Efficient, Safe, and Probably Approximately Complete Learning of Action Models

Roni Stern
Ben Gurion University of the Negev
Be'er Sheva, Israel
roni.stern@gmail.com

Brendan Juba
Washington University in St. Louis
St. Louis, MO, 63130 USA
bjuba@wustl.edu

Abstract

In this paper we explore the theoretical boundaries of planning in a setting where no model of the agent's actions is given. Instead of an action model, a set of successfully executed plans are given and the task is to generate a plan that is *safe*, i.e., guaranteed to achieve the goal without failing. To this end, we show how to learn a conservative model of the world in which actions are guaranteed to be applicable. This conservative model is then given to an off-the-shelf classical planner, resulting in a plan that is guaranteed to achieve the goal. However, this reduction from a model-free planning to a model-based planning is not complete: in some cases a plan will not be found even when such exists. We analyze the relation between the number of observed plans and the likelihood that our conservative approach will indeed fail to solve a solvable problem. Our analysis show that the number of trajectories needed scales gracefully.

1 Introduction

In classical planning problems, a model of the acting agent and its relationship to the relevant world is given in a formal planning description language, e.g., the classical STRIPS model [Fikes and Nilsson, 1971] or PDDL [McDermott *et al.*, 1998]. Planning algorithms (planners) use this model to generate a plan for achieving a given goal condition. Creating a planning domain model, however, is acknowledged as a notoriously hard knowledge engineering task. This has motivated much work on *learning* such knowledge.

One such well-studied approach is to learn a domain model by observing the agent's interactions with the environment. The problems that arise in such approaches, however, are frequently intractable [Kearns and Valiant, 1994; Daniely and Shalev-Shwartz, 2016]. An alternative approach that is commonly used in *reinforcement learning* is to skip the model-learning phase and directly learn how to act by observing the agent's past actions and observations, and by guiding the agent towards performing exploratory actions [Kearns *et al.*, 2002, inter alia]. In most prior work, the agent may fail to execute a planned action. When this occurs, the agent replans, possibly refining an underlying domain or action model it has learned. Thus, the agent learns from both positive and negative examples.

In this work we address a different setting, in which such execution failures must be avoided. This setting is relevant when execution failures are very costly or when the agent has limited computation capabilities, and thus does not have the capability to re-plan after the plan it has tried to execute has failed. Consider, for example, a team of nano-bots deployed inside a human body for medical target identification and drug delivery [Cavalcanti *et al.*, 2007]. Re-planning is likely not possible in such nano-bots, and, of course, failing to cure the disease is undesirable. Thus, the planning task we focus on in this paper is how to find a plan that is *safe*, i.e., is guaranteed to achieve the goal, in a setting where a domain model is not available. We call this problem the *safe model-free planning problem*.

Since performing actions that might fail is not allowed, exploration actions cannot be performed. The only source of information available is a set of *trajectories* of previously executed plans. First, we show how to learn a set of actions that the agent can use from the given trajectories. For every such action a, we bound the set of predicates that are a's preconditions and effects. This bounded action model is then used to construct a classical planning problem such that a solution to it is a solution to our model-free problem. This approach to solve the model-free planning problem is sound and can be very efficient, since current classical planners are very efficient. However, it is not complete, as the planning problem that uses the learned action model might not be solvable even if the underlying model-free planning problem is. Nonetheless, we prove that under some assumptions, the probability of this occurring decreases quasi-linearly with the number of observed trajectories.

This positive result comes in contrast to the hardness of other tasks related to model learning. For example, learning to predict labels computed by finite-state machines [Kearns and Valiant, 1994] or even DNF formulas [Daniely and Shalev-Shwartz, 2016] is believed to be computationally intractable. Thus, we cannot hope to predict the values that fluents will take merely on the basis of the fact that these can be computed by a finite-state machine or a DNF. Similarly, even the problems of finding memoryless policies [Littman, 1994] or finite-state policies [Meuleau *et al.*, 1999] in simple environments is computationally intractable. Finally, in the

standard interactive learning model, simple examples (that would be captured by a STRIPS environment, for example) are known to force a learner to explore an exponential number of paths in any reasonable measure of the environment's complexity [Kakade, 2003, Section 8.6].

The key difference between the model learning we propose and these hardness results is that we limit our attention to the learning of STRIPS domain models in a PAC ("Probably Approximately Correct") sense: we do not aim to learn an accurate action model, only one that is sufficient for finding a safe plan in most cases. We introduce and desribe this PAC-like setting in Section 4.

2 Problem Definition

The setting we address is a a STRIPS planning domain, represented using the SAS$^+$ formalism [Bäckström and Nebel, 1995]. A planning domain in SAS+ is defined by the tuple $\mathcal{D} = \langle \mathcal{X}, \mathcal{O} \rangle$, where

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of state variables, each associated with a finite domain $Dom(x_i)$.
- $\mathcal{O}$ is a set of actions, where each action $a \in \mathcal{O}$ is defined by a tuple $\langle \mathit{pre}(a), \mathit{eff}(a) \rangle$, where $\mathit{pre}(a)$ and $\mathit{eff}(a)$ are assignments of values to state variables, i.e., a set of assignments of the form $x_i = v$ where $v \in Dom(x_i)$. We refer to $\mathcal{O}$ and its associated sets of preconditions and effects as the *action model* of the domain.

A state is also a set of assignments of the form $x_i = v$ such that every variable in $\mathcal{X}$ is assigned a single value from its corresponding domain. As a shorthand notation, if a state s contains an assignment $x_i = v$ we will write $s(x_i) = v$. A planning problem in SAS$^+$ is defined by the tuple $\Pi = \langle \mathcal{D}, s_\mathcal{I}, s_\mathcal{G} \rangle$, where $s_\mathcal{I}$ is the initial state and $s_\mathcal{G}$ is a partial assignment of some state variables that defines the goal. A state s is a goal state if $s_\mathcal{G} \subseteq s$. For an action a and a state s we denote by $apply(s, a)$ the state resulting from applying a on state s. A solution to an SAS$^+$ planning problem is a *plan*, which is a sequence of actions $a_1, \ldots, a_m$ such that $s_\mathcal{G} \subseteq apply(\ldots apply(s, a_1), a_2) \ldots a_m)$.

The key challenge we address in this paper is how to solve a SAS$^+$ planning problem $\Pi = \langle \mathcal{D}, s_\mathcal{I}, s_\mathcal{G} \rangle$ without having the action model of $\mathcal{D}$. Instead, a set of *trajectories* are assumed to be given.

Definition 1 (Trajectory) *A trajectory $T = \langle s_1, a_1, s_2, a_2, \ldots, a_{n-1}, s_n \rangle$ is an alternating sequence of states $(s_1, \ldots, s_n)$ and actions $(a_1, \ldots, a_n)$ that starts and ends with a state.*

A trajectory represents a successful execution of a sequence of actions by the agent. A set of trajectories may be obtained, for example, by monitoring the acting agent when it is controlled manually by a human operator. The states in the given trajectories are assumed to be fully observable.

Finally, we can define the *safe model-free planning* problem, which is the focus of this work.

Definition 2 (Safe model-free planning) *Let $\Pi = \langle \langle \mathcal{X}, \mathcal{O} \rangle, s_\mathcal{I}, s_\mathcal{G} \rangle$ be a planning problem and let $\mathcal{T} = \{\mathcal{T}^1, \ldots, \mathcal{T}^m\}$ be a set of trajectories in the planning domain $\langle \mathcal{X}, \mathcal{O} \rangle$. The input to a safe model-free planning problem is the tuple $\langle \mathcal{X}, s_\mathcal{I}, s_\mathcal{G}, \mathcal{T} \rangle$ and the task is to generate a plan π that is a solution to Π. We denote this safe model-free planning problem as $\Pi_\mathcal{T}$.*

3 Conservative Planning

To solve the safe model-free planning problem, we propose to learn a conservative action model, and then use it to find sound plans.

Following prior work on learning action models [Wang, 1995; 1994; Walsh and Littman, 2008], we partition every observed trajectory $\langle s_1, a_1, s_2, \ldots, s_{n+1} \rangle \in \mathcal{T}$ into a set of *action triplets*, where each action triplet is of the form $\langle s_i, a_i, s_{i+1} \rangle$. Let $\mathcal{T}(a)$ be all the action triplets for action a. A state s and s' are called pre- and post-state of a, respectively, if there is an action triplet $\langle s, a, s' \rangle$. Following Walsh and Littman [2008] and Wang [1994; 1995], we observe that from the set of trajectories $\mathcal{T}$ we can "bound" the set of predicates in an action's preconditions and effects, as follows.

$$\emptyset \subseteq \mathit{pre}(a) \subseteq \bigcap_{\langle s,a,s' \rangle \in \mathcal{T}(a)} s \quad (1)$$

$$\bigcup_{\langle s,a,s' \rangle \in \mathcal{T}(a)} s' \setminus s \subseteq \mathit{eff}(a) \subseteq \bigcap_{\langle s,a,s' \rangle \in \mathcal{T}(a)} s' \quad (2)$$

Equation 1 holds because a value assignment $(x_i = v)$ cannot be a precondition of a if it is not in every pre-state of a, and thus only value assignments that exists in all the pre-states of a may be preconditions of a (hence $\mathit{pre}(a) \subseteq \bigcap_{\langle s,a,s' \rangle \in \mathcal{T}(a)} s$). On the other hand, the fact that a state variable x_i happened to have the same value v in all the pre-states of the observed trajectories does not necessarily mean that $(x_i = v)$ is a precondition of a. It may even be the case that a has no preconditions at all, and thus the "lower bound" on an action's precondition is $\emptyset$.

Equations 2 holds because a value assignment $(x_i = v)$ cannot be an effect of a if it is not in every post-state of a, and thus only value assignments that exists in all the post-states of a may be preconditions of a (hence $\mathit{eff}(a) \subseteq \bigcap_{\langle s,a,s' \rangle \in \mathcal{T}(a)} s'$). On the other hand, every variable that has a value in the post-state that is different from the pre-state then it must be an effect (hence $\bigcup_{\langle s,a,s' \rangle \in \mathcal{T}(a)} s' \setminus s$). We denote the "upper bound" of the preconditions by $\mathit{pre}^U_\mathcal{T}(a)$ and the "lower bound" of the effects by $\mathit{eff}^L_\mathcal{T}(a)$.

3.1 Compilation to Classical Planning

Next, we use the bounds in Equations 1 and 2 to compile a safe model-free planning problem $\Pi_\mathcal{T} = \langle \mathcal{X}, s_\mathcal{I}, s_\mathcal{G}, \mathcal{T} \rangle$ to a classical SAS$^+$ problem $F(\Pi_\mathcal{T})$, such that a solution to $F(\Pi_\mathcal{T})$ is a solution to $\Pi_\mathcal{T}$, i.e., it is a solution for the underlying planning problem Π. $F(\Pi_\mathcal{T})$ is defined as follows. It has exactly the same set of state variables ($\mathcal{X}$), start state ($s_\mathcal{I}$), and goal ($s_\mathcal{G}$) as $\Pi_\mathcal{T}$. The actions of $F(\Pi_\mathcal{T})$ is the set of all the actions seen in an observed trajectory. We denote this set of actions by $A(\mathcal{T})$. The preconditions of an action a in $F(\Pi_\mathcal{T})$ are defined as the "upper bound" estimate given in Equation 1 ($\mathit{pre}^U_\mathcal{T}(a)$) and the effects of a in $F(\Pi_\mathcal{T})$ are

defined to be the "lower bound" estimate given in Equation 2 ($\mathit{eff}^L_{\mathcal{T}}(a)$).

Definition 3 (Safe) *An action model $\mathcal{O}'$ is safe with respect to an action model $\mathcal{O}$ if for every state s and action a it holds that (1) if a is applicable in s according to $\mathcal{O}'$ then it is also applicable in s according to $\mathcal{O}$, and (2) applying a to s results in exactly the same state when using either $\mathcal{O}'$ or $\mathcal{O}$.*

Theorem 1 *The action model in $F(\Pi_{\mathcal{T}})$ is safe with respect to the action model of Π.*

Proof: Let $\mathcal{O}_{\Pi}$ and $\mathcal{O}_{F(\Pi_{\mathcal{T}})}$ be the action models of Π and $F(\Pi_{\mathcal{T}})$, respectively. Since a is applicable in s according to $\mathcal{O}_{F(\Pi_{\mathcal{T}})}$, it means that $\mathit{pre}^U_{\mathcal{T}}(a) \subseteq s$ and consequently a is also applicable in s according to $\mathcal{O}_{\Pi}$, since $\mathit{pre}(a) \subseteq \mathit{pre}^U_{\mathcal{T}}$ (Equation 1).

Now, let s'_{Π} be the state resulting from applying a on s according to $\mathcal{O}_{\Pi}$, and let v denote the value of a state variable x_i in s'_{Π}, i.e., $v = s'_{\Pi}(x_i)$. Since $\mathit{eff}^L_{\mathcal{T}}(a) \subseteq \mathit{eff}(a)$, then according to $\mathcal{O}_{F(\Pi_{\mathcal{T}})}$ either $(x_i = v)$ is an effect of a or a has no effect on x_i. If the former is true then $s'_{F(\Pi_{\mathcal{T}})}(x_i) = v = s'_{\Pi}(x_i)$. Otherwise, it means that in the observed trajectories, applying a never changed the value of x_i, i.e., x_i was equal to v in both the pre-state and post-state. By definition, this means that $(x_i = v)$ is a precondition of a according to $\mathcal{O}_{F(\Pi_{\mathcal{T}})}$, and thus $s(x_i) = s'_{F(\Pi_{\mathcal{T}})}(x_i) = v = s'_{\Pi}(x_i)$. Thus, the effects of a on s will be the same in both action models, and hence a $s'_{\Pi} = s'_{F(\Pi_{\mathcal{T}})}$. ■

Corollary 1 (Soundness) *Every solution to $F(\Pi_{\mathcal{T}})$ is also a solution to $\Pi_{\mathcal{T}}$*

Corollary 1 is a direct result of Theorem 1, and its practical implication is the following algorithm for solving any safe model-free planning problem $\Pi_{\mathcal{T}}$: compile it to a classical planning problem $F(\Pi_{\mathcal{T}})$, run an off-the-shelf classical planner, and return the resulting plan. We refer to this algorithm as the *conservative model-free planner*. The *conservative model-free planner* is sound, but it is not complete. There can be planning problems that have a solution but the observed trajectories are not sufficient to induce a corresponding compiled planning problem that is solvable. As an extreme example, if we do not receive any observed trajectories, the compiled planning problem will not have any actions in its action model and thus will not be able to solve anything. In the next section we show that the required number of trajectories is actually reasonable.

Figure 1 illustrates how to generate $F(\Pi_{\mathcal{T}})$ from observed trajectories in a simple logistic-like domain with one truck, one package, and three possible location A, B, and C. The state variables are TruckAt, with domain A, B, and C, and PackageAt with domain A, B, C, and T, where T represents that the package is on the truck. The possible actions are $\text{Move}_{X,Y}$, Pickup_X, and Unload_X, for every $X, Y \in \{A, B, C\}$.

The three tables on the left-hand side, T_1, T_2, and T_3 are three observed trajectories, where the column T represents the value of the state variable TruckAt and the column P represents the value of the state variable PackageAt. For example, T_1 represents a trajectory where the truck starts at A, moves to B, picks up the package, moves to C, and unloads the package there. The tables on the right-hand side of Figure 1 show the action model learned from these trajectories. For didactic reasons, we show the action model learned given just T_1 ($F(\Pi_{\{T_1\}})$), then the action model learned given T_1 and T_2 ($F(\Pi_{\{T_1,T_2\}})$), and finally the action model learned using all three trajectories.

As can be observed, given only T_1 we do not have any knowledge of many actions such as Pick_A, Pick_C, etc. Also, the preconditions learned for the actions $\text{Move}_{A,B}$ and $\text{Move}_{B,C}$ are too restrictive, requiring that the package is at some location (while clearly a Move action only requires knowing the truck's location). However, given T_1 and T_2, these redundant preconditions are removed, and thus task that can be achieved with the actions $\text{Move}_{A,B}$, $\text{Move}_{B,C}$, Pick_B, and Unload_C will be found by our conservative model-free planner.

4 Learning to Perform Safe Planning

In general, we cannot guarantee that *any finite number of trajectories* will suffice to obtain *precisely* the underlying action model. This is because, for example, if some action never appears in a trajectory, we may not know its effects; or, if an action is only used in a limited variety of states, it may be impossible to distinguish its preconditions. Consequently, we cannot guarantee a complete solution to the model-free safe planning problem. However, as the number of trajectories increases, we can hope to learn enough of the actions accurately enough to be able to find plans for most goals in practice. This gives raise to a statistical view of the model-free safe planning task (Definition 2) that follows the usual statistical view of learning, along the lines of Vapnik and Chervonenkis [1971] and Valiant [1984].

Definition 4 (Safe Model-Free Learning-to-Plan) *We suppose that there is an arbitrary, unknown (prior) probability distribution D over triples of the form $\langle s_{\mathcal{I}}, s_{\mathcal{G}}, T\rangle$, where $s_{\mathcal{I}}$ is a state, $s_{\mathcal{G}}$ is a goal condition, and T is a trajectory that starts in $s_{\mathcal{I}}$ and ends in a state that satisfies $s_{\mathcal{G}}$, and all trajectories are applicable in a fixed planning domain $\mathcal{D} = \langle \mathcal{X}, \mathcal{O}\rangle$. In the safe model-free learning-to-plan task, we are given a set of triplets $\left\langle s^{(1)}_{\mathcal{I}}, s^{(1)}_{\mathcal{G}}, T^{(1)}\right\rangle, \ldots, \left\langle s^{(m)}_{\mathcal{I}}, s^{(m)}_{\mathcal{G}}, T^{(m)}\right\rangle$ drawn independently from D, and a new SAS$^+$ planning problem $\Pi = \langle \mathcal{D}, s_{\mathcal{I}}, s_{\mathcal{G}}\rangle$ such that the initial state and goal condition $(s_{\mathcal{I}}, G)$ are from some $(s_{\mathcal{I}}, s_{\mathcal{G}}, T)$ drawn from D. The task is to either output a plan for Π or, with probability at most ϵ, return that no plan was found.*

Remarks on the task formulation

We stress that D is arbitrary, and thus the conditional distribution of trajectories given a start and goal state, $D(T|s_{\mathcal{I}}, s_{\mathcal{G}})$, can also be any arbitrary distribution. For example, D could be the distribution of trajectories obtained by running some (unknown, sophisticated) planning algorithm on input $(s_{\mathcal{I}}, s_{\mathcal{G}})$, or produced by hand by a human domain expert. More generally, the adversarial choice of D in our model subsumes a model in which a trajectory $T(s_{\mathcal{I}}, s_{\mathcal{G}})$ is nondeterministically and adversarially chosen to be paired with the start state $s_{\mathcal{I}}$ and goal $s_{\mathcal{G}}$. (Indeed, the distributions used in

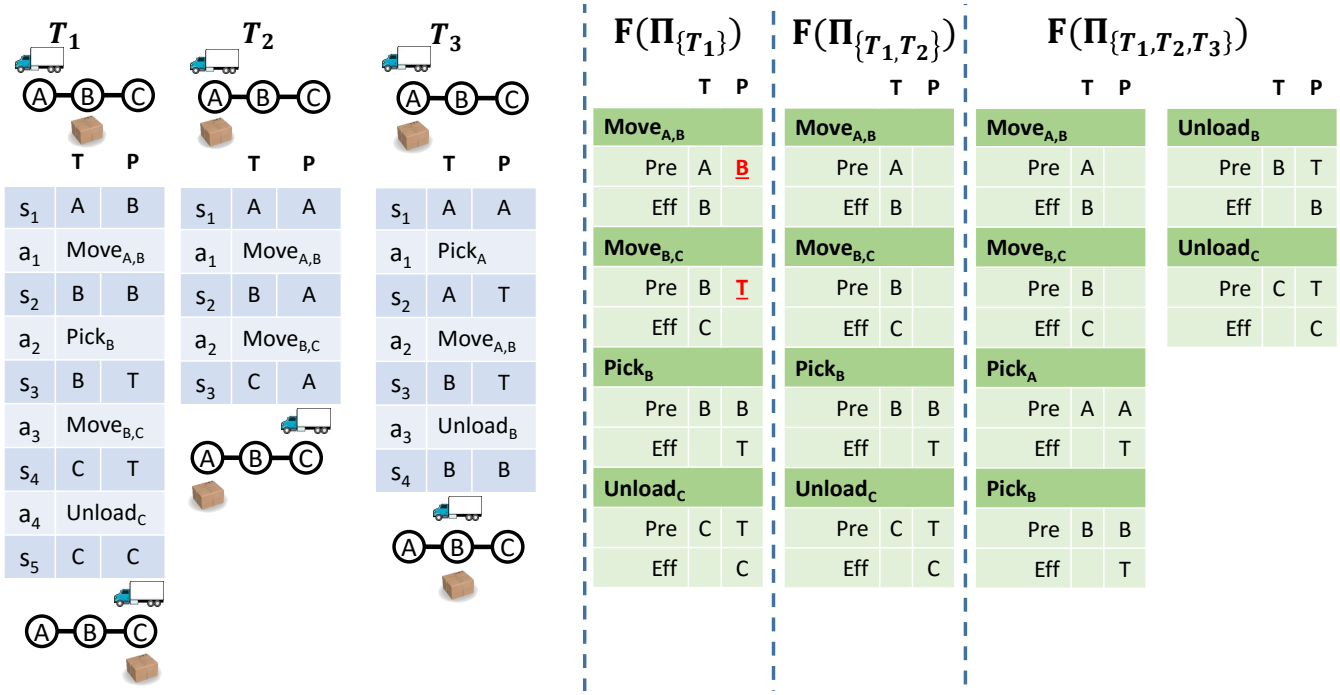


Figure 1: An example of creating $F(\Pi_{\mathcal{T}})$ from observed trajectories in a simple logistics-like domain. The left-hand side of the figure shows the three given trajectories T_1, T_2, and T_3. The right hand side of the figure shows our conservative action model, learned using different subsets of the trajectories T_1, T_2, and T_3 together.

such a case satisfy the stronger restriction that $D(T|s_{\mathcal{I}}, s_{\mathcal{G}})$ produces a deterministic outcome $T(s_{\mathcal{I}}, s_{\mathcal{G}})$, which does not necessarily hold in our model.)

We also note that our conservative model-free planner does not actually require knowledge of the goals $s_{\mathcal{G}}$ associated with the trajectories drawn from D. Thus, our approach actually solves a more demanding task that does not provide the goals to the learning algorithm. But, such a distribution over goals is nevertheless a central feature in our notion of "approximate completeness," and features prominently in the analysis as we discuss next.

Analysis of learning

A key question is how our certainty that a plan can be generated for a new start and goal state grows with the number of trajectories. Let log and ln denote $\log_2$ and $\log_e$, respectively, and let d denote the largest number of values for a state variable.

Theorem 2 *Using the conservative model-free planner, it is sufficient to observe* $m \geq \frac{(2\ln d)|A|}{\epsilon}(|\mathcal{X}| + \log\frac{2|A|}{\delta})$ *trajectories to solve the safe model-free learning-to-plan problem with probability* $1-\delta$.

Proof Outline. First, Lemma 1 shows that the set of actions used by our conservative model-free planner ($A(\mathcal{T})$) is sufficient to solve a randomly drawn problem with high probability. Then, Lemma 2 shows that under certain conditions, the preconditions we learned for these actions ($pre^U_{\mathcal{L}}$) are not too conservative, i.e., they are adequate for finding a plan with high probability. Finally, we prove that with high probability these conditions over the action model we learned indeed hold.

Lemma 1 *Let* A_ϵ *be the set of actions such that each action* $a \in A_\epsilon$ *appears in a trajectory sampled from* D *with probability at least* $\frac{\epsilon}{2|A|}$ *and let* $A(\mathcal{T})$ *be the set of every action that appeared in a trace. The probability that all the actions* $a \in A_\epsilon$ *appear in* $A(\mathcal{T})$ *is at least* $1-\delta/2$.

Proof: By definition, the probability that an action $a \in A_\epsilon$ does not exist in a trajectory drawn from D is $1-\frac{\epsilon}{2|A|}$. Since the observed trajectories $\mathcal{T}$ are drawn independently from D we have that the probability that $a \notin A(\mathcal{T})$ is $(1-\frac{\epsilon}{2|A|})^m \leq e^{-\frac{\epsilon\cdot m}{2|A|}}$, using the inequality $1-x \leq e^{-x}$. Since we assume in Theorem 2 that $m \geq \frac{(2\ln d)|A|}{\epsilon}(|\mathcal{X}|+\log\frac{2|A|}{\delta})$ which is larger than $\frac{(2\ln 2)|A|}{\epsilon}\log\frac{2|A|}{\delta}$, we have that the probability that $a \notin A(\mathcal{T})$ is at most

$$e^{-\frac{\epsilon\cdot m}{2|A|}} \leq e^{-\frac{\epsilon}{2|A|}\cdot\frac{(2\ln 2)|A|}{\epsilon}\log\frac{2|A|}{\delta}} = e^{-\ln\frac{2|A|}{\delta}} = \frac{\delta}{2|A|} \quad (3)$$

Hence, by a union bound over $a \in A_\epsilon$ (noting $|A_\epsilon| \leq |A|$), we have that $A_\epsilon \subseteq A(\mathcal{T})$ with probability $1-\delta/2$ as needed. ■

Stated informally, Lemma 1 says that with high probability we will observe all the "useful" actions, i.e., the actions used in many trajectories. However, we may have learned preconditions for these actions that are too conservative, preventing the planner from finding a plan even if one exists. We next define a property of action models that states that this does not occur frequently.

Definition 5 (Adequate) *We call an action model* ϵ-adequate *if, with probability at most* $\epsilon/2$*, we sample a trajectory* T *from* D *such that* T *contains an action triplet* $\langle s, a, s' \rangle$ *where* $a \in A_\epsilon$ *and* s *does not satisfy* $pre^U_{\mathcal{T}}$.

We say that an action model is ϵ-*inadequate* if it is not ϵ-adequate. An equivalent way to define the notion of an ϵ-*adequate* action model is that with probability at most $\epsilon/2$ a trajectory is sampled in which an action $a \in A_\epsilon$ is invoked on a state that does not satisfy the conservative preconditions of a we learned from the given set of trajectories ($\mathcal{T}$).

Lemma 2 *If the learned action model is* ϵ*-adequate and* $A_\epsilon \subseteq A(\mathcal{T})$*, then with probability* $1-\epsilon$ *our conservative model-free planner will find a plan for a start-goal pair* $(s_\mathcal{I}, s_\mathcal{G})$ *sampled from* D.

Proof: Let T be the (unknown) trajectory sampled for $(s_\mathcal{I}, s_\mathcal{G})$. The probability that T uses an action that is not in A_ϵ is at most $|A| \cdot \frac{\epsilon}{2|A|} = \epsilon/2$. Thus, T contains only actions known to our planner with probability at least $1-\epsilon/2$, since we assumed that $A_\epsilon \subseteq A(\mathcal{T})$. Since the action model is ϵ-adequate then with probability $1-\epsilon/2$ the learned preconditions are satisfied on all of the states in T. Thus, by a union bound, we find that with probability $1-\epsilon$, our planner could at least find $\mathcal{T}$. Hence it will find a trajectory from $s_\mathcal{I}$ to $s_\mathcal{G}$, as required. ■

Lemma 3 *The action model used by our conservative model-free planner is* ϵ*-adequate with probability at least* $1-\delta/2$.

Proof: Whether an action model is ϵ-adequate or not depends on the assignment of preconditions to actions. Since there are $|\mathcal{X}|$ state variables each with at most d values, then there are $d^{|\mathcal{X}|}$ possible assignments of preconditions for an individual action and a total of $d^{|\mathcal{X}|\cdot|A|}$ possible preconditions assignments for an action model . Let BAD be the subset of these action model preconditions assignments that are not ϵ-adequate. Clearly, BAD has size at most $d^{|\mathcal{X}|\cdot|A|}$.

Consider a particular assignment of preconditions in an ϵ-inadequate action model $\mathcal{O}_{BAD}$. Since $\mathcal{O}_{BAD}$ is ϵ-inadequate, it has a set of state-action pairs (s, a) associated with it such that $a \in A_\epsilon$ and a cannot be applied to s according to $\mathcal{O}_{BAD}$. The action model $\mathcal{O}_{BAD}$ can only be learned by our algorithm if none of these state-action pairs were observed in the given trajectories $\mathcal{T}$. On the other hand, by the definition of inadequacy the probability of having a state-action pair from that list in a trajectory drawn from D is at least $\epsilon/2$. Thus, the probability that our algorithm will learn a particular preconditions assignment of an ϵ-inadequate action model is at most $(1-\epsilon/2)^m$. Since $m \geq \frac{(2\ln d)|A|}{\epsilon}(|\mathcal{X}| + \log\frac{2|A|}{\delta}) > \frac{2\ln d}{\epsilon}(|A|\cdot|\mathcal{X}| + \log\frac{2}{\delta})$, then $(1-\epsilon/2)^m$ is smaller than

$$e^{-\frac{\epsilon}{2}\cdot m} \leq e^{-(\ln d\cdot(|A|\cdot|\mathcal{X}|+\log\frac{2}{\delta}))} \leq e^{-\ln d\cdot|A|\cdot|\mathcal{X}|}\cdot e^{-\ln\frac{2}{\delta}} \quad (4)$$

which is at most $\frac{\delta}{2\cdot d^{|\mathcal{X}|\cdot|A|}}$. Thus, by a union bound over this set of inadequate assignments BAD, the probability that any inadequate assignment of preconditions could be output is at most $\delta/2$. Thus, with probability $1-\delta$, the algorithm indeed produces an assignment of preconditions that is neither unsafe for $A(\mathcal{T})$ nor inadequate for A_ϵ, as needed. ■

	$P(\Pi)$	$\neg P(\Pi)$	Priors
$S(\Pi)$	$1-\epsilon$	ϵ	μ
$\neg S(\Pi)$	0	1	$1-\mu$
Marginals	$\mu(1-\epsilon)$	$1-\mu(1-\epsilon)$	

Table 1: The table shows the priors, conditional probabilities, and marginals obtained with probability $1-\delta$ when using our conservative model-free planner when it is given $m \geq \frac{(2\ln d)|A|}{\epsilon}(|\mathcal{X}| + \log\frac{2|A|}{\delta})$ trajectories.

4.1 Unsolvable Instances

The implication of Theorem 2 is that by observing a number of trajectories that is quasi-linear in the number of actions and the number of state variables, we expect our safe model-free planner to be complete with high probability, in the sense that if a solution exists it will be found. But what if some of the drawn problem instances are not solvable?

Corollary 2 *If the probability of drawing a solvable start-goal pair from* D *is* μ *then it is sufficient to observe* $m \geq \frac{(2\ln d)|A|}{\epsilon}(|\mathcal{X}| + \log\frac{2|A|}{\delta})$ *trajectories (of solvable instances) to guarantee that with probability of at least* $1-\delta$ *our conservative model-free planner will solve a start-goal pair drawn from* D *with probability at least* $\mu\cdot(1-\epsilon)$

Proof: Let $S(\Pi)$ be true or false if a given planning problem Π is solvable or unsolvable, respectively, and let $P(\Pi)$ be true or false if our planner returns a solution to Π or not respectively. We aim to bound $Pr(P(\Pi))$. Since our planner is sound, $P(\Pi) \rightarrow S(\Pi)$ and so

$$Pr(P(\Pi)) = Pr(P(\Pi) \wedge S(\Pi)) = Pr(P(\Pi)|S(\Pi))\cdot Pr(S(\Pi))$$

According to Theorem 2, $Pr(P(\Pi)|S(\Pi)) \geq 1-\epsilon$ and $Pr(S(\Pi)) = \mu$ by definition. ■

Table 1 shows the priors, conditional probabilities, and marginals use by the proof of Corollary 2 The first row shows the probabilities $Pr(P(\Pi)|S(\Pi))$, $Pr(\neg P(\Pi)|S(\Pi))$, abd $Pr(S(\Pi))$; the second row shows the probabilities $Pr(P(\Pi)|\neg S(\Pi))$, $Pr(\neg P(\Pi)|\neg S(\Pi))$, and $Pr(\neg S(\Pi))$; and the last row shows the marginal probabilities $Pr(P(\Pi))$ and $Pr(\neg P(\Pi))$.

Corollary 2 and Table 1 are valuable in that they provides a relationship between μ, ϵ, δ, and m. Thus, we can increase m to satisfy more demanding values of μ, ϵ, and δ and different types of error bounds. For example, consider an application that requires bounding, by some γ, the probability that our planner outputs incorrectly that no plan exists. In other words, an application that requires

$$Pr(S(\Pi)|\neg P(\Pi)) \leq \gamma$$

Using Bayes' rule and the values from Table 1, this means that

$$\frac{\epsilon\cdot\mu}{1-(1-\epsilon)\cdot\mu} \leq \gamma \Leftrightarrow \epsilon \leq \frac{\gamma\cdot(1-\mu)}{\mu\cdot(1+\gamma)}$$

Plugging $\frac{\gamma\cdot(1-\mu)}{\mu\cdot(1+\gamma)}$ into the sample complexity instead of ϵ in Theorem 2 will give the required number of trajectories to obtain a bound of γ on the probability of incorrectly outputting that a problem is not solvable.

4.2 Limited Planner Capabilities

The given trajectories $\mathcal{T}$ are presumably generated by some planning entity. Since planning in general is a hard problem, it may be the case that the planner that generated the given set of trajectories has drawn a solvable problem from D but was just not able to solve it due to memory or time constraints.

Learning from such a set of trajectories does not enable bounding the probability of solving problems in general. What can be obtained in such cases is to bound the solving capabilities of our conservative model-free planner with respect to the capabilities of the planner that generated the observed trajectories. Thus, instead of having μ represent the probability that an instance is solvable, we will have μ represent the probability that an instance is solvable by the original planner. The rest of the analysis follows exactly the same as in the previous section.

5 Related Work

Our work relates to several well-studied types of problems: planning under uncertainty, reinforcement learning, and domain model learning.

Planning under uncertainty. In common models for planning under uncertainty, such as Markov Decision Problems (MDP) and Partially Observable MDPs (POMDP), the uncertainty stems from the stochastic nature of the world or from imperfect sensors that prevent full observability of the agent's state. Our setting is different in that our uncertainty only stems from not knowing the agent's action model.

Reinforcement learning. Reinforcement learning algorithms learn how to act by interacting with the environment. Thus, they are designed for a trial-and-error approach to learn the domain and/or how to plan in it. Our task is to generate a plan that *must* work, so a trial-and-error approach is not sufficient.

Domain model learning. Most prior work on learning a domain model in general or a STRIPS action model from observed trajectories, such as ARMS [Yang *et al.*, 2007] and LOCM [Cresswell *et al.*, 2013], learn approximate models that do not guarantee safety. Hence, such work generally also involves some form of trial-and-error as well, iteratively requesting more example trajectories or interacting directly with the environment to refine the learned model [Mourão *et al.*, 2012; Wang, 1994; 1994; Walsh and Littman, 2008; Levine and DeJong, 2006; Jiménez *et al.*, 2013]. In addition, most works learn from both positive and negative examples – observing successful and failed trajectories, while we only require successful trajectories to be provided.

Another key difference is that unlike our work, most prior works do not provide statistical guarantees on the soundness of the plan generated with their learned model. An exception to this is the work of Walsh and Littman [2008], that also discussed the problem of learning STRIPS operators from observed trajectories and provided theoretical bounds on the sample complexity – the number of interactions that may fail until the resulting planner is sound and complete. By contrast, we do not assume any planning and execution loop and do not allow failed interactions. Hence, we aim for a planning algorithm that is guaranteed to be sound, at the cost of completeness. This difference affects their approach to learning. They attempted to follow an optimistic assumption about the preconditions and effects of the learned actions, in an effort to identify inaccuracies in their action model. By contrast, we are forced to take a pessimistic approach, as we aim for a successful execution of the plan rather than information gathering to improve the action model.

6 Conclusions

This paper deals with a planning problem in which the planner agent has no knowledge about its actions. Instead of an action model, the planner is given a set of observed trajectories of successfully executed plans. In this setting we introduced the *safe model-free planning* problem, in which the task is to find a plan that is guaranteed to reach the goal, i.e., there is no tolerance for execution failure. This type of problem is important in cases where failure is costly or in cases where the agent has no capability to replan during execution.

We showed how to use the given set of trajectories to learn about the agent's actions, bounding the set of predicates in the actions' preconditions and effects. Then, we proposed a conservative approach to solve the safe model-free problem that is based on a translation to a classical planning problem. This solution is sound but is not complete, as it may fail to find a solution even if one exists. However, we prove that under some assumptions the likelihood of finding a solution with this approach grows linearly with the number of predicates and quasi-linearly with the number of actions.

Future directions for safe model-free planning include studying how to address richer underlying planning models including parametrized actions, conditional effects, stochastic action outcomes, and partial observability. While some of these more complex action models can be compiled away (e.g., a problem with conditional effects can be compiled to a problem without conditional effects [Nebel, 2000]), the resulting problem can be significantly larger. A particularly interesting direction is how to learn there lifted action model, i.e., what can be learned from a trajectory with an action $a(x)$ on the action model of $a(y)$, where a is a parameterized action and x and y are different values for the same parameter.

Acknowledgements

B. Juba was partially supported by an AFOSR Young Investigator Award. R. Stern was partially supported by the Cyber Security Research Center at BGU.

References

[Bäckström and Nebel, 1995] Christer Bäckström and Bernhard Nebel. Complexity results for SAS+ planning. *Computational Intelligence*, 11(4):625–655, 1995.

[Cavalcanti *et al.*, 2007] Adriano Cavalcanti, Bijan Shirinzadeh, Robert A Freitas Jr, and Tad Hogg. Nanorobot architecture for medical target identification. *Nanotechnology*, 19(1):015103, 2007.

[Cresswell *et al.*, 2013] Stephen N. Cresswell, Thomas L. McCluskey, and Margaret M. West. Acquiring planning domain models using LOCM. *The Knowledge Engineering Review*, 28(02):195–213, 2013.

[Daniely and Shalev-Shwartz, 2016] Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limtations on learning DNF's. In *Proceedings of the 29th Conference on Computational Learning Theory*, volume 49 of *JMLR Workshops and Conference Proceedings*, pages 1–16. 2016.

[Fikes and Nilsson, 1971] Richard E Fikes and Nils J Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[Jiménez *et al.*, 2013] Sergio Jiménez, Fernando Fernández, and Daniel Borrajo. Integrating planning, execution, and learning to improve plan execution. *Computational Intelligence*, 29(1):1–36, 2013.

[Kakade, 2003] Sham M. Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, University of London, 2003.

[Kearns and Valiant, 1994] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

[Kearns *et al.*, 2002] Michael Kearns, Yishay Mansour, and Andrew Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Mach. Learn.*, 49(2):193–208, 2002.

[Levine and DeJong, 2006] Geoffrey Levine and Gerald DeJong. Explanation-based acquisition of planning operators. In *ICAPS*, pages 152–161, 2006.

[Littman, 1994] Michael L. Littman. Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, volume 3, page 238. MIT Press, 1994.

[McDermott *et al.*, 1998] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL-the planning domain definition language. Technical report, AIPS '98 - The Planning Competition Committee, 1998.

[Meuleau *et al.*, 1999] Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 417–426. Morgan Kaufmann Publishers Inc., 1999.

[Mourão *et al.*, 2012] Kira Mourão, Luke S Zettlemoyer, Ronald Petrick, and Mark Steedman. Learning STRIPS operators from noisy and incomplete observations. In *UAI*, pages 614–623, 2012.

[Nebel, 2000] Bernhard Nebel. On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, 12:271–315, 2000.

[Valiant, 1984] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 18(11):1134–1142, 1984.

[Vapnik and Chervonenkis, 1971] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[Walsh and Littman, 2008] Thomas J. Walsh and Michael L. Littman. Efficient learning of action schemas and web-service descriptions. In *the National Conference on Artificial Intelligence (AAAI)*, pages 714–719, 2008.

[Wang, 1994] Xuemei Wang. Learning planning operators by observation and practice. In *the International Conference on Artificial Intelligence Planning Systems (AIPS)*, pages 335–340, 1994.

[Wang, 1995] Xuemei Wang. Learning by observation and practice: An incremental approach for planning operator acquisition. In *ICML*, pages 549–557, 1995.

[Yang *et al.*, 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.