Learning Safe Action Models with Partial Observability

Anonymous submission

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.

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, Celorrio, and Onaindia 2019; Stern and Juba 2017; Juba, Le, and Stern 2021; Cresswell, McCluskey, and West 2013; Wu, Yang, and Jiang 2007). A prominent example is FAMA (Aineto, Celorrio, and Onaindia 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, Le, and Stern 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, Le, and Stern 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-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 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, Celorrio, and Onaindia 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.

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 com-

plete assignment of values to all fluents, i.e., $s : F \to \{\text{true}, \text{false}\}$. 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 $p[\ell] = \text{true}$, and $p[\ell] = \text{false}$ the fact that $p[f] = \text{false}$ and $p[f] = \text{true}$, respectively. We say that a literal ℓ is in a partial state p, denoted $\ell \in p$, if $p[\ell] = \text{true}$. Similarly, if $p[\ell] = \text{false}$ we say that ℓ is not in s, denoted $\ell \notin s$. An action a is defined by a tuple $\langle \text{name}(a), \text{pre}(a), \text{eff}(a) \rangle$ where $\text{name}(a)$ is a unique identifier of the action and $\text{pre}(a)$ and $\text{eff}(a)$ 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 $\text{pre}(a)$ 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 $\text{eff}(a)$ then $\text{eff}(a)[f] = a(s)[f]$, (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, Le, and Stern (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, Palacios, and Geffner 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 nondeterministic 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, McCluskey, and West 2013; Yang, Wu, and Jiang 2007; Aineto, Celorrio, and Onaindia 2019; Juba, Le, and Stern 2021). Algorithms from the LOCM family (Cresswell and Gregory 2011; Cresswell, McCluskey, and West 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, Celorrio, and Onaindia 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, Le, and Stern 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 1. A safe model-free planning problem is 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).

Partial Information SAM Learning

Following prior work (Stern and Juba 2017; Juba, Le, and Stern 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 2 (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 1 (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 (Masking function). A trajectory masking function O 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 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

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

$\mathcal{T}_D$ 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 2. 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 $M_{\text{PI-SAM}}$ such that with probability at least $1-\delta$, a problem drawn from $\mathcal{P}_D$ is not solvable with $M_{\text{PI-SAM}}$ with probability at most ϵ.

Definition 5 (Adequate). An action model M is ϵ-adequate if, w.p. 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 $\text{pre}_M(a)$ or 2. there is a literal in $s' \setminus s$ but not in $\text{eff}_M(a)$.

Lemma 1. The action model returned by PI-SAM Learning given m trajectories (as specified in Theorem 2) is ϵ-adequate with probability at least $1-\delta$.

A proof of Lemma 1 appears in the appendix. We now prove Theorem 2.

Proof. When PI-SAM deletes a literal from $\text{pre}(a)$, 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 $M_{\text{PI-SAM}}$, it can also be taken in M^*. Conversely, since $M_{\text{PI-SAM}}$ is ϵ-adequate, 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 $M_{\text{PI-SAM}}$. The first condition ensures that the preconditions of $M_{\text{PI-SAM}}$ allow the action to be executed, and the second condition guarantees that $M_{\text{PI-SAM}}$ obtains the same states on each transition. Thus, with probability $1-\epsilon$, the goal is achievable under $M_{\text{PI-SAM}}$ using the plan. □

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 1). 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 $\text{pre}(a)$. 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 1). 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.

[2]The planner need not be deterministic.

Algorithm 1: EPI-SAM: Learning Effects

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

```
foreach literal ℓ do
    CNF_eff(ℓ) ← ∅
    foreach action a do  Add to CNF_eff(ℓ): {¬IsEff(ℓ, a) ∨ ¬IsEff(¬ℓ, a)}
    foreach trajectory T ∈ 𝒯 do
        foreach index i ∈ {1, ..., |T|} where ℓ ∈ T.s_i do
            T' ← max. prefix of T.s_i where ℓ is masked
            if ℓ ∉ T'.s_0 then  Add to CNF_eff(ℓ): {IsEff(ℓ, T'.a_1) ∨ ··· ∨ IsEff(ℓ, T'.a_|T'|)}
            Add to CNF_eff(ℓ): {¬IsEff(¬ℓ, T'.a_|T'|)}
            foreach j = 1 to |T'| − 1 do
                Add to CNF_eff(ℓ): {¬IsEff(¬ℓ, T'.a_j) ∨ IsEff(ℓ, T'.a_{j+1}) ∨ ··· ∨ IsEff(ℓ, T'.a_|T'|)}
            end
        end
    end
end
return {CNF_eff(ℓ)}_ℓ
```

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 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 $s_0[\ell] = \text{false}$, $s_2[\ell] = \text{true}$, 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 $\text{CNF}_{\text{eff}}(\ell)$, 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 $\text{IsEff}(\ell, a)$, representing whether literal ℓ is an effect of action a. In the second part (learning preconditions), EPI-SAM creates a set of literals $\text{pre}(a)$ 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.

Learning Effects To learn effects, EPI-SAM extends PI-SAM rules 2 and 3 (Observation 1) 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 3 (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 3. Initially, $\text{CNF}_{\text{eff}}(\ell)$ 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 $\text{CNF}_{\text{eff}}(\ell)$ 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 $\text{pre}(a)$ 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 $s[l] = \text{false}$, and (2) no set of action effects is consistent with $\mathcal{T}$ when we additionally set $s[l] = \text{true}$ 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 $s[l] = \text{true}$ 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]

Algorithm 2: EPI-SAM: Learning Preconditions

```
   Input  : Partially observed trajectories 𝒯
   Output: Precondition pre(a) for each action a
13 foreach action a do pre(a) ← all literals
14 foreach action a, literal ℓ do
15     if ∃ ⟨s, a, s'⟩ ∈ T ∈ 𝒯 where ¬ℓ ∈ s then
16         Remove ℓ from pre(a)
             Continue to the next (a, ℓ) pair
17     𝒯_{a,ℓ} ← AssumePrecondition(a, ℓ, 𝒯) ; A_irr ← ∅
       while ∃a' ∉ A_irr where Irrelevant(a',ℓ,𝒯_{a,ℓ}) do
18         foreach ⟨s, a', s'⟩ in T ∈ 𝒯_{a,ℓ} do
19             if s[ℓ] and s'[ℓ] are inconsistent then
20                 Remove ℓ from pre(a)
                   Continue to the next (a, ℓ) pair
21             else
22                 if s[ℓ] = ? then s[ℓ] ← s'[ℓ]
23                 Remove ⟨s, a', s'⟩ from T
24             end
25         end
26     end
27 end
28 return {pre(a)}_a
```

Algorithm 2 lists the pseudo-code 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 $\text{pre}(a)$ 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

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

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, $\{\text{CNF}_{\text{eff}}(\ell)\}_\ell$ and $\{\text{pre}_a\}_a$, 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 $f_{\text{IsEff}(a,\ell)}$ 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 ($\{\text{CNF}_{\text{eff}}(\ell)\}_\ell$), replacing every literal $\text{IsEff}(a, \ell)$ with the corresponding fluent $f_{\text{IsEff}(a,\ell)}$. 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, $\text{pre}(a)$. 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 $f_{\text{IsEff}(a,\ell)}$ is true then ℓ is an effect of a. Note that conditional effects are supported by many classical and conformant planners (Bonet 2010; Grastien, Scala, and Kessler 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 $\text{pre}_A(a)$ and $\text{eff}_A(a)$ 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 $\Phi_{\text{eff}} = \bigwedge_\ell \text{CNF}_{\text{eff}}(\ell)$, by setting $\text{IsEff}(\ell, a)$ 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. Proofs of the lemmas and theorems given below are in the appendix.

Lemma 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 3. For every action a in A_{SAM} and literal ℓ, it holds that $\ell \in \text{pre}_{A_{\text{SAM}}}(a)$ if and only if there exists an action model A consistent with $\mathcal{T}$ where $\ell \in \text{pre}_A(a)$.

Theorem 4. EPI-SAM returns a safe plan.

Theorem 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 6. Given a set of trajectories $\mathcal{T}$, EPI-SAM runs in time $\mathcal{O}\Big(|A| \cdot |\mathcal{F}| \cdot \sum_{a \in \mathcal{A}} |\mathcal{T}(a)|\Big)$.

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

[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	$\eta = 0.3$						$\eta = 0.1$					
		$\|\mathcal{T}\|$	P(pre)	R(pre)	P(eff)	R(eff)	T(sec)	$\|\mathcal{T}\|$	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 $\eta = 0.3$.

number of 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 tives/negatives (TP,FP,TN,FN) but compute TP, FP, TN, and FN differently. For preconditions, TP 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

Results and Discussion We performed experiments using PI-SAM and EPI-SAM*, a simplified (unsafe) ver in precision and recall directly to EPI-SAM. EPI-SAM* ery 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 $\neg\text{IsEff}(a, \ell)$ is satisfiable. EPI-SAM* outputs a classical action model instead of

	$\|T\| = 3$		$\|T\| = 5$		$\|T\| = 7$	
	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

Table 2: Results on Blocks with $\eta = 0.3$.

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 3: # of transitions needed to learn the preconditions

a conformant plan. Nevertheless, observe that the infer 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

As a baseline, we compared our algorithms to FAMA (Aineto, Celorrio, and Onaindia 2019), a modern algomark 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 pendent runs. Columns "P(pre)", "R(pre)", "P(eff)", and "R(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 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 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 perfor input. The results are shown in Table 2. We also com learn the preconditions (i.e., $P(\text{pre})$ and $R(\text{pre}) = 1.0$) 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 3. 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 2. The source code of the experiments will

rithm (Juba, Le, and Stern 2021) to partially observable work on general observations. In practice, we can choose observation sets (e.g., whether they satisfy the bounded concealment assumption or not). For future work, we

References

Aineto, D.; Celorrio, S.; and Onaindia, E. 2019. Learning action models with minimal observability. Artificial Intelligence, 275: 104–137.

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

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

Brafman, R.; and Shani, G. 2012. A multi-path compilation approach to contingent planning. In AAAI Conference on Artificial Intelligence.

Cresswell, S.; and Gregory, P. 2011. Generalised domain model acquisition from action traces. In International Conference on Automated Planning and Scheduling (ICAPS), 42–49.

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2013. Acquiring planning domain models using LOCM. The Knowledge Engineering Review, 28(2): 195–213.

Grastien, A.; Scala, E.; and Kessler, F. B. 2017. Intelligent Belief State Sampling for Conformant Planning. In IJCAI, 4317–4323.

Hoffmann, J.; and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In ICAPS, volume 2005.

Juba, B.; Le, H. S.; and Stern, R. 2021. Safe Learning of Lifted Action Models. In International Conference on Principles of Knowledge Representation and Reasoning (KR), 379–389.

Juba, B.; and Stern, R. 2022. Learning Probably Approximately Complete and Safe Action Models for Stochastic Worlds. In AAAI Conference on Artificial Intelligence.

Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. S.; and Koenig, S. 2020. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 1898–1900.

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

McDermott, D. 2000. The 1998 AI Planning Systems Competition. AI Magazine, 21(2): 13.

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

Mordoch, A.; Portnoy, D.; Stern, R.; and Juba, B. 2022. 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).

Stern, R.; and Juba, B. 2017. Efficient, Safe, and Probably Approximately Complete Learning of Action Models. In the International Joint Conference on Artificial Intelligence (IJCAI), 4405–4411.

Wu, K.; Yang, Q.; and Jiang, Y. 2007. ARMS: An automatic knowledge engineering tool for learning action models for AI planning. The Knowledge Engineering Review, 22(2): 135–152.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted MAX-SAT. Artificial Intelligence, 171(2-3): 107–143.