the masked literals. Nevertheless, the returned plan π must be *safe*, in the sense that π is a sequence of actions that are applicable in I according to the real action model A and ends up in a goal state. We make the following simplifying assumptions. Actions have deterministic effects. The preconditions and effects of actions are conjunctions of literals, as opposed to more complex logical statements, such as conditional effects. The form of partial observability defined above embodies the assumption that observations are noiseless: the value of a literal that is not masked is assumed to be correct. These assumptions are reasonable when planning in digital/virtual environments, such as video games, or environments that have been instrumented with reliable sensors, such as warehouses designed to be navigated by robots [Li et al., 2020].

3 Partial Information SAM Learning

Following prior work [Stern and Juba, 2017, Juba et al., 2021], we first learn an action model from the given trajectories, and then use a planner to solve the given planning problem. We aim to learn an action model that is *safe*.

Definition 3.1 (Safe Action Model)**.** An action model $\hat{A}$ is safe w.r.t an action model A if (1) for every action $a \in \hat{A}$ and state s if a is applicable in s according to $\hat{A}$ then it is also applicable in s according to A, and (2) for every goal G, if a plan achieves G according to $\hat{A}$ then it also achieves G according to A. Safety of/w.r.t is defined analogously for a fixed problem and its goal G.

The first learning algorithm we propose is called Partial Information SAM (PI-SAM). PI-SAM is based on the following observation.

Observation 3.2 (PI-SAM Rules)**.** *For any action triplet $\langle s, a, s' \rangle$ and literal ℓ*

Rule 1 [not a precondition]. If $(\ell \in s) \wedge (s[\ell] \neq ?)$ then $\neg\ell$ is not a precondition of a.
Rule 2 [an effect]. If $(\ell \notin s) \wedge (\ell \in s') \wedge (s[\ell] \neq ?) \wedge (s'[\ell] \neq ?)$ then ℓ is an effect of a.
Rule 3 [not an effect]. If $(\ell \notin s') \wedge (s'[\ell] \neq ?)$ then ℓ is not an effect of a.

PI-SAM applies rules 1 and 2 in almost the same way as SAM Learning. For every action a observed in some trajectory, we first assume that it has no effects and its preconditions consist of all possible literals. Then, for every transition $\langle s, a, s' \rangle$ and each literal ℓ observed in both pre- and post-states, i.e., $(s[\ell] \neq ?) \wedge (s'[\ell] \neq ?)$, we apply Rule 1 to remove preconditions and apply Rule 2 to add effects.

PI-SAM runs in $\mathcal{O}\Big(\sum_{a\in\mathcal{A}}|\mathcal{T}(a)|\cdot|\mathcal{F}|\Big)$, where $\mathcal{T}(a)$ is the set of transitions in $\mathcal{T}$ with action a.[1] PI-SAM also returns a safe action model, following the same reasoning given for the fully observable case [Stern and Juba, 2017]. Note that PI-SAM essentially uses the SAM learning rules, except that they are only applied for literals observed in both pre- and post-states. This may seem unintuitive, since Rule 1 does not require that a literal l is observed in a post-state to infer that it cannot be a precondition. To see why this modification is needed, consider running PI-SAM on a single trajectory with a single transition $\langle s, a, s'\rangle$ where $\ell \notin s$ and $s'[\ell] = ?$. Since the value of l is masked in s', we cannot apply Rule 3, and thus PI-SAM will assume l is not an effect of a. However, we cannot know if l is an effect of a or not. Thus, even though we can infer that l is not a precondition of a, returning an action model that allows a in such states may yield an unsafe action model.

Sample Complexity Analysis Learning a non-trivial safe action model without any restrictions on how the partially observable trajectories have been generated is impossible. To see this, consider the case where the value of some fluent f is always masked. Since we never observe the value of f, then for every action a we can never be certain if its preconditions include f, $\neg f$, or neither. Thus, we can never have a safe action model that allows action a to be applied. This example highlights that some assumption about how the partially observable trajectories were created is necessary in order to guarantee efficient learning of a safe action model. We propose such an assumption, based on the definition of a *masking function*.

Definition 3.3 (Masking function)**.** A trajectory masking function O is a function that maps a trajectory T to a partially observable trajectory $O(T)$ where (1) $T.a = O(T).a$, (2) $|T| = |O(T)|$, and (3) $\forall i : T.s_i$ is consistent with $O(T).s_i$.

An example of a masking function is *random masking*, which masks the value of each fluent with some fixed, independent probability. Without loss of generality. we assume the set of trajectories $\mathcal{T}$ were created by applying some masking function O on fully observable trajectories. Next, we introduce the following assumption about masking functions, adapted from Michael's theory of learning from partial information Michael [2010]:

Definition 3.4 (Bounded Concealment Assumption)**.** A masking function satisfies the *η-bounded concealment assumption* in an environment if for every literal that is not a precondition of an action, when that action is taken and the literal is false, then the corresponding fluent is observed in both the pre- and post-states with probability at least η.

As an example of a masking function that satisfies a bounded concealment assumption, consider a random masking function, where every literal is masked with a fixed independent probability α. Thus, each literal is observed in both the pre- and post-states with probability α^2 on each transition, i.e., such cases feature α^2-bounded concealment. Next, we analyze the relation between the number of trajectories given to PI-SAM and the ability of the action model it returns to solve new problems in the same domain, under the bounded concealment assumption. Let $\mathcal{P}_D$ be a probability distribution over solvable planning problems in a domain D. Let $\mathcal{T}_D$ be a probability distribution over pairs $\langle P, T\rangle$ given by drawing a problem P from $\mathcal{P}(D)$, using a sound and complete planner to generate a plan for P, and setting T to be the trajectory from following this plan.[2]

Theorem 3.5. *Under η-bounded concealment, given $m \geq \frac{1}{\epsilon\cdot\eta}(2\ln 3|A|\cdot|\mathcal{F}| + \ln\frac{1}{\delta})$ trajectories sampled from $\mathcal{T}_D$, PI-SAM returns a safe action model $M_{PI\text{-}SAM}$ such that with probability at least $1-\delta$, a problem drawn from $\mathcal{P}_D$ is not solvable with $M_{PI\text{-}SAM}$ with probability at most ϵ.*

Definition 3.6 (Adequate)**.** An action model M is *ϵ-adequate* if, with probability at most ϵ, a trajectory T sampled from $\mathcal{T}_D$ contains an action triplet $\langle s, a, s'\rangle$ where 1. s does not satisfy $pre_M(a)$ or 2. there is a literal in $s' \setminus s$ but not in $eff_M(a)$.

Lemma 3.7. *The action model returned by PI-SAM Learning given m trajectories (as specified in Theorem 3.5) is ϵ-adequate with probability at least $1-\delta$.*

A proof of Lemma 3.7 appears in the appendix. *Proof of Theorem 3.5.* When PI-SAM deletes a literal from $pre(a)$, it observed a triplet $\langle s, a, s'\rangle$ where l is false in s. Thus, whenever action a can be taken in some state under $M_{PI\text{-}SAM}$, it can also be taken in M^*. Conversely, since $M_{PI\text{-}SAM}$ is ϵ-adequate, with probability at least $1-\epsilon$ the sequence of actions appearing in the trajectory associated with

[1] Assuming one can access $\mathcal{T}(a)$ in $O(1)$.

[2] The planner need not be deterministic.

Algorithm 1 EPI-SAM: Learning Effects

Input : Partially observed trajectories $\mathcal{T}$
Output : $CNF_{\mathit{eff}}(\ell)$ for each literal l

foreach *literal* ℓ **do**
  $\mathit{CNF}_{\mathit{eff}}(\ell) \leftarrow \emptyset$
  foreach *action* a **do** Add to $\mathit{CNF}_{\mathit{eff}}(\ell)$: $\{\neg \mathit{IsEff}(\ell, a) \vee \neg \mathit{IsEff}(\neg\ell, a)\}$
  foreach *trajectory* $T \in \mathcal{T}$ **do**
    foreach *index* $i \in \{1, \ldots, |T|\}$ *where* $\ell \in T.s_i$ **do**
      $T' \leftarrow$ max. prefix of $T.s_i$ where ℓ is masked
      if $\ell \notin T'.s_0$ **then** Add to $\mathit{CNF}_{\mathit{eff}}(\ell)$: $\{\mathit{IsEff}(\ell, T'.a_1) \vee \cdots \vee \mathit{IsEff}(\ell, T'.a_{|T'|})\}$
      Add to $\mathit{CNF}_{\mathit{eff}}(\ell)$: $\{\neg \mathit{IsEff}(\neg\ell, T'.a_{|T'|})\}$
      foreach $j = 1$ *to* $|T'| - 1$ **do**
        Add to $\mathit{CNF}_{\mathit{eff}}(\ell)$: $\{\neg \mathit{IsEff}(\neg\ell, T'.a_j) \vee \mathit{IsEff}(\ell, T'.a_{j+1}) \vee \cdots \vee \mathit{IsEff}(\ell, T'.a_{|T'|})\}$
      end
    end
  end
end
return $\{\mathit{CNF}_{\mathit{eff}}(\ell)\}_\ell$

a draw from $\mathcal{T}_D$ is a valid plan in $M_{\textit{PI-SAM}}$. The first condition ensures that the preconditions of $M_{\textit{PI-SAM}}$ allow the action to be executed, and the second condition guarantees that $M_{\textit{PI-SAM}}$ obtains the same states on each transition. Thus, with probability $1 - \epsilon$, the goal is achievable under $M_{\textit{PI-SAM}}$ using the plan. □

4 Extended PI-SAM (EPI-SAM)

The PI-SAM algorithm is easy to implement and outputs an action model that can be used by any planner designed to solve classical planning problems. Yet, it only uses transitions where there are literals that are observed in both pre- and post-states. For example, consider an action a, a literal ℓ, and three transitions $\langle s_1, a, s'_1\rangle$, $\langle s_2, a, s'_2\rangle$, and $\langle s_3, a, s'_3\rangle$ where ℓ is not observed in any state except s_1, s'_2, and s'_3 in which its values are *false*, *false*, and *true*, respectively. Since ℓ was observed to be false in s_1, we can deduce it is not a precondition of a (Rule 1 in Observation 3.2). Since ℓ is never observed in both pre- and post-states of the same transition, the PI-SAM algorithm still does not remove ℓ from $\mathit{pre}(a)$. However, considering the value of ℓ in s'_2 and s'_3, we can deduce that neither ℓ nor $\neg\ell$ are effects of a (Rule 2 and 3 in Observation 3.2). Thus, it is possible to apply a in states without ℓ and maintain our safety property. Next, we propose the Extended PI-SAM (EPI-SAM) learning algorithm, which is able to make such inferences.

EPI-SAM relies on several key observations. The first observation is that learning of the effects of actions and learning their preconditions can be done separately, because we can never be certain that a literal is a precondition of an action. The second observation is that limiting the output of EPI-SAM to a classical planning action model limits the scope of safe model-free planning problems we can solve. For example, if we observe a trajectory $(s_0, a_1, s_1, a_2, s_2)$, where $s_0[\ell] = \textit{false}$, $s_2[\ell] = \textit{true}$, and ℓ is masked in s_1, we cannot discern which action — a_1 or a_2 — achieved ℓ, but we can learn that at least one of them has done so. While classical planning action models cannot capture this knowledge directly, such uncertainty can be compiled into a non-classical planning problem.

Based on these observations, EPI-SAM has the following parts: learning effects, learning preconditions, and compilation to non-classical planning. In the first part (learning effects), EPI-SAM creates a Conjunctive Normal Form (CNF) formula for each literal ℓ, denoted by $\mathit{CNF}_{\mathit{eff}}(\ell)$, which describes conditions for sequences of actions that achieve ℓ in the problems returned by EPI-SAM. The literals of this CNF are of the form $\mathit{IsEff}(\ell, a)$, representing whether literal ℓ is an effect of action a. In the second part (learning preconditions), EPI-SAM creates a set of literals $\mathit{pre}(a)$ for each action a that describes the preconditions of a in the returned problems. In the third part (compilation to non-classical planning), EPI-SAM creates a conformant planning problem using the output of the previous two parts. This conformant planning problem is constructed so that any (strong) solution to this problem is a safe solution to the actual planning problem. We describe these in detail next.

Algorithm 2 EPI-SAM: Learning Preconditions

Input : Partially observed trajectories $\mathcal{T}$
Output : Precondition *pre*(a) for each action a

foreach *action* a **do** *pre*$(a) \leftarrow$ all literals
foreach *action* a, *literal* ℓ **do**
  if $\exists \langle s, a, s' \rangle \in T \in \mathcal{T}$ *where* $\neg\ell \in s$ **then**
    Remove ℓ from *pre*(a)
    Continue to the next (a, ℓ) pair
  $\mathcal{T}_{a,\ell} \leftarrow$ **AssumePrecondition**$(a, \ell, \mathcal{T})$; $A_{irr} \leftarrow \emptyset$
  while $\exists a' \notin A_{irr}$ *where* ***Irrelevant***$(a', \ell, \mathcal{T}_{a,\ell})$ **do**
    foreach $\langle s, a', s' \rangle$ *in* $T \in \mathcal{T}_{a,\ell}$ **do**
      if $s[\ell]$ *and* $s'[\ell]$ *are inconsistent* **then**
        Remove ℓ from *pre*(a)
        Continue to the next (a, ℓ) pair
      else
        if $s[\ell] = ?$ **then** $s[\ell] \leftarrow s'[\ell]$
        Remove $\langle s, a', s' \rangle$ from T
      end
    end
  end
end
return $\{pre(a)\}_a$

Learning Effects To learn effects, EPI-SAM extends PI-SAM rules 2 and 3 (Observation 3.2) from rules over transitions to rules over *sub-trajectories*. A trajectory T' is a *sub-trajectory* of trajectory T, denoted $T' \subseteq T$, if it is a consecutive subsequence of T, i.e., there exists i and j where $i < j$ such that $T'.s_0 = T.s_i$ and for every $k \in \{1, \ldots, |T'|\}$ we have $T'.s_k = T.s_{i+k}$ and $T'.a_k = T.a_{i+k}$.

Observation 4.1 (EPI-SAM Rules)**.** *For any sub-trajectory T' of a trajectory in $\mathcal{T}$ that ends in a state where literal l is not masked, i.e., where $T'.s_{-1}[l] \neq ?$, then*

Rule 1 [an effect]. If $l \in T'.s_{-1}$ and $l \notin T'.s_0$ then $\exists a \in T'.a$ that has l as an effect.
Rule 2 [not an effect]. If $l \in T.s_{-1}$ then $\neg l$ is not an effect of $T'.a_{-1}$
Rule 3 [not deleted].If $l \in T'.s_{-1}$ and $\neg l$ is an effect of an action $T'.a_i$ then $\exists i' > i$ that has l as an effect.

Algorithm 1 lists the pseudo-code for effects learning in EPI-SAM, which builds on the EPI-SAM rules in Observation 4.1. Initially, $CNF_{\mathit{eff}}(\ell)$ contains a single clause for every action a that ensures the effects of a are mutually exclusive (line 2). Then, we implement the EPI-SAM rules by going over every trajectory T and every state $T.s_i$ in which ℓ is not masked. For each such pair of trajectory and state, we extract the longest sub-trajectory $T' \subseteq T$ that ends in $T.s_i$ and where ℓ is masked in all other states in T' (line 5). If a literal ℓ was false at the first state of T', then we add to $CNF_{\mathit{eff}}(\ell)$ a clause to ensure that ℓ is an effect of some action a_i (EPI-SAM Rule 1). Then, we add a clause to ensure that $\neg\ell$ is not an effect of the last action in T' (EPI-SAM Rule 2). Finally, we add a clause to ensure that if $\neg\ell$ was an effect of any action $a \in T'.a$ then some action in T' after that action must have had ℓ as an effect (EPI-SAM Rule 3).

Learning Preconditions EPI-SAM starts by assuming for every action a that it has all literals as preconditions. Then, it removes a literal l from the set of preconditions of an action a if and only if assuming l is a precondition of *pre*(a) is inconsistent with $\mathcal{T}$. There are two possible ways in which the assumption that l is a precondition of a can be inconsistent with the observations: (1) there is a transition $\langle s, a, s' \rangle$ in $\mathcal{T}$ where $s[l] =$ *false*, and (2) no set of action effects is consistent with $\mathcal{T}$ when we additionally set $s[l] =$ *true* for every transition $\langle s, a, s' \rangle$ in $\mathcal{T}$. The former corresponds to PI-SAM Rule 1, which can be easily verified in linear time. The latter can be checked by setting $s[l] =$ *true* in the relevant transitions, running EPI-SAM's effect-learning part (Algorithm 1) on the resulting set of trajectories, and checking if the resulting CNF is satisfiable. This check can be done by calling any SAT solver. Fortunately, it is also possible to perform this satisfiability check in polynomial time. This is because assumptions about which action achieves literal l are independent of any assumption about which actions achieve any other literal except $\neg l$.[3]

[3]This independence fails when conditional effects are allowed.

Algorithm 2 lists the pseudo-code of EPI-SAM's precondition learning part. Like PI-SAM, EPI-SAM initially assumes that the preconditions of every action include all literals. Then, EPI-SAM iterates over every pair of action a and literal ℓ to check if ℓ can be removed from the set of preconditions assumed for a. The first way EPI-SAM attempts to remove ℓ from $pre(a)$ is by checking if it violates PI-SAM Rule 1 (lines 15-16). The second way is by using a proof-by-contradiction approach, checking if assuming ℓ is a precondition of a leads to a contradiction with the observations and every possible assumption about actions' effects. EPI-SAM performs this check by performing the following steps. First, it creates a copy of the set of trajectories $\mathcal{T}$ where ℓ is set to be true in every state where a is applied (the **AssumePrecondition** call in line 17). This set of modified trajectories is denoted by $\mathcal{T}_{a,\ell}$ in Algorithm 2. Then, EPI-SAM iteratively searches for actions that are *irrelevant* for the value of ℓ. An action a is said to be irrelevant for the value of ℓ if we can infer that neither ℓ nor $\neg\ell$ are effects of a. We do this by invoking PI-SAM Rule 2 for both ℓ and $\neg\ell$. That is, action a' is identified as irrelevant to ℓ if there are two transitions $\langle s_1, a', s'_1\rangle$ and $\langle s_2, a', s'_2\rangle$ where ℓ is not masked in their post-states and it has different values, i.e., $(s'_1[\ell] \neq ?) \wedge (s'_2[\ell] \neq ?) \wedge (s'1[\ell] \neq s'_2[\ell])$. A contradiction is identified if there exists a transition $\langle s, a', s'\rangle$ where a' is an irrelevant action but the value of ℓ in s and in s' is inconsistent, i.e., unmasked and different (line 19). If a' is irrelevant but the values of s and s' are consistent, then we propagate the value of s' to s and remove the transition $\langle s, a', s'\rangle$ from $\mathcal{T}_{a,\ell}$ (lines 22-23). [4]

Compilation to Non-Classical Planning Next, EPI-SAM creates a *conformant planning* problem Π_{SAM} based on the outputs of the previous EPI-SAM parts, $\{CNF_{eff}(\ell)\}_\ell$ and $\{pre_a\}_a$, and the available knowledge of the underlying planning problem Π. A conformant planning problem is defined by a tuple $\langle F, O, A, I, G\rangle$ where F, A, I, and G are the set of fluents, actions, initial state, and goals, as in a classical planning problem, except that A may include non-deterministic and conditional effects, and I is a set of possible initial states defined by a formula over F. O is the subset of fluents in F that are observable. The set of fluents in Π_{SAM} includes all fluents in Π and an additional fluent $f_{IsEff(a,\ell)}$ for every action a and literal ℓ. All fluents from Π are observable in Π_{SAM} and all others are not. The initial state formula in Π_{SAM} sets the values of all observable fluents according to their initial values in Π. In addition, it includes all the clauses in the CNFs returned by EPI-SAM ($\{CNF_{eff}(\ell)\}_\ell$), replacing every literal $IsEff(a, \ell)$ with the corresponding fluent $f_{IsEff(a,\ell)}$. The action model of Π_{SAM} includes all actions observed in $\mathcal{T}$. For each action a, we set its preconditions to the set of preconditions learned for it by EPI-SAM's learning preconditions part, $pre(a)$. All the effects of a are *conditional effects*. A conditional effect of an action is an effect (i.e., a partial state) that is only applied if a specified condition holds. For each action a and literal ℓ, we add a conditional effect such that if $f_{IsEff(a,\ell)}$ is true then ℓ is an effect of a. Note that conditional effects are supported by many classical and conformant planners [Bonet, 2010, Grastien et al., 2017]. If the agent executing the plan can observe the values of fluents during execution and react, then the above compilation can be used almost as-is to construct a contingent planning problem instead of a conformant planning problem. The output of a contingent planning algorithm is a plan tree, branching over the observed values during execution, which can be more efficient than the linear plan returned for the respective conformant planning problem.

Theoretical Properties Next, we analyze EPI-SAM theoretically, showing that it is safe, runs in polynomial time, and it is the strongest algorithm for solving safe model-free planning problems, in the sense that any algorithm able to solve a problem that cannot be solved by EPI-SAM cannot also be safe. Throughout this analysis, we denote by A^* the action model of the underlying problem, and denote by $pre_A(a)$ and $eff_A(a)$ the set of preconditions and effects, respectively, of an action a according to an action model A. Observe that every classical action model A corresponds to an assignment σ_A to the formula $\Phi_{eff} = \bigwedge_\ell CNF_{eff}(\ell)$, by setting $IsEff(\ell, a)$ to true if ℓ is an effect of a for each literal ℓ and action a. Similarly, every satisfying assignment of Φ_{eff} describes the effects of a classical action model. Lemmas and theorems given below either without a proof or with a proof sketch are formally proven in the appendix.

Lemma 4.2. *If a classical action model A is consistent with $\mathcal{T}$ then σ_A is a satisfying assignment of Φ_{eff}. Conversely, every satisfying assignment σ to Φ_{eff} describes the effects of at least one classical action model that is consistent with $\mathcal{T}$.*

Lemma 4.3. *For every action a in A_{SAM} and literal ℓ, it holds that $\ell \in pre_{A_{SAM}}(a)$ if and only if there exists an action model A consistent with $\mathcal{T}$ where $\ell \in pre_A(a)$.*

[4] If $\exists \langle s', a'', s''\rangle \in T$, then removing $\langle s, a', s'\rangle$ implicitly adds the transition $\langle s, a'', s''\rangle$.

Domain	Algorithm	$\eta = 0.3$						$\eta = 0.1$					
		$\lvert\mathcal{T}\rvert$	P(*pre*)	R(*pre*)	P(*eff*)	R(*eff*)	T(sec)	$\lvert\mathcal{T}\rvert$	P(*pre*)	R(*pre*)	P(*eff*)	R(*eff*)	T(sec)
Blocks	FAMA	3	0.90	0.90	1.00	0.89	13	6	0.90	**0.85**	0.90	0.85	60
(5,4,2,2)	PI-SAM	3	**1.00**	0.90	1.00	**0.95**	9	6	**1.00**	0.83	**1.00**	0.85	43
	EPI-SAM*	3	**1.00**	**0.92**	1.00	0.95	-	6	**1.00**	**0.85**	**1.00**	**0.88**	-
Depot	FAMA	5	0.80	0.85	0.90	1.00	17	8	0.80	0.80	0.90	1.00	60
(6,5,4,2)	PI-SAM	5	**1.00**	0.85	**1.00**	1.00	12	8	**1.00**	0.82	**1.00**	1.00	53
	EPI-SAM*	5	**1.00**	0.85	**1.00**	1.00	-	8	**1.00**	**0.83**	**1.00**	1.00	-
Ferry	FAMA	3	0.85	1.00	1.00	1.00	9	6	0.80	**1.00**	0.85	**1.00**	35
(5,3,2,2)	PI-SAM	3	**1.00**	1.00	1.00	1.00	5	6	**1.00**	0.94	**1.00**	0.90	27
	EPI-SAM*	3	**1.00**	1.00	1.00	1.00	-	6	**1.00**	0.95	**1.00**	0.90	-
Floortile	FAMA	5	0.84	0.80	0.79	0.80	18	9	0.87	0.82	0.80	0.83	60
(10,7,2,4)	PI-SAM	5	**1.00**	0.87	**1.00**	0.87	15	9	**1.00**	0.85	**1.00**	0.85	50
	EPI-SAM*	5	**1.00**	**0.89**	**1.00**	**0.90**	-	9	**1.00**	**0.87**	**1.00**	**0.87**	-
Gripper	FAMA	5	1.00	1.00	1.00	1.00	8	10	1.00	1.00	1.00	1.00	30
(4,3,2,3)	PI-SAM	5	1.00	1.00	1.00	1.00	5	10	1.00	1.00	1.00	1.00	24
	EPI-SAM*	5	1.00	1.00	1.00	1.00	-	10	1.00	1.00	1.00	1.00	-
Hanoi	FAMA	1	0.85	1.00	1.00	1.00	1	1	0.81	1.00	1.00	1.00	60
(3,1,2,3)	PI-SAM	1	**1.00**	1.00	1.00	1.00	1	1	**1.00**	1.00	1.00	1.00	15
	EPI-SAM*	1	**1.00**	1.00	1.00	1.00	-	1	**1.00**	1.00	1.00	1.00	-
Npuzzle	FAMA	1	1.00	1.00	1.00	1.00	1	1	0.83	1.00	1.00	1.00	23
(3,1,2,3)	PI-SAM	1	1.00	1.00	1.00	1.00	1	1	**1.00**	1.00	1.00	1.00	17
	EPI-SAM*	1	1.00	1.00	1.00	1.00	-	1	**1.00**	1.00	1.00	1.00	-
Parking	FAMA	6	0.85	0.85	1.00	1.00	13	8	0.83	**0.85**	0.90	1.00	60
(5,4,2,3)	PI-SAM	6	**1.00**	**0.88**	1.00	1.00	8	8	**1.00**	0.83	**1.00**	1.00	49
	EPI-SAM*	6	**1.00**	**0.88**	1.00	1.00	-	8	**1.00**	**0.85**	**1.00**	1.00	-
Sokoban	FAMA	2	1.00	1.00	1.00	1.00	8	5	1.00	1.00	1.00	1.00	40
(4,2,3,5)	PI-SAM	2	1.00	1.00	1.00	1.00	6	5	1.00	1.00	1.00	1.00	33
	EPI-SAM*	2	1.00	1.00	1.00	1.00	-	5	1.00	1.00	1.00	1.00	-
Transport	FAMA	5	0.77	0.80	0.80	0.90	14	9	0.80	0.80	0.84	0.90	60
(5,3,2,5)	PI-SAM	5	**1.00**	0.83	**1.00**	0.90	9	9	**1.00**	0.80	**1.00**	0.90	48
	EPI-SAM*	5	**1.00**	**0.85**	**1.00**	**0.92**	-	9	**1.00**	**0.83**	**1.00**	**0.92**	-

Table 1: Empirical precision and recall results under random masking with $\eta = 0.1$ and $\eta = 0.3$.

Theorem 4.4. *EPI-SAM returns a safe plan.*

Theorem 4.5 (Strength)**.** *The problem Π_{SAM} returned by EPI-SAM is the strongest safe problem formulation, in the sense that if an action model A is not safe with respect to Π_{SAM}, then there exists an action model A' consistent with $\mathcal{T}$ such that A is not safe with respect to A'.*

Theorem 4.6. *Given a set of trajectories $\mathcal{T}$, EPI-SAM runs in time $\mathcal{O}\Big(|A| \cdot |\mathcal{F}| \cdot \sum_{a \in \mathcal{A}} |\mathcal{T}(a)|\Big)$.*

5 Experiments

We evaluate our algorithms' performance experimentally on the IPC [McDermott, 2000] domains listed in Table 1. The tuple listed under each domain details the number of lifted fluents, lifted actions, maximal arity of fluents, and maximal arity of actions in that domain. For each domain, we generated problems using the generators provided by the IPC learning tracks and solved them using the true action model and an off-the-shelf planner. In the resulting trajectories, we masked some states using *random masking* with masking probability $\eta = 0.1$ and $\eta = 0.3$.

Metrics A common approach to comparing action models is by computing the precision and recall of the learned action model with respect to which literals appear in the real action model. However, this syntactic measure has three limitations. First, it requires the evaluated action models to use the same fluents and action names. Second, it gives the same "penalty" for every mistake in the learned model. Third, domains may have distinct but semantically-equivalent action models. For example, in Npuzzle, we could have a precondition that the tile we are sliding into the empty position is not an empty position. This precondition is not necessary, as there is only ever one empty position in any puzzle. Thus, either formulation of the domain is adequate for planning purposes, but a syntactic measure of correctness will penalize one of the two formulations. Instead, we introduce and use *empirically-based precision and recall* measures, which are based on comparing the number of

	$\lvert T\rvert = 3$		$\lvert T\rvert = 5$		$\lvert T\rvert = 7$	
	PI-SAM	FAMA	PI-SAM	FAMA	PI-SAM	FAMA
P(pre)	**1.00**	0.90	**1.00**	0.87	**1.00**	0.90
R(pre)	0.90	0.90	**0.92**	0.88	**0.93**	0.90
P(eff)	1.00	1.00	**1.00**	0.95	1.00	1.00
R(eff)	**0.95**	0.89	**0.96**	0.87	**0.96**	0.90

Alg.	SAM ($\eta = 1.0$)	PI-SAM ($\eta = 0.3$)	PI-SAM ($\eta = 0.1$)
Hanoi	1	10	95
Npuzzle	1	9	92
Ferry	4	42	355
Gripper	5	51	476
Sokoban	6	55	563

Table 2: (Left) Results on Blocks with $\eta = 0.3$. (Right) # of transitions needed to learn the preconditions

transitions that are valid or invalid according to the learned action model ($\hat{A}$) and the true action model (A). The empirical precision and recall measures are defined according to the number of true/false positives/negatives (TP,FP,TN,FN) but compute TP, FP, TN, and FN differently. For preconditions, TP is the number of transitions that are valid according to both $\hat{A}$ and A, FP is the number of transitions that are valid according to $\hat{A}$ but not A, TN is the number of transitions that are invalid according to $\hat{A}$ and A, and FN is the number of transitions that are valid according to A and but not $\hat{A}$. TP, FP, TN, and FN for effects are computed similarly.

Results and Discussion We performed experiments using PI-SAM and EPI-SAM*, a simplified (unsafe) version of EPI-SAM. Recall that EPI-SAM does not return a classical action model, and the conformant planning formulation it produces involves explicitly reasoning about the various possible states that could occur in trajectories using the uncertain action model. As such, it does not make sense to apply state-wise measures of precision and recall directly to EPI-SAM. EPI-SAM* uses unit propagation to determine the effects of every action in the CNF returned by Algorithm 1, by checking if assuming literal l is an effect of action a if the CNF formula extended by $\neg \mathit{IsEff}(a, \ell)$ is satisfiable. EPI-SAM* outputs a classical action model instead of a conformant plan. Nevertheless, observe that the inferences obtained by unit propagation are sound and are a subset of those obtainable in EPI-SAM's formulation. Thus, since EPI-SAM is safe, the precision and recall for EPI-SAM* provide a lower bound on the performance of EPI-SAM.

As a baseline, we compared our algorithms to FAMA [Aineto et al., 2019], a modern algorithm for learning action models under partial observability. We ran those three algorithms on our benchmark domains. For each domain, we computed the *empirical precision* (P) and *recall* (R) separately for the preconditions (*pre*) and effects (*eff*). Table 1 lists the results of our experiments, averaged over three independent runs. Columns "$P(\mathit{pre})$", "$R(\mathit{pre})$", "$P(\mathit{eff})$", and "$R(\mathit{eff})$" show the empirical precision and recall for preconditions and effects for every evaluated algorithm. $|\mathcal{T}|$ is determined as the point that FAMA started decreasing performance (i.e. precision-recall) or reaching the time limit. We limited the running time of each algorithm to 60 seconds. Column "T" is the runtime of each algorithm in seconds. Since EPI-SAM* is unsafe, we do not report its runtime. Since PI-SAM and EPI-SAM*, by definition, never remove a literal that is an actual precondition from the preconditions or add a literal that is not an actual effect, their empirical precision is perfect for both preconditions and effects, as opposed to FAMA, which does not always achieve this. PI-SAM tends to have a higher empirical recall under lower masking probability (high η), while FAMA tends to obtain higher recall under higher masking probability (low η). EPI-SAM* generally outperforms both. Note that FAMA's performance may decrease as more input is given, while PI-SAM cannot. To demonstrate this, we picked a domain (Blocks) and recorded their performance as given an increasing number of trajectories as input. The results are shown in Table 2(left). We also compared the number of transitions required to correctly learn the preconditions (i.e., $P(\mathit{pre})$ and $R(\mathit{pre}) = 1.0$) when using PI-SAM with $\eta \in \{0.1, 0.3\}$ and when having full observability and using SAM. The results are shown in Table 2 (right). As expected, the number of transitions required scales inversely with the random masking probability η^2, which verifies the tightness of the bound in Theorem 3.5. The source code of the experiments will be made available upon acceptance.

6 Conclusion and Future Work

We proposed two algorithms for learning safe action models in domains with partial observability. The first algorithm, PI-SAM, extends the SAM learning algorithm [Juba et al., 2021] to partially observable domains and outputs classical planning action models. The second algorithm, EPI-SAM, provides the outputs in the form of conformant planning problems, but can work on general observations. In practice, we can choose either PI-SAM or EPI-SAM, depending on the specific observation sets (e.g., whether they satisfy the bounded concealment assumption or not). For future work, we aim to extend safe action model learning to more complicated domains, such as domains with stochastic effects, numeric state variables, etc.

References

Diego Aineto, Sergio Celorrio, and Eva Onaindia. Learning action models with minimal observability. *Artificial Intelligence*, 275:104–137, 05 2019.

Alexandre Albore, Héctor Palacios, and Hector Geffner. A translation-based approach to contingent planning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

Blai Bonet. Conformant plans and beyond: Principles and complexity. *Artificial Intelligence*, 174(3): 245–269, 2010.

Ronen Brafman and Guy Shani. A multi-path compilation approach to contingent planning. In *AAAI Conference on Artificial Intelligence*, 2012.

Stephen Cresswell and Peter Gregory. Generalised domain model acquisition from action traces. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 42–49, 2011.

Stephen N Cresswell, Thomas L McCluskey, and Margaret M West. Acquiring planning domain models using locm. *The Knowledge Engineering Review*, 28(2):195–213, 2013.

Alban Grastien, Enrico Scala, and Fondazione Bruno Kessler. Intelligent belief state sampling for conformant planning. In *IJCAI*, pages 4317–4323, 2017.

Jörg Hoffmann and Ronen Brafman. Contingent planning via heuristic forward search with implicit belief states. In *ICAPS*, volume 2005, 2005.

Brendan Juba and Roni Stern. Learning probably approximately complete and safe action models for stochastic worlds. In *AAAI Conference on Artificial Intelligence*, 2022.

Brendan Juba, Hai S. Le, and Roni Stern. Safe learning of lifted action models. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 379–389, 2021.

Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W Durham, TK Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1898–1900, 2020.

Stephen M Majercik and Michael L Littman. Contingent planning under uncertainty via stochastic satisfiability. *Artificial Intelligence*, 147(1):119–162, 2003.

Drew McDermott. The 1998 AI planning systems competition. *AI Magazine*, 21(2):13, June 2000.

Loizos Michael. Partial observability and learnability. *Artificial Intelligence*, 174(11):639–669, 2010.

Argaman Mordoch, Daniel Portnoy, Roni Stern, and Brendan Juba. Collaborative multi-agent planning with black-box agents by learning action models. In *Learning with Strategic Agents (LSA) Workshop in the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2022.

Roni Stern and Brendan Juba. Efficient, safe, and probably approximately complete learning of action models. In *the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4405–4411, 2017.

Kangheng Wu, Qiang Yang, and Yunfei Jiang. ARMS: An automatic knowledge engineering tool for learning action models for ai planning. *The Knowledge Engineering Review*, 22(2):135–152, 2007.

Qiang Yang, Kangheng Wu, and Yunfei Jiang. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence*, 171(2-3):107–143, 2007.

Algorithm 3 Partial Information SAM Learning Algorithm (PI-SAM)

Input : Partially Observed Trajectories $\mathcal{T}$
Output : (*pre*, *eff*) for a safe action model

foreach *action* a **do**
  $\mathit{eff}(a) \leftarrow \emptyset$
  $\mathit{pre}(a) \leftarrow$ all parameter-bound literals
  foreach *transition* $\langle s, a, s' \rangle$ **do**
    foreach *literal* $l \in \mathit{pre}(a)$ **do**
      if $\neg l$ *is unmasked and* $\neg l \in s$ **then**
        Remove l from $\mathit{pre}(a)$
    end
    foreach *literal* $l \in s' \setminus s$ *that is unmasked in* s *and in* s' **do**
      Add l to $\mathit{eff}(a)$
    end
  end
end
Return $\langle \mathit{pre}, \mathit{eff} \rangle$

7 Supplementary Material

7.1 PI-SAM Sample Complexity Analysis

Definition 7.1 (Bounded Concealment Assumption)**.** A masking function satisfies the η-*bounded concealment assumption* in an environment if for every literal that is not a precondition of an action, when that action is taken and the literal is false, then the corresponding fluent is observed in both the pre- and post-states with probability at least η.

Theorem 7.2. *Under η-bounded concealment, given*

$$m \geq \frac{1}{\epsilon \cdot \eta}(2 \ln 3|A|.|\mathcal{F}| + \ln \frac{1}{\delta})$$

trajectories sampled from $\mathcal{T}_D$, with probability at least $1 - \delta$, PI-SAM Learning Algorithm (Algorithm 3) returns a safe action model $M_{\textit{PI-SAM}}$ such that a problem drawn from $\mathcal{P}_D$ is not solvable with $M_{\textit{PI-SAM}}$ with probability at most ϵ.

To prove the theorem, we use the following definition of an *adequate* action model:

Definition 7.3 (Adequate)**.** An action model M is ϵ-*adequate* if, with probability at most ϵ, a trajectory T sampled from $\mathcal{T}_D$ contains an action triplet $\langle s, a, s' \rangle$ where either

1. s does not satisfy $\mathit{pre}_M(a)$ or
2. there is a literal in $s' \setminus s$ but not in $\mathit{eff}_M(a)$.

Proof of Theorem 7.2. We first argue that PI-SAM returns an ϵ-adequate action model with probability $1 - \delta$: indeed, consider any action model $\tilde{M}$ that is *not* ϵ-adequate: then either

1. with probability at least ϵ, trajectories sampled from $\mathcal{T}_D$ contain a triplet $\langle s, a, s' \rangle$ for which s does not satisfy $\mathit{pre}_{\tilde{M}}(a)$, or
2. with probability at least ϵ, trajectories sampled from $\mathcal{T}_D$ contain a triplet $\langle s, a, s' \rangle$ for which there is a literal in $s' \setminus s$ but not in $\mathit{eff}_{\tilde{M}}(a)$.

In the first case, note that since $\langle s, a, s' \rangle$ is a valid transition under the true action model M^*, the literal for which $\mathit{pre}_{\tilde{M}}(a)$ is violated cannot be in $\mathit{pre}_{M^*}(a)$. Therefore, by η-bounded concealment, the violated precondition literal in $\mathit{pre}_{\tilde{M}}(a)$ is observed with probability at least η when such a transition occurs; thus, with probability at least $\eta \cdot \epsilon$ overall, the literal is observed and deleted from $\mathit{pre}_{M_{\textit{PI-SAM}}}(a)$. Since PI-SAM never adds precondition literals back, this ensures that $M_{\textit{PI-SAM}} \neq \tilde{M}$.

Similarly, in the second case, if $l \in s' \setminus s$, $l \in \mathit{eff}_{M^*}(a)$. Thus, η-bounded concealment ensures that l is observed in both s and s' with probability at least η when such a transition occurs. So, overall with probability $\eta \cdot \epsilon$, the trajectory contains a triple $\langle s, a, s' \rangle$ where l is observed and $l \in s' \setminus s$. When this happens, l is added to $\mathit{eff}_{\textit{PI-SAM}}(a)$, and we again get $M_{\textit{PI-SAM}} \neq \tilde{M}$ since PI-SAM never removes literals from the effects.

Algorithm 4 EPI-SAM: Learning Effects

Input : Partially observed trajectories $\mathcal{T}$
Output : $CNF_{\mathit{eff}}(\ell)$ for each literal l

foreach *literal* ℓ **do**
  $\mathit{CNF}_{\mathit{eff}}(\ell) \leftarrow \emptyset$
  foreach *action* a **do**
    Add to $\mathit{CNF}_{\mathit{eff}}(\ell)$: $\{\neg \mathit{IsEff}(\ell, a) \vee \neg \mathit{IsEff}(\neg\ell, a)\}$
  end
  foreach *trajectory* $T \in \mathcal{T}$ **do**
    foreach *index* $i \in \{1, \ldots, |T|\}$ *where* $\ell \in T.s_i$ **do**
      $T' \leftarrow$ max. prefix of $T.s_i$ where ℓ is masked
      if $\ell \notin T'.s_0$ **then**
        Add to $\mathit{CNF}_{\mathit{eff}}(\ell)$: $\{\mathit{IsEff}(\ell, T'.a_1) \vee \cdots \vee \mathit{IsEff}(\ell, T'.a_{|T'|})\}$
      Add to $\mathit{CNF}_{\mathit{eff}}(\ell)$: $\{\neg \mathit{IsEff}(\neg\ell, T'.a_{|T'|})\}$
      foreach $j = 1$ *to* $|T'| - 1$ **do**
        Add to $\mathit{CNF}_{\mathit{eff}}(\ell)$: $\{\neg \mathit{IsEff}(\neg\ell, T'.a_j) \vee \mathit{IsEff}(\ell, T'.a_{j+1}) \vee \cdots \vee \mathit{IsEff}(\ell, T'.a_{|T'|})\}$
      end
    end
  end
end
return $\{\mathit{CNF}_{\mathit{eff}}(\ell)\}_\ell$

Thus, in either case, the probability of obtaining a trajectory that ensures that $\tilde{M}$ is not output is at least $\eta \cdot \epsilon$ on each example. Since the examples are drawn independently, the probability that we do not obtain such an example after m draws is at most $(1 - \eta \cdot \epsilon)^m \leq e^{-\eta \cdot \epsilon \cdot m}$. For m as stated in the claim, this is at most

$$e^{-2 \ln 3 |A|.|\mathcal{F}| - \ln \frac{1}{\delta}} = \frac{\delta}{3^{2|A|.|\mathcal{F}|}}.$$

Note that there are only $3^{2|A|.|\mathcal{F}|}$ possible consistent sets of fluents for the action a (for each fluent, each precondition or effect will either contain that fluent, or its negation, or neither of them), and hence $3^{2|A|.|\mathcal{F}|}$ possible action models, given by effects and preconditions for each action. There are, in particular, at most this many action models that are not ϵ-adequate. So, by a union bound over all such action models, the probability that PI-SAM returns any of them is at most δ.

We note that PI-SAM only deletes a literal from $\mathit{pre}(a)$ when a triplet $\langle s, a, s' \rangle$ is observed where l is false in s, and hence cannot be a precondition of a in M^*. Thus, whenever action a can be taken in some state under $M_{\mathit{PI\text{-}SAM}}$, it can also be taken in M^*. Conversely, when $M_{\mathit{PI\text{-}SAM}}$ is ϵ-adequate, we have that with probability at least $1 - \epsilon$ the sequence of actions appearing in the trajectory associated with a draw from $\mathcal{T}_D$ is a valid plan in $M_{\mathit{PI\text{-}SAM}}$: the first condition ensures that the preconditions of $M_{\mathit{PI\text{-}SAM}}$ allow the given action to be executed, and the second condition guarantees that $M_{\mathit{PI\text{-}SAM}}$ obtains the same states on each transition. Thus, with probability $1 - \epsilon$, the goal is achievable under $M_{\mathit{PI\text{-}SAM}}$ using the plan. □

7.2 EPI-SAM Theoretical Properties with Proofs

Observation 7.4 (EPI-SAM Rules)**.** *For any sub-trajectory T' of a trajectory in $\mathcal{T}$ that ends in a state where literal l is not masked, i.e., where $T'.s_{-1}[l] \neq ?$, then*

- *Rule 1 [an effect]. If $l \in T'.s_{-1}$ and $l \notin T'.s_0$ then $\exists a \in T'.a$ that has l as an effect.*
- *Rule 2 [not an effect]. If $l \in T.s_{-1}$ then $\neg l$ is not an effect of $T'.a_{-1}$*
- *Rule 3 [must not delete]. If $l \in T'.s_{-1}$ and $\neg l$ is an effect of some action $T'.a_i$ then $\exists i' > i$ that has l as an effect.*

Next, we show that EPI-SAM is safe, runs in polynomial time, and it is the strongest algorithm for solving safe model-free planning problems, in the sense that any algorithm able to solve a problem that cannot be solved by EPI-SAM cannot also be safe. Throughout this analysis, we denote by A^* the action model of the underlying problem, and denote by $\mathit{pre}_A(a)$ and $\mathit{eff}_A(a)$ the set of preconditions

Algorithm 5 EPI-SAM: Learning Preconditions

Input : Partially observed trajectories $\mathcal{T}$
Output : Precondition *pre*(a) for each action a

16 **foreach** *action* a **do** *pre*$(a) \leftarrow$ all literals
17 **foreach** *action* a, *literal* ℓ **do**
18 **if** $\exists \langle s, a, s' \rangle \in T \in \mathcal{T}$ *where* $\neg\ell \in s$ **then**
19 Remove ℓ from *pre*(a)
  Continue to the next (a, ℓ) pair
20 $\mathcal{T}_{a,\ell} \leftarrow$ **AssumePrecondition**$(a, \ell, \mathcal{T})$
  $A_{irr} \leftarrow \emptyset$
  while $\exists a' \notin A_{irr}$ *where* ***Irrelevant***$(a', \ell, \mathcal{T}_{a,\ell})$ **do**
21 **foreach** $\langle s, a', s' \rangle$ *in* $T \in \mathcal{T}_{a,\ell}$ **do**
22 **if** $s[\ell]$ *and* $s'[\ell]$ *are inconsistent* **then**
23 Remove ℓ from *pre*(a)
  Continue to the next (a, ℓ) pair
24 **else**
25 **if** $s[\ell] = ?$ **then** $s[\ell] \leftarrow s'[\ell]$
26 Remove $\langle s, a', s' \rangle$ from T
27 **end**
28 **end**
29 **end**
30 **end**
31 **return** $\{pre(a)\}_a$

and effects, respectively, of an action a according to an action model A. Observe that every classical action model A corresponds to an assignment σ_A to the formula $\Phi_{\mathit{eff}} = \bigwedge_\ell \mathit{CNF}_{\mathit{eff}}(\ell)$, by setting $\mathit{IsEff}(\ell, a)$ to true if ℓ is an effect of a for each literal ℓ and action a. Similarly, every satisfying assignment of Φ_{eff} describes the effects of a classical action model.

Lemma 7.5. *If a classical action model A is consistent with $\mathcal{T}$ then σ_A is a satisfying assignment of Φ_{eff}. Conversely, every satisfying assignment σ to Φ_{eff} describes the effects of at least one classical action model that is consistent with $\mathcal{T}$.*

Sketch of proof. Consider the clausal encoding of the STRIPS axioms, instantiated at each step of each trajectory in $\mathcal{T}$. This CNF, denoted $\mathit{CNF}_{\mathcal{T}}$ is defined over variables of the form $\mathit{IsEff}(l, a)$, $\mathit{IsPre}(l, a)$, and $\mathit{State}(l, i, T)$, representing that l is a precondition of a, l is an effect of a, and $l = \mathit{true}$ in the i^{th} state of trajectory T, respectively. This CNF includes the following clauses for every transition $\langle s_{i-1}, a_i, s_i \rangle$ in every trajectory $T \in \mathcal{T}$:

- (C1) $\neg\mathit{IsPre}(l, a_i) \vee \mathit{State}(l, i-1, T)$
- (C2) $\neg\mathit{IsEff}(l, a_i) \vee \mathit{State}(l, i, T)$
- (C3) $\mathit{IsEff}(l, a_i) \vee \neg\mathit{State}(l, i-1, T) \vee \mathit{State}(l, i, T)$

By construction, a satisfying assignment to $\mathit{CNF}_{\mathcal{T}}$ corresponds to the effects of an action model and the complete trajectories for this action model, given the values observed in the trajectories of $\mathcal{T}$. Moreover, the action model with these effects and no preconditions is consistent with $\mathcal{T}$.

Let $\mathit{CNF}_{\mathcal{T}}(\ell)$ be the formula containing all the clauses in $\mathit{CNF}_{\mathcal{T}}$ containing literals for a single fluent literal ℓ. Note that the clauses of $\mathit{CNF}_{\mathcal{T}}$ only contain literals for a single fluent literal, so $\mathit{CNF}_{\mathcal{T}}$ is satisfiable iff for every ℓ the formula $\mathit{CNF}_{\mathcal{T}}(\ell)$ is satisfiable. The final part of our proof will show that the CNF returned by EPI-SAM, $\mathit{CNF}_{\mathit{eff}}(\ell)$, is satisfiable iff $\mathit{CNF}_{\mathcal{T}}(\ell)$ is satisfiable. To this end, we rely on the refutation-completeness of resolution and examine which clauses may appear in a refutation of $\mathit{CNF}_{\mathcal{T}}(\ell)$. The $\mathit{IsPre}(a, \ell)$ literals, appearing only negatively, cannot appear in a refutation. Thus, any refutation will be based on clauses of types C2 and C3. Two types of proofs can be created from such clauses. The first requires observing the value of ℓ in enough states such that we have contradicting unit clauses with *IsEff* literals for some action a_i. That is, we have transitions $\langle s_i, a_i, s'_i \rangle$ and $\langle s_j, a_i, s'_j \rangle$ where l is observable in states s'_i, s_{j-1}, and s_j with values *false*, *true*, and *false*, respectively. This option is implemented in line 19 of Algorithm 5. The second type of proof requires using resolution to eliminate at least one *State* literal. Reordering the applications of the resolution rule on these literals to the beginning of the proof, we see that we must create clauses that correspond to consecutive runs of unobserved literals using the resolution rule on clauses of

type C3 for each step, beginning with either an observed literal or with using clauses of type C2 to eliminate the first $\mathit{State}(\ell, i, T)$ literal. These are, respectively, the clauses of $\mathit{CNF}_{\mathit{eff}}(l)$ created on lines 8 and 10 in Algorithm 4.

Lemma 7.6. *For every action a in A_{SAM} and literal ℓ, it holds that $\ell \in \mathit{pre}_{A_{SAM}}(a)$ if and only if there exists an action model A consistent with $\mathcal{T}$ where $\ell \in \mathit{pre}_{A}(a)$.*

Proof. We first prove that if EPI-SAM removes a literal ℓ from $\mathit{pre}(a)$, then there exists a transition $\langle s, a, s' \rangle$ in $\mathcal{T}$ where ℓ is false, and hence cannot be in $\mathit{pre}_{A*}(a)$. EPI-SAM removes ℓ from $\mathit{pre}(a)$ in two places in Algorithm 5: line 19 and line 23. The correctness of line 19 is immediate: if ℓ is observed to be false in a state where a has been applied then it cannot be a precondition of a (PI-SAM Rule 1). Before removing a precondition due to line 23, EPI-SAM creates a set of trajectories $\mathcal{T}_{\ell,a}$ that assumes ℓ was true whenever a was taken, and detects the set of actions A_{irr} that cannot affect the value of ℓ in any action model consistent with $\mathcal{T}_{\ell,a}$. Because of the frame axioms, the value of ℓ gets propagated in any transition that includes an action in A_{irr}. ℓ is only removed in line 23 if this propagation results in a state where ℓ has contradicting values. As this occurs for any action model consistent with $\mathcal{T}_{\ell,a}$, this implies that ℓ cannot be true in every state where a was applied, and thus cannot be a precondition of a in any action model consistent with $\mathcal{T}$.

Next, we prove that if ℓ has not been deleted from $\mathit{pre}(a)$ by EPI-SAM, then there exists an action model A consistent with $\mathcal{T}$ where $\ell \in \mathit{pre}_{A}(a)$. Consider the subset of A_{irr} that includes only actions that have been in a transition where the value of ℓ is not masked. For each action a' in this set, we are guaranteed that this value of ℓ is always the same, denoted $v(a', \ell)$. Otherwise a' would have been added to A_{irr}. The action model created by assigning $v(a', \ell)$ as an effect of a' for each of these actions is consistent with $\mathcal{T}_{l,a}$. Therefore, there exists an action model where ℓ is a precondition of a that is consistent with $\mathcal{T}$. □

Theorem 7.7. *EPI-SAM returns a safe action model.*

Proof. Let A^* denotes the action model of the underlying planning problem. Due to Lemma 7.6, every action applicable according to A_{SAM} is also applicable according to A^*. Consider a goal G and a (strong) plan to achieve it π_{SAM} created by a conformant planner given A_{SAM}. This means π_{SAM} achieves G for any action model that satisfies the $\{\mathit{CNF}_\ell\}_\ell$. Due to Lemma 7.5, we know that this means π_{SAM} achieves G according to any action model consistent with $\mathcal{T}$. Thus, π_{SAM} also achieves G according to A^*, as required. □

Theorem 7.8 (Strength)**.** *The action model A_{SAM} returned by EPI-SAM is the strongest safe action model, in the sense that if an action model A is not safe with respect to A_{SAM}, then there exists an action model A' consistent with $\mathcal{T}$ such that A is not safe with respect to A'.*

Proof. By contradiction, assume that A_{bad} is an action model that is not safe with respect to A_{SAM}, but it is safe with respect to any action model consistent with $\mathcal{T}$. This means that either there exists a literal ℓ that is in $\mathit{pre}_{A_{SAM}}$ but not in $\mathit{pre}_{A_{bad}}$ or a plan π_{bad} that achieves some goal G according to A_{bad} but not according to A_{SAM}. The first condition cannot hold due to Lemma 7.6: for any precondition assumed by A_{SAM} there exists an action model consistent with $\mathcal{T}$ that requires it. For the second condition, suppose that there is a plan under A_{bad} that is allowed by the EPI-SAM action model, but for which EPI-SAM does not achieve the goal. This means (by Lemma 7.5) that there was some action model consistent with $\mathcal{T}$ under which the goal was not achieved. The other action model is therefore not safe. □

Theorem 7.9. *Given a set of trajectories $\mathcal{T}$, the EPI-SAM learning runs in time*

$$\mathcal{O}\Big(|A| \cdot |\mathcal{F}| \cdot \sum_{a \in \mathcal{A}} |\mathcal{T}(a)|\Big)$$

where A is the set of actions, $\mathcal{F}$ is the set of literals.

Proof. The EPI-SAM algorithm consists of two parts: learning the effects and learning the preconditions. For learning the effects (algorithm 4), the algorithm has to iterate over all literals to create a CNF formula for each literal. The first inner loop (line 3-4) iterates through all actions to add mutually exclusive clauses for each action to the CNF, while the second inner loop (line 5-12) goes through

every transition in each trajectory to add clauses to the CNF based on Rule 1, 3 of Observation 7.4. Thus, the total time complexity of Algorithm 4 is $\mathcal{O}\Big(|\mathcal{F}|(|A| + \sum_{a\in\mathcal{A}} |\mathcal{T}(a))|\Big)$.

The second part, learning the preconditions (Algorithm 5), iterates over all actions and literals to build the precondition (in form of conjunction) for each action. In each inner loop, each literal in each transition of each trajectory is examined $O(1)$ times—in the first loop we create one clause for each step, and it is set to true or deleted at most once in the second loop. We can perform the bookkeeping in the second loop in linear time overall by suitable data structures: we maintain a linked list over the occurrences of a given action, all with a reference to a common structure for the action that records which settings of l appear in the post-state, and we record each of the unobserved runs of a literal with a linked list. Then checking if an action should be deleted takes $O(1)$ time and deleting the occurrences of an action takes $O(1)$ time per occurrence. The data structures likewise take $O(1)$ time per each occurrence of a literal to initialize. The algorithm has to go through every transition in each trajectory to build the data structures and perform the check/delete operations. Thus, the total time complexity of Algorithm 5 is $\mathcal{O}\Big(|A| \cdot |\mathcal{F}| \cdot \sum_{a\in\mathcal{A}} |\mathcal{T}(a)|\Big)$. □