Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Reasoning, Learning & Decision-Making Track

Action Model Learning with Guarantees

Diego Aineto and **Enrico Scala**
Department of Information Engineering, University of Brescia, Italy
{diego.ainetogarcia, enrico.scala}@unibs.it

Abstract

This paper studies the problem of action model learning with full observability. Following the learning by search paradigm by Mitchell, we develop a theory for action model learning based on version spaces that interprets the task as search for hypotheses that are consistent with the learning samples. Our theoretical findings are instantiated in an online algorithm that maintains a compact representation of all solutions of the problem. Among this range of solutions, we bring attention to action models approximating the actual transition system from below (sound models) and from above (complete models). We show how to manipulate the output of our learning algorithm to build deterministic and non-deterministic formulations of the sound and complete models and prove that, given enough examples, both formulations converge into the very same true model. Our experiments reveal their usefulness over a range of planning domains.

1 Introduction

The engineering of action models is complex and error-prone, presenting one of the main bottlenecks in the application of model-based reasoning (Kambhampati 2007). Automating this process holds the promise of enabling AI planning (Ghallab, Nau, and Traverso 2004) even for those problems in which the model is unknown. Action model learning tackles this problem by computing an approximation of a domain's dynamics from demonstrations.

Over the last two decades, much of the research on action model learning has concentrated on learning under partial observability, investigating the application of different techniques with the aim of either improving the expressiveness of the learnt models or handling more incomplete and noisy demonstrations. Against this trend, *Safe Action Model (SAM) Learning* (Stern and Juba 2017; Juba, Le, and Stern 2021; Juba and Stern 2022; Mordoch, Juba, and Stern 2023) is a family of algorithms that takes a step back to study the fully observable setting from a theoretical standpoint that emphasizes the properties of the learnt model. Specifically, SAM is concerned with learning *safe* models – those allowing an agent to safely execute actions all the way to the goal.

This paper deepens this theoretical investigation through the lens of version spaces (Mitchell 1982). We focus on classical planning models, and develop a framework for learning action preconditions and effects by maintaining a version space comprising all hypotheses *consistent* with the demonstrations. The computed version space provides an efficient representation of *all* solutions to the action model learning problem. Among these solutions, those at the boundaries have special properties. At one end lies pessimism, while at the other, optimism. The pessimistic form leads to the construction of *sound* models – ones that always generate plans guaranteed to work with the true model, in line with the *safe* property studied by SAM. However, this approach will often discard valid plans. Conversely, the optimistic form yields *complete* models, which may not always generate valid plans but ensure the existence of a plan if one exists for the true model. This paper represents the first exploration into the concept of complete models within the context of action model learning. By introducing and investigating complete models alongside sound models, we provide an agent with more comprehensive reasoning capabilities. Indeed, not only can the agent demonstrate the existence of a valid plan when there is one induced by the sound model, but also prove the absence of any valid solution when there is none for the complete one.

The main contribution of our work is theoretical. Our investigation precisely establishes rules that heavily exploit the structure of the hypothesis space in order to learn all the solutions to an action model learning problem. These rules are operationalized into an online algorithm, which produces a compact representation of the solution set. Subsequently, we show how to manipulate this representation to derive both sound and complete action model formulations that, given enough demonstrations, converge to the very same true model. Importantly, the sound and the complete models ensure these properties after any number of demonstrations and whilst learning is undergoing. These models constitute incumbent solutions with complementary properties that can be leveraged while convergence to the true model has not been achieved or is infeasible. The resulting learning framework, namely VSLAM, subsumes approaches offering either only sound models (e.g., SAM (Stern and Juba 2017)) or just consistent models (e.g., FAMA (Aineto, Celorrio, and Onaindia 2019)), providing a unified account that can compute complete models, too. In order to understand the practical benefit of VSLAM, we conducted an experimental evaluation across various domains. Our findings demonstrate that VLSAM provides the agent with better reasoning ca-

pabilities than previous learning methods. This is because complete models offer plans sooner (sound models can at times be too pessimistic), and exploit negative demonstrations, too.

The paper is organized as follows: We start off with background material on action model learning and version spaces. Then we delve into building a precise mapping between these two worlds (Section 3) outlining several theoretical results that we leverage in Section 4 to build sound and complete action models. Section 5 clarifies how to extrapolate our theory to learn lifted representations. Then, we practically evaluated our approach in Section 6, and conclude with related work and discussion (sections 7 and 8).

2 Preliminaries

This section presents the basic notions around action model learning and version spaces.

2.1 Action Model Learning

An action model is a description of the capabilities of some agent, system or environment. In this work, we focus on learning deterministic action models with conjunctive preconditions (McDermott and others 1998), as defined below.

Definition 1 (Action Model)**.** *An action model is a tuple* $M = \langle F, A, \mathsf{pre}, \mathsf{eff}\rangle$ *where:*

- $F = \{f_1, \ldots, f_n\}$ *is a finite set of Boolean state variables called* fluents*. A positive (resp. negative)* literal *is* $l = f$ *(resp.* $l = \neg f$*) and its completement is* $\bar{l} = \neg l$*. We denote the set of all literals by* L*.*
- A *is a finite set of labels called* actions*.*
- $\mathsf{pre} : A \to 2^L$ *defines the* precondition $\mathsf{pre}(a) \subseteq L$ *for all* $a \in A$*.*
- $\mathsf{eff} : A \to 2^L$ *defines the* effect $\mathsf{eff}(a) \subseteq L$ *for all* $a \in A$*.*

An action model succinctly represents a transition system where a state s is a full assignment over F, represented by a subset of L without conflicting values. For instance, if $F = \{p_1, p_2\}$, then $L = \{p_1, \neg p_1, p_2, \neg p_2\}$ and a well-formed state s can be $\{p_1, \neg p_2\}$. We denote by S the state space induced by F. Formally, an action model $M = \langle F, A, \mathsf{pre}, \mathsf{eff}\rangle$ induces the transition system $\mathcal{T}_M = \{\langle s, a, s'\rangle \in S \times A \times S \mid \mathsf{pre}(a) \subseteq s \wedge s' = (s \setminus \overline{\mathsf{eff}(a)}) \cup \mathsf{eff}(a)\}$ where $\overline{\mathsf{eff}(a)} = \{\bar{l} \mid l \in \mathsf{eff}(a)\}$. This definition is equivalent to the successor function defined for classical planning using add and delete list explicitly (Ghallab, Nau, and Traverso 2004).

Action models are widely used in AI planning to formulate reachability problems over the induced transition system. A *classical planning problem* $P = \langle M, s_0, G\rangle$ is a reachability problem defined by combining an action model $M = \langle F, A, \mathsf{pre}, \mathsf{eff}\rangle$ with an *initial state* $s_0 \in S$ and *goal condition* $G \subseteq L$. A solution for P is a sequence of actions $\pi = (a_1, \ldots a_n)$ known as *plan*; the *execution* of π in s_0 yields an interleaved sequence $\langle s_0, a_1, s_1, a_2, s_2, \ldots, a_n, s_n\rangle$ that alternates actions and states iteratively reached by applying the actions one after the other. A plan is valid if every *transition* $\langle s_i, a_{i+1}, s_{i+1}\rangle$ belongs to the transition system $\mathcal{T}_M$, and it solves P if $G \subseteq s_n$. We denote the set of solution plans for P by $\Pi(P)$.

Action model learning is about computing the action model of an agent from *demonstrations* of its capabilities. Hereinafter, we denote by M^* the *true action model* of the agent and assume that it complies with Definition 1. Demonstrations are collected from executions of M^*, e.g., a plan or random walk, and represented similarly to transitions.

Definition 2 (Demonstration)**.** *A demonstration is a triple* $d = \langle s, a, s'\rangle$ *consisting of a pre-state* $s \in S$*, an action* $a \in A$*, and a post-state* $s' \in S \cup \{\perp\}$*.*

In this work, we consider *positive demonstrations*, those transitions of $\mathcal{T}_{M^*}$, and *negative demonstrations*, those representing the failure of executing action a in state s which we indicate by using a $\perp$ post-state.

An action model learning problem takes as input a set of fluents F, a set of actions A and a set of demonstrations D. The aim of action model learning is to find an action model that is *consistent* with all the demonstrations in D. It is worth noting that the space of models that can be synthesised given F and A is finite, i.e, $\mathcal{M} = \{\langle F, A, \mathsf{pre}, \mathsf{eff}\rangle \mid \forall a \in A : \mathsf{pre}(a) \in 2^L \wedge \mathsf{eff}(s) \in 2^L\}$.

Definition 3 (Action Model Learning Problem)**.** *An action model learning problem is a tuple* $\langle F, A, D\rangle$ *where* F *is a set of fluents,* A *is a set of actions, and* D *is a set of demonstrations. A solution to* $\langle F, A, D\rangle$ *is an action model* $M = \langle F, A, \mathsf{pre}, \mathsf{eff}\rangle$ *such that:*

1. *for all positive demonstrations* $\langle s, a, s'\rangle \in D$*, it holds that* $\mathsf{pre}(a) \subseteq s$ *and* $s' = (s \setminus \overline{\mathsf{eff}(a)}) \cup \mathsf{eff}(a)$*;*
2. *for all negative demonstrations* $\langle s, a, \perp\rangle \in D$*, it holds that* $\mathsf{pre}(a) \not\subseteq s$*.*

We denote by $\mathcal{M}_D$ *the set of solutions of* $\langle F, A, D\rangle$*, i.e., the subset of the model space* $\mathcal{M}$ *that satisfies (1) and (2) and is, therefore, consistent with* D*.*

2.2 Version Spaces

We adopt the notation and definitions introduced in later extensions of the version spaces framework (Lau et al. 2003).

Definition 4 (Hypothesis and Hypothesis Spaces)**.** *A* hypothesis *is a function* $h : I \to O$*. A* hypothesis space $\mathcal{H}$ *is a set of functions with the same domain and range.*

Definition 5 (Learning Example)**.** *A learning example* ϵ *is a pair* $(i, o) \in I \times O$*. A hypothesis* h *is consistent with a learning example* $\epsilon = (i, o)$ *if and only if* $h(i) = o$*.*

Definition 6 (Version Space)**.** *Given a hypothesis space* $\mathcal{H}$ *and a set of learning examples* E*, the* version space $\mathcal{V}_{\mathcal{H},E}$ *is the subset of* $\mathcal{H}$ *that is consistent with all examples in* E*. We will often omit the subscripts if the hypothesis space and learning set are clear from the context.*

When the hypothesis space is partially ordered relative to some partial order relation $\leq$, a version space $\mathcal{V}$ can be efficiently represented in terms of its least upper bound, called $\mathcal{U}$ boundary, and its greatest lower bound, called $\mathcal{L}$ boundary (Lau, Domingos, and Weld 2000). It can be proven that all hypotheses that belong to the boundaries or lie between them in the partial order are *consistent*.

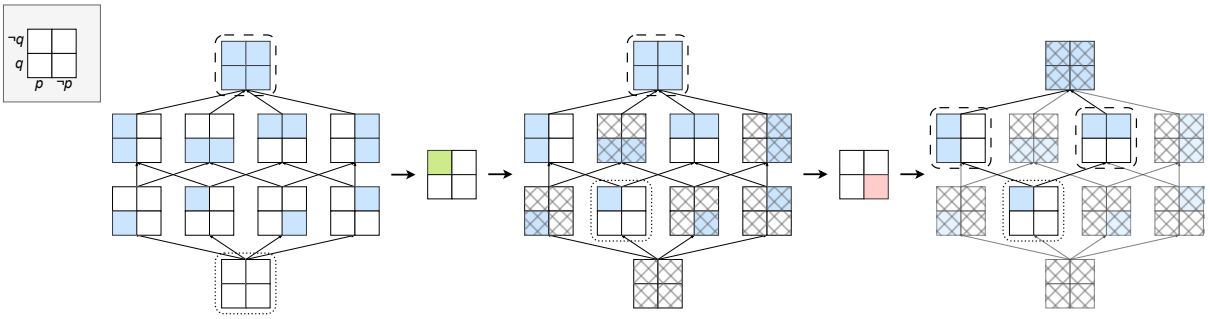

Figure 1: Example of version space learning for the task of learning a purely-conjunctive formula over two propositional variables. Examples consistent with the true formula are colored in green, while inconsistent ones are in red. Hypotheses are colored in blue, and those belonging to the $\mathcal{L}$ and $\mathcal{U}$ boundaries are denoted with dotted and dashed lines, respectively.

Definition 7 (Compact Representation of a Version Space)**.** *Let $\mathcal{L}_{\mathcal{H},E}$ and $\mathcal{U}_{\mathcal{H},E}$ be the boundaries of the version space $\mathcal{V}_{\mathcal{H},E}$. The hypotheses in $\mathcal{V}_{\mathcal{H},E}$ are $\{h \in \mathcal{H} \mid \exists (h_{\mathcal{L}}, h_{\mathcal{U}}) \in \mathcal{L}_{\mathcal{H},E} \times \mathcal{U}_{\mathcal{H},E} : h_{\mathcal{L}} \leq h \leq h_{\mathcal{U}}\}$.*

As an example, consider the task of learning a purely-conjunctive formula defined over two propositional variables p and q, i.e., formulas like $p \wedge \neg q$ and $\neg p$. We represent these formulas as sets (e.g., $\{p, \neg q\}$ and $\{\neg q\}$), and define a partial order such that, given two formulas ϕ and ϕ', $\phi \leq \phi'$ iff $\phi \supseteq \phi'$. Hence, $\{p, \neg q\} \leq \{p\}$ and $\{p, \neg q\} \leq \{\neg q\}$, but $\{p, \neg q\} \not\leq \{\neg p\}$. We represent the tautology `true` with an empty set $\{\}$, and the contradiction `false` with $\{p, \neg p, q, \neg q\}$, since p and $\neg p$ cannot be true at the same time (same for q and $\neg q$).

Figure 1 visually illustrates the concepts of the version spaces framework for this example, representing the 4 possible assignments over p and q as a 2×2 grid. In this graphical representation, a hypothesis (i.e., a conjunction) is a convex region of cells that we color in blue. For instance, the bottom-half of the grid represents $\{q\}$. The hypothesis space is represented as hypotheses connected via arrows denoting the partial order relation. A learning example is a cell of the grid, i.e., a full assignment over the variables, and it is labeled "green", if it satisfies the target conjunction, or "red", otherwise. A hypothesis is consistent with a green learning example if it contains it (i.e., it is a subset), whereas it is consistent with a red learning example if it does not contain it (not a subset). For instance, the hipothesis $\{p\}$ (left half of the grid) is consistent with the "green" example $\{p, \neg q\}$ (top-left cell), but the hypothesis $\{\neg p\}$ (right half) is not.

Version Space Learning (Mitchell 1982) is an online algorithm that maintains a version space by updating its boundaries each time a new learning example is observed. In Figure 1, the initial version space (before seeing any example) is shown on the left. It contains the whole hypothesis space, since nothing is inconsistent yet. Note that every hypothesis is a subset of $\{p, \neg p, q, \neg q\}$ and a superset of $\{\}$. Therefore, we can compactly represent the whole hypothesis space by its boundaries, setting $\mathcal{L}$ to contain the contradiction $\{p, \neg p, q, \neg q\}$ and $\mathcal{U}$ to contain the tautology $\{\}$. In the figure, the hypotheses constituting the $\mathcal{L}$ and $\mathcal{U}$ boundaries are denoted using dotted and dashed lines, respectively. When the "green" example $\{p, \neg q\}$ comes, some of the hypotheses, those crossed out in the figure, are not consistent with it. Therefore, we update the $\mathcal{L}$ boundary, which now contains the hypothesis $\{p, \neg q\}$. This update removes the inconsistent hypotheses from the version space since they no longer lie between the boundaries. The next example to arrive is a "red" one, indicating that $\{\neg p, q\}$ does not satisfy the target conjunction. The only hypothesis in the current version space that is inconsistent with it is the hypothesis that contains everything, the tautology $\{\}$. To remove it, we update the $\mathcal{U}$ boundary which now consists of two hypotheses, $\{p\}$ and $\{\neg q\}$. At this point, we still have not learnt the conjunction, but we can assert that it must be either $\{p, \neg q\}$, $\{p\}$ or $\{\neg q\}$. Observe that the $\mathcal{L}$ boundary is the most pessimistic hypothesis, a minimal frontier that encloses only the observed green examples. On the other hand, $\mathcal{U}$ is the most optimistic hypothesis, a maximal frontier only keeping outside the red examples.

3 Version Space Learning for Action Models

This section proposes a novel framework for action model learning based on version spaces. Roughly, our approach learns the precondition and effect of an action by computing a version space of its preconditions and a version space of its effects. We start by defining the hypothesis spaces and the update functions. Then, we present our algorithm.

3.1 The Hypothesis Space

For simplicity, with abuse of notation, in our context a hypothesis h is a subset of L that represents a function. When h is a *precondition hypothesis*, h represents the *applicability* function $App_h : S \to \{0, 1\}$ defined as $App_h(s) = h \subseteq s$. On the other hand, if h is an *effect hypothesis*, h represents the *successor* function $Suc_h : S \to S$ defined as $Suc_h(s) = (s \setminus \overline{h}) \cup h$.

Let $a \in A$, with $\mathcal{H}^a_p = 2^L$ we define the hypothesis space of a's precondition, while with $\mathcal{H}^a_e = 2^L$ the hypothesis space of its effects. We order our hypothesis spaces using

a set inclusion relation. Specifically, given two precondition hypotheses h_1 and h_2 in $\mathcal{H}^a_p$, $h_1 \leq h_2$ iff $h_1 \supseteq h_2$, and given two effect hypotheses h_1 and h_2 in $\mathcal{H}^a_e$, $h_1 \leq h_2$ iff $h_1 \subseteq h_2$. Note the opposite direction of the inclusion relation.

Considering these hypothesis spaces, the learning examples will be pairs $(s, b) \in S \times \{0, 1\}$ for the preconditions, and pairs $(s, s') \in S \times S$ for the effects. Learning examples are implicitly given by the demonstrations. A positive demonstration $\langle s, a, s' \rangle$ entails the learning example $(s, 1)$ for the precondition and the learning example (s, s') for the effect. On the other hand, a negative demonstration $\langle s, a, \bot \rangle$ entails only the learning example $(s, 0)$ for the precondition. Hereinafter, D_p and D_e denote the learning examples entailed by a set of demonstrations D for $\mathcal{H}^a_p$ and $\mathcal{H}^a_e$, respectively. A hypothesis $h \in \mathcal{H}^a_p$ is consistent with a learning example (s, b) iff $App_h(s) = b$. Analogously, a hypothesis $h \in \mathcal{H}^a_e$ is consistent with a learning example (s, s') iff $Suc_h(s) = s'$.

The following theorem shows that, when the learning examples come from the same set of demonstrations D, any model M built using preconditions and effects from the learnt version spaces is a solution to the action model learning problem $\langle F, A, D \rangle$, i.e., $M \in \mathcal{M}_D$.

Theorem 1. *Let $\mathcal{V}_{\mathcal{H}^a_p, D_p}$ and $\mathcal{V}_{\mathcal{H}^a_e, D_e}$ be the version spaces of preconditions and effects of $a \in A$. The action model $M = \langle F, A, \mathsf{pre}, \mathsf{eff} \rangle$ belongs to $\mathcal{M}_D$ if and only if $\forall a \in A$: $\mathsf{pre}(a) \in \mathcal{V}_{\mathcal{H}^a_p, D_p}$ and $\mathsf{eff}(a) \in \mathcal{V}_{\mathcal{H}^a_e, D_e}$.*

Proof. Let $d = \langle s, a, s' \rangle$ be a positive demonstration in D that entails the learning examples $(s, 1) \in D_p$ and $(s, s') \in D_e$. From the definition of version space, $(\mathsf{pre}(a), \mathsf{eff}(a))$ belongs to $\mathcal{V}_{\mathcal{H}^a_p, D_p} \times \mathcal{V}_{\mathcal{H}^a_e, D_e}$ iff $App_{\mathsf{pre}(a)}(s) = 1$ and $Suc_{\mathsf{eff}(a)}(s) = s'$ or, equivalently, iff $\mathsf{pre}(a) \subseteq s$ and $s' = (s \setminus \overline{\mathsf{eff}(a)}) \cup \mathsf{eff}(a)$. Now, let $d = \langle s, a, \bot \rangle$ be a negative demonstration in D and $(s, 0) \in D_p$ the entailed learning example. Again, $\mathsf{pre}(a) \in \mathcal{V}_{\mathcal{H}^a_p, D_p}$ if and only if $App_{\mathsf{pre}(a)}(s) = 0$, i.e., iff $\mathsf{pre}(a) \not\subseteq s$. Therefore, if $\forall a \in A : \mathsf{pre}(a) \in \mathcal{V}_{\mathcal{H}^a_p, D_p} \wedge \mathsf{eff}(a) \in \mathcal{V}_{\mathcal{H}^a_e, D_e}$, M satisfies conditions (1) and (2) of Definition 3, i.e., $M \in \mathcal{M}_D$; otherwise, $M \notin \mathcal{M}_D$. □

3.2 Initializing and Updating the Version Space

The initialization of the version space learning algorithm sets the version space to contain the whole hypothesis space, i.e., $\mathcal{V}_{\mathcal{H}^a_p, \emptyset} = \mathcal{H}^a_p$ and $\mathcal{V}_{\mathcal{H}^a_e, \emptyset} = \mathcal{H}^a_e$. This is done by setting the $\mathcal{L}$ and $\mathcal{U}$ boundaries to contain the minimal and maximal elements of the hypothesis space, respectively. In our problem, for all actions $a \in A$, $\mathcal{L}_{\mathcal{H}^a_p, \emptyset} = \{L\}$, $\mathcal{U}_{\mathcal{H}^a_p, \emptyset} = \{\emptyset\}$, $\mathcal{L}_{\mathcal{H}^a_e, \emptyset} = \{\emptyset\}$, and $\mathcal{U}_{\mathcal{H}^a_e, \emptyset} = \{L\}$. Indeed, by using Definition 7, it is easy to see that both bounds on the respective version spaces will yield $\mathcal{V}_{\mathcal{H}^a_p, \emptyset} = \mathcal{V}_{\mathcal{H}^a_e, \emptyset} = 2^L$, i.e., every hypothesis is consistent provided we are given no demonstrations. Note that, by Theorem 1 and the definition of version spaces, these boundaries allow to compactly represent the full space of action models $\mathcal{M}$.

As illustrated in Figure 1, updating a version space involves extending the $\mathcal{L}$ boundary or shrinking the $\mathcal{U}$ boundary. This is done by modifying the hypothesis that constitute the boundaries or removing them. In our context, where hypotheses are sets, the update consists in finding the smallest superset or the largest subset that is consistent with the new demonstration. The next theorem shows how the boundaries for the version space of a's preconditions $\mathcal{V}_{\mathcal{H}^a_p}$ are updated. Intuitively, we make the $\mathcal{L}$ boundary weaker when we see a positive demonstration by removing any literal not in its pre-state, whilst we tighten the $\mathcal{U}$ boundary by adding some literal not in the pre-state of a negative demonstration. Also, observe that the hypothesis in $\mathcal{L}$ is only refined or removed so this boundary will at most be a singleton. This matches the well-known result by Mitchell (1982) regarding the learning of purely-conjunctive formulas.

Theorem 2 (Update rules for $\mathcal{V}_{\mathcal{H}^a_p}$)**.** *Let $\mathcal{L}_{\mathcal{H}^a_p, D_p}$ and $\mathcal{U}_{\mathcal{H}^a_p, D_p}$ be the boundaries of a version space $\mathcal{V}_{\mathcal{H}^a_p, D_p}$ and d a demonstration. The updated version space $\mathcal{V}_{\mathcal{H}^a_p, D'_p}$, with $D' = D \cup \{d\}$, is given by the following rules.*
If $d = \langle s, a, s' \rangle$ is a positive demonstration:

- ***RUP.*** *Remove inconsistent hypotheses from $\mathcal{U}_{\mathcal{H}^a_p, D_p}$:*

$$\mathcal{U}_{\mathcal{H}^a_p, D'_p} \coloneqq \{h_\mathcal{U} \mid h_\mathcal{U} \in \mathcal{U}_{\mathcal{H}^a_p, D_p} \wedge h_\mathcal{U} \subseteq s\}$$

- ***ULP.*** *Update hypotheses in $\mathcal{L}_{\mathcal{H}^a_p, D_p}$:*

$$\mathcal{L}_{\mathcal{H}^a_p, D'_p} \coloneqq \{h_\mathcal{L} \cap s \mid h_\mathcal{L} \in \mathcal{L}_{\mathcal{H}^a_p, D_p}\}$$

If $d = \langle s, a, \bot \rangle$ is a negative demonstration:

- ***RLP.*** *Remove inconsistent hypotheses from $\mathcal{L}_{\mathcal{H}^a_p, D_p}$:*

$$\mathcal{L}_{\mathcal{H}^a_p, D'_p} \coloneqq \{h_\mathcal{L} \mid h_\mathcal{L} \in \mathcal{L}_{\mathcal{H}^a_p, D_p} \wedge h_\mathcal{L} \not\subseteq s\}$$

- ***UUP.*** *Update hypotheses in $\mathcal{U}_{\mathcal{H}^a_p, D_p}$:*
  Let $h_\mathcal{L} \in \mathcal{L}_{\mathcal{H}^a_p, D_p}$,

$$\mathcal{U}_{\mathcal{H}^a_p, D'_p} \coloneqq \{h_\mathcal{U} \mid h_\mathcal{U} \in \mathcal{U}_{\mathcal{H}^a_p, D_p} \wedge h_\mathcal{U} \not\subseteq s\} \cup \\ \cup \{h_\mathcal{U} \cup \{l\} \mid h_\mathcal{U} \in \mathcal{U}_{\mathcal{H}^a_p, D_p} \wedge h_\mathcal{U} \subseteq s \ \wedge l \in h_\mathcal{L} \setminus s\}$$

Proof. When $d = (s, a, s')$ is a positive demonstration, a hypothesis h is inconsistent iff $h \not\subseteq s$. Rule RUP removes an upper bound $h_\mathcal{U} \in \mathcal{U}_{\mathcal{H}^a_p, D_p}$ if $h_\mathcal{U} \not\subseteq s$ and, in doing so, removes any hypothesis h such that $h_\mathcal{U} \subset h$ from the version space. Observe that, since $h_\mathcal{U} \not\subseteq s$, any superset of $h_\mathcal{U}$ will also be inconsistent. Rule ULP raises the lower bound from $h_\mathcal{L}$ to $h_\mathcal{L} \cap s$. Indeed, $h_\mathcal{L} \cap s \subseteq s$ and all subsets of $h_\mathcal{L} \cap s$ are also consistent. Note that, if $h_\mathcal{L}$ already satisfied $h_\mathcal{L} \subseteq s$, this rule causes no change, i.e., $h_\mathcal{L} = h_\mathcal{L} \cap s$; otherwise, $h_\mathcal{L} \cap s$ is the largest subset that is consistent since $\forall l \in h_\mathcal{L} \setminus (h_\mathcal{L} \cap s)$ it holds that $l \notin s$ so all other subsets h such that $h_\mathcal{L} \cap s \subset h \subseteq h_\mathcal{L}$ are inconsistent.

In the case that $d = \langle s, a, \bot \rangle$, a hypothesis h is inconsistent if $h \subseteq s$. Rule RLP removes the lower bound $h_\mathcal{L} \in \mathcal{L}_{\mathcal{H}^a_p, D}$ if $h_\mathcal{L} \subseteq s$ which also removes all its subsets from the version space. Indeed, for any subset h of $h_\mathcal{L}$ it holds that $h \subseteq h_\mathcal{L} \subseteq s$ and, therefore, h is inconsistent. Rule UUP shrinks an upper bound $h_\mathcal{U} \in \mathcal{U}_{\mathcal{H}^a_p, D}$ to the set $\{h_\mathcal{U} \cup \{l\} \mid l \in h_\mathcal{L} \setminus s\}$ if $h_\mathcal{U} \subseteq s$. Note that $\forall l \in h_\mathcal{L} \setminus s : h_\mathcal{U} \cup \{l\} \not\subseteq s$ so all the new upper bounds are the smallest supersets of $h_\mathcal{U}$ that are consistent. □

As an example of our rules, consider an environment described through facts p_1 and p_2, i.e., $F = \{p_1, p_2\}$, and with a single action a. Assume to have at some point $\mathcal{L}_{\mathcal{H}^a_p,D_p} = \{\{p_1, \neg p_2\}\}$ and $\mathcal{U}_{\mathcal{H}^a_p,D_p} = \{\emptyset\}$. Now, say one gets the two demonstrations $d_1 = \langle\{p_1, p_2\}, a, \{\neg p_1, p_2\}\rangle$ and $d_2 = \langle\{\neg p_1, p_2\}, a, \bot\rangle$, in this very order. d_1 leaves $U_{\mathcal{H}^a_p,D_p}$ unaltered (by RUP) while it weakens the lower bound, i.e., $\mathcal{L}_{\mathcal{H}^a_p,D_p} = \{\{p_1\}\}$ (by ULP); indeed it is easy to see that the safe application of a only requires p to be true. Instead, d_2 leaves $\mathcal{L}_{\mathcal{H}^a_p,D_p}$ intact (by RLP) while it enforces that $U_{\mathcal{H}^a_p,D_p} = \{\{p_1\}\}$ (by UUP). Next, we see what happens if we see d_1 and d_2 in reverse order. After d_2, we will have $U_{\mathcal{H}^a_p,D_p}$ made up of two elements, i.e., $\{p_1\}$ and $\{\neg p_2\}$. This is the effect of the second part of the UUP rule. In this case we are basically inferring that, at this moment, only for the pre-state of d_2 we are sure that the action is not applicable. RLP will produce no change in $\mathcal{L}_{\mathcal{H}^a_p,D_p}$. After d_1, both $U_{\mathcal{H}^a_p,D_p}$ and $\mathcal{L}_{\mathcal{H}^a_p,D_p}$ will remain with only $\{p_1\}$ as an element. With both sequencing of demonstrations, we observe that the version space converged and no further demonstrations are needed.

Next, we move on to the learning of effects. Before presenting the update rules, we introduce the following lemma that gives us a better handle on the version space of effects as it enables reasoning about consistency and inconsistency of a given hypothesis provided some positive demonstration in terms of set inclusion.

Lemma 1. *Let $d = \langle s, a, s'\rangle$ be a positive demonstration, and $h \in \mathcal{H}^a_e$. h is consistent with d, i.e., $s' = (s \setminus \overline{h}) \cup h$ iff $s' \setminus s \subseteq h \subseteq s'$.*

Proof Sketch. Starting with $s' \setminus s \subseteq h \subseteq s' \implies s' = (s \setminus \overline{h}) \cup h$. By contradiction, we assume the antecedent is true while the consequent is false. If $s' \neq (s \setminus \overline{h}) \cup h$ then either (1) $s' \not\subseteq (s \setminus \overline{h}) \cup h$ or (2) $s' \not\supseteq (s \setminus \overline{h}) \cup h$. For (1), we can use algebra of sets to simplify $s' \setminus ((s \setminus \overline{h}) \cup h) \neq \emptyset$ into $\emptyset \neq \emptyset$, a contradiction. For (2), we rewrite $((s \setminus \overline{h}) \cup h) \setminus s' \neq \emptyset$ into $(s \setminus s') \setminus \overline{h} \neq \emptyset$. Since $\overline{s' \setminus s} = s \setminus s'$ and $s' \setminus s \subseteq h$, we have that $s \setminus s' \subseteq \overline{h}$ and we arrive at a contradiction $\emptyset \neq \emptyset$.

Similarly, for the proof of $s' = (s \setminus \overline{h}) \cup h \implies s' \setminus s \subseteq h \subseteq s'$, we show that both (3) $s' \setminus s \not\subseteq h$ and (4) $h \not\subseteq s'$ lead to contradictions. For (3), consider that $s' = (s \setminus \overline{h}) \cup h$ implies $s' \subseteq (s \setminus \overline{h}) \cup h$, which can be rewritten as $(s' \setminus s) \subseteq h$. Hence, (3) cannot be true. Finally, (4) is very obviously false given that s' is the union of $(s \setminus \overline{h})$ and h. □

The update rules for the version space of effects, presented in the next theorem, leverage Lemma 1. The intuition for these rules is that, whenever we get a new positive demonstration $\langle s, a, s'\rangle$ we update the upper bound to be a subset of s' and the lower bound to be a superset of $s' \setminus s$. By Lemma 1, any hypothesis between the updated bounds will also be consistent. Note that our rules do not increase the cardinality of the boundaries, so both $\mathcal{L}$ and $\mathcal{U}$ will at most contain one hypothesis.

Algorithm 1 VSLAM

Input Action Model Learning problem $\langle F, A, D\rangle$
Output $\mathcal{L}_{\mathcal{H}^a_p}, \mathcal{U}_{\mathcal{H}^a_p}, \mathcal{L}_{\mathcal{H}^a_e}$ and $\mathcal{U}_{\mathcal{H}^a_e}$ for all $a \in A$

1: **for** $a \in A$ **do** ▷ Inizialisation
2: $\quad\mathcal{L}_{\mathcal{H}^a_p} \coloneqq \{L\}$
3: $\quad\mathcal{U}_{\mathcal{H}^a_p} \coloneqq \{\emptyset\}$
4: $\quad\mathcal{L}_{\mathcal{H}^a_e} \coloneqq \{\emptyset\}$
5: $\quad\mathcal{U}_{\mathcal{H}^a_e} \coloneqq \{L\}$
6: **for** $\langle s, a, s'\rangle \in D$ **do** ▷ Online loop
7: $\quad$**if** s' is not $\bot$ **then** ▷ positive demonstration
8: $\quad\quad\mathcal{U}_{\mathcal{H}^a_p} \coloneqq RUP(\mathcal{U}_{\mathcal{H}^a_p}, (s, 1))$
9: $\quad\quad\mathcal{L}_{\mathcal{H}^a_p} \coloneqq ULP(\mathcal{L}_{\mathcal{H}^a_p}, (s, 1))$
10: $\quad\quad\mathcal{L}_{\mathcal{H}^a_e} \coloneqq ULE(\mathcal{L}_{\mathcal{H}^a_e}, (s, s'))$
11: $\quad\quad\mathcal{U}_{\mathcal{H}^a_e} \coloneqq UUE(\mathcal{U}_{\mathcal{H}^a_e}, (s, s'))$
12: $\quad$**else** ▷ negative demonstration
13: $\quad\quad\mathcal{L}_{\mathcal{H}^a_p} \coloneqq RLP(\mathcal{L}_{\mathcal{H}^a_p}, (s, 0))$
14: $\quad\quad\mathcal{U}_{\mathcal{H}^a_p} \coloneqq UUP(\mathcal{U}_{\mathcal{H}^a_p}, (s, 0))$
return $(\mathcal{L}_{\mathcal{H}^a_p}, \mathcal{U}_{\mathcal{H}^a_p}, \mathcal{L}_{\mathcal{H}^a_e}, \mathcal{U}_{\mathcal{H}^a_e})$

Theorem 3 (Update rules for $\mathcal{V}_{\mathcal{H}^a_e}$)**.** *Let $\mathcal{L}_{\mathcal{H}^a_e,D_e}$ and $\mathcal{U}_{\mathcal{H}^a_e,D_e}$ be the boundaries of a version space $\mathcal{V}_{\mathcal{H}^a_e,D_e}$ and $d = \langle s, a, s'\rangle$ a positive demonstration. The updated version space $\mathcal{V}_{\mathcal{H}^a_e,D'_e}$, with $D' = D \cup \{d\}$, is given by the following rules:*

- ***ULE.*** *Update hypotheses in $\mathcal{L}_{\mathcal{H}^a_e,D_e}$:*

$$\mathcal{L}_{\mathcal{H}^a_e,D'_e} \coloneqq \{h_\mathcal{L} \cup (s' \setminus s) \mid h_\mathcal{L} \in \mathcal{L}_{\mathcal{H}^a_e,D_e} \wedge h_\mathcal{L} \subseteq s'\}$$

- ***UUE.*** *Update hypotheses in $\mathcal{U}_{\mathcal{H}^a_e,D_e}$:*

$$\mathcal{U}_{\mathcal{H}^a_e,D'_e} \coloneqq \{h_\mathcal{U} \cap s' \mid h_U \in \mathcal{U}_{\mathcal{H}^a_e,D_e} \wedge s' \setminus s \subseteq h_\mathcal{U}\}$$

Proof. Both rules will only keep consistent hypotheses by Lemma 1. Indeed each hypothesis h is such that $s' \setminus s \subseteq h \subseteq s'$. Moreover, ULE computes the smallest superset of $h_\mathcal{L}$ that is consistent. Indeed, $\forall l \in (h_\mathcal{L} \cup (s' \setminus s)) \setminus h_\mathcal{L} : l \in s' \setminus s$ so removing any newly added literal from the hypothesis would make it inconsistent. Instead, for UUE we have that, if $h_\mathcal{U}$ was already a subset of s', UUE produces no change; otherwise, $h_\mathcal{U} \cap s' \subseteq s'$ is the largest subset of $h_\mathcal{U}$ that is consistent. □

To illustrate these rules, we resume our example environment with $F = \{p_1, p_2\}$ but this time considering the learning of effects. Assume that our starting point is $\mathcal{L}_{\mathcal{H}^a_e,D_e} = \{\{\neg p_1\}\}$ and $\mathcal{U}_{\mathcal{H}^a_e,D_e} = \{\{\neg p_1, \neg p_2\}\}$, and we receive demonstration $d_1 = \langle\{p_1, p_2\}, a, \{\neg p_1, p_2\}\rangle$. Rule ULE leaves $\mathcal{L}_{\mathcal{H}^a_e,D_e}$ unaltered, the lower bound is already a superset of $\{\neg p_1, p_2\} \setminus \{p_1, p_2\} = \{\neg p_1\}$. In contrast, by applying rule UUE, we refine the current hypothesis in $\mathcal{U}_{\mathcal{H}^a_e,D_e}$, $\{\neg p_1, \neg p_2\}$, into $\{\neg p_1, \neg p_2\} \cap \{\neg p_1, p_2\} = \{\neg p_1\}$. At this point, both boundaries have converged and we can safely assert that $\{\neg p_1\}$ is the true effect of action a.

3.3 The VSLAM Algorithm

In this section we present VSLAM, our algorithm for action model learning, outlined in Algorithm 1. VSLAM takes as

input an action model learning problem $\langle F, A, D\rangle$ and returns the boundaries of the version space of preconditions and of effects for each action in A.

The pseudocode for VSLAM is a straightforward instantiation of the initialization and update rules presented in the previous section. First, from lines 1 to 5, VSLAM initializes the version spaces associated to each action. Then, from lines 6 to 16, VSLAM processes all demonstrations in D in an online fashion and uses the induced learning examples to update the boundaries of the version spaces by applying the updates rules presented in theorems 2 and 3.

VSLAM inherits some important properties from its version spaces foundation. First, by Theorem 1, the boundaries computed by VSLAM capture exactly all models in $\mathcal{M}_D$, i.e., all the solutions to $\langle F, A, D\rangle$. Second, and in stark difference w.r.t. existing action model learning approaches (e.g., FAMA (Aineto, Celorrio, and Onaindia 2019) and SAM (Stern and Juba 2017)), VSLAM can detect when learning has concluded (i.e., it has learnt the true model) by checking convergence of the version space. A version space *converges* when only one consistent hypothesis remains, i.e., when both boundaries are singletons and contain the same hypothesis. Under our working assumption that the true model M^* follows the syntax of Definition 1, theorems 2 and 3 ensure that no consistent hypothesis is discarded and, therefore, it follows that if only one hypothesis remains it must match the true model M^*.

Corollary 1 (Convergence)**.** *Let $M^* = \langle F, A, \mathsf{pre}, \mathsf{eff}\rangle$ be the true model. If $\mathcal{L}_{\mathcal{H}^a_p, D_p} = \mathcal{U}_{\mathcal{H}^a_p, D_p} = \{hp\}$, then $hp = \mathsf{pre}(a)$, and if $\mathcal{L}_{\mathcal{H}^a_e, D_e} = \mathcal{U}_{\mathcal{H}^a_e, D_e} = \{he\}$, then $he = \mathsf{eff}(a)$.*

Lastly, VSLAM can also detect if our working assumptions are violated by checking for collapse of the version space. A version space *collapses* when it becomes empty, i.e., when no consistent hypothesis remains. This indicates that the learning examples are noisy or that the hypothesis space does not contain the true model.

4 Sound and Complete Action Models

In the previous section we have shown how to compute all solutions of an action model learning problem. Now, we put the focus on learnt models that guarantee some formal property of interest. In particular, we consider *soundness*[1] and *completeness*, and we show how to manipulate the computed version spaces to build models that guarantee such properties with respect to the true model.

4.1 Sound and Complete Action Models

A model M is *sound* w.r.t. another model M' if every transition of M is also a transition of M'. In contrast, a model M is *complete* w.r.t. another model M' if every transition of M' is also a transition of M.

Definition 8 (Soundness)**.** *Let M and M' be two action models, we say that M is* sound *w.r.t. M' iff $\mathcal{T}_M \subseteq \mathcal{T}_{M'}$.*

Definition 9 (Completeness)**.** *Let M and M' be two action models, we say that M is* complete *w.r.t. M' iff $\mathcal{T}_M \supseteq \mathcal{T}_{M'}$.*

[1] Soundness has previously been referred to as "safeness" (Stern and Juba 2017)

The main objective of this work is to compute action models that are sound or complete with respect to the true model M^*. The motivation for this is that the soundness and completeness properties carry over to solution plans computed with such models, allowing us operate with guarantees even when the true model remains unknown.

Theorem 4. *Let M^* be the true model, and M and M' a sound and a complete model w.r.t. M^*. It follows that:*

- *any solution plan for $P = \langle M, s_0, G\rangle$ is valid for $P^* = \langle M^*, s_0, G\rangle$, too, i.e., $\Pi(P) \subseteq \Pi(P^*)$.*
- *any solution plan for $P^* = \langle M^*, s_0, G\rangle$ is valid for $P' = \langle M', s_0, G\rangle$, too, i.e., $\Pi(P') \supseteq \Pi(P^*)$.*

Proof. A plan $\pi = (a_1, a_2, \ldots, a_n)$ in $\Pi(P)$ is one that induces an execution $(s_0, a_1, s_1, a_2, s_2, \ldots, a_n, s_n)$ such that every transition $\langle s_i, a_i, s_{i+1}\rangle$ belongs to $\mathcal{T}_M$. Since $\mathcal{T}_M \subseteq T_{M^*}$, all such transitions will also belong to T_{M^*} and, therefore, π belongs to $\Pi(P^*)$, too. By the same reasoning, any plan in $\Pi(P^*)$ will belong to $\Pi(P')$. □

An immediate corollary of the theorem is that $\mathcal{T}_M \subseteq \mathcal{T}_{M^*} \subseteq \mathcal{T}_{M'}$ and, consequently, $\Pi(P) \subseteq \Pi(P^*) \subseteq \Pi(P')$. In other words, a sound action model underapproximates the transition system of the true model and its solution plans, whereas a complete model overapproximates them. It is then natural to look for models that provide tighter approximations. The following subsections describe how to compute sound and complete models that approximate as tightly as possible the true model.

4.2 Building Sound Models from a Version Space

Let us start by summarizing our current progress and goals. We are presented with the fluents and actions of the true model M^* alongside a set of demonstrations, i.e., an action model learning problem $\langle F, A, D\rangle$, and our objective is to build a model that is sound w.r.t. M^*. From $\langle F, A, D\rangle$ we are able to derive all consistent models $\mathcal{M}_D$, e.g., by applying VSLAM, and we know that M^* is in $\mathcal{M}_D$. With our next theorem, we establish that the only way to build a model guaranteed to be sound w.r.t. M^* is by ensuring soundness w.r.t. all models in $\mathcal{M}_D$. Additionally, we demonstrate that we can use the version spaces computed by VSLAM, and more precisely their lower boundaries $\mathcal{L}_{\mathcal{H}^a_p, D_p}$ and $\mathcal{L}_{\mathcal{H}^a_e, D_e}$, to build a sound model w.r.t. M^*. Further, such a sound model is optimal, in the sense that no better underapproximation of M^* can be built while ensuring its soundness.

Theorem 5. *Let $\langle F, A, D\rangle$ be an action model learning problem and M^* be the true model. The action model $M = \langle F, A, \mathsf{pre}, \mathsf{eff}\rangle$ such that $\mathsf{pre}(a) \in \mathcal{L}_{\mathcal{H}^a_p, D_p}$ and $\mathsf{eff}(a) \in \mathcal{L}_{\mathcal{H}^a_e, D_e}$ for all $a \in A$ is sound w.r.t. M^* and there exists no other model M' ensuring soundness w.r.t. M^* such that $\mathcal{T}_{M'} \supset \mathcal{T}_M$.*

Proof. Let $\mathcal{T}^{\cap}_{\mathcal{M}_D} = \bigcap_{M' \in \mathcal{M}_D} \mathcal{T}_{M'}$ denote the intersection of all transition systems induced by consistent models $\mathcal{M}_D$. First, we show that the best sound model one can build is one that induces the transition system $\mathcal{T}^{\cap}_{\mathcal{M}_D}$, and then we prove that $\mathcal{T}_M = \mathcal{T}^{\cap}_{\mathcal{M}_D}$.

$\mathcal{T}^{\cap}_{\mathcal{M}_D}$ is trivially sound, i.e., $\mathcal{T}^{\cap}_{\mathcal{M}_D} \subseteq \mathcal{T}_{M^*}$ since the intersection of sets is always a subset of each set in the intersection. Next, observe that any superset $\mathcal{T}'$ of $\mathcal{T}^{\cap}_{\mathcal{M}_D}$ will contain at least one transition that does not belong to at least one model in $\mathcal{M}_D$; if such model happens to be the true model M^*, then $\mathcal{T}' \not\subseteq \mathcal{T}_{M^*}$ so any superset of $\mathcal{T}^{\cap}_{\mathcal{M}_D}$ would not be sound. Therefore, the best sound model that can be built is one that captures exactly all transitions in $\mathcal{T}^{\cap}_{\mathcal{M}_D}$.

(Soundness) Next, we prove that $\mathcal{T}_M \subseteq \mathcal{T}^{\cap}_{\mathcal{M}_D}$. By contradiction, assume that there exists a transition $t = \langle s, a, s' \rangle$ such that $t \in \mathcal{T}_M$ but $t \notin \mathcal{T}^{\cap}_{\mathcal{M}_D}$. For $t \notin \mathcal{T}^{\cap}_{\mathcal{M}_D}$ to be true, there must exist a model $M' = \langle F, A, \mathsf{pre}', \mathsf{eff}' \rangle$ in $\mathcal{M}_D$ such that either (1) a is not applicable in s, or (2) its execution results in a state $s'' \neq s'$. By construction, it holds that $\mathsf{pre}(a)' \subseteq \mathsf{pre}(a)$. Consequently, if $\mathsf{pre}(a) \subseteq s$ then $\mathsf{pre}(a)' \subseteq s$ and case (1) does not hold. For case (2), we know that $\mathsf{eff}'(a) \setminus \mathsf{eff}(a) \subseteq \mathsf{pre}(a)$ so it follows that $\mathsf{eff}'(a) \setminus \mathsf{eff}(a) \subseteq s$. Meaning that, any difference between $\mathsf{eff}(a)$ and $\mathsf{eff}'(a)$ is already part of state s and the execution of a cannot result in different states.

(Optimality) We prove $\mathcal{T}_M \supseteq \mathcal{T}^{\cap}_{\mathcal{M}_D}$. For this, simply observe that, by Theorem 1, $M \in \mathcal{M}_D$. Therefore, $\mathcal{T}^{\cap}_{\mathcal{M}_D}$ is a subset of any of its intersecting sets including $\mathcal{T}_M$. Finally, since $\mathcal{T}_M \subseteq \mathcal{T}^{\cap}_{\mathcal{M}_D}$ and $\mathcal{T}_M \supseteq \mathcal{T}^{\cap}_{\mathcal{M}_D}$, it follows that $\mathcal{T}_M = \mathcal{T}^{\cap}_{\mathcal{M}_D}$. □

4.3 Building Complete Models from a Version Space

This time we move the focus to building models that are complete w.r.t. the true model M^*. Analogously to the sound case, we will see that to guarantee completeness w.r.t. M^*, we have to be complete w.r.t. all models in $\mathcal{M}_D$. However, note that such a model must be able to produce any of the transitions generated by any of the consistent models in $\mathcal{M}_D$. Intuitively, the actions of a complete model should be applicable in any state where it is applicable according to a consistent model, and its execution should generate all possible post-states generated under any consistent model. It is easy to see that working under the limits of Definition 1 leads to two problems. First, a model that produces multiple possible post-states is, by definition, non-deterministic. And second, using conjunctive preconditions may easily lead to a weak precondition that accepts more pre-states than necessary. Therefore, in order to build a complete model, we target a more expressive planning model that accommodates disjunctive preconditions and non-deterministic effects.

Definition 10 (Non-deterministic Action Model)**.** *A non-deterministic action model is a tuple* $M = \langle F, A, \mathsf{Pre}, \mathsf{Eff} \rangle$ *where:*

- F *and* A *are finite sets of fluents and actions as given in Definition* 1.
- $\mathsf{Pre} : A \to 2^{2^L}$ *defines the* set of preconditions $\mathsf{Pre}(a) \subseteq 2^L$ *of each action* $a \in A$.
- $\mathsf{Eff} : A \to 2^{2^L}$ *defines the* set of effects $\mathsf{Eff}(a) \subseteq 2^L$ *of each action* $a \in A$.

The main difference with respect to Definition 1 is that here each action is associated to a set of preconditions and a set of effects. A non-deterministic action model $M = \langle F, A, \mathsf{Pre}, \mathsf{Eff} \rangle$ induces the transition system $\mathcal{T}_M \subseteq S \times A \times S$ consisting of all transitions $\langle s, a, s' \rangle$ that satisfy $\exists p \in \mathsf{Pre}(a) : p \subseteq s$ and $\exists e \in \mathsf{Eff}(a)$ such that $s' = (s \setminus \overline{e}) \cup e$.

Using this new formulation, we leverage once again the version spaces computed by VSLAM to build a complete model w.r.t. the true model M^* that constitutes the tightest overapproximation of M^* that can be built without compromising completeness.

Theorem 6. *Let* $\langle F, A, D \rangle$ *be an action model learning problem and* M^* *be the true model. The non-deterministic action model* $M = \langle F, A, \mathsf{Pre}, \mathsf{Eff} \rangle$ *such that* $\mathsf{Pre}(a) = \mathcal{U}_{\mathcal{H}^a_p, D_p}$ *and* $\mathsf{Eff}(a) = \mathcal{V}_{\mathcal{H}^a_e, D_e}$ *for all* $a \in A$ *is complete w.r.t.* M^* *and there exists no other model* M' *ensuring completeness w.r.t.* M^* *such that* $\mathcal{T}_{M'} \subset \mathcal{T}_M$.

Proof. Let $\mathcal{T}^{\cup}_{\mathcal{M}_D} = \bigcup_{M' \in \mathcal{M}_D} \mathcal{T}_{M'}$ denote the union of all transition systems induced by consistent models $\mathcal{M}_D$. First, we show that the best complete model one can build is one that induces the transition system $\mathcal{T}^{\cup}_{\mathcal{M}_D}$, and then we prove that $\mathcal{T}_M = \mathcal{T}^{\cup}_{\mathcal{M}_D}$.

$\mathcal{T}^{\cup}_{\mathcal{M}_D}$ is trivially complete, i.e., $\mathcal{T}^{\cup}_{\mathcal{M}_D} \supseteq \mathcal{T}_{M^*}$ since the union of sets is always a superset of each set in the union. Next, observe that any subset $\mathcal{T}'$ of $\mathcal{T}^{\cup}_{\mathcal{M}_D}$ will be missing at least one transition that belongs to at least one model in $\mathcal{M}_D$; if such model happens to be the true model M^*, then $\mathcal{T}' \not\supseteq \mathcal{T}_{M^*}$ so any subset of $\mathcal{T}^{\cup}_{\mathcal{M}_D}$ would not be complete. Therefore, the best complete model that can be built is one that induces exactly $\mathcal{T}^{\cup}_{\mathcal{M}_D}$.

(Completeness) We start by proving that $\mathcal{T}_M \supseteq \mathcal{T}^{\cup}_{\mathcal{M}_D}$. By contradiction, assume that there exists a transition $t = \langle s, a, s' \rangle$ such that $t \in \mathcal{T}^{\cup}_{\mathcal{M}_D}$ but $t \notin \mathcal{T}_M$. Then, $\mathcal{M}_D$ must include a model $M' = \langle F, A, \mathsf{pre}', \mathsf{eff}' \rangle$ such that $\mathsf{pre}'(a) \subseteq s$ and $s' = (s \setminus \overline{\mathsf{eff}'(a)}) \cup \mathsf{eff}'(a)$, and either (1) $\forall p \in \mathsf{Pre}(a) : p \not\subseteq s$, or (2) $\forall e \in \mathsf{Eff}(a) : s' \neq (s \setminus \overline{e}) \cup e$. Since $\mathsf{Pre}(a) = \mathcal{U}_{\mathcal{H}^a_p, D_p}$ and $\mathsf{pre}'(a) \in \mathcal{V}_{\mathcal{H}^a_p, D_p}$, there must exists a $p \in \mathsf{Pre}(a)$ such that $p \subseteq \mathsf{pre}'(a)$, and if $\mathsf{pre}'(a) \subseteq s$ is true then so must be $p \subseteq s$. This falsifies case (1). For case (2), observe that $\mathsf{eff}'(a) \in \mathcal{V}_{\mathcal{H}^a_e, D_e}$ and $\mathsf{Eff}(a) = \mathcal{V}_{\mathcal{H}^a_e, D_e}$. Therefore, $\mathsf{eff}'(a) \in \mathsf{Eff}(a)$ and case (2) cannot be true.

(Optimality) We prove $\mathcal{T}_M \subseteq \mathcal{T}^{\cup}_{\mathcal{M}_D}$. By contradiction, assume that $t \in \mathcal{T}_M$ but $t \notin \mathcal{T}^{\cup}_{\mathcal{M}_D}$. Then, it must be true that $\exists p \in \mathsf{Pre}(a) : p \subseteq s$ and $\exists e \in \mathsf{Eff}(a) : s' = (s \setminus \overline{e}) \cup e$ and $\mathcal{M}_D$ cannot contain a model $M' = \langle F, A, \mathsf{pre}', \mathsf{eff}' \rangle$ such that $\mathsf{pre}'(a) \subseteq s$ and $s' = (s \setminus \overline{\mathsf{eff}'(a)}) \cup \mathsf{eff}'(a)$. However, this cannot be true since $\mathsf{Pre}(a) \subseteq \mathcal{V}_{\mathcal{H}^a_p, D}$ and $\mathsf{Eff}(a) = \mathcal{V}_{\mathcal{H}^a_e, D}$, so $\mathcal{M}_D$ contains a model M' with $\mathsf{pre}'(a) = p$ and $\mathsf{eff}'(a) = e$. Finally, since $\mathcal{T}_M \supseteq \mathcal{T}^{\cup}_{\mathcal{M}_D}$ and $\mathcal{T}_M \subseteq \mathcal{T}^{\cup}_{\mathcal{M}_D}$ are true, we have that $\mathcal{T}_M = \mathcal{T}^{\cup}_{\mathcal{M}_D}$. □

5 Learning Lifted Representations

Action models are typically represented with a lifted representation using declarative languages such as PDDL (Fox and Long 2003). We briefly explain how our approach copes with such a representation.

For starters, the learning problem is presented in a lifted manner, i.e., $\langle F^{\uparrow}, A^{\uparrow}, D \rangle$, and a solution is a lifted action model $M^{\uparrow} = \langle F^{\uparrow}, A^{\uparrow}, \text{pre}^{\uparrow}, \text{eff}^{\uparrow} \rangle$. $F^{\uparrow}$ and $A^{\uparrow}$ are, respectively, the *lifted fluents* and *lifted actions*, both of the form $name(?v_1 \ldots ?v_n)$ where $name$ is an identifier and $(?v_1 \ldots ?v_n)$ is a list of parameters. As an example, in the BLOCKS domain (Slaney and Thiébaux 2001), the lifted fluents are $F^{\uparrow}$ = {`hand-empty()`, `holding(?b1)`, `on-table(?b1)`, `clear(?b1)`, `on(?b1 ?b2)`}, and the lifted actions are $A^{\uparrow}$ = {`pick-up(?b)`, `put-down(?b)`, `stack(?top ?bot)`, `unstack(?top ?bot)`}. The precondition $\text{pre}^{\uparrow}(a^{\uparrow})$ and effect $\text{eff}^{\uparrow}(a^{\uparrow})$ of a lifted action $a^{\uparrow} \in A^{\uparrow}$ are defined as a set of literals whose parameters are bound to the parameters of $a^{\uparrow}$. For instance, the precondition of `stack` is $\text{pre}^{\uparrow}$(`stack`) = {`holding(?top)`, `clear(?bot)`} and its effect is $\text{eff}^{\uparrow}$(`stack`) = {¬`holding(?top)`, ¬`clear(?bot)`, `hand-empty()`, `clear(?top)`, `on(?top ?bot)`}.

Since preconditions and effects are parameter-bound literals, the hypothesis space is no longer 2^L for all actions and, therefore, potentially smaller. Indeed, each lifted action will have a different hypothesis space depending on its parameters. In BLOCKS, the parameter-bound literals for `pick-up(?b)` and `put-down(?b)` are {`hand-empty()`, `holding(?b)`, `clear(?b)`, `on-table(?b)`}, while for `stack(?top ?bot)` and `unstack(?top ?bot)`, they are {`hand-empty()`, `holding(?top)`, `holding(?bot)`, `on-table(?top)`, `on-table(?bot)`, `clear(?top)`, `clear(?bot)`, `on(?top ?bot)`, `on(?bot ?top)`} Similarly, learning examples must also be given as sets of parameter-bound literals. To achieve this, we "lift" the demonstrations by substituting each object in a literal by the action parameter it is bound to. Doing so, we obtain demonstrations like d = ⟨{`on-table(?b)`, `clear(?b)`, `hand-empty()`, ¬`holding(?b)`}, `pick-up(?b)`, {¬`on-table(?b)`, ¬`clear(?b)`, ¬`hand-empty()`, `holding(?b)`}⟩ where both pre-state and post-state are sets of parameter-bound literals. Once the hypothesis space and learning examples are fixed, we can apply our update rules off-the-shelf to learn a version space in this lifted representation and construct lifted models following the previous defined procedures. This procedure is very similar to SAM's approach, described by Juba, Le, and Stern (2021).

6 Experimental Evaluation

We compare VSLAM against SAM (Juba, Le, and Stern 2021) and FAMA (Aineto, Celorrio, and Onaindia 2019), evaluating the usability of the models computed by each system over an ongoing learning process. That is, we evaluate the performances of the sound model by SAM, the consistent model by FAMA, and the sound and the complete models by VSLAM. Note that both SAM and VSLAM return the same sound model. Our objective is to understand when and to what extent the complete models that we highlight in this work are beneficial. The code and benchmarks can be found at https://github.com/daineto/VSLAM/tree/KR2024.

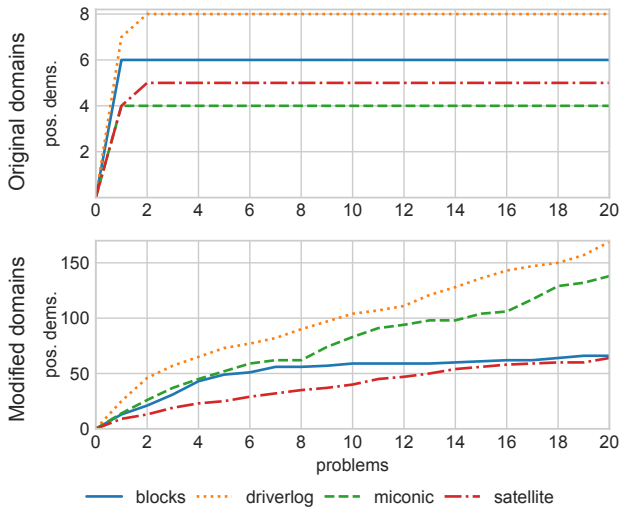

Figure 2: Distinct positive demonstrations collected with the original (top) and modified domains (bottom).

DOMAIN	A	F	POS	NEG
BLOCKS	4 (3)	5 (2)	66	882
DRIVERLOG	6 (8)	6 (2)	169	903
MICONIC	4 (4)	8 (2)	138	381
SATELLITE	5 (8)	8 (2)	64	166

Table 1: Domains, features and collected demonstrations.

6.1 Benchmarks

We focus on 4 domains from the International Planning Competition (McDermott 2000): BLOCKS, SATELLITE, MICONIC and DRIVERLOG. We collect 20 random problems for each domain using available generators (Seipp, Torralba, and Hoffmann 2022), and solve them with LAMA (Richter and Westphal 2010). We collect positive demonstrations from transitions of the solution plans. Negative ones are generated by randomly picking non applicable actions throughout the states traversed by solution plans.

We slightly extend the planning domains by adding extra parameters to the actions. This gives us a larger hypothesis space, and therefore a slower learning process that can be used to better appreciate the behavior of the different techniques. As Figure 2 shows, this simple modification increases by 10 to 30 times the number of *distinct* positive demonstrations and reduces the overlap between plans. Note that, with the original domains, the number of demonstrations saturates after the second problem, making all techniques behave the same. Table 1 details the features of our benchmarks: columns 2 and 3 displays the number of lifted actions and fluents with their maximal number of parameters; columns 4 and 5 report on the number of positive (POS) and negative (NEG) demonstrations collected.

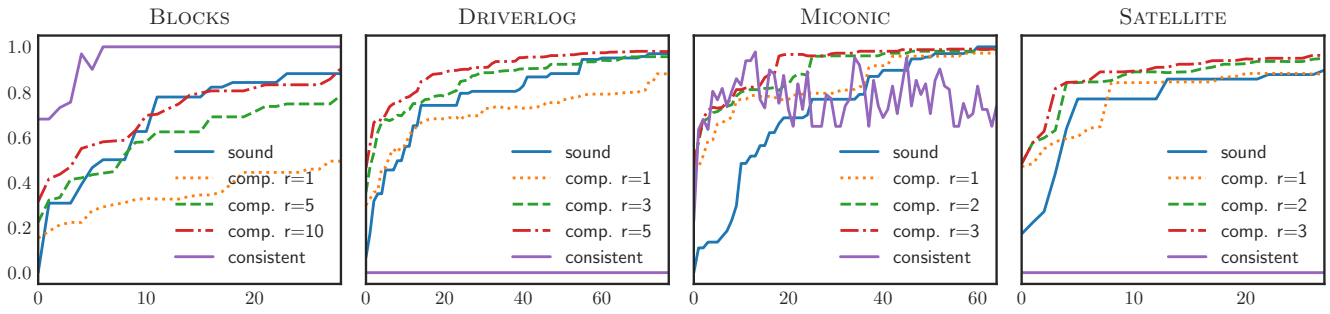

Figure 3: F1-score (y-axis) of the sound, complete, and consistent action models as the training demonstrations increase (x-axis).

6.2 Evaluation criteria

Following previous works on action model learning (Lamanna et al. 2021; Aineto, Celorrio, and Onaindia 2019), we use the *f1-score*, the harmonic mean of *precision* and *recall*. We interpret these metrics over a test set of demonstrations by having the learnt action model label them as positive or negative with respect to its transition system. *Precision* degrades with *false positives*, i.e., when the model accepts a negative demonstration as part of its transition system. Conversely, *recall* degrades with *false negatives*, i.e., when the model rejects a positive demonstration. Indeed, a sound action model will always have perfect *precision* but lower *recall*, meaning that it accepts few transitions but all of them are correct. In contrast, a complete model will have perfect recall yet low precision, since it will often accept transitions that do not belong to the true model. The *f1-score* provides us with a very good proxy for understanding the usability of our models. We split the collected demonstrations, using half for learning and half for testing, and measure the f1-score of the learnt models as more demonstrations are processed. The complete model is evaluated simulating scenarios where negative demonstrations are seen at different rates. We do this by using a ratio r of negative to positive demonstrations. For instance, a ratio $r = 2$ indicates that the demonstrations set D contains 2 negative demonstrations for every positive one. With this setting, we aim at understanding the impact of the distribution of our dataset.

6.3 Results

Figure 3 presents our results; the x-axis reports on the number of growing positive demonstrations. For each positive demonstration, the learning systems also see r negative ones. We represent such an information with different curves. We observe that the sound action model performs better in more imbalanced domains (in terms of positive and negative demonstrations) such as BLOCKS. Instead, the complete model seems to be effective over more balanced domains like MICONIC and SATELLITE. This comes with no surprise since more imbalanced distributions correlate to stricter preconditions and such models are closer (in the hypothesis space) to the sound model. The opposite holds true for the complete model. Generally, the complete model has the advantage in the earlier stages of the learning process, but in our experiment is later outperformed by the sound model after more demonstrations have been processed. VSLAM approach would take the best of sound and complete learning approaches. Therefore, if the learning is at the early stage, it is with the complete model that it achieves the highest effectiveness. Regarding consistent models, we see FAMA dominating in BLOCKS. This is because FAMA follows a restricted definition of action model, e.g., no negated literals in the precondition, so, after a few demonstrations, the only consistent solution becomes the true model. The results in MICONIC are more interesting and show one of the weaknesses of using an approach like FAMA. That is, there is no guarantee that using more demonstrations will lead to a better model. Indeed, we see the performance of the consistent model sway, while the sound and complete models always improve with more demonstrations. FAMA runs out of memory in the other two domains, so it did not produce any consistent model. Overall, the sound and complete models exhibit complementary performance depending on the domain characteristics, how far into the learning we are, and how accessible positive and negative demonstrations are. This is a compelling argument to learn both together, rather than only the sound one (SAM).

7 Related Work

Research on action model learning has produced a wide variety of sophisticated learning approaches – different surveys can be found were written by Jiménez et al. (2012), Arora et al. (2018), and Aineto, Jiménez, and Onaindia (2022). Starting with the pioneering works of ARMS (Yang, Wu, and Jiang 2007) and SLAF (Amir and Chang 2008), research in this field has been quite prolific. Approaches like LAMP (Zhuo et al. 2010) investigated the learning of more expressive action models, while others like FAMA (Aineto, Celorrio, and Onaindia 2019) and AMAN (Zhuo and Kambhampati 2013) focused on learning from incomplete or noisy demonstrations. We can even find approaches that actively seek the demonstrations that will help them learn faster (Lamanna et al. 2021; Verma, Marpally, and Srivastava 2021).

Broadly, most learning approaches can be classified, according to their notion of solution, into those that compute a model consistent with the demonstrations (Cresswell, McCluskey, and West 2013; Aineto, Celorrio, and Onaindia

2019; Bonet and Geffner 2020), or those that target the action model that maximizes some objective or fitness function (Yang, Wu, and Jiang 2007; Kučera and Barták 2018; Zhuo et al. 2010; Zhuo and Kambhampati 2013; Mourao et al. 2012). While we follow the former interpretation, our approach is one of the few, alongside SLAF (Amir and Chang 2008), which is able to compute all solutions to the problem. SLAF computes a CNF formula representing all possible transitions that can be regarded as a form of version space. However, this formula does not offer the compactness of our boundaries nor can be easily manipulated and, indeed, the only way to extract from it a concrete solution model is using a SAT solver. Unlike us, SLAF handles partial observability, a feature that we expect to support in the future following similar extensions for version spaces.

The approach that we regard as the closest to our own is SAM (Stern and Juba 2017) for its focus on safe (sound) models, learning setting and, interestingly, a foundational connection. The authors of SAM link their approach to Valiant's elimination algorithm (Valiant 1984), an algorithm that Mitchell himself describes as the subproblem of computing the $\mathcal{L}$ boundary in version space learning (Mitchell 1982). We revive this connection in the action model learning setting. Indeed, SAM focuses on one extreme of the spectrum of solution models, those guaranteeing soundness, which are tied to the $\mathcal{L}$ boundary. On the other extreme we find the complete models that we highlight in this work.

8 Conclusions and Future Work

This paper proposes an approach to learning action models from first principles. We do so by exploiting version spaces to a great extent. One of the main benefits of our approach is the ability to learn in an integrated and comprehensive way sound models as by Stern and Juba (2017) together with complete models, providing therefore a great deal of flexibility. Indeed, our framework enables an agent not only to learn from positive demonstrations but also from failures. Empirically, we observed that with this facility in place, an agent can start learning something useful for reasoning already with a few number of examples.

We have a number of future works on our way. First, we want to tackle the learning of more expressive action models. SAM has been extended along several dimensions, for instance planning with numeric information (Mordoch, Juba, and Stern 2023) and planning under partial observability (Le, Juba, and Stern 2024). Theoretically speaking, provided a formal characterisation of the given language, and therefore of the given induced hypothesis spaces, we argue that the challenge for VSLAM is about finding the right set of update rules. In this direction, we think that looking into *version space algebra* (Lau et al. 2003) which supports more expressive hypothesis spaces would equip us with the necessary tools. Second, we want to study VSLAM in the context of optimal planning. Note, indeed, that the optimal solution cost from a complete model lowers bound the true solution cost. We can use this to assess the quality of sound solutions.

Acknowledgments

Diego Aineto and Enrico Scala were supported by Climate Change AI project (No. IG-2023-174).

References

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

Aineto, D.; Jiménez, S.; and Onaindia, E. 2022. A comprehensive framework for learning declarative action models. *Journal of Artificial Intelligence Research* 74:1091–1123.

Amir, E., and Chang, A. 2008. Learning partially observable deterministic action models. *Journal of Artificial Intelligence Research* 33:349–402.

Arora, A.; Fiorino, H.; Pellier, D.; Métivier, M.; and Pesty, S. 2018. A review of learning planning action models. *The Knowledge Engineering Review* 33:e20.

Bonet, B., and Geffner, H. 2020. Learning first-order symbolic representations for planning from the structure of the state space. In *ECAI*, 2322–2329.

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.

Fox, M., and Long, D. 2003. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: theory and practice*. Elsevier.

Jiménez, S.; De La Rosa, T.; Fernández, S.; Fernández, F.; and Borrajo, D. 2012. A review of machine learning for automated planning. *The Knowledge Engineering Review* 27(4):433–467.

Juba, B., and Stern, R. 2022. Learning probably approximately complete and safe action models for stochastic worlds. In *AAAI*, 9795–9804.

Juba, B.; Le, H. S.; and Stern, R. 2021. Safe learning of lifted action models. In *KR*, 379–389.

Kambhampati, S. 2007. Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models. In *AAAI*, 1601–1605.

Kučera, J., and Barták, R. 2018. Louga: learning planning operators using genetic algorithms. In *KEPS*, 124–138.

Lamanna, L.; Saetti, A.; Serafini, L.; Gerevini, A.; and Traverso, P. 2021. Online learning of action models for PDDL planning. In *IJCAI*, 4112–4118.

Lau, T.; Wolfman, S. A.; Domingos, P.; and Weld, D. S. 2003. Programming by demonstration using version space algebra. *Machine Learning* 53:111–156.

Lau, T. A.; Domingos, P. M.; and Weld, D. S. 2000. Version space algebra and its application to programming by demonstration. In *ICML*, 527–534.

Le, H. S.; Juba, B.; and Stern, R. 2024. Learning safe action models with partial observability. In *AAAI*, 20159–20167.

McDermott, D., et al. 1998. The planning domain definition language manual. Technical report, Technical Report 1165, Yale Computer Science, 1998.(CVC Report 98-003).

McDermott, D. M. 2000. The 1998 AI planning systems competition. *AI magazine* 21(2):35–35.

Mitchell, T. M. 1982. Generalization as search. *Artificial intelligence* 18(2):203–226.

Mordoch, A.; Juba, B.; and Stern, R. 2023. Learning safe numeric action models. In *AAAI*, 12079–12086.

Mourao, K.; Zettlemoyer, L. S.; Petrick, R.; and Steedman, M. 2012. Learning strips operators from noisy and incomplete observations. *arXiv preprint arXiv:1210.4889*.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Seipp, J.; Torralba, Á.; and Hoffmann, J. 2022. PDDL generators. https://doi.org/10.5281/zenodo.6382173.

Slaney, J., and Thiébaux, S. 2001. Blocks world revisited. *Artificial Intelligence* 125(1-2):119–153.

Stern, R., and Juba, B. 2017. Efficient, safe, and probably approximately complete learning of action models. In *AAAI*, 4405–4411.

Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM* 27(11):1134–1142.

Verma, P.; Marpally, S. R.; and Srivastava, S. 2021. Asking the right questions: Learning interpretable action models through query answering. In *AAAI*, 12024–12033.

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.

Zhuo, H. H., and Kambhampati, S. 2013. Action-model acquisition from noisy plan traces. In *IJCAI*, 2444–2450.

Zhuo, H. H.; Yang, Q.; Hu, D. H.; and Li, L. 2010. Learning complex action models with quantifiers and logical implications. *Artificial Intelligence* 174(18):1540–1569.