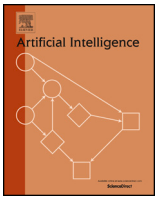
Artificial Intelligence 339 (2025) 104256

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Lifted action models learning from partial traces

Leonardo Lamanna [a,*], Luciano Serafini [a], Alessandro Saetti [b], Alfonso Emilio Gerevini [b], Paolo Traverso [a]

[a] Center for Augmented Intelligence, Fondazione Bruno Kessler, Trento, Italy
[b] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy

ARTICLE INFO

Keywords:
Planning
Learning
Action model
Partial observability
Plan traces

ABSTRACT

For applying symbolic planning, there is the necessity of providing the specification of a symbolic action model, which is usually manually specified by a domain expert. However, such an encoding may be faulty due to either human errors or lack of domain knowledge. Therefore, learning the symbolic action model in an automated way has been widely adopted as an alternative to its manual specification. In this paper, we focus on the problem of learning action models offline, from an input set of partially observable plan traces. In particular, we propose an approach to: *(i)* augment the observability of a given plan trace by applying predefined logical rules; *(ii)* learn the preconditions and effects of each action in a plan trace from partial observations before and after the action execution. We formally prove that our approach learns action models with fundamental theoretical properties, not provided by other methods. We experimentally show that our approach outperforms a state-of-the-art method on a large set of existing benchmark domains. Furthermore, we compare the effectiveness of the learned action models for solving planning problems and show that the action models learned by our approach are much more effective w.r.t. a state-of-the-art method.[1]

1. Introduction

Automated planning techniques require the specification of planning domains through action models, i.e., a set of preconditions and a set of effects for each action. However, the manual specification of an action model is often an inaccurate, time-consuming, and error-prone task. The automated learning of action models is widely recognised as a key and compelling challenge to overcome these difficulties. To ensure the generality and re-usability of an action model, preconditions and effects are specified with lifted schemes, which are independent from the specific set of objects involved in each planning domain. Several works have addressed the task of learning action models and have provided important results from different perspectives and according to different assumptions, see, e.g., [1–3,9,12,26,27,29,30,32,23]. A common assumption is to learn action models from an input set of *traces* that provide information about a sequence of successfully executed actions and information about the states before and after each execution. Some works address the problem of learning action models from *partial traces*, i.e., traces in which the value of some state variable is unknown, and some of the executed actions might be missing. Though this problem is intuitively clear, its formal specification is far

* Corresponding author.
E-mail address: llamanna@fbk.eu (L. Lamanna).

[1] The source code is available in [21].

https://doi.org/10.1016/j.artint.2024.104256
Received 22 December 2023; Received in revised form 11 November 2024; Accepted 11 November 2024

Available online 15 November 2024

from being obvious, and it has not been completely addressed by previous works. Some important aspects are still missing, including the criteria for determining the output action model among different possibilities, and the criteria for evaluating the efficacy of the learned model.

Given a set of partial traces, the problem is to learn an action model that can generate a trace consistent with the input partial trace. However, in general such an action model is not unique. Indeed, two action models could differently complete the partial trace with the missing actions and the missing values for state variables. Therefore, we need a criterion to select one model among all the models that could generate a completion of the given partial trace. Concerning the criteria for evaluating the quality of the learned model, several works have adopted the standard measures of precision and recall of the preconditions and effects. Another important issue to consider is how the model can be used in solving planning problems, finding executable and valid plans w.r.t. the ground truth model. In this paper, we provide the following contributions:

- We propose to consider the *minimal action models* among all the possible models that are capable of generating the input set of partial traces. To characterize what a minimal action model is for a partial trace, we define a partial order on the set of action models. Intuitively a minimal model for a partial trace is the "smallest" model consistent with the trace, i.e., the model that contains only the necessary effects to justify the changes observed in the trace. We will see that a partial trace can have more than one minimal model leaving a certain room for uncertainty in the action model learning.
- We provide a sound and complete set of *completion rules* for deriving *all the minimal models* of a set of partially observable traces. We prove that the minimal models of a set of partial traces can be derived by the iterative application of these rules until the partial traces and the learned model are not changed. From the set of minimal models, we can extract all the preconditions of the actions that are possibly part of the action model and all the effects that are necessarily part of the model.
- We define an algorithm, called $\mathsf{OffLAM}_{\mathsf{PT}}$, which iteratively applies the set of completion rules, and which is guaranteed to terminate and to compute all the minimal models of a set of partial traces.
- We evaluate $\mathsf{OffLAM}_{\mathsf{PT}}$ on a benchmark of 18 planning domains adopted in the past International Planning Competitions (IPCs). We show that our algorithm outperforms a state-of-the-art system in almost all the tasks using the standard evaluation of precision and recall on preconditions and effects. We also evaluate the efficacy of the action model learned by $\mathsf{OffLAM}_{\mathsf{PT}}$ for solving problems and computing valid solutions. This evaluation shows that, even in terms of efficacy, the proposed algorithm outperforms a state-of-the-art approach in all the cases.

The paper is structured as follows. Section 2 summarizes the existing work on the action model learning; Section 3 gives the necessary background; Section 4 describes the completion rules for the derivation of all the minimal action models consistent with a set of traces with partially observable states and fully observable actions; Section 5 shows that the iterative application of the completion rules meets several fundamental theoretical properties about the computation of the minimal models; Section 6 presents algorithm $\mathsf{OffLAM}_{\mathsf{PT}}$; Section 7 introduces an additional rule to treat traces having partially observable actions; Section 8 presents the results of our experimental study; finally, Section 9 gives the conclusions and mentions the future work.

2. Related work

The task of learning an action model can be formulated for different sources of input (e.g., labelled directed graphs encoding the structure of the search space for one or more problems [8], or camera images observed during the execution of one or more plans [4–6]). This work focuses on the task of learning action models from traces containing a symbolic representation of the traversed states and executed actions. Moreover, the approaches to action model learning can be classified into two main classes, namely: *offline* and *online*. Offline approaches assume that a set of traces is provided before learning the model; on the contrary, online approaches determine the action to execute while learning the model. Specifically, the determined action is executed by either a real agent or a simulator, which returns failure or a set of observations on the state of the world obtained from the execution of the action. Some prominent approaches to online action model learning are [10,22,20]. These approaches progressively learn an action model by using an approximation of the ground truth model to compute plans that reach several informative states. The observations derived from the execution of the computed plans allow these approaches to refine the model utilised for planning. In this paper, we propose an offline approach. For this reason, we restrict the discussion on the related work to the approaches that fall in this category.

Offline approaches address the problem of model learning with different assumptions on the observability of states and actions in the input traces. Stern and Juba [28] propose a methodology for learning grounded action models from a set of *fully observable* traces. Notably, the learned action model is guaranteed to be safe, i.e., the plans computed using the learned action model are sound. The methodology has been extended into an approach, SAM, for learning lifted action models [18]. The learned action models are still safe, but their computation requires an additional assumption called "injective action binding", i.e., the assumption that every action parameter is replaced with a different object. A follow-up work has extended SAM to deal with ground actions that do not meet the injective action binding assumption [18]. Further works have extended SAM for learning probabilistic action models, and action models with numerical variables [19,25]. All these works assume input traces with fully observable states and actions, while our approach deals with partially observable traces.

Bachor and Behnke [7] study the complexity of the action model learning problem from a set of fully observable traces. They introduce a partial ordering between action models to give a minimality notion for action models so that a model smaller than a safe model is safe. We give a similar notion, with the difference that we do not distinguish between positive and negative effects. As a consequence, our approach can derive unsafe action models.

In the rest of this section, we focus on learning from *partially observable traces*. Action Relation Modeling System (ARMS) is one of the first approaches proposed for learning action models, encoded using the STRIPS language, from traces with partially observable states and fully observable actions [30]. For learning action models, ARMS builds a weighted propositional satisfiability problem (weighted MAX-SAT), and solves it using a MAX-SAT solver. The constraints of the SAT problem are extracted from states and actions in the input traces. ARMS guarantees that the learned action models are approximately correct and concise; the notion of correctness and conciseness are given according to some metrics used for the evaluation (i.e. rates of error and redundancy).

Another approach learning from the same kind of traces as ARMS is EPI-SAM [24]. EPI-SAM deals with the uncertainty on which action in a sequence has a certain effect by using disjunctions in the definition of the initial state and conditional effects. Specifically, it outputs a conformant planning problem such that a strong plan solving such a problem is guaranteed to be sound for the ground truth model. Differently from EPI-SAM, $\text{OffLAM}_{\text{PT}}$ can learn from sets of traces having partially observable actions and outputs an action model that can be used by classical planners.

CAMA learns action models from a set of traces and a noisy initial action model derived by human annotations [31]. Similarly to ARMS, CAMA encodes the learning problem as a weighted MAX-SAT problem constructed from a set of soft constraints based on the human annotations and the input traces. Afterward, CAMA solves the problem using a weighted MAX-SAT solver and converts the solution of the solver to an action model.

Learning Object Centred Models (LOCM) learns an action model from a set of traces with fully observable actions and unobservable states. It groups the problem objects in sets such that each object behaves in the same way as any other object in its set [12]. Afterwards, it assembles the transition behaviour of each group of objects, which is determined by a finite-state machine, assembles the coordination between transitions of different groups of objects, and the relationships between objects of different groups. Notably, LOCM does not require input knowledge about the planning domain to be learned (e.g., the set of predicate names), but it does not deal with static knowledge (e.g., precondition propositions that are not in the set of effects of any action), which needs to be explicitly specified. A follow-up work extends LOCM for learning a wider range of domains by allowing an object's behaviour to be represented by more than one finite-state machine [11]. Finally, LOP extends LOCM for dealing with static knowledge, but requires as input a set of *optimal* plan traces [15].

As far as we know, FAMA is the only action model learner working with the same kind of input traces as $\text{OffLAM}_{\text{PT}}$. Indeed, similarly to $\text{OffLAM}_{\text{PT}}$, FAMA can learn action models from a set of traces with both partially observable states and actions. It learns action models by transforming the learning task into a classical planning task. FAMA works with different kinds of inputs, from a set of complete traces to partial traces containing only the initial and final states (without intermediate actions or states). With respect to FAMA, $\text{OffLAM}_{\text{PT}}$ guarantees to compute all the "smallest" models consistent with the input traces. Moreover, we will show that in practice the models computed using our approach are better than those computed by FAMA in many ways.

The planning community has also proposed approaches for learning action models encoded into a language more expressive than STRIPS. For instance, LAMP outputs action models having quantifiers and logical implications [33]; SLAF outputs models with action effects having universal quantifiers [3]. Our work focuses on learning action models encoded into the PDDL language by using the STRIPS assumption for action preconditions and effects.

3. Preliminaries

Let $\mathcal{P}$ be a set of predicate names with associated arity, of a first order language and $\mathcal{O}$ be a finite set of operator names with associated arity. Predicates and operators of arity n are called n-ary predicates and n-ary operators. Given a set $\mathcal{X}$ of n distinct symbols (constants or variables), let $\mathcal{P}(\mathcal{X})$ be the set of atomic formulas $p(x_1, \dots, x_m)$ obtained by applying the m-ary predicate $p \in \mathcal{P}$ to any m-tuple of symbols $\boldsymbol{x} \in \mathcal{X}^m$. For instance, if $\mathcal{P}$ contains the single binary predicate on, and $\mathcal{X} = \{x_1, x_2, x_3\}$ then $\mathcal{P}(\mathcal{X})$ is the set of 9 atoms $\{\texttt{on}(x_i, x_j) \mid 1 \leq i, j \leq 3\}$. Moreover, $\mathcal{L}it(\mathcal{P}(\mathcal{X}))$ denotes the set of literals with atoms in $\mathcal{P}(\mathcal{X})$.

Definition 1 *(Action schema).* An *action schema* for an n-ary operator name $op \in \mathcal{O}$ on the set of predicates $\mathcal{P}$ is a tuple $\langle \mathsf{par}(op), \mathsf{pre}(op), \mathtt{eff}(op) \rangle$, where $\mathsf{par}(op)$ is a tuple of n distinct variables $\boldsymbol{x} = (x_1, \dots, x_n)$, $\mathsf{pre}(op)$ is a set of atoms in $\mathcal{P}(\mathcal{X})$ and $\mathtt{eff}(op)$ is a consistent set of literals in $\mathcal{L}it(\mathcal{P}(\mathcal{X}))$ with $\mathcal{X} = \{x_1, \dots, x_n\}$.

$\mathsf{pre} : \mathcal{O} \to 2^{\mathcal{P}(\mathcal{X})}$ and $\mathtt{eff} : \mathcal{O} \to 2^{\mathcal{L}it(\mathcal{P}(\mathcal{X}))}$ are functions that associate operators with their sets of preconditions and effects, respectively. Therefore, sets $\mathsf{pre}(op)$ and $\mathtt{eff}(op)$ contain the preconditions and the effects of the operator op; $\mathtt{eff}^+(op)$ and $\mathtt{eff}^-(op)$ denote the set of positive and negative literals in $\mathtt{eff}(op)$, and are called positive and negative effects of op. Without loss of generality, we assume that operators have no negative precondition, and that operators do not make true atoms that they require to be true in their preconditions, i.e., $\mathtt{eff}(op) \cap \mathsf{pre}(op) = \emptyset$.[2]

Definition 2 *(Ground action).* The *ground action* $a = op(c_1, \dots, c_n)$ of an n-ary operator name $op \in \mathcal{O}$ w.r.t. constants $c_1, \dots, c_n$ is the pair $\langle \mathsf{pre}(a), \mathtt{eff}(a) \rangle$, where $\mathsf{pre}(a)$ (resp. $\mathtt{eff}(a)$) is obtained by replacing the i-th parameter of $\mathsf{par}(op)$ in $\mathsf{pre}(op)$ (resp. $\mathtt{eff}(op)$) with c_i, for $i = 1, \dots, n$.

[2] An operator with $p(\boldsymbol{x})$ as a negative precondition can be translated into an operator without such a precondition by introducing a dummy predicate, say *not-*$p(\boldsymbol{x})$, substituting $\neg p(\boldsymbol{x})$ with *not-*$p(\boldsymbol{x})$ in the preconditions, adding *not-*$p(\boldsymbol{x})$ to the action effects that make $p(\boldsymbol{x})$ false, and adding $\neg$*not-*$p(\boldsymbol{x})$ to the action effects that make $p(\boldsymbol{x})$ true. An operator with $p(\boldsymbol{x})$ as both precondition and effect can be simply translated into the same operator but without the $p(\boldsymbol{x})$ effect.

For simplicity, we adopt the injective binding assumption in the presentation of our approach, i.e., we restrict the ground actions to $op(c_1, \dots, c_n)$ with $c_i \neq c_j$ for $1 \leq i < j \leq n$. We use notation $\boldsymbol{c}$ to denote a tuple of constants. Moreover, we use the term *lifted*, as the opposite of *grounded*, to refer to expressions and actions where constants have been replaced with parameters.

Definition 3 *(Action model).* An *action model* $\mathcal{M}$ is a triple $\langle \mathcal{P}, \mathcal{O}, \mathcal{H} \rangle$ where $\mathcal{P}$ is a set of predicate names with their arity, $\mathcal{O}$ is a set of operator names with their arity, and $\mathcal{H}$ maps every operator name $op \in \mathcal{O}$ into an action schema.

In the rest of the paper, $\mathtt{pre}_{\mathcal{M}}$ and $\mathtt{eff}_{\mathcal{M}}$ denote functions $\mathtt{pre}$ and $\mathtt{eff}$ for action model $\mathcal{M}$, respectively. For every literal l, $\overline{l}$ denotes the complementary literal defined as follows: if l is $p(\boldsymbol{c})$ then $\overline{l}$ is $\neg p(\boldsymbol{c})$, if l is $\neg p(\boldsymbol{c})$ then $\overline{l}$ is $p(\boldsymbol{c})$. For a set of literals L, $\overline{L}$ denotes the set of literals consisting of the complementary of each literal in L, i.e., $\overline{L} = \{\overline{l} \mid l \in L\}$.

Definition 4 *(Finite-state machine of an action model).* The *finite-state machine* of an action model $\mathcal{M} = \langle \mathcal{P}, \mathcal{O}, \mathcal{H} \rangle$ for a set C of constants is the triple $\mathcal{M}(C) = \langle S, \mathcal{A}, \delta \rangle$ where the set of states S is the set of maximally consistent sets of literals in $\mathcal{L}it(\mathcal{P}(C))$, i.e., S is the set of $s \subseteq \mathcal{L}it(\mathcal{P}(C))$, such that for every $l \in \mathcal{L}it(\mathcal{P}(C))$, either $l \in s$ or $\overline{l} \in s$. $\mathcal{A}$ is the set of all possible ground actions of each n-ary operator name in $\mathcal{O}$ on any n-tuple of constants in C; $\delta \subseteq S \times \mathcal{A} \times S$ is a transition relation such that $(s, a, s') \in \delta$ if and only if $\mathtt{pre}(a) \subseteq s$ and $s' = s \setminus \overline{\mathtt{eff}(a)} \cup \mathtt{eff}(a)$.

We used both terms "action model" and "planning domain" interchangeably. We also use the term ground model to indicate the finite-state machine of the model for some set of constants. The proper meaning can be determined from the context.

Under total observability, a state can be unambiguously represented as a set of atoms (positive literals); if an atom is not present in a state, it is assumed to be false in that state. Under partial observability, instead, the absence of an atom and its negation should be considered as an absence of evidence about the atom's truth value. In other words, under partial observability, a complete description of a state contains for every atom $p(\boldsymbol{c})$ either $p(\boldsymbol{c})$ or $\neg p(\boldsymbol{c})$, and a *partial state* is a proper subset of a complete state.

Definition 5 *(Planning problem).* A *planning problem* for a planning domain $\mathcal{M}$ and a set of constant C is a triple $\langle \mathcal{M}(C), s_0, g \rangle$ where $\mathcal{M}(C)$ is the finite-state machine of $\mathcal{M}$ for C, s_0 is a state of $\mathcal{M}(C)$, called the *initial state*, and g is a propositional formula over the predicates of $\mathcal{M}$ and constants in C, called the *goal*.

Definition 6 *(Plan and executable plan).* A *plan* π is a finite sequence of ground actions. A plan $(a_1, \dots, a_n)$ is *executable* in $\mathcal{M}(C)$ from a state s_0 if $(s_{i-1}, a_i, s_i) \in \delta$ for $i \in \{1, \dots, n\}$. In this case, we say that π executed in s_0 *reaches* s_n, or that s_n is reachable from s_0 by π.

Definition 7 *(Trace).* Given an executable plan $\pi = (a_1, \dots, a_n)$, the sequence $(s_0, a_1, s_1), \dots, (s_{n-1}, a_n, s_n)$ is called a *plan trace* of π, or simply a *trace*.

Definition 8 *(Valid plan).* A plan π is *valid* for a planning problem $\langle \mathcal{M}(C), s_0, g \rangle$, if it is executable from s_0 in $\mathcal{M}(C)$ and reaches a state s that satisfies g.

Definition 9 *(Consistency with a trace).* An action model $\mathcal{M}$ is *consistent with a trace* $\boldsymbol{t}$ (in short $\boldsymbol{t}$ is a trace of $\mathcal{M}$) if $\boldsymbol{t}$ is a trace of $\mathcal{M}(C)$, where C is the set of constants in $\boldsymbol{t}$.

A trace of a plan contains the complete description of each state generated by executing the actions in the plan; i.e., every s_i contains all the ground literals that are true at timestep i. However, in many cases, such a complete description might not be available. This lack of knowledge is modelled with *partial traces*. In a partial trace information about the truth value in a state or about the executed actions can be partial. In the following, the wildcard symbol $\star$ can be replaced by either an action, an operator, or an argument of an action.

Definition 10 *(Partial state and partial action).* A partial state $\hat{s}$ of a state s is a subset of s. A partial action $\hat{a}$ of an action a is either $\star$ or an expression obtained by replacing one or more parts (operator name and/or argument) of a with $\star$.

The worst case is when in an input trace $\hat{a} = \star$, because in this case neither the action name nor its arguments are observed, i.e., action a is completely non observable. In our experimental study, when input traces contain partial actions, we will focus on such a worst case.

Definition 11 *(Partial trace).* A *partial trace* $\hat{\boldsymbol{t}}$ of a trace $\boldsymbol{t}$ is obtained by replacing some state s_i of $\boldsymbol{t}$ with a partial state $\hat{s}_i$ and/or some action a_i of $\boldsymbol{t}$ with a partial action $\hat{a}_i$. If $\hat{\boldsymbol{t}}$ is a partial trace of $\boldsymbol{t}$, then we say that $\boldsymbol{t}$ is *a completion of* $\hat{\boldsymbol{t}}$.

Definition 12 *(Consistency with a partial trace).* An action model $\mathcal{M}$ is *consistent with a partial trace* $\hat{\boldsymbol{t}}$ if it is consistent with some completion of $\hat{\boldsymbol{t}}$.

We are now ready to formally define the problem of learning an action model from an input set of partial traces. We assume that the knowledge contained in such a set of traces is sound, i.e., it cannot happen that in a trace an action a makes an atom true while a makes the same atom false in another trace.

Definition 13 *(Action model learning problem).* Let $\hat{\mathcal{T}} = \{\hat{t}^{(1)}, \dots, \hat{t}^{(n)}\}$ be a set of partial traces such that there is an (unknown) ground truth model $\mathcal{M}^\star$ consistent with $\hat{t}^{(i)}$, for $1 \leq i \leq n$. The problem of *action model learning* is the task of finding an action model $\mathcal{M}$ that is the most similar to $\mathcal{M}^\star$.

We address this problem by looking for all the "smallest" models that are consistent with an input set of traces. This requires the definition of a partial order on the set of action models. Intuitively, a model $\mathcal{M}_1$ is smaller than a model $\mathcal{M}_2$ if the transitions outgoing from a state for $\mathcal{M}_1$ are a subset of those for $\mathcal{M}_2$, and the changes in the state resulting from the execution of an action in a state for $\mathcal{M}_1$ are a subset of that for $\mathcal{M}_2$. Let us formally define what we mean by the smallest model.

Definition 14 *(Partial order on action models).* Let $\mathcal{M}(C) = (S, \mathcal{A}, \delta)$, $\mathcal{M}'(C) = (S, \mathcal{A}, \delta')$, and p be a predicate. $\mathcal{M}(C)$ is *smaller than or equal to* $\mathcal{M}'(C)$ w.r.t. p, denoted by $\mathcal{M}(C) \preceq_p \mathcal{M}'(C)$, if $(s_0, a, s_1) \in \delta$ implies that there exists $(s'_0, a, s'_1) \in \delta'$ for all tuples of constants $\boldsymbol{c}$ of C such that

1. $p(\boldsymbol{c}) \in s'_0 \Rightarrow p(\boldsymbol{c}) \in s_0$;
2. $p(\boldsymbol{c}) \in s'_0 \cap s'_1 \Rightarrow p(\boldsymbol{c}) \in s_0 \cap s_1$;
3. $\neg p(\boldsymbol{c}) \in s'_0 \cap s'_1 \Rightarrow \neg p(\boldsymbol{c}) \in s_0 \cap s_1$.

Furthermore, $\mathcal{M} \preceq_p \mathcal{M}'$ if for all set of constants C, $\mathcal{M}(C) \preceq_p \mathcal{M}'(C)$; $\mathcal{M} \preceq \mathcal{M}'$, if for all predicates p, $\mathcal{M} \preceq_p \mathcal{M}'$.

One can easily prove that $\preceq_p$ and $\preceq$ are partial orders. The definition of $\mathcal{M} \preceq \mathcal{M}'$ is similar to the definition of the fact that $\mathcal{M}$ is a safe action model w.r.t. $\mathcal{M}'$ given in [18] (Def. 4) with the only difference that we do not require that the effects of an action are the same in $\mathcal{M}$ and $\mathcal{M}'$ when the action is applied to the same state, but only that the effects of the action in $\mathcal{M}$ are fewer than or the same as those in $\mathcal{M}'$. Indeed, conditions 2 and 3 of Definition 14 impose that the literals that are not changed by action a in $\mathcal{M}'$ are a subset of the literals that are also not changed in $\mathcal{M}$.

Example 1. Assume that $\delta = \{(s_0, op_1(\boldsymbol{c}), s_1), (s_0, op_2(\boldsymbol{c}), s_2)\}$, $\delta' = \{(s'_0, op_1(\boldsymbol{c}), s'_1), (s'_1, op_2(\boldsymbol{c}), s'_2)\}$, δ and δ' are respectively the transition functions of $\mathcal{M}(C)$ and $\mathcal{M}'(C)$, and let $s_0 = \{\neg p(\boldsymbol{c}), q(\boldsymbol{c})\}$, $s_1 = \{p(\boldsymbol{c}), q(\boldsymbol{c})\}$, $s_2 = \{\neg p(\boldsymbol{c}), \neg q(\boldsymbol{c})\}$, $s'_0 = \{p(\boldsymbol{c}), q(\boldsymbol{c})\}$, $s'_1 = \{p(\boldsymbol{c}), q(\boldsymbol{c})\}$ and $s'_2 = \{p(\boldsymbol{c}), q(\boldsymbol{c})\}$, where $\boldsymbol{c}$ is a sequence of constants in C. The truth-value of literals $p(\boldsymbol{c})$ and $q(\boldsymbol{c})$ is not changed by action $op_1(\boldsymbol{c})$ and $op_2(\boldsymbol{c})$ in $\mathcal{M}'(C)$, whereas it is changed in $\mathcal{M}(C)$, therefore $\mathcal{M}(C) \not\preceq_p \mathcal{M}'(C)$ and $\mathcal{M}(C) \not\preceq_q \mathcal{M}'(C)$. On the contrary, since we also have that all the positive literals of s_0 and s_1 are contained in s'_0 and s'_1, $\mathcal{M}'(C) \preceq_p \mathcal{M}(C)$, $\mathcal{M}'(C) \preceq_q \mathcal{M}(C)$, and hence $\mathcal{M}'(C) \preceq \mathcal{M}(C)$.

Proposition 1. *$\mathcal{M} \preceq_p \mathcal{M}'$ if and only if for any operator $op \in \mathcal{O}$*

1. $p(\boldsymbol{x}) \in \mathsf{pre}'(op(\boldsymbol{x})) \Rightarrow p(\boldsymbol{x}) \in \mathsf{pre}(op(\boldsymbol{x}))$,
2. $p(\boldsymbol{x}) \in \mathtt{eff}(op(\boldsymbol{x})) \Rightarrow p(\boldsymbol{x}) \in \mathtt{eff}'(op(\boldsymbol{x}))$,
3. $\neg p(\boldsymbol{x}) \in \mathtt{eff}(op(\boldsymbol{x})) \Rightarrow \neg p(\boldsymbol{x}) \in \mathtt{eff}'(op(\boldsymbol{x}))$,

where pre *and* eff *(resp.* pre′ *and* eff′*) are the functions for preconditions and effects of $\mathcal{M}$ (resp. $\mathcal{M}'$).*

Proof. We start by proving the only if direction: Suppose that $\mathcal{M} \preceq_p \mathcal{M}'$.

1. Let us show that $p(\boldsymbol{x}) \in \mathsf{pre}'(op(\boldsymbol{x})) \Rightarrow p(\boldsymbol{x}) \in \mathsf{pre}(op(\boldsymbol{x}))$. Let $p(\boldsymbol{x}) \in \mathsf{pre}'(op(\boldsymbol{x}))$, and suppose by contradiction that $p(\boldsymbol{x}) \notin \mathsf{pre}(op(\boldsymbol{x}))$. Let s_0 be a state in which all the literals are true with the exception of $p(\boldsymbol{c})$, for some tuple of constants $\boldsymbol{c}$ in C. Since $p(\boldsymbol{x}) \notin \mathsf{pre}(op(\boldsymbol{x}))$, we have that $op(\boldsymbol{c})$ is applicable in s_0, and therefore $(s_0, op(\boldsymbol{c}), s_1) \in \delta$ for some s_1. But in $\mathcal{M}'(C)$, as $p(\boldsymbol{x}) \in \mathsf{pre}'(op(\boldsymbol{x}))$, $op(\boldsymbol{c})$ is not applicable in every state s'_0 that satisfies condition 1 of Definition 14, and therefore there is no such $(s'_0, op(\boldsymbol{c}), s'_1) \in \delta'$, which contradicts the fact that $\mathcal{M} \preceq_p \mathcal{M}'$.
2. Let us prove that the second condition of the proposition is true, i.e., that $p(\boldsymbol{x}) \in \mathtt{eff}(op(\boldsymbol{x})) \Rightarrow p(\boldsymbol{x}) \in \mathtt{eff}'(op(\boldsymbol{x}))$. Let $p(\boldsymbol{x}) \in \mathtt{eff}(op(\boldsymbol{x}))$, and suppose by contradiction that $p(\boldsymbol{x}) \notin \mathtt{eff}'(op(\boldsymbol{x}))$. Since $p(\boldsymbol{x}) \in \mathtt{eff}(op(\boldsymbol{x}))$, because of our definition of action schema, $p(\boldsymbol{x}) \notin \mathsf{pre}(op(\boldsymbol{x}))$. By the previous point, we have that $p(\boldsymbol{x}) \notin \mathsf{pre}'(op(\boldsymbol{x}))$. Let s_0 the state in which all the literals are true except $p(\boldsymbol{c})$ for some tuple of constants $\boldsymbol{c}$ of C. We have that in $\mathcal{M}(C)$ $op(\boldsymbol{c})$ is applicable in s_0. This implies that there exists $(s_0, op(\boldsymbol{c}), s_1) \in \delta$ and since $p(\boldsymbol{c}) \in \mathtt{eff}(op(\boldsymbol{c}))$, $\neg p(\boldsymbol{c}) \notin s_0 \cap s_1$. For every transition $(s'_0, op(\boldsymbol{c}), s'_1)$ such that $\neg p(\boldsymbol{c}) \in s'_0$, since $p(\boldsymbol{c}) \notin \mathtt{eff}'(op(\boldsymbol{c}))$ we have that $\neg p(\boldsymbol{c}) \in s'_0 \cap s'_1$, which contradicts condition 3 of Definition 14, and hence contradicts the fact that $\mathcal{M} \preceq_p \mathcal{M}'$.
3. The same argument used to prove the second condition applies to the third one, where s_0 is a state in which all literals are true, which leads us to contradict condition 2 of Definition 14.

Let us now prove the opposite direction, i.e., that if $\mathcal{M}$ and $\mathcal{M}'$ satisfy properties 1–3 of Proposition 1 then $\mathcal{M} \preceq_p \mathcal{M}'$. Let $(s_0, op(\boldsymbol{c}), s_1) \in \delta$, and s'_0 be the state in which all and only the atoms in $\mathsf{pre}'(op(\boldsymbol{c}))$ are true, therefore $op(\boldsymbol{c})$ is applicable in s'_0, i.e., there exists $(s'_0, op(\boldsymbol{c}), s'_1) \in \delta'$ for some s'_1. By condition 1 of Proposition 1, we have that all atoms in $\mathsf{pre}'(op(\boldsymbol{c}))$ are in $\mathsf{pre}(op(\boldsymbol{c}))$, hence they must be true in s_0. Therefore, condition 1 of Definition 14 is satisfied by construction of s'_0. Concerning condition 2 of Definition 14, we have that, if $p(\boldsymbol{c})$ is true in s_0 and s'_0 and $p(\boldsymbol{c}) \in s'_0 \cap s'_1$ for some tuple of constants $\boldsymbol{c}$, then $\neg p(\boldsymbol{x}) \notin \mathsf{eff}'(op(\boldsymbol{x}))$ and therefore by condition 2 of Proposition 1 $\neg p(\boldsymbol{x}) \notin \mathsf{eff}(op(\boldsymbol{x}))$, which implies that $p(\boldsymbol{c}) \in s_0 \cap s_1$. A similar argument can be done for condition 3 of Proposition 1. □

An immediate consequence of Proposition 1 is the following proposition.

Proposition 2. *$\mathcal{M} \preceq \mathcal{M}'$ if and only if for any operator $op \in \mathcal{O}$*

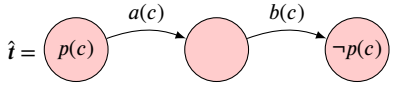
1. $\mathsf{pre}'(op(\boldsymbol{x})) \subseteq \mathsf{pre}(op(\boldsymbol{x}))$;
2. $\mathsf{eff}(op(\boldsymbol{x})) \subseteq \mathsf{eff}'(op(\boldsymbol{x}))$.

The above propositions state that the relationship between two action models does not depend on a set of constants. Indeed, it is sufficient to consider the preconditions and effects of the operators. A second consequence of the above proposition is that the binary relation $\preceq$ is a partial order on the set of action models with fixed predicates and operators. This allows us to provide the following definition.

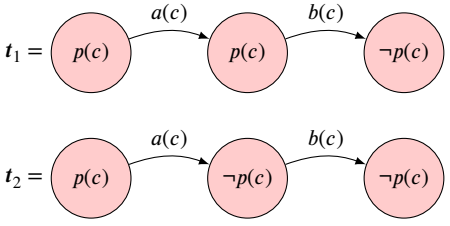
Definition 15 *(Minimal action model for a set of partial traces).* An action model $\mathcal{M}$ is a *minimal model* for a set of partial traces $\hat{\mathcal{T}}$ if $\mathcal{M}$ is consistent with all $\hat{\boldsymbol{t}} \in \hat{\mathcal{T}}$, and for all $\mathcal{M}'$ consistent with all $\hat{\boldsymbol{t}} \in \hat{\mathcal{T}}$, it is not true that $\mathcal{M}' \prec \mathcal{M}$. The set of minimal models for a set of partial traces $\hat{\mathcal{T}}$ is denoted by $\mathbb{M}_{\text{min}}(\hat{\mathcal{T}})$.

As a first remark, one can see that partial traces can have more than one minimal model. Consider the following example.

Example 2. Let $\hat{\boldsymbol{t}}$ be the following partial trace



where each circle represents a partial state that contains the literals shown in the circle, and each edge represents an action. This partial trace shows that $p(c)$ becomes false between the first and the third state, but since we do not observe the truth value of $p(c)$ in the second state, it could be that either $p(c)$ is falsified by the first action (i.e., $a(c)$) or by the second action (i.e., $b(c)$) or by both. If p is the only predicate and c is the only constant, then the only two possible completions $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ of the partial trace $\hat{\boldsymbol{t}}$ are the following traces:



The smallest model consistent with $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ are respectively the models $\mathcal{M}_1$ and $\mathcal{M}_2$ that are shown in the following table:

Models	$a(x)$		$b(x)$	
	pre	eff	pre	eff
$\mathcal{M}_1$	$p(x)$		$p(x)$	$\neg p(x)$
$\mathcal{M}_2$	$p(x)$	$\neg p(x)$		

$\mathcal{M}_1$ is minimal since pre cannot be further extended, and if we remove $\neg p(x)$ from the effects of b the model will not be consistent with $\boldsymbol{t}_1$. Similarly, $\mathcal{M}_2$ is minimal because if we add $p(x)$ to the precondition of $b(x)$ or we remove it from the negative effect of $a(x)$ we obtain a model that is not consistent with $\boldsymbol{t}_2$.

Furthermore $\mathcal{M}_1 \not\prec \mathcal{M}_2$ since $\mathsf{eff}_1(b(x)) \not\subseteq \mathsf{eff}_2(b(x))$ and $\mathcal{M}_2 \not\prec \mathcal{M}_1$ since $\mathsf{eff}_2(a(x)) \not\subseteq \mathsf{eff}_1(a(x))$, where $\mathsf{eff}_i(op(\boldsymbol{x}))$ indicates the set of effects of $op(\boldsymbol{x})$ in model $\mathcal{M}_i$. Therefore, $\mathcal{M}_1$ and $\mathcal{M}_2$ are both minimal for $\hat{\boldsymbol{t}}$.

For total traces, instead, there is only one minimal model, as it is stated by the following proposition.

Proposition 3. *For every set of total traces $\mathcal{T}$, $\mathbb{M}_{\min}(\mathcal{T})$ contains only one model.*

Proof. For every operator $op \in \mathcal{O}$ we define a model $\mathcal{M}$ consistent with $\mathcal{T}$ as follows

$$\mathsf{pre}(op(\boldsymbol{x})) = \mathcal{P}(\boldsymbol{x}) \setminus \{p(\boldsymbol{x}) \mid (s, op(\boldsymbol{c}), s') \in \mathcal{T}, \neg p(\boldsymbol{c}) \in s\}$$

$$\mathsf{eff}(op(\boldsymbol{x})) = \{l(\boldsymbol{x}) \in \mathcal{L}it(\boldsymbol{P}(\boldsymbol{x})) \mid (s, op(\boldsymbol{c}), s') \in \mathcal{T}, l(\boldsymbol{c}) \in s' \setminus s\}$$

It is easy to show that this is the only minimal model consistent with $\mathcal{T}$. □

Definition 16. The *cautious model* for a set of partial trace $\hat{\mathcal{T}}$ is the model $\mathcal{M}_c$ where for each $op \in \mathcal{O}$:

$$\mathsf{pre}_{\mathcal{M}_c}(op) = \bigcup_{\mathcal{M} \in \mathbb{M}_{\min}(\hat{\mathcal{T}})} \mathsf{pre}_{\mathcal{M}}(op), \qquad \mathsf{eff}_{\mathcal{M}_c}(op) = \bigcap_{\mathcal{M} \in \mathbb{M}_{\min}(\hat{\mathcal{T}})} \mathsf{eff}_{\mathcal{M}}(op)$$

where $\mathsf{pre}_{\mathcal{M}}(op)$ and $\mathsf{eff}_{\mathcal{M}}(op)$ are the sets of preconditions and effects of op in $\mathcal{M}$.

Notably, if an action a is applicable in a state s according to the cautious model, then it is applicable in s according to all the models compatible with $\hat{\mathcal{T}}$, including the ground truth models $\mathcal{M}^\star$. Similarly, the changes performed by a on s according to the cautious model are done also by all the models compatible with $\hat{\mathcal{T}}$ and therefore by $\mathcal{M}^\star$. However, this does not guarantee that a plan executable in the cautious model is also executable in $\mathcal{M}^\star$. Consider the following example.

Example 3. Suppose that in $\mathcal{M}^\star$ an atom, $p(x)$, is a negative effect and a positive precondition of two actions $a(x)$ and $b(x)$, respectively. Suppose also that in the cautious model, $p(x)$ is not included in the negative effects of $a(x)$. While the plan $(a(c), b(c))$ is executable in the cautious model from an initial state where $p(c)$ is true, it is not executable in $\mathcal{M}^\star$ because $p(c)$ becomes false after the execution of $a(c)$, making $b(c)$ inapplicable.

The cautious model preserves the single transitions of the ground truth model, but the changes in the resulting state of a transition for the cautious model can be a subset of those of the ground truth model. This might compromise the executability of sequences of transitions in the ground truth model because of negative effects. Some approaches (e.g., [7,24,28]) compute safe action models that preserve the executability of sequences of transitions.

In our work, we focus on the computation of the minimal action models, which are the smallest models consistent with the set of input traces. However, we acknowledge there exist other models interesting to learn, such as, e.g., safe models [28]. Notably, the minimal action models can be useful even for deriving a safe model, as well as for the cautious model. Indeed, assuming that preconditions and goals are not negative, it is easy to see that a safe action model can be defined from the minimal action models as shown in Definition 16 for preconditions and positive effects, and considering all the negative effects that are consistent with the input traces as the set of negative effects.

4. Learning action models from partial traces

In this section, we present a methodology for learning an action model, which is based on a set of *completion rules*. The application of the completion rules gradually refines the action model based on the evidence in the traces and extends the partial traces using the action models under construction. This iterative process continues until no rules are applicable. Subsequently, we compute both the minimal models and the cautious model from the resulting learned model. We conclude the section by proving that the method is sound and complete, meaning that it identifies all and only the minimal action models compatible with a set of partial traces

The completion rules are applied on a structure defined by a 4-tuple $(\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}})$, where:

- for every $op \in \mathcal{O}$, $\mathsf{pre}_?(op)$ is a superset of the preconditions of op in every (minimal) model of $\hat{\mathcal{T}}$. The elements of $\mathsf{pre}_?(op)$ are called *potential preconditions*;
- for every $op \in \mathcal{O}$, $\mathsf{eff}_!(op)$ is a subset of the effects of op in every (minimal) model of $\hat{\mathcal{T}}$. The elements of $\mathsf{eff}_!(op)$ are called *definite (positive/negative) effects*;
- for every $op \in \mathcal{O}$, $\mathsf{eff}_?(op)$ is a set of literals disjoint from $\mathsf{eff}_!(op)$, such that $\mathsf{eff}_!(op) \cup \mathsf{eff}_?(op)$ is a superset of the effects of op in every (minimal) model of $\hat{\mathcal{T}}$. The elements of $\mathsf{eff}_?(op)$ are called *potential (positive/negative) effects*;
- $\hat{\mathcal{T}}$ is a set of partial traces.

$\mathsf{pre}_? : \mathcal{O} \to 2^{\mathcal{P}(\mathcal{X})}$, $\mathsf{eff}_! : \mathcal{O} \to 2^{\mathcal{L}it(\mathcal{P}(\mathcal{X}))}$ and $\mathsf{eff}_? : \mathcal{O} \to 2^{\mathcal{L}it(\mathcal{P}(\mathcal{X}))}$ are functions that associate operators with their sets of potential preconditions, definite effects and potential effects, respectively, while $\mathsf{eff}_{!?}(op)$ denotes the union of $\mathsf{eff}_!(op)$ and $\mathsf{eff}_?(op)$.

Intuitively, the tuple $(\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}})$ provides a lower-bound and an upper-bound of the action models consistent with $\hat{\mathcal{T}}$, the lower-bound being $(\mathsf{pre}_?, \mathsf{eff}_!)$ and the upper-bound being $(\emptyset, \mathsf{eff}_{!?})$. We formalize this concept in the following definition.

Definition 17 *($\mathcal{M} \models (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$).* An action model $\mathcal{M}$ *satisfies* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$, in symbols $\mathcal{M} \models (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$, if $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}$, $\mathtt{pre}_{\mathcal{M}}(op) \subseteq \mathtt{pre}_?(op)$, and $\mathtt{eff}_!(op) \subseteq \mathtt{eff}_{\mathcal{M}}(op) \subseteq \mathtt{eff}_{!?}(op)$, for any operator op in $\mathcal{O}$.

The completion rules transform a 4-tuple $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ into either another 4-tuple or two alternative 4-tuples that preserve the satisfiability relation defined above. More precisely, the rules can have one of the following three forms:

$$\frac{(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})}{(\mathtt{pre}'_?, \mathtt{eff}'_!, \mathtt{eff}'_?, \hat{\mathcal{T}})} \tag{1}$$

$$\frac{(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})}{(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}')} \tag{2}$$

$$\frac{(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})}{(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}') \;\big|\; (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}'')} \tag{3}$$

The rules of form (1) update the sets of potential preconditions, potential and definite effects by observing the literals in the states involved in the transitions contained in $\hat{\mathcal{T}}$. The rules of form (2), instead, update the partial traces $\hat{\mathcal{T}}$ with the information contained in the potential preconditions, potential and definite effects. Finally, the rules of form (3) update a partial state in $\hat{\mathcal{T}}$ where an atom $p(\boldsymbol{c})$ is not observed by considering the two cases in which $p(\boldsymbol{c})$ is true or $p(\boldsymbol{c})$ is false, which produce the two alternative sets of partial traces $\hat{\mathcal{T}}'$ and $\hat{\mathcal{T}}''$.

4.1. Rules for learning preconditions and effects

The rule of type (1) consists in applying for every transition $(\hat{s}, op(\boldsymbol{c}), \hat{s}')$ that appears in some trace $\hat{t} \in \hat{\mathcal{T}}$ (in short $(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}}$) the following transformations on the $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ 4-tuple:

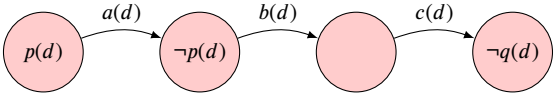
$$\frac{(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}}}{\mathtt{eff}_!(op) \leftarrow \mathtt{eff}_!(op) \cup \{l(\boldsymbol{x}) \mid \overline{l(\boldsymbol{c})} \in \hat{s} \text{ and } l(\boldsymbol{c}) \in \hat{s}'\}} \tag{4}$$

$$\frac{(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}}}{\mathtt{eff}_?(op) \leftarrow \mathtt{eff}_?(op) \setminus \{l(\boldsymbol{x}) \mid l(\boldsymbol{c}) \in \mathtt{eff}_!(op) \text{ or } \overline{l(\boldsymbol{c})} \in \hat{s}'\}} \tag{5}$$

$$\frac{(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}}}{\mathtt{pre}_?(op) \leftarrow \mathtt{pre}_?(op) \setminus \{p(\boldsymbol{x}) \mid \neg p(\boldsymbol{c}) \in \hat{s}\}} \tag{6}$$

which augment the set of definite effects, and reduce the sets of potential preconditions and potential effects. In particular, rule (4) adds $p(\boldsymbol{x})$ (resp. $\neg p(\boldsymbol{x})$) to the definite effects of op, if $\hat{\mathcal{T}}$ contains an execution of op where $\neg p(\boldsymbol{c})$ (resp. $p(\boldsymbol{c})$) is observed in the state immediately before the execution of $op(\boldsymbol{c})$ and $p(\boldsymbol{c})$ (resp. $\neg p(\boldsymbol{c})$) is observed in the state immediately after. Rule (5) removes $p(\boldsymbol{x})$ (resp. $\neg p(\boldsymbol{x})$) from the potential effects of op if it is in the set of definite effects of op or the opposite literal $\neg p(\boldsymbol{x})$ (resp. $p(\boldsymbol{x})$) is in the state $\hat{s}'$ resulting from the execution of $op(\boldsymbol{c})$. Finally, rule (6) removes $p(\boldsymbol{x})$ from the potential precondition of op if $\neg p(\boldsymbol{c})$ is in the state from which $op(\boldsymbol{c})$ is executed. Basically, rules (4)–(6) are the same defined by Juba et al. (see Observation 1 in [18]). Notice that these rules construct sets of *lifted* literals for operator op from the set of ground literals observed in the traces. The mapping among the constants involved in the literals observed in the traces and the variables used for the definition of op is unique and can be easily determined because of the adoption of the injective binding assumption.

Example 4. Consider the following partial trace:



From the first transition we have that $p(d)$ is a negative effect of action $a(d)$, and therefore rule (4) adds $\neg p(x)$ to the set of definite effect of operator $a(x)$; the second transition instead tells us that $p(d)$ cannot be a precondition of action $b(d)$ and therefore rule (6) removes $p(x)$ from the potential preconditions of operator $b(x)$; finally, the third transition implies that $q(d)$ cannot be a positive effect of action $c(d)$, and therefore rule (5) removes $p(x)$ from the potential effects of operator $c(x)$.

4.2. Rules for propagating (non) effects

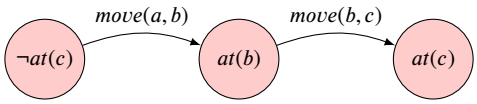
The rule of type (2) consists in applying for every transition $(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}}$ the following transformations on the $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ 4-tuple:

$$\frac{(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}}}{\hat{s}' \leftarrow \hat{s}' \cup \mathtt{eff}_!(op(\boldsymbol{c}))} \tag{7}$$

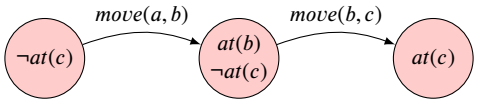
$$\frac{(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}}}{\begin{array}{l}\hat{s} \leftarrow \hat{s} \cup (\hat{s}' \setminus \mathtt{eff}_{!?}(op(\boldsymbol{c}))) \\ \hat{s}' \leftarrow \hat{s}' \cup (\hat{s} \setminus \overline{\mathtt{eff}_{!?}(op(\boldsymbol{c}))})\end{array}} \tag{8}$$

which extends the states of $\hat{\mathcal{T}}$ that occur immediately before and after the execution of operator op with literals according to the set of definite and potential effects of op. In particular, rule (7) simply extends the state obtained from the execution of the action $op(\boldsymbol{c})$ with the definite effects of the action. Rule (8) implements the inertia principle that states the property that the states before and after an action $op(\boldsymbol{c})$ agree on all the literals that are not changed by $op(\boldsymbol{c})$, i.e., that are not in the potential or definite effects of the action. Indeed, for every action $op(\boldsymbol{c})$, $\mathtt{eff}_!(op(\boldsymbol{c}))$ contains the effects that we have learned so far, while $\mathtt{eff}_?(op(\boldsymbol{c}))$ contains the literals that we have not excluded from the set of possible effects so far. So, if an atom is not in $\mathtt{eff}_{!?}(op(\boldsymbol{c}))$ it is certainly not changed by $op(\boldsymbol{c})$, and therefore it should have the same truth value in the states before and after the execution of $op(\boldsymbol{c})$.

Example 5. Consider the following partial trace:



Since action $move(a, b)$ does not involve constant c, rule (8) propagates literal $\neg at(c)$ from the first partial state to the second, obtaining the partial trace:



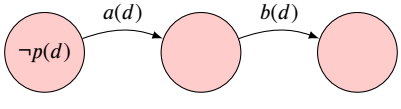
From this updated trace, rule (4) adds $at(y)$ to the set of definite effect of operator $move(x, y)$.

One would expect a similar rule for potential preconditions to extend the state prior to the execution of an action with its preconditions. However, $\mathtt{pre}_?(op)$ are preconditions that have not been verified yet. Formally, the set of preconditions of an action $op(\boldsymbol{c})$ in the ground truth model is a subset of $\mathtt{pre}_?(op(\boldsymbol{c}))$. The only way to find out that an atom needs to be a precondition of $op(\boldsymbol{c})$ is trying to perform the action from a state where the atom is false and obtain a failure. Since in our setting there is no trace containing action failures, we have no evidence to state that an atom *must be* a precondition of an action. The reader interested in methods that learn preconditions from failures can see paper [22].

4.3. Rules for case reasoning on unobserved atoms

The above rules are not sufficient to complete the set of partial traces $\hat{\mathcal{T}}$ in order to obtain fully observable traces. One possible situation happens with partial traces where the truth value of some atom $p(\boldsymbol{c})$ changes after executing a sequence of more than one action but we do not have enough clues to decide which action in the sequence affects $p(\boldsymbol{c})$. An instance of such a partial trace is provided in Example 2. To deal with these situations, we have to reason by cases, i.e., we have to consider two alternatives: $p(\boldsymbol{c})$ is/is not affected by the first action. Another situation that requires reasoning by case is described in the following example:

Example 6. Consider the following partial trace:



and assume that $p(d) \in \mathtt{pre}_?(b(d))$. If $p(d)$ is true in the second state of the trace, $p(x)$ can be added to the set of definite effects of operator $a(x)$. If, instead, $p(d)$ is false in the second state of the trace, then $p(x)$ is not a definite positive effect, and $p(x)$ can be removed from the set of potential preconditions of $a(x)$. These two cases correspond to two models that are incomparable w.r.t. the order $\prec$, and therefore they are both minimal.

The rules of type (3) cope with the situations shown in Examples 2 and 6 by splitting the search of the minimal models in two "parallel" cases, as stated by the following transformations:

$$\frac{\begin{array}{c}(\hat{s}_i, a_{i+1}, \hat{s}_{i+1}), \dots, (\hat{s}_{k-1}, a_k, \hat{s}_k) \in \hat{\mathcal{T}} \\ \{p(\boldsymbol{c}), \neg p(\boldsymbol{c})\} \cap (\hat{s}_j \cup \mathtt{pre}_?(a_j)) = \emptyset \text{ for } i < j < k \\ \neg p(\boldsymbol{c}) \in \hat{s}_i \text{ and } p(\boldsymbol{c}) \in \hat{s}_k \cup \mathtt{pre}_?(a_k)\end{array}}{\hat{s}_{i+1} \leftarrow \hat{s}_{i+1} \cup \{p(\boldsymbol{c})\} \;\big|\; \hat{s}_{i+1} \leftarrow \hat{s}_{i+1} \cup \{\neg p(\boldsymbol{c})\}} \tag{9}$$

$$\frac{\begin{array}{c}(\hat{s}_i, a_{i+1}, \hat{s}_{i+1}), \dots, (\hat{s}_{k-1}, a_k, \hat{s}_k) \in \hat{\mathcal{T}} \\ \{p(\boldsymbol{c}), \neg p(\boldsymbol{c})\} \cap (\hat{s}_j \cup \mathtt{pre}_?(a_j)) = \emptyset \text{ for } i < j < k \\ \neg p(\boldsymbol{c}) \in \hat{s}_k \text{ and } p(\boldsymbol{c}) \in \hat{s}_i \cup \mathtt{pre}_?(a_{i+1})\end{array}}{\hat{s}_{k-1} \leftarrow \hat{s}_{k-1} \cup \{p(\boldsymbol{c})\} \;\big|\; \hat{s}_{k-1} \leftarrow \hat{s}_{k-1} \cup \{\neg p(\boldsymbol{c})\}} \tag{10}$$

In particular, for every transition $(\hat{s}_i, a_{i+1}, \hat{s}_{i+1})$ in $\hat{\mathcal{T}}$, if $\neg p(\boldsymbol{c})$ is in $\hat{s}_i$, the truth value of $p(\boldsymbol{c})$ is unknown in every state $\hat{s}_j$ with $i < j < k$, and $p(\boldsymbol{c})$ is in either $\hat{s}_k$ or the set of potential precondition of an action a_k, then rule (9) derives from $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ two new tuples $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_1)$ and $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_2)$ such that $p(c)$ is in the state $\hat{s}_{i+1}$ of $\hat{\mathcal{T}}_1$ and $\neg p(c)$ is in the state $\hat{s}_{i+1}$ of $\hat{\mathcal{T}}_2$. Intuitively, rule (9) makes two different assumptions on the truth value of $p(c)$ in $\hat{s}_{i+1}$ from which one can derive different conclusions for the effects of action a_{i+1}: the first tuple implies a set of models where $p(\boldsymbol{c})$ is in the definite positive effects of a_{i+1}, while the second tuple implies another set of models where $p(\boldsymbol{c})$ is not in the definite and potential positive effects of a_{i+1}.

Similarly, for every transition $(\hat{s}_{k-1}, a_k, \hat{s}_k)$ in $\hat{\mathcal{T}}$, rule (10) derives from $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ two new tuples $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_1)$ and $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_2)$ such that $p(c)$ is in the state $\hat{s}_{k-1}$ of $\hat{\mathcal{T}}_1$ and $\neg p(c)$ is in the state $\hat{s}_{k-1}$ of $\hat{\mathcal{T}}_2$. Therefore, rule (10) makes two different assumptions on the truth value of $p(c)$ in $\hat{s}_{k-1}$ from which one derives different conclusions for the preconditions and effects of action a_k: the first tuple implies a set of models for which $p(\boldsymbol{c})$ is in the definite negative effects of a_k, while the second tuple implies another set of models for which it is not in the potential preconditions of a_k. These two sets of models are incomparable w.r.t. the order $\prec$, and therefore we need to consider both of them for the computation of the minimal models.

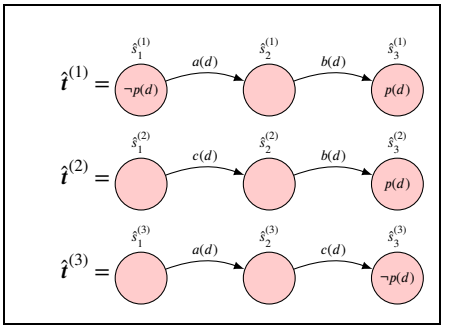
Definition 18 *(Completion tree).* A *completion tree for a set of traces* $\hat{\mathcal{T}}$ is a tree with root $(\mathtt{pre}^0_?, \mathtt{eff}^0_!, \mathtt{eff}^0_?, \hat{\mathcal{T}})$ where each node is a tuple $(\mathtt{pre}'_?, \mathtt{eff}'_!, \mathtt{eff}'_?, \hat{\mathcal{T}}')$ obtained by applying the rules (4)–(10) and, for any operator $op \in \mathcal{O}$, we have that

- $\mathtt{pre}^0_?(op) = \mathcal{P}(\boldsymbol{x})$ where $\mathcal{P}$ is the set of predicates with their arity occurring in $\hat{\mathcal{T}}$;
- $\mathtt{eff}^0_?(op) = \mathcal{L}it(\mathcal{P}(\boldsymbol{x}))$ where $\mathcal{L}it(\mathcal{P})$ is the set of literals with their arity occurring in $\hat{\mathcal{T}}$;
- $\mathtt{eff}^0_!(op) = \emptyset$.

Intuitively, at the beginning of the search of the minimal models, we have not yet ruled out any predicate (literal) from the set of the potential preconditions (effects), and hence the set of potential preconditions of an operator includes all the possible predicates, and the set of potential effects includes all the possible literals. Similarly, at the beginning, we have not included any literal to the set of the definite effects of an operator yet, and therefore such a set is empty. The application of rules (4)–(8) connects a node to exactly one child in the tree, while the application of rules (9)–(10) connects a node to two children. Let us see an example of completion tree.

Example 7. Consider the following set of partial traces $\hat{\mathcal{T}}_0 = \left\{\hat{\boldsymbol{t}}^{(1)}, \hat{\boldsymbol{t}}^{(2)}, \hat{\boldsymbol{t}}^{(3)}\right\}$.

$\hat{\mathcal{T}}_0 =$



At the beginning, the set of preconditions of each action is $p(x)$, the set of definite effects is empty, and the set of potential effects is $\{p(x), \neg p(x)\}$. The root of the completion tree is $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ where sets $\mathtt{pre}_?$, $\mathtt{eff}_!$, and $\mathtt{eff}_?$ are defined as shown in the next table.

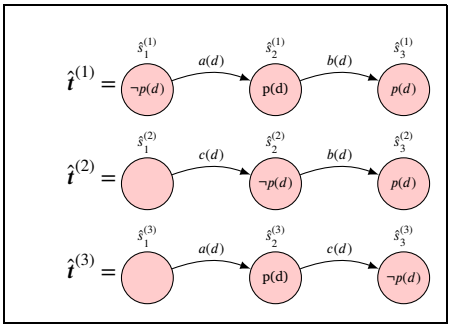
$a(x)$			$b(x)$			$c(x)$			Traces
$\texttt{pre}_?$	$\texttt{eff}_!$	$\texttt{eff}_?$	$\texttt{pre}_?$	$\texttt{eff}_!$	$\texttt{eff}_?$	$\texttt{pre}_?$	$\texttt{eff}_!$	$\texttt{eff}_?$	
$p(x)$	$\emptyset$	$p(x), \neg p(x)$	$p(x)$	$\emptyset$	$p(x), \neg p(x)$	$p(x)$	$\emptyset$	$p(x), \neg p(x)$	$\hat{\mathcal{T}}_0$

By applying rules (5) to the second transition of $\hat{\boldsymbol{t}}^{(1)}$ and $\hat{\boldsymbol{t}}^{(3)}$, we remove $\neg p(x)$ and $p(x)$ from the potential effects of operators b and c, respectively. Furthermore we apply rule (6) to the first transition of $\hat{\boldsymbol{t}}^{(1)}$ to remove $p(x)$ from the potential preconditions of operator a. The only applicable rules are then rule (9) to $\hat{\boldsymbol{t}}^{(1)}$ and rule (10) to $\hat{\boldsymbol{t}}^{(2)}$ and $\hat{\boldsymbol{t}}^{(3)}$. Specifically, by applying rule (9) to $\hat{\boldsymbol{t}}^{(1)}$, we obtain two sets of traces. After applying all possible rules for propagating effects and learning potential preconditions, definite and potential effects, we end up with two nodes of the completion tree representing the two tuples defined in the next table and figure.

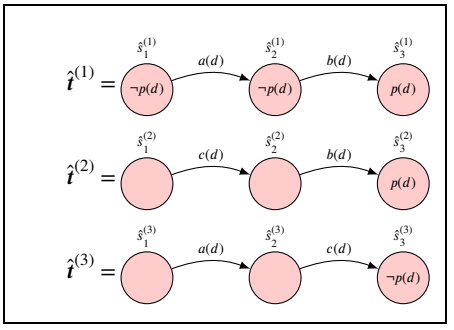
$a(x)$			$b(x)$			$c(x)$			Traces
$\texttt{pre}_?$	$\texttt{eff}_!$	$\texttt{eff}_?$	$\texttt{pre}_?$	$\texttt{eff}_!$	$\texttt{eff}_?$	$\texttt{pre}_?$	$\texttt{eff}_!$	$\texttt{eff}_?$	
$\emptyset$	$p(x)$	$\emptyset$	$\emptyset$	$p(x)$	$\emptyset$	$p(x)$	$\neg p(x)$	$\emptyset$	$\hat{\mathcal{T}}_1$
$\emptyset$	$\emptyset$	$p(x), \neg p(x)$	$\emptyset$	$p(x)$	$\emptyset$	$p(x)$	$\emptyset$	$\neg p(x)$	$\hat{\mathcal{T}}_2$

where $\hat{\mathcal{T}}_1$ and $\hat{\mathcal{T}}_2$ are defined as follows.

$\hat{\mathcal{T}}_1 =$

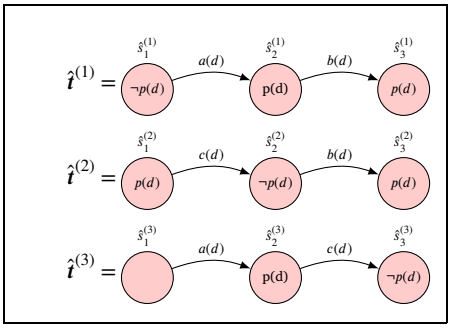


$\hat{\mathcal{T}}_2 =$

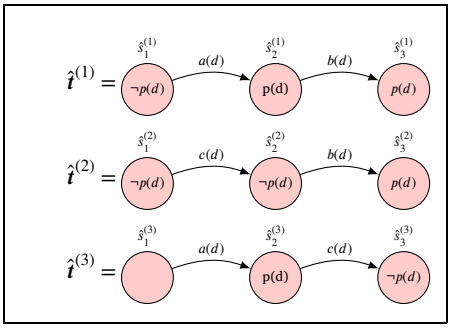


Since $p(d)$ is in the current set of potential preconditions of $c(d)$, we can apply rule (10) to $\hat{\boldsymbol{t}}^{(2)}$ in $\hat{\mathcal{T}}_1$ obtaining the following two set of traces $\hat{\mathcal{T}}_{11}$ and $\hat{\mathcal{T}}_{12}$.

$\hat{\mathcal{T}}_{11} =$



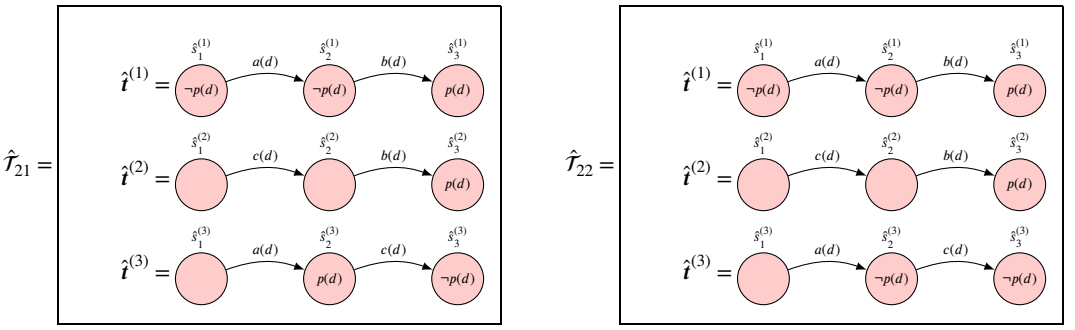
$\hat{\mathcal{T}}_{12} =$



After applying all possible rules, we end up with two leaf nodes of the completion tree representing the two tuples defined in the next table.

$a(x)$			$b(x)$			$c(x)$			Traces
$\texttt{pre}_?$	$\texttt{eff}_!$	$\texttt{eff}_?$	$\texttt{pre}_?$	$\texttt{eff}_!$	$\texttt{eff}_?$	$\texttt{pre}_?$	$\texttt{eff}_!$	$\texttt{eff}_?$	
$\emptyset$	$p(x)$	$\emptyset$	$\emptyset$	$p(x)$	$\emptyset$	$p(x)$	$\neg p(x)$	$\emptyset$	$\hat{\mathcal{T}}_{11}$
$\emptyset$	$p(x)$	$\emptyset$	$\emptyset$	$p(x)$	$\emptyset$	$\emptyset$	$\neg p(x)$	$\emptyset$	$\hat{\mathcal{T}}_{12}$

Similarly, we can apply rule (10) to $\hat{\boldsymbol{t}}^{(3)}$ of $\hat{\mathcal{T}}_2$ obtaining the following partial traces:



Finally, by applying all the possible rules we obtain two other leaf nodes of the completion tree representing the two tuples defined in the next table.

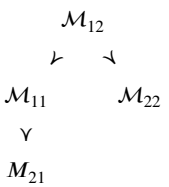
$a(x)$			$b(x)$			$c(x)$			Traces
$\mathtt{pre}_?$	$\mathtt{eff}_!$	$\mathtt{eff}_?$	$\mathtt{pre}_?$	$\mathtt{eff}_!$	$\mathtt{eff}_?$	$\mathtt{pre}_?$	$\mathtt{eff}_!$	$\mathtt{eff}_?$	
$\emptyset$	$\emptyset$	$p(x)$	$\emptyset$	$p(x)$	$\emptyset$	$p(x)$	$\neg p(x)$	$\emptyset$	$\hat{\mathcal{T}}_{21}$
$\emptyset$	$\emptyset$	$\neg p(x)$	$\emptyset$	$p(x)$	$\emptyset$	$\emptyset$	$\emptyset$	$\neg p(x)$	$\hat{\mathcal{T}}_{22}$

As one can see from the previous example, the application of rules (4)–(10) can still end up in a set of traces which are not complete. However, we have now enough information to derive the minimal models of the initial partial traces $\hat{\mathcal{T}}$. Indeed, as we prove in the next section, if $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}')$ is a leaf node of the completion tree for $\hat{\mathcal{T}}$, then: *(i)* the model $\mathcal{M}$ represented by the leaf, i.e., having precondition function $\mathtt{pre}_{\mathcal{M}} = \mathtt{pre}_?$ and effect function $\mathtt{eff}_{\mathcal{M}} = \mathtt{eff}_!$, is a model consistent with $\hat{\mathcal{T}}$; *(ii)* all the minimal models for $\hat{\mathcal{T}}$ are in the set of models represented by the leaves of the tree.

Example 8. The minimal models for the four sets of partial traces corresponding to the four leaves of the completion tree of Example 7 are the following.

Models	$a(x)$		$b(x)$		$c(x)$	
	pre	eff	pre	eff	pre	eff
$\mathcal{M}_{11}$	$\emptyset$	$p(x)$	$\emptyset$	$p(x)$	$p(x)$	$\neg p(x)$
$\mathcal{M}_{12}$	$\emptyset$	$p(x)$	$\emptyset$	$p(x)$	$\emptyset$	$\neg p(x)$
$\mathcal{M}_{21}$	$\emptyset$	$\emptyset$	$\emptyset$	$p(x)$	$p(x)$	$\neg p(x)$
$\mathcal{M}_{22}$	$\emptyset$	$\emptyset$	$\emptyset$	$p(x)$	$\emptyset$	$\emptyset$

Each model is minimal for the set of traces of the corresponding leaf, but not all of them are minimal w.r.t. the initial traces $\hat{\mathcal{T}}_0$. Indeed we have the following partial order.



Therefore, only $\mathcal{M}_{21}$ and $\mathcal{M}_{22}$ are minimal models for $\hat{\mathcal{T}}_0$. As we will see later, we can also say that there is no model different from $\mathcal{M}_{21}$ and $\mathcal{M}_{22}$ that is minimal for $\hat{\mathcal{T}}_0$.

5. Soundness and completeness of the completion rules

The main objective of this section is to formally prove that the set of leaves of a completion tree for a set of partial traces $\hat{\mathcal{T}}$ contains all the minimal models for $\hat{\mathcal{T}}$.

Lemma 1 *(Soundness of definite effects). If* $(\mathtt{pre}_?, \mathtt{eff}'_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ *is derived by applying rule* (4) *to a node of a completion tree representing* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$*, then for every action model* $\mathcal{M}$*,* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ *if and only if* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}'_!, \mathtt{eff}_?, \hat{\mathcal{T}})$.

Proof. ($\Longrightarrow$) Let $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$, and let $\langle \hat{s}_i, op(\boldsymbol{c}), \hat{s}_{i+1} \rangle \in \hat{\mathcal{T}}$ be the premise of rule (4). If the application of the rule adds literal $l(\boldsymbol{x})$ to $\mathtt{eff}_!(op)$, then $\overline{l(\boldsymbol{c})} \in \hat{s}_i$ and $l(\boldsymbol{c}) \in \hat{s}_{i+1}$. If s_i and s_{i+1} are completions of $\hat{s}_i$ and $\hat{s}_{i+1}$, respectively, then from the fact that $s_{i+1} = s_i \cup \mathtt{eff}_{\mathcal{M}}(op(\boldsymbol{c})) \setminus \overline{\mathtt{eff}_{\mathcal{M}}(op(\boldsymbol{c}))}$ we can conclude that $l(\boldsymbol{x}) \in \mathtt{eff}_{\mathcal{M}}(op)$, and therefore $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}'_!, \mathtt{eff}_?, \hat{\mathcal{T}})$.
($\Longleftarrow$) If $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}'_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ then $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$, since $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}$, $\mathtt{pre}_?$ and $\mathtt{eff}_?$ are not changed, $\mathtt{eff}_! \subseteq \mathtt{eff}'_! \subseteq \mathtt{eff}_{\mathcal{M}}$, and $\mathtt{eff}_{\mathcal{M}} \subseteq \mathtt{eff}'_! \cup \mathtt{eff}_? = \mathtt{eff}_! \cup \mathtt{eff}_?$. The latter equality holds because $\mathtt{eff}'_! \setminus \mathtt{eff}_! \subseteq \mathtt{eff}_?$. Indeed, in the root node of the completion tree $\mathtt{eff}_?$ is $\mathcal{L}it(\mathcal{P}(\boldsymbol{x}))$, and a literal $l(\boldsymbol{c})$ can be removed from $\mathtt{eff}_?(op)$ only if (i) $l(\boldsymbol{c})$ has already been added to $\mathtt{eff}_!(op)$, or (ii) $\overline{l(\boldsymbol{c})} \in \hat{s}_{i+1}$; in this latter case rule (4) could not add $l(\boldsymbol{c})$ to $\mathtt{eff}'_!(op)$. □

Lemma 2 *(Soundness of potential effects). If* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}'_?, \hat{\mathcal{T}})$ *is derived by applying rule* (5) *to a node of a completion tree representing* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$*, then for every action model* $\mathcal{M}$*,* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ *if and only if* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}'_?, \hat{\mathcal{T}})$.

Proof. ($\Longrightarrow$) Let $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$, and let $\langle \hat{s}_i, op(\boldsymbol{c}), \hat{s}_{i+1} \rangle \in \hat{\mathcal{T}}$ be the premise of rule (5). Suppose that the rule removes $l(\boldsymbol{x})$ from $\mathtt{eff}_?(op)$. Then $l(\boldsymbol{c}) \in \mathtt{eff}_!(op)$ or $\overline{l(\boldsymbol{c})} \in \hat{s}_{i+1}$. In the first case, $l(\boldsymbol{c})$ has already been added to $\mathtt{eff}_!(op)$ by rule (4). Therefore, $\mathtt{eff}_{!?}(op) = \mathtt{eff}'_{!?}(op)$, and we can say that $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}'_?, \hat{\mathcal{T}})$. In the second case, if s_i and s_{i+1} are completions of $\hat{s}_i$ and $\hat{s}_{i+1}$, respectively, then, from the fact that $s_{i+1} = s_i \cup \mathtt{eff}_{\mathcal{M}}(op(\boldsymbol{c})) \setminus \overline{\mathtt{eff}_{\mathcal{M}}(op(\boldsymbol{c}))}$ we can conclude that $p(\boldsymbol{x}) \notin \mathtt{eff}_{\mathcal{M}}(op)$, and therefore that $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}'_?, \hat{\mathcal{T}})$.
($\Longleftarrow$) If $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}'_?, \hat{\mathcal{T}})$ then $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$, since $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}$, $\mathtt{pre}_?$ and $\mathtt{eff}_!$ are not changed, and $\mathtt{eff}_! \subseteq \mathtt{eff}_{\mathcal{M}} \subseteq \mathtt{eff}_! \cup \mathtt{eff}'_? \subseteq \mathtt{eff}_! \cup \mathtt{eff}_?$. The latter inclusion relationship holds because $\mathtt{eff}'_? \subseteq \mathtt{eff}_?$. □

Lemma 3 *(Soundness of potential preconditions). If* $(\mathtt{pre}'_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ *is derived by applying rule* (6) *to a node of a completion tree representing* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$*, then for every action model* $\mathcal{M}$*,* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ *if and only if* $\mathcal{M} \vDash (\mathtt{pre}'_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$.

Proof. ($\Longrightarrow$) Let $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$, and let $\langle \hat{s}_i, op(\boldsymbol{c}), \hat{s}_{i+1} \rangle \in \hat{\mathcal{T}}$ be the premise of rule (6). Suppose that the rule removes $p(\boldsymbol{x})$ from $\mathtt{pre}_?(op(\boldsymbol{x}))$. Then $\neg p(\boldsymbol{c}) \in \hat{s}_i$. If s_i is a completion of $\hat{s}_i$, then since $op(\boldsymbol{c})$ can be executed in s_i if and only if $\mathtt{pre}(op(\boldsymbol{c})) \subseteq s_i$, we can conclude that $p(\boldsymbol{x}) \notin \mathtt{pre}_{\mathcal{M}}(op)$, and therefore that $\mathcal{M} \vDash (\mathtt{pre}'_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$.
($\Longleftarrow$) If $\mathcal{M} \vDash (\mathtt{pre}'_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ then $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ since $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}$, $\mathtt{eff}_!$ and $\mathtt{eff}_?$ are not changed, and $\mathtt{pre}_{\mathcal{M}} \subseteq \mathtt{pre}'_? \subseteq \mathtt{pre}_?$. □

Lemma 4 *(Soundness of propagating effects). If* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}')$ *is derived by applying rule* (7) *to a node of a completion tree representing* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$*, then for every action model* $\mathcal{M}$*,* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ *if and only if* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}')$.

Proof. ($\Longrightarrow$) Let $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$, and let $\langle \hat{s}_i, op(\boldsymbol{c}), \hat{s}_{i+1} \rangle \in \hat{\mathcal{T}}$ be the premise of rule (7). If the rule adds $l(\boldsymbol{c})$ to $\hat{s}_{i+1}$ in $\hat{\mathcal{T}}$, then $l(\boldsymbol{x}) \in \mathtt{eff}_!(op)$, which implies that $l(\boldsymbol{x}) \in \mathtt{eff}_{\mathcal{M}}(op)$. Let $\mathcal{T}$ be a completion of $\hat{\mathcal{T}}$ generated by $\mathcal{M}$; such a $\mathcal{T}$ exists since $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}$. The completion of $\hat{s}_{i+1}$ in $\mathcal{T}$ must contain $l(\boldsymbol{c})$ because $l(\boldsymbol{c}) \in \mathtt{eff}_{\mathcal{M}}(op)$. Therefore $\mathcal{T}$ is also a completion of $\hat{\mathcal{T}}'$, and therefore $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}'$.
($\Longleftarrow$) If $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}')$ then $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ since $\mathtt{pre}_?$, $\mathtt{eff}_!$ and $\mathtt{eff}_?$ are not changed, and $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}'$, which is a completion of $\hat{\mathcal{T}}$, and therefore $\mathcal{M}$ is also consistent with $\hat{\mathcal{T}}$. □

Lemma 5 *(Soundness of propagating non effects). If* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}')$ *is derived by applying rule* (8) *to a node of a completion tree representing* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$*, then for every action model* $\mathcal{M}$*,* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ *if and only if* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}')$.

Proof. ($\Longrightarrow$) Let $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$, and let $\langle \hat{s}, op(\boldsymbol{c}), \hat{s}' \rangle \in \hat{\mathcal{T}}$ be the premise of rule (8). Suppose that the rule adds $l(\boldsymbol{c})$ to $\hat{s}'$. This implies that $l(\boldsymbol{c}) \in \hat{s}$ and $\overline{l(\boldsymbol{x})} \notin \mathtt{eff}_{!?}(op)$. This in turn implies that $\overline{l(\boldsymbol{x})} \notin \mathtt{eff}_{\mathcal{M}}(op)$. Let $\mathcal{T}$ be a completion of $\hat{\mathcal{T}}$ generated by $\mathcal{M}$, and let s and s' be the completion of $\hat{s}$ and $\hat{s}'$, respectively, in $\mathcal{T}$. The fact that $l(\boldsymbol{c}) \in \hat{s} \subseteq s$ and that $\overline{l(\boldsymbol{x})} \notin \mathtt{eff}_{\mathcal{M}}(op)$ implies that $l(\boldsymbol{c}) \in s'$. Therefore, $\mathcal{T}$ is also a completion of $\hat{\mathcal{T}}'$ generated by $\mathcal{M}$. A similar argument can be done if $l(\boldsymbol{c})$ is added to s.
($\Longleftarrow$) If $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}')$ then $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ since $\mathtt{pre}_?$, $\mathtt{eff}_!$ and $\mathtt{eff}_?$ are not changed, and $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}'$, which is a completion of $\hat{\mathcal{T}}$, and therefore $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}$. □

Lemma 6 *(Soundness of case reasoning). If* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_1)$ *and* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_2)$ *are derived by applying rule* (9) *or* (10) *to a node of a completion tree representing* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$*, then for every action model* $\mathcal{M}$*,* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ *if and only if* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_1)$ *or* $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_2)$.

Proof. We prove the case of rule (9). The proof for rule (10) is similar.
($\Longrightarrow$) Let $\mathcal{T}$ be the completion of $\hat{\mathcal{T}}$ according to $\mathcal{M}$. Then the completion of $\hat{s}_{i+1}$ contains either $p(\boldsymbol{c})$ or $\neg p(\boldsymbol{c})$, which implies that it is either a completion of $\hat{\mathcal{T}}_1$ or of $\hat{\mathcal{T}}_2$. From this we can conclude that either $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_1)$ or $\mathcal{M} \vDash (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_2)$.

($\Longleftarrow$) If $\mathcal{M} \models (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}_i)$ for $i \in \{1,2\}$, from the fact that every completion of $\hat{\mathcal{T}}_i$ is a completion of $\hat{\mathcal{T}}$, we can conclude that $\mathcal{M} \models (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$. □

Since we are interested in finding the *minimal models* of a set of partial traces, it is crucial to determine how we can obtain these models from the leaves of the completion tree. First of all, recall that the leaves of a completion tree can contain non-minimal models of the initial partial trace. For instance, the completion tree of Example 7 contains four leaves that correspond to four action models consistent with the initial set of partial traces, but not all of them are minimal. From the partial order shown in Example 8, one can see that only two of them are minimal models. However, we can prove that all the minimal models are among the models represented by the leaves of the completion tree. In order to prove this completeness result, as the set of traces in a tuple represented by a leave of the completion tree may be partial, we introduce a way to complete these traces to obtain fully observable traces. For this purpose, we define the following transformations.

$$\frac{(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}} \quad l(\boldsymbol{c}) \in \hat{s} \quad \{l(\boldsymbol{c}), \overline{l(\boldsymbol{c})}\} \cap \hat{s}' = \emptyset}{\hat{s}' \leftarrow \hat{s}' \cup \{l(\boldsymbol{c})\}} \qquad \frac{(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}} \quad l(\boldsymbol{c}) \in \hat{s}' \quad \{l(\boldsymbol{c}), \overline{l(\boldsymbol{c})}\} \cap \hat{s} = \emptyset}{\hat{s} \leftarrow \hat{s} \cup \{l(\boldsymbol{c})\}} \tag{11}$$

$$\frac{\hat{\boldsymbol{t}} \in \hat{\mathcal{T}}, \{p(\boldsymbol{c}), \neg p(\boldsymbol{c})\} \cap \hat{s} = \emptyset, \ \forall \hat{s} \in \hat{\boldsymbol{t}}}{\hat{s} \leftarrow \hat{s} \cup \{p(\boldsymbol{c})\}, \ \forall \hat{s} \in \hat{\boldsymbol{t}}} \tag{12}$$

If $l(\boldsymbol{c}) \in \hat{s}$ for some state $\hat{s}$ in a trace of $\hat{\mathcal{T}}$, then rule (11) propagates this literal forward and backward along the trace. Note that it cannot happen that $l(\boldsymbol{c}) \in \hat{s}$ and $\overline{l(\boldsymbol{c})} \in \hat{s}'$ for some states $\hat{s}$ and $\hat{s}'$ of the same trace in $\hat{\mathcal{T}}$, because in this case rules (9) and (10) would be applicable and therefore $\hat{\mathcal{T}}$ would not be part of the tuple in a leaf node of the completion tree. If neither $p(\boldsymbol{c})$ nor $\neg p(\boldsymbol{c})$ is in $\hat{s}$ for any state $\hat{s}$ of a trace, then rule (12) arbitrarily adds $p(\boldsymbol{c})$ to every $\hat{s}$ of that trace. It is easy to see that the application of rules (11) and (12) to the partial traces in a leaf of a completion tree derives a fully observable set of traces.

Lemma 7. *Assume that* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ *is a leaf node of a completion tree, and* $\mathcal{T}$ *is the completion of* $\hat{\mathcal{T}}$ *according to rules* (11) *and* (12)*. Then, for every transition* $(s, op(\boldsymbol{c}), s') \in \mathcal{T}$ *that changes the truth value of* $l(\boldsymbol{c})$*, it must be* $l(\boldsymbol{c}) \in \hat{s}$ *and* $\overline{l(\boldsymbol{c})} \in \hat{s}'$ *where* $\hat{s}$ *and* $\hat{s}'$ *are the partial states of* $\hat{\mathcal{T}}$ *that have been completed in* s *and* s'*, respectively, with rules* (11) *and* (12)*.*

Proof. We prove the case in which the truth value of $p(\boldsymbol{c})$ is changed from true to false; the converse case can be proved in the same way. Rules (11) and (12) never introduce transitions that change the truth of literals along a trace of $\hat{\mathcal{T}}$, as $\hat{\mathcal{T}}$ is part of the tuple represented by a leaf of the completion tree. This implies that, if $p(\boldsymbol{c}) \in s$ and $\neg p(\boldsymbol{c}) \in s'$, then $p(\boldsymbol{c}) \in \hat{s}$ and $\neg p(\boldsymbol{c}) \in \hat{s}'$. Suppose by contradiction that $p(\boldsymbol{c}) \notin \hat{s}$ or $\neg p(\boldsymbol{c}) \notin \hat{s}'$. In the latter case, $\neg p(\boldsymbol{c}) \notin \hat{s}''$ for any partial state $\hat{s}''$ occurring after $\hat{s}'$ in the trace, otherwise the rule (10) would have been applied. Therefore state $\hat{s}'$ can only be filled by the application of rules (11) and (12) applied on the transition $(\hat{s}, op(\boldsymbol{c}), \hat{s}')$, but these rules would add $p(\boldsymbol{c})$ to s', contradicting the assumption that $\neg p(\boldsymbol{c}) \in s'$. An analogous argument can be done for the case in which $p(\boldsymbol{c}) \notin \hat{s}$. □

Lemma 8. *If* $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ *is a leaf node of a completion tree and every partial state* $\hat{s} \in \hat{\mathcal{T}}$ *is not contradictory, i.e.,* $\{p(\boldsymbol{c}), \neg p(\boldsymbol{c})\} \not\subseteq \hat{s}$*, then the model* $\mathcal{M}$ *having precondition function* $\mathtt{pre}_{\mathcal{M}} = \mathtt{pre}_?$ *and effect function* $\mathtt{eff}_{\mathcal{M}} = \mathtt{eff}_!$ *is a model consistent with* $\hat{\mathcal{T}}$*.*

Proof. Let $\mathcal{T}$ be the results of the application of rules (11) and (12) to $\hat{\mathcal{T}}$. The applications of rules (11) and (12) complete every trace of $\hat{\mathcal{T}}$ and do not introduce contradictory states. We show that $\mathcal{M}$ generates $\mathcal{T}$. For every $(s, op(\boldsymbol{c}), s') \in \mathcal{T}$, for every atom $p(\boldsymbol{c})$ we consider the four cases deriving from the combination of $p(\boldsymbol{c}) \in s$ or $\neg p(\boldsymbol{c}) \in s$ and $p(\boldsymbol{c}) \in s'$ or $\neg p(\boldsymbol{c}) \in s'$.

1. If $p(\boldsymbol{c}) \in s$ and $\neg p(\boldsymbol{c}) \in s'$, then by Lemma 7 $p(\boldsymbol{c}) \in \hat{s}$ and $\neg p(\boldsymbol{c}) \in \hat{s}'$, where $\hat{s}$ and $\hat{s}'$ are the partial states in $\hat{\mathcal{T}}$ corresponding to s and s' respectively. Therefore, since $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ is a leaf node, $\neg p(\boldsymbol{x})$ has already been added to $\mathtt{eff}_!(op)$ by rule (4).
2. If $\neg p(\boldsymbol{c}) \in s$ and $p(\boldsymbol{c}) \in s'$, we can reason as in previous case and show that $p(\boldsymbol{x})$ is part of $\mathtt{eff}_!(op)$.
3. If $p(\boldsymbol{c}) \in s$ and $p(\boldsymbol{c}) \in s'$, then $\neg p(\boldsymbol{x}) \notin \mathtt{eff}_!(op)$ because, if $\neg p(\boldsymbol{x})$ were in $\mathtt{eff}_!(op)$, rule (7) would have added $\neg p(\boldsymbol{c})$ to $\hat{s}'$. Therefore, $\neg p(\boldsymbol{c}) \in s'$ and state s' would contain a contradiction.
4. If $\neg p(\boldsymbol{c}) \in s$ and $\neg p(\boldsymbol{c}) \in s'$, for the effects we can reason as in the previous case and show that $p(\boldsymbol{x}) \notin \mathtt{eff}_!(op)$. Furthermore, concerning the set of preconditions, we distinguish two cases. First, suppose that $\neg p(\boldsymbol{c}) \in \hat{s}$. Then rule (6) has already removed $p(\boldsymbol{x})$ from $\mathtt{pre}_?(op)$. In the other case, when $\neg p(\boldsymbol{c}) \notin \hat{s}$, then $\neg p(\boldsymbol{c})$ has been added to s by rule (11). But this means that, even in such a case, $p(\boldsymbol{x})$ has already been removed from $\mathtt{pre}_?(op)$, otherwise one of the two rules (9) and (10) would be applicable, while rule (11) can be only applied to the tuples represented by the leaves of the completion tree. Therefore, we can conclude that $p(\boldsymbol{x}) \notin \mathtt{pre}_?(op)$. □

The next theorem states that rules (4)–(10) are sound w.r.r. the input set of partial traces $\hat{\mathcal{T}}$, i.e., the models derived using these rules are consistent with $\hat{\mathcal{T}}$.

Algorithm 1 $\text{OffLAM}_{\text{PT}}$.

Require: $\hat{\mathcal{T}}$ set of partial traces
1: $(\mathcal{P}, \mathcal{O}, \text{pre}_?, \text{eff}_!) \leftarrow$ ComputeMinimalModels($\hat{\mathcal{T}}$)
2: $(\mathcal{P}, \text{pre}, \text{eff}) \leftarrow$ ComputeCautiousModel($\mathcal{P}, \mathcal{O}, \text{pre}_?, \text{eff}_!$)
3: **return** $(\mathcal{P}, \mathcal{O}, \text{pre}, \text{eff})$

Theorem 1 *(Soundness of the derivation rules). If* $(\text{pre}_?, \text{eff}_!, \text{eff}_?, \hat{\mathcal{T}}')$ *is a leaf node of the completion tree for* $\hat{\mathcal{T}}$ *and every partial state* $\hat{s} \in \hat{\mathcal{T}}'$ *is not contradictory, i.e.,* $\{p(\boldsymbol{c}), \neg p(\boldsymbol{c})\} \not\subseteq \hat{s}$*, then the model* $\mathcal{M}$ *having precondition function* $\text{pre}_{\mathcal{M}} = \text{pre}_?$ *and effect function* $\text{eff}_{\mathcal{M}} = \text{eff}_!$ *is a model consistent with the initial partial traces* $\hat{\mathcal{T}}$.

Proof. Lemma 8 states that $\mathcal{M}$ is consistent with the partial traces $\hat{\mathcal{T}}'$ in the leaf node. By the "$\Longleftarrow$" direction of Lemmas (4)–(6), $\mathcal{M}$ is also consistent with the traces in the parent node of the leaf; hence, by iterating this reasoning, we can say that $\mathcal{M}$ is consistent with the partial traces $\hat{\mathcal{T}}$ in the root node of the completion tree. □

Lemma 9. *If* $(\text{pre}_?, \text{eff}_!, \text{eff}_?, \hat{\mathcal{T}})$ *is a leaf node of a completion tree and every partial state* $\hat{s} \in \hat{\mathcal{T}}$ *is not contradictory, i.e.,* $\{p(\boldsymbol{c}), \neg p(\boldsymbol{c})\} \not\subseteq \hat{s}$*, then the model* $\mathcal{M}$ *having precondition function* $\text{pre}_{\mathcal{M}} = \text{pre}_?$ *and effect function* $\text{eff}_{\mathcal{M}} = \text{eff}_!$ *is the only minimal model for* $\hat{\mathcal{T}}$.

Proof. Theorem 1 states that $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}$. Let $\mathcal{M}'$ be another model consistent with $\hat{\mathcal{T}}$. First, we show that $\text{pre}_{\mathcal{M}'} \subseteq \text{pre}_{\mathcal{M}} = \text{pre}_?$. By contradiction, suppose that this is not the case, i.e., there is a $p(\boldsymbol{x}) \in \text{pre}_{\mathcal{M}'}(op)$ such that $p(\boldsymbol{x}) \notin \text{pre}_?(op)$. This means that $p(\boldsymbol{x})$ has been removed from $\text{pre}_?(op)$ by the application of rule (6), which implies that there is a transition $(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}}$ such that $\neg p(\boldsymbol{c}) \in \hat{s}$, which implies in turn that, according to $\mathcal{M}'$, $op(\boldsymbol{c})$, is not applicable in any completion of $\hat{s}$, which contradicts the fact that $\mathcal{M}'$ is consistent with $\hat{\mathcal{T}}$. Then, we show that $\text{eff}_! = \text{eff}_{\mathcal{M}} \subseteq \text{eff}_{\mathcal{M}'}$. By contradiction, suppose that this is not the case, i.e., there is a literal $p(\boldsymbol{x}) \in \text{eff}_!(op)$ such that $p(\boldsymbol{x}) \notin \text{eff}_{\mathcal{M}'}(op)$. This means that $p(\boldsymbol{x})$ has been added to $\text{eff}_!(op)$ by the application of rule (4), which implies that there is a transition $(\hat{s}, op(\boldsymbol{c}), \hat{s}') \in \hat{\mathcal{T}}$ such that $\neg p(\boldsymbol{c}) \in \hat{s}$ and $p(\boldsymbol{c}) \in \hat{s}'$, which in turn implies that $\neg p(\boldsymbol{c}) \in s$ and $p(c) \in s'$, which contradicts the fact that $\mathcal{M}'$ is consistent with $\hat{\mathcal{T}}$. Therefore, since there cannot exist a model consistent with $\hat{\mathcal{T}}$ with a precondition set larger than $\text{pre}_{\mathcal{M}}$ *or* with an effect set smaller than $\text{eff}_{\mathcal{M}}$, we can conclude that $\mathcal{M}$ is the *only* minimal model for $\hat{\mathcal{T}}$. □

The previous lemma states that $\mathcal{M}$ is the only minimal model for a leaf node $(\text{pre}_?, \text{eff}_!, \text{eff}_?, \hat{\mathcal{T}}')$, but this does not necessarily mean that $\mathcal{M}$ is minimal w.r.t. the initial traces $\hat{\mathcal{T}}$, as pointed out in Example 8. However, we can show that all the minimal models of $\hat{\mathcal{T}}$ are represented by some leaf of the completion tree for $\hat{\mathcal{T}}$, i.e., all the minimal models can be computed by applying rules (4)–(10).

Theorem 2 *(Completeness of the completion rules). For every minimal model* $\mathcal{M}$ *of a set of partial traces* $\hat{\mathcal{T}}$*, there is a leaf of the completion tree of* $\hat{\mathcal{T}}$*, labelled with* $(\text{pre}_?, \text{eff}_!, \text{eff}_?, \hat{\mathcal{T}}')$ *having precondition function* $\text{pre}_{\mathcal{M}} = \text{pre}_?$ *and effect function* $\text{eff}_{\mathcal{M}} = \text{eff}_!$.

Proof. If $\mathcal{M}$ is a minimal model for $\hat{\mathcal{T}}$, then there must exists a leaf node $(\text{pre}_?, \text{eff}_!, \text{eff}_?, \hat{\mathcal{T}}')$ of the completion tree for $\hat{\mathcal{T}}$ where $\text{pre}_{\mathcal{M}} = \text{pre}_?$ and $\text{eff}_{\mathcal{M}} = \text{eff}_!$. Indeed, if $\mathcal{M}$ is a minimal model for $\hat{\mathcal{T}}$, then $\mathcal{M}$ is consistent with $\hat{\mathcal{T}}$. Since all models consistent with $\hat{\mathcal{T}}$ satisfy $(\text{pre}^0_?, \text{eff}^0_!, \text{eff}^0_?, \hat{\mathcal{T}})$, then $\mathcal{M}$ satisfies $(\text{pre}^0_?, \text{eff}^0_!, \text{eff}^0_?, \hat{\mathcal{T}})$. Every model satisfying $(\text{pre}^0_?, \text{eff}^0_!, \text{eff}^0_?, \hat{\mathcal{T}})$ also satisfies at least one of the leaf nodes $(\text{pre}_?, \text{eff}_!, \text{eff}_?, \hat{\mathcal{T}}')$, because of the "$\Longrightarrow$" direction of Lemmas 1–5 and the "or" in the claim of Lemma 6. Therefore, there must exists a leaf node $(\text{pre}_?, \text{eff}_!, \text{eff}_?, \hat{\mathcal{T}}')$ such that $\mathcal{M}$ satisfies the node. By Lemma 9, there exists only one minimal model $\mathcal{M}'$ consistent with $\hat{\mathcal{T}}'$ such that $\text{pre}_{\mathcal{M}'} = \text{pre}_?$ and $\text{eff}_{\mathcal{M}'} = \text{eff}$. Since $\mathcal{M}$ is minimal and consistent with $\hat{\mathcal{T}}'$ then $\mathcal{M} = \mathcal{M}'$. □

6. Algorithm $\text{OffLAM}_{\text{PT}}$

This section describes a sound and complete algorithm, called $\text{OffLAM}_{\text{PT}}$ (Offline Learning of Action Models from Partial Traces), which computes the set of minimal action models and the cautious model for a set of partial traces $\hat{\mathcal{T}}$. In this section, we focus on traces with totally observable actions and partially observable states. In the next section, we will show how to modify $\text{OffLAM}_{\text{PT}}$ to treat traces with both partially observable states and actions.

Algorithm 1 shows the pseudo-code of $\text{OffLAM}_{\text{PT}}$. In line 1, $\text{OffLAM}_{\text{PT}}$ invokes algorithm ComputeMinimalModels to compute *all* the minimal models for an input set of partial traces $\hat{\mathcal{T}}$ (see Definition 15); in line 2, $\text{OffLAM}_{\text{PT}}$ invokes algorithm ComputeCautiousModel to compute the cautious model for $\hat{\mathcal{T}}$ (see Definition 16). Finally, the algorithm returns the set of predicates with their arity, and the set of operators with their arity, preconditions and effects (line 3).

Given an input set of partial traces, $\hat{\mathcal{T}}$, basically, ComputeMinimalModels iteratively applies the completion rules presented in Section 4; the iterative application of these rules constructs the completion tree, from which one can derive all the minimal models. However, ComputeMinimalModels does not construct the completion tree as depicted in Examples 7 and 8, but it constructs a compact representation of the tree. In particular, the algorithm sequentially applies all the completion rules with the exception of rules (9) and (10) that generate two sub-trees in the completion tree. Since a large number of expansion rules can be applied in both the sub-trees, instead of splitting the tree into two sub-trees with rules (9) and (10), ComputeMinimalModels uses two alternative transformations. In place of rule (9), the algorithm adopts the following rule:

Algorithm 2 COMPUTEMINIMALMODELS.

Require: $\hat{\mathcal{T}}$ set of partial traces
1: $\mathcal{P} \leftarrow \{p \mid p(\boldsymbol{c}) \in \hat{\mathcal{T}} \vee \neg p(\boldsymbol{c}) \in \hat{\mathcal{T}}\}$
2: $\mathcal{O} \leftarrow \{op \mid op(\boldsymbol{c}) \in \hat{\mathcal{T}}\}$
3: **for** $op \in \mathcal{O}$ **do**
4: $\quad \mathsf{pre}_?(op) \leftarrow \mathcal{P}(\boldsymbol{x})$
5: $\quad \mathsf{eff}_?(op) \leftarrow \mathcal{L}it(\mathcal{P}(\boldsymbol{x}))$
6: $\quad \mathsf{eff}_!(op) \leftarrow \emptyset$
7: **end for**
8: **repeat**
9: $\quad$ **repeat**
10: $\quad\quad$ Apply rules (4)–(8) to $(\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}})$
11: $\quad$ **until** $(\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}})$ does not change
12: $\quad$ Apply rules (13)-(14) to $(\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}})$
13: **until** $(\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}})$ does not change
14: **return** $(\mathcal{P}, \mathcal{O}, \mathsf{pre}_?, \mathsf{eff}_!)$

$$\frac{\begin{array}{c}(\hat{s}_i, a_{i+1}, \hat{s}_{i+1}) \dots (\hat{s}_{k-1}, a_k, \hat{s}_k) \in \hat{\mathcal{T}} \\ \{p(\boldsymbol{c}), \neg p(\boldsymbol{c})\} \cap (\hat{s}_j \cup \mathsf{pre}_?(a_j)) = \emptyset \ \forall j \text{ s.t. } i < j < k \\ \neg p(\boldsymbol{c}) \in \hat{s}_i \text{ and } p(\boldsymbol{c}) \in \hat{s}_k \cup \mathsf{pre}_?(a_k)\end{array}}{\begin{array}{c}\hat{s}_{i+1} \leftarrow \hat{s}_{i+1} \cup \{p_1(\boldsymbol{c}), \neg p_2(\boldsymbol{c})\} \\ (\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}}) \leftarrow (\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}})[p/\{p_1, p_2\}]\end{array}} \tag{13}$$

where notation $(\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}})[p/\{p_1, \dots, p_n\}]$ means that every literal with p as predicate in potential preconditions, definite effects, potential preconditions or set of traces is replaced with n literals, $p_1, \dots, p_n$. For example, $\{p(x,y)\}[p/p_1, p_2]$ is equal to: $\{p_1(x,y), p_2(x,y)\}$. Intuitively, the two alternative assumptions made by rule (9) on the truth value of $p(\boldsymbol{c})$ in $\hat{s}_{i+1}$ are both in the (single) consequence of (13), as $p_1(\boldsymbol{c})$ in $\hat{s}_{i+1}$ represents the assumption that the truth value of $p(\boldsymbol{c})$ is true in $\hat{s}_{i+1}$, while $\neg p_2(\boldsymbol{c})$ in $\hat{s}_{i+1}$ represents the opposite assumption. Therefore, the two different conclusions that can be subsequently derived on the effects of action a_{i+1} from the assumptions made by rule (9) can be still both derived by using p_1 and p_2 in place of p.

Similarly, in place of rule (10) the algorithm adopts the following rule:

$$\frac{\begin{array}{c}(\hat{s}_i, a_{i+1}, \hat{s}_{i+1}) \dots (\hat{s}_{k-1}, a_k, \hat{s}_k) \in \hat{\mathcal{T}} \\ \{p(\boldsymbol{c}), \neg p(\boldsymbol{c})\} \cap (\hat{s}_j \cup \mathsf{pre}_?(a_j)) = \emptyset \ \forall j \text{ s.t. } i < j < k \\ \neg p(\boldsymbol{c}) \in \hat{s}_k \text{ and } p(\boldsymbol{c}) \in \hat{s}_i \cup \mathsf{pre}_?(a_{i+1})\end{array}}{\begin{array}{c}\hat{s}_{k-1} \leftarrow \hat{s}_{k-1} \cup \{p_1(\boldsymbol{c}), \neg p_2(\boldsymbol{c})\} \\ (\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}}) \leftarrow (\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}})[p/\{p_1, p_2\}]\end{array}} \tag{14}$$

Intuitively, both the two alternative assumptions made by rule (10) on the truth value of $p(\boldsymbol{c})$ in $\hat{s}_{k-1}$ are in the consequence of (14), because $p_1(\boldsymbol{c})$ in $\hat{s}_{k-1}$ represents the assumption that the truth value of $p(\boldsymbol{c})$ is true in $\hat{s}_{k-1}$, while $\neg p_2(\boldsymbol{c})$ in $\hat{s}_{k-1}$ represents the opposite assumption.

We are now ready to describe the algorithm used by $\mathsf{OffLAM}_{\mathsf{PT}}$ for the computation of the minimal models consistent with a set of partial traces; its pseudo code is shown in Algorithm 2. Lines 1 and 2 of the algorithm define the sets of predicates $\mathcal{P}$ and operators $\mathcal{O}$ on the basis of the input set of partial traces $\hat{\mathcal{T}}$. Loop 3–7 defines $\mathsf{pre}_?$, $\mathsf{eff}_!$, and $\mathsf{eff}_?$ for the root of the completion tree for $\hat{\mathcal{T}}$. Loop 8–12 applies the rules (4)–(8) and (13)-(14) to the tuple $(\mathsf{pre}_?, \mathsf{eff}_!, \mathsf{eff}_?, \hat{\mathcal{T}})$. These operations are repeated until a fix point is reached. For the sake of efficiency, before making any assumption on the truth value of an atom in the set of partial traces (line 12), we derive all the possible potential preconditions and potential/definite effects from the current set of partial traces, and extend the partial traces on the basis of the action models under construction (loop 9–11). Finally, line 14 of the algorithm returns the set of predicates with their arity, $\mathcal{P}$, the set of operators with their arity, $\mathcal{O}$, the function associating operators with their own potential preconditions, $\mathsf{pre}_?$, and the function associating operators with their own definite effects, $\mathsf{eff}_!$.

Note that the computation of the minimal models could be done one predicate at a time. However, our rules (4)–(8) and (13)–(14) are defined on single transitions, rather than single predicates. Therefore, for the sake of presentation, in loop 8–13, Algorithm 2 updates the potential preconditions and effects, the definite effects, and the partial traces by considering all the predicates in $\mathcal{P}$ together.

The output returned by COMPUTEMINIMALMODELS($\hat{\mathcal{T}}$) is defined on signatures of predicates that are different from the ones in the original partial traces $\hat{\mathcal{T}}$. In particular, for every predicate p in $\hat{\mathcal{T}}$, the output returned by COMPUTEMINIMALMODELS($\hat{\mathcal{T}}$) uses p_σ, where σ is a string in $\{1,2\}^*$. If σ is the empty string ϵ, then no split rule (13) or (14) has been applied on the atoms with predicate p; on the contrary, if $\sigma \neq \epsilon$, then some split rule has been applied to an atom of the form $p(\boldsymbol{c})$ for some constant $\boldsymbol{c}$. To extract all the minimal models for $\hat{\mathcal{T}}$ from the output of COMPUTEMINIMALMODELS($\hat{\mathcal{T}}$), it is sufficient to select, for every predicate p, one particular string σ, substitute p_σ with p, and drop all the other predicates having signatures derived from p.

Theorem 3 *(Termination).* COMPUTEMINIMALMODELS *terminates.*

Algorithm 3 COMPUTECAUTIOUSMODEL.

Require: $\mathcal{P}, \mathcal{O}, \mathtt{pre}_?, \mathtt{eff}_!$
1: **repeat**
2: **for** $op \in \mathcal{O}$ **do**
3: $\mathtt{pre}_?(op) \leftarrow \bigcup_{i\in\{1,2\}} \mathtt{pre}_?[p_{\sigma i}/p_\sigma](op)$
4: $\mathtt{eff}_!(op) \leftarrow \bigcap_{i\in\{1,2\}} \mathtt{eff}_![p_{\sigma i}/p_\sigma](op)$
5: **end for**
6: $\mathcal{P} \leftarrow \bigcup_{i\in\{1,2\}} \mathcal{P}[p_{\sigma i}/p_\sigma]$
7: **until** $\mathcal{P}$ is not changed
8: **return** $(\mathcal{P}, \mathtt{pre}_?, \mathtt{eff}_!)$

Proof. At every iteration, COMPUTEMINIMALMODELS either extends some state $\hat{s}$ of $\hat{\mathcal{T}}$ with a ground literal, or adds some lifted literals to $\mathtt{eff}_!(op)$, or removes some lifted literal from $\mathtt{pre}_?(op)$ or $\mathtt{eff}_?(op)$, for some op. Since the number of partial states in $\hat{\mathcal{T}}$ are finite, and the number of ground and lifted literals are finite, after a finite number of steps the process must converge to a fixpoint, and the algorithm terminates. □

Theorem 4 *(Computational complexity). The time complexity of* COMPUTEMINIMALMODELS *is* $O((2^{Len(\hat{\mathcal{T}})} \cdot |\hat{\mathcal{T}}| \cdot Len(\hat{\mathcal{T}}))^3)$*, where* $Len(\hat{\mathcal{T}})$ *is the sum of the length of all the input partial traces, i.e., the total number of transitions in* $\hat{\mathcal{T}}$*, and* $|\hat{\mathcal{T}}|$ *is the size of the input partial traces, i.e., the sum of the number of literals and actions in* $\hat{\mathcal{T}}$*.*

Proof. Lines 1–2 require iterating over all literals and actions in the input set of traces, which requires $O(|\hat{\mathcal{T}}|)$ operations. Loop 3–7 iterates $|\mathcal{O}|$ times, and the most expensive operation in the loop requires $\Theta(|\mathcal{P}|)$ operations. Therefore, the total number of operations required by loop 3–7 is $\Theta(|\mathcal{O}| \cdot |P|) \leq \Theta(Len(\hat{\mathcal{T}}) \cdot |\hat{\mathcal{T}}|)$. As for the loop 8–13, the most expensive rules are (13)-(14), because they apply to pairs of transitions, while rules (4)–(8) apply to single transitions. Moreover, it is important to note that all rules in loop 8–13 apply to the set of literals augmented by rules (13)-(14). For each atom, these rules introduce up to $O(2^{Len(\hat{\mathcal{T}})})$ dummy atoms into the current set of traces. Therefore, in the worst case the number of atoms in the set of augmented traces is up to $O(2^{Len(\hat{\mathcal{T}})} \cdot |\mathcal{P}(C)|) \leq O(2^{Len(\hat{\mathcal{T}})} \cdot |\hat{\mathcal{T}}|)$. For each atom, rules (13)-(14) scan the states and the precondition sets of actions included between a pair of transitions. Hence, in the worst case, the computational cost of these rules is $O((2^{Len(\hat{\mathcal{T}})} \cdot |\hat{\mathcal{T}}|))^2 \cdot (Len(\hat{\mathcal{T}}))^2)$. In the worst case, each iteration of the loop 8–13 either adds or removes a (dummy) atom in a (partial) state, precondition or effect sets in the current set of traces. Therefore, the number of iterations of loop 8–13 is $O(2^{Len(\hat{\mathcal{T}})} \cdot |\hat{\mathcal{T}}| \cdot Len(\hat{\mathcal{T}}))$, and hence its time complexity is $O((2^{Len(\hat{\mathcal{T}})} \cdot |\hat{\mathcal{T}}| \cdot Len(\hat{\mathcal{T}}))^3)$. □

Algorithm 3 shows the pseudo-code of COMPUTECAUTIOUSMODEL. Intuitively, the algorithm applies the operations in Definition 16 with the signatures of predicates returned by COMPUTEMINIMALMODELS. Notation $\mathcal{X}[p_{\sigma i}/p_\sigma]$ indicates the set $\mathcal{X}$ obtained by changing $p_{\sigma i}$ to p_σ, for some string σi. For example, if both $p_{\sigma 1}(\boldsymbol{x})$ and $p_{\sigma 2}(\boldsymbol{x})$ are in $\mathtt{eff}_!(op)$, then they are both substituted in $\mathtt{eff}_!(op)$ with $p_\sigma(\boldsymbol{x})$. This is iteratively repeated so that $p(\boldsymbol{x})$ is included into $\mathtt{eff}_!(op)$ if $\mathtt{eff}_!(op)$ initially contains all the signatures introduced by COMPUTEMINIMALMODELS for p (Line 4). Similarly, $p(\boldsymbol{x})$ is included into $\mathtt{pre}_?(op)$ if $\mathtt{pre}_?(op)$ initially contains at least one signature introduced by COMPUTEMINIMALMODELS for p (Line 3). Moreover, all the signature introduced by COMPUTEMINIMALMODELS for p in set $\mathcal{P}$ are replaced with p (Line 6). At the end, COMPUTECAUTIOUSMODEL returns the restored set $\mathcal{P}$, and the revised functions $\mathtt{pre}_?$ and $\mathtt{eff}_!$ (Line 8). Essentially, the output of the algorithm states an action model having $\mathtt{pre}_?$ as precondition function, and $\mathtt{eff}_!$ as effect function. Such an action model is the cautious model for the set of partial traces given as input to $\mathsf{OffLAM}_{\mathsf{PT}}$.

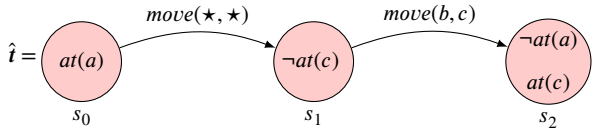
7. Learning action models from partial traces with partial actions

In this section, we focus on traces that contain partial information about the executed actions. In particular, we extend the set of completion rules and the algorithms presented in Section 6 to cope with these traces with partially observable actions. A partially observed action $\hat{a}$ is a regular expression that describes a set of ground actions with terminal symbols $\mathcal{C} \cup \mathcal{O} \cup D$ where D is the set of delimiters containing "(", ")", and ",". A partial action matches a set of ground actions. For instance, the expression $\star$ matches all the ground actions; $\star(a, \star)$ matches the ground actions that apply some binary operator in $\mathcal{O}$ to the constant a and some other constants in $\mathcal{C}$, e.g., $move(a, b)$, and $move(a, c)$. Given a partial transition $\langle \hat{s}, \hat{a}, \hat{s}' \rangle$, $\hat{a}$ can be grounded with a subset of actions that match $\hat{a}$, depending on the specific partial transition and from the model that we have learned so far. We therefore define the set $A_{\langle \hat{s},\hat{a},\hat{s}' \rangle}(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?)$ of ground actions that are compatible with $\langle \hat{s}, \hat{a}, \hat{s}' \rangle$ w.r.t. $\langle \mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_? \rangle$ as:

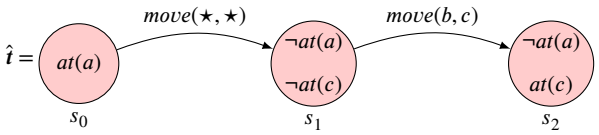
$$A_{\langle \hat{s},\hat{a},\hat{s}' \rangle}(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?) = \left\{ op(\boldsymbol{c}) \;\middle|\; \begin{array}{l} op(\boldsymbol{c}) \text{ matches with } \hat{a} \\ \text{if } l(\boldsymbol{x}) \in \mathtt{eff}_!(op) \text{ then } \overline{l(\boldsymbol{c})} \notin \hat{s}' \\ \text{if } l(\boldsymbol{x}) \notin \mathtt{eff}_{!?}(op) \text{ then } \overline{l(\boldsymbol{c})} \notin \hat{s} \text{ or } l(\boldsymbol{c}) \notin \hat{s}' \\ \text{if } l(\boldsymbol{d}) \in \hat{s} \text{ and } \overline{l(\boldsymbol{d})} \in \hat{s}' \text{ then } \boldsymbol{d} \subseteq \boldsymbol{c} \end{array} \right\} \tag{15}$$

Intuitively, $A_{\langle \hat{s},\hat{a},\hat{s}' \rangle}(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?)$ is the set of *candidate actions* of a model larger than or equal to $(\mathtt{pre}_?, \mathtt{eff}_!)$ and smaller than or equal to $(\emptyset, \mathtt{eff}_{!?})$ that can produce a transition from a state s to s', such that $\hat{s} \subseteq s$ and $\hat{s}' \subseteq s'$.

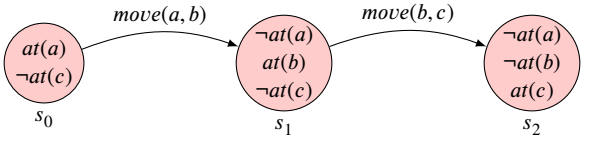
Example 9. Consider the following partial trace with partial information about the first action in the trace.



Suppose that $\texttt{eff}_!(move(x,y))$ is empty. The trace contains the three constants a, b, and c. By the first condition in Formula (15), we can say that $A_{\langle s_0, move(\star,\star), s_1\rangle}(\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?)$ is the set of all the possible ground actions, i.e., {move(a,b), move(a,c), move(b,a), move(b,c), move(c,a), move(c,b)}. By applying rule (6) to the second transition of $\hat{t}$, we remove predicate $at(y)$ from $\texttt{pre}_?(move(x,y))$. Moreover, by applying rule (8) to the second transition of $\hat{t}$, we add $\neg at(a)$ to state s_1 of $\hat{t}$ obtaining the following trace:



By applying rule (4) to the second transition of $\hat{t}$, we add $at(y)$ to set $\texttt{eff}_!(move(x,y))$. Therefore, by the second condition in Formula (15), $A_{\langle s_0, move(\star,\star), s_1\rangle}(\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?)$ is the set of ground actions that matches $move(\star,\star)$ such that $at(a)$ and $at(c)$ do not belong to their set of definite effects, as these predicates are false in s_1 and $at(y)$ is a definite effect of $move(x,y)$. This rules out actions $move(\star,a)$ and $move(\star,c)$ from $A_{\langle s_0, move(\star,\star), s_1\rangle}(\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?)$, and therefore we are left with $move(a,b)$ and $move(c,b)$. Moreover, the truth value of predicate $at(a)$ in s_1 is different from s_0 and the only action involving constant a in these latter two actions is $move(a,b)$. So, by the fourth condition in Formula (15), we can say that $A_{\langle s_0, move(\star,\star), s_1\rangle}(\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?) = \{move(a,b)\}$, and hence the first action in $\hat{t}$ is necessarily $move(a,b)$. This also enables further reasoning. Indeed, we can apply rule (4) to the first transition of $\hat{t}$ by adding $\neg at(x)$ to $\texttt{eff}_!(move(x,y))$, apply rule (7) to the first and second transition of $\hat{t}$, and apply rule (8) to the second transition of $\hat{t}$ by deriving the following trace.



From which we get that the final model is $\texttt{pre}_?(move(x,y)) = \{at(x)\}$ and $\texttt{eff}(move(x,y)) = \{\neg at(x), at(y)\}$.

The previous example shows that the reasoning on the partial actions contributes to extend the partial traces and discover further knowledge about the action model. In the example, the set $A_{\langle \hat{s}, \hat{a}, \hat{s}'\rangle}(\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?)$ ends up with only a single element; however, in general, such a set could contain more than one action. To deal with this situation, we extend the set of rules provided for partially observable states with a new rule.

$$\frac{\langle \hat{s}, \hat{a}, \hat{s}'\rangle \in \hat{\mathcal{T}} \qquad A_{\langle \hat{s}, \hat{a}, \hat{s}'\rangle}(\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?) = \{a_1, \ldots, a_n\}}{\hat{a} \leftarrow a_1 \mid \hat{a} \leftarrow a_2 \mid \cdots \mid \hat{a} \leftarrow a_n} \tag{16}$$

This rule splits the search for a minimal model in as many cases as the number of actions that are compatible with the transition $\langle \hat{s}, \hat{a}, \hat{s}'\rangle$ according to the already learned approximated model $(\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?)$. Consequently, partial actions is a source of complexity, as in the worst case, when no information is available in states s and s', the set can contain the entire set of ground actions, and we have to consider $O(|\mathcal{O}| \times |C|^m)$ cases, where m is the maximum arity of the operator contained in the partial traces. Since rule (16) considers all the possible cases, we have the following Lemma.

Lemma 10 *(Partially observable rule soundness). If* $\{(\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?, \hat{\mathcal{T}}_i)\}_{i=1}^n$ *is derived by applying rule* (16) *to a node of a completion tree* $(\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?, \hat{\mathcal{T}})$*, then for every action model* $\mathcal{M}$*,* $\mathcal{M} \vDash (\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?, \hat{\mathcal{T}})$ *if and only if* $\mathcal{M} \vDash (\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?, \hat{\mathcal{T}}_i)$ *for some* $1 \le i \le n$*.*

Proof. To prove the lemma it is enough to show that any completion of $\hat{t}$ instantiate partial action $\hat{a}_{i+1}$ in $\hat{t}$ with at least one action a in $A_{(\hat{s}_i, a_{i+1}, \hat{s}_{i+1})}(\texttt{pre}_?, \texttt{eff}_!\texttt{eff}_?)$. Suppose by contradiction that there is a completion of $\hat{a}$ with an action $op(\boldsymbol{c})$ such that $op(\boldsymbol{c}) \notin A_{(\hat{s}_i, a_{i+1}, \hat{s}_{i+1})}(\texttt{pre}_?, \texttt{eff}_!, \texttt{eff}_?)$. This implies that one of the following conditions holds:

- $l(\boldsymbol{x}) \in \texttt{eff}_!(op)$ and $\overline{l(\boldsymbol{c})} \in \hat{s}_{i+1}$. This implies that $l(\boldsymbol{c}) \in \texttt{eff}_!(op(\boldsymbol{c}))$ and therefore $l(\boldsymbol{c}) \in s_{i+1}$ for all the completions of $\hat{t}$, contradicting the fact that $\overline{l(\boldsymbol{c})} \in \hat{s}_{i+1} \subseteq s_{i+1}$.

- $l(\boldsymbol{x}) \notin \mathtt{eff}_{!?}(op)$ and $\overline{l(\boldsymbol{c})} \in \hat{s}$ and $l(\boldsymbol{c}) \in \hat{s}'$. This is also impossible because the effects of op in all the models $\mathcal{M}$ that satisfy $(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})$ cannot contain $l(\boldsymbol{x})$ as effect, while in every completion of the transition $(\hat{s}_i, \hat{a}_{i+1}, \hat{s}_{i+1})$ the value of $l(\boldsymbol{c})$ changes, which is a contradiction.
- $l(\boldsymbol{d}) \in \hat{s}$ and $\overline{l(\boldsymbol{d})} \in \hat{s}$ and $\boldsymbol{d} \nsubseteq \boldsymbol{c}$. This implies that some literal that contains a constant not in the parameters of $op(\boldsymbol{c})$ changes. But this contradicts the assumption that if an action changes the truth value of a literal, then all the constants that appear in the literals are arguments of the actions. □

The above Lemma guarantees that Theorems 1 and 2 also hold in the case of partially observable actions.

To apply rule (16), we can follow the same approach used for rules (9) and (10). Indeed, instead of splitting the completion tree into several sub-trees, we duplicate the set of predicates for which we have not taken a decision about being a potential precondition or a definite effect. Formally, given a subset of ground actions $\mathcal{A}' \subseteq \mathcal{A}$,

$$\mathcal{P}_{\mathcal{A}'}(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?) = \{p \in \mathcal{P} \mid p(\boldsymbol{c}) \in \mathtt{pre}_?(\mathcal{A}') \text{ or } \{p(\boldsymbol{c}), \neg p(\boldsymbol{c})\} \cap \mathtt{eff}_!(\mathcal{A}') = \emptyset\}$$

where $\mathtt{pre}_?(\mathcal{A}') = \bigcup_{a \in \mathcal{A}'} \mathtt{pre}_?(a)$ and $\mathtt{eff}_!(\mathcal{A}') = \bigcap_{a \in \mathcal{A}'} \mathtt{eff}_!(a)$. In place of rule (16), $\mathsf{OffLAM}_{\mathsf{PT}}$ adopts the following rule:

$$\frac{\begin{array}{c}\langle \hat{s}_i, \hat{a}_{i+1}, \hat{s}_{i+1} \rangle \in \hat{\boldsymbol{t}}, \hat{\boldsymbol{t}} \in \hat{\mathcal{T}} \\ A_{\langle \hat{s}_i, \hat{a}_{i+1}, \hat{s}_{i+1} \rangle}(\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?) = \{a_1, \ldots, a_n\} \\ \mathcal{P}_{\{a_1, \ldots, a_n\}} = \{p^1, p^2, \ldots, p^m\}\end{array}}{\begin{array}{c}\hat{\mathcal{T}} \leftarrow \hat{\mathcal{T}} \setminus \{\hat{\boldsymbol{t}}\} \cup \{\hat{\boldsymbol{t}}[\hat{a}_{i+1}/a_1], \ldots, \hat{\boldsymbol{t}}[\hat{a}_{i+1}/a_n]\} \\ (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}}) \leftarrow (\mathtt{pre}_?, \mathtt{eff}_!, \mathtt{eff}_?, \hat{\mathcal{T}})[p^1/\{p^1_1, \ldots, p^1_n\}, \ldots, p^m/\{p^m_1, \ldots, p^m_n\}]\end{array}} \tag{17}$$

Rule (17) creates n copies of the trace in $\hat{\boldsymbol{t}}$ that contains transition $(\hat{s}_i, \hat{a}_{i+1}, \hat{s}_{i+1})$. In each copy of the trace, the partially observed action $\hat{a}_{i+1}$ is instantiated with a ground action that is compatible with $\hat{a}_{i+1}$. Let p be a predicate that has not been already removed from the potential preconditions, or added to the definite effects of all the actions compatible with $\hat{a}_{i+1}$. Then every occurrence of a literal with predicate equal to p is replaced with n copies of the literal, where p is replaced with $p_1, \ldots, p_n$. For example, suppose that the set of actions that are initially compatible with partial action $move(\star, \star)$ is 3 and $at(y)$ has not been already added to $\mathtt{eff}_!(move(x, y))$. Let a and b be the constants mentioned in the partial traces. Then, $at(a)$ is replaced with $at_1(a)$, $at_2(a)$, and $at_3(a)$, and similarly for $at(b)$.

The pseudo-code of Algorithm ComputeMinimalModels for traces with both partially observable states and actions is the same as in Algorithm 2 with the addition of rule (17) into Line 12. Furthermore, Algorithm 3 is modified to consider $p_1, \ldots, p_n$ instead of p_1 and p_2. If we applied rule (17) to traces that contain a large amount of partially observed actions, the number of dummy predicates added to the trace would blowup. Therefore, the application of this rule can be much more expensive than others; for instance, while an application of rules (13) and (14) introduces 2 dummy predicates, an application of rule (17) introduces up to $|\mathcal{A}| \cdot |\mathcal{P}| \cdot 2^{Len(\hat{\mathcal{T}})}$ dummy predicates. For this reason, in the implementation of ComputeMinimalModels($\hat{\mathcal{T}}$), we apply rule (17) only in certain circumstances. Specifically, when the set of possible actions shares the same operator, we apply rule (17) only once for every branch of the completion tree. Consequently, the implementation of the algorithm does not guarantee completeness in the computation of all the minimal models for $\hat{\mathcal{T}}$, i.e., some false preconditions could not be discharged from the set of potential preconditions, or some true effect could not be added to the set of definite effect. However, notice that soundness is still guaranteed, as of Lemma 10.

8. Experiments

In this section, we present and discuss the results of an experimental study that we conducted with the aim of: *(i)* evaluating the quality of the learned models for different settings and observability degrees of the plan traces; *(ii)* comparing the quality and the efficacy of the learned models w.r.t. a state-of-the-art approach to learning action models from partially observable plan traces.

8.1. Experimental setting

We experimented $\mathsf{OffLAM}_{\mathsf{PT}}$ on 18 classical planning domains adopted in the past International Planning Competitions (IPCs). The plan traces used for our experiments are built as follows. For each domain, we randomly generated 10 small or medium size problem instances. Table 1 shows the domains used for our experiments and the size of the generated problems. Then, we computed a plan for these problems by using the FastDownward planning system [16] with a lazy greedy best-first search and a context-enhanced additive heuristic [17,13]. For each FastDownward's plan $(a_1, \ldots, a_n)$, we construct a trace $\boldsymbol{t} = (s_0, a_1, s_1), \ldots, (s_{n-1}, a_n, s_n)$, where each state s_i in the trace contains all the (positive/negative) literals that hold at time-step i. We randomly select a time step i between 0 and $n - 10$, and extract the sub-trace $\boldsymbol{t}' = (s_i, a_{i+1}, s_{i+1}), \ldots, (s_{i+9}, a_{i+10}, s_{i+10})$ from $\boldsymbol{t}$. Finally, we constructed a number of additional traces by removing literals and/or actions from $\boldsymbol{t}'$. Specifically, we considered the ten degrees of observability in set $\Omega = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$, and three different settings of the input traces: *(i)* traces with partially observable states and fully observable actions; *(ii)* traces with fully observable states and partially observable actions; *(iii)* traces with partially observable states and actions. When states in traces are partially observable, for each literal l in each state of $\boldsymbol{t}'$ we randomly remove l with a probability equal to $1 - \omega$, $\forall \omega \in \Omega$. Therefore, the higher the observability degree, the lower the number of removed literals. When $\omega = 1$, no literal is removed, the trace remains complete, and the observability is full. Similarly, when actions in traces are

Table 1
Number of object types (2nd column), number of predicates (3rd column), number of operators (4th column), maximum arity for predicates (5th column), maximum arity for operators (6th column), range of object number (7nd column), range of ground actions (8rd column) and range of ground atoms (9th column).

Domain	#types	$\|\mathcal{P}\|$	$\|\mathcal{O}\|$	max arity $\mathcal{P}$	max arity $\mathcal{O}$	$\|C\|$	$\|\mathcal{A}\|$	$\|\mathcal{P}(C)\|$
Blocksworld	1	5	4	2	2	[3, 8]	[18, 128]	[16, 81]
Driverlog	6	6	6	2	4	[9, 27]	[22, 740]	[29, 246]
Ferry	2	5	3	2	2	[4, 13]	[12, 136]	[14, 110]
Grid	3	9	5	2	4	[14, 43]	[98, 1924]	[81, 393]
Hanoi	3	3	1	2	3	[5, 10]	[38, 328]	[32, 122]
Miconic	2	6	4	2	2	[3, 12]	[4, 52]	[7, 48]
N-puzzle	2	3	1	2	3	[17, 49]	[192, 1920]	[105, 705]
Parking	2	5	4	2	3	[6, 18]	[60, 2448]	[24, 187]
Satellite	4	8	5	2	4	[17, 32]	[156, 731]	[67, 188]
Transport	7	5	3	2	5	[17, 30]	[408, 1824]	[70, 216]
Depots	10	6	5	2	4	[17, 54]	[27, 3384]	[26, 408]
Elevators	6	8	6	2	5	[10, 18]	[188, 1462]	[67, 209]
Gold-miner	1	12	7	2	2	[10, 43]	[73, 456]	[63, 310]
Gripper	4	4	3	3	4	[6, 20]	[12, 1155]	[8, 171]
Nomystery	6	6	3	3	6	[7, 21]	[18, 224]	[28, 139]
Sokoban	3	4	2	3	5	[24, 58]	[80, 448]	[96, 364]
Spanner	6	6	3	2	4	[7, 24]	[5, 42]	[13, 55]
Tpp	8	7	4	3	7	[9, 27]	[36, 18784]	[33, 687]

partially observable, for each action a in t' we randomly remove a with a probability equal to $1-\omega$, $\forall \omega \in \Omega$. When both states and actions in traces are partially observable, we randomly remove both literals and actions as mentioned. Therefore, for each considered setting of the input traces, observability degree, and domain, we derived a set of ten (partial) traces from which we learn an action model.

We focus our experimental study on the cautious models learned by $\mathsf{OffLAM}_{\mathsf{PT}}$. We evaluate both the quality and the efficacy of the learned models. As for the quality, we compared the learned models with the ground truth model by using the well-known metrics precision and recall [2]. Given a model $\mathcal{M}$ and a ground truth model $\mathcal{M}^\star$, the precision and recall for preconditions of $\mathcal{M}$ w.r.t. $\mathcal{M}^\star$ (denoted by P_{pre} and R_{pre}) are defined as follows:

$$P_{\mathsf{pre}} = \frac{\sum_{op} |\mathsf{pre}(op) \cap \mathsf{pre}^\star(op)|}{\sum_{op} |\mathsf{pre}(op)|}, \qquad R_{\mathsf{pre}} = \frac{\sum_{op} |\mathsf{pre}(op) \cap \mathsf{pre}^\star(op)|}{\sum_{op} |\mathsf{pre}^\star(op)|},$$

where $\mathsf{pre}(op)$ and $\mathsf{pre}^\star(op)$ are the precondition sets of op in $\mathcal{M}$ and in $\mathcal{M}^\star$, respectively. Intuitively, P_{pre} measures the (relative) amount of *extra* preconditions in $\mathcal{M}$ w.r.t. $\mathcal{M}^\star$; R_{pre} measures the (relative) amount of *missing* preconditions in $\mathcal{M}$ w.r.t. $\mathcal{M}^\star$. The lower these amounts, the greater and better the metrics. If the values of precision and recall for pre are 1, then the set of preconditions in the learned model is exactly the same as in the ground truth model. Similarly, we define precision and recall for $\mathtt{eff}^-$ and $\mathtt{eff}^+$. The *overall* precision P and recall R of $\mathcal{M}$ w.r.t. $\mathcal{M}^\star$ are defined considering pre, $\mathtt{eff}^-$, and $\mathtt{eff}^+$ together:

$$P = \frac{\sum_{op} |\mathsf{pre}(op) \cap \mathsf{pre}^\star(op)| + |\mathtt{eff}^+(op) \cap \mathtt{eff}^{\star +}(op)| + |\mathtt{eff}^-(op) \cap \mathtt{eff}^{\star -}(op)|}{\sum_{op} |\mathsf{pre}(op)| + |\mathtt{eff}^+(op)| + |\mathtt{eff}^-(op)|},$$

where $\mathtt{eff}^{+/-}(op)$ and $\mathtt{eff}^{\star +/-}(op)$ are the sets of positive/negative effects in $\mathcal{M}$ and $\mathcal{M}^\star$, respectively. The overall recall is defined similarly.

Evaluating action models in terms of their amounts of unnecessary (or missing) knowledge is appropriate when, e.g., the learned models need to be validated by humans, and hence the learned models should not contain unnecessary knowledge (as is typically the case for the ground truth models considered in our experiments). On the other hand, the impact of unnecessary or missing knowledge can be significantly different on the efficacy of the learned models for computing valid plans. For example, suppose that in the learned model there is a missing precondition for an operator, which is implied by a different learned precondition for the same operator; then the missing precondition has no impact on the validity of the computed plans with the learned model. Therefore, as for the efficacy, we also compared the learned models in terms of percentage of valid plans w.r.t. the ground truth model. In particular, we constructed a set of test problems different from the set of problems used to generate the traces and hence to learn the model. We run a state-of-the-art planning system with this set of test problems and the learned models, and validated the computed plans w.r.t. the ground truth model. The higher the percentage of valid plans w.r.t. the ground truth model, the more effective the learned model used to compute those plans. Similarly, we measured the percentage of plans lost by using the learned models among those computed by a state-of-the-art planning system using the ground truth model. The lower the percentage of lost plans, the more permissive the learned models.

All the experiments have been conducted on a CPU Apple M1 Pro with 16 GB of RAM, and a CPU time limit of 1000 seconds.

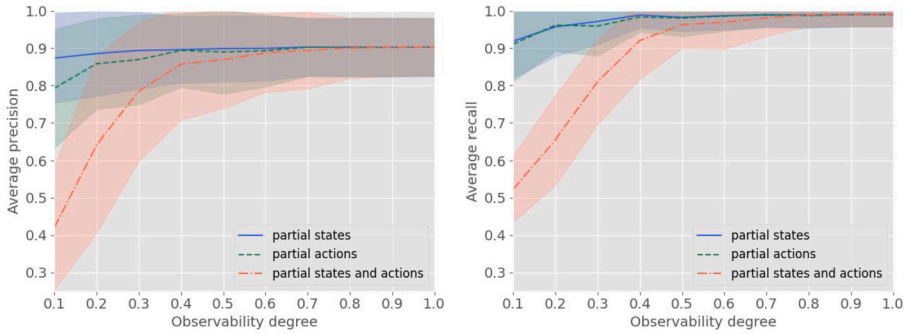


Fig. 1. Average precision and recall of the OffLAM$_{PT}$'s models for different degrees of observability, and for our traces with partial states, traces with partial actions, and traces with both partial states and actions. The shaped area represents the standard deviation.

8.2. Quality of the learned models

The models computed by OffLAM$_{PT}$ are evaluated for three different settings of traces: *(i)* partially observable states and fully observable actions; *(ii)* fully observable states and partially observable actions; and *(iii)* partially observable states and actions. For each setting, Fig. 1 shows the average precision and recall. The curves represent the values averaged over the 18 action models learned for the considered benchmark domains, whereas the shaped area around each curve represents the standard deviation.

The results in Fig. 1 indicate that when traces have partially observable states and fully observable actions, the quality of the learned models is high. Since OffLAM$_{PT}$ defines the cautious model from the sets of potential preconditions and definite effects, the model does not miss any precondition but can contain extra preconditions, and does not contain any extra effect but can miss some effect. Therefore, the results in Fig. 1 indicate that the number of extra preconditions and missing effects in the learned models is low. More specifically, while the recall is very often equal to 1, the precision is more frequently lower than 1, because in our learned models there are very few missing effects but they sometimes contain some extra preconditions. The reason why this happens is that certain literals, such as the static ones, cannot be ruled out from the set of the potential preconditions, as their negation never appears in the set of input traces. Remarkably, the performance is good even when the observability rate is low (e.g. 0.1), which means that the rules used by OffLAM$_{PT}$ are quite effective in inferring or making assumptions on the truth value of the missing literals in the partial states of the traces, and therefore refining the sets of action preconditions and effects. The results for traces with partially observable actions and fully observable states are similar. The quality of the learned models becomes poor only when both states and actions are partially observable and the observability degree is lower than 0.4. With this setting, the amount of missing evidence is large, and the rules of OffLAM$_{PT}$ fail to infer useful knowledge to refine the sets of action preconditions and effects.

The standard deviation of the precision and recall achieved by OffLAM$_{PT}$ is comparable for the three considered settings of traces, and does not drastically change for different observability degrees. The performance of the approach is more affected by the nature of the domain, and the amount of useful evidence in the input traces. For example, when the traces have partially observable states and fully observable actions, the overall precision of the model learned for the Blocksworld domain is always 1, while it ranges from 0.48 to 0.72 for the Gold-miner domain. Similarly, the range of the overall recall for Blocksworld is smaller than for Gold-miner: the overall recall for Blocksworld ranges from 0.93 (with observability degree equal to 0.1) to 1, while for Gold-miner it ranges from 0.71 to 0.95. In particular, the models learned for Blocksworld contain no extra preconditions and almost always no missing effects (only one missing effect when the observability degree is equal to 0.1), while in the models learned for Gold-miner the number of extra preconditions ranges from 15 to 83 and the number of missing effects ranges from 2 to 20.

Appendix A shows the detailed results about the precision and recall of OffLAM$_{PT}$ domain by domain. For traces with either partial states or partial actions, OffLAM$_{PT}$ is usually capable to draw all the useful knowledge from the traces when the observability degree is greater than 0.3. Indeed, for all domains but three, OffLAM$_{PT}$ achieved the same best performance as with fully observable traces when the observability degree is 0.4 (or greater). As expected, the picture deteriorates for traces with both partial states and partial actions, since, for all domains but three, OffLAM$_{PT}$ achieved the best performance only when the observability degree is 0.6 (or greater). Another remarkable result is that for domains Blocksworld, Gripper, and Miconic, OffLAM$_{PT}$ is capable of deriving exactly the same action model as the ground truth model even when the observability is poor.

8.3. Performance w.r.t. an increasing number of input traces

We also investigate the scalability of OffLAM$_{PT}$ and the quality of the learned models when the number of input traces becomes large. We considered a number of input traces ranging from 10 to 100. Such a set of traces was randomly generated as follows: for each of the 10 problems of every considered domain, in addition to the FastDownward's plan π solving the problem, we generated nine traces by performing random walks starting from the first nine states of π. From the trajectories obtained by the random walks, we derived the input traces for our experiment as described in Section 8.1.

For this experiment, we focus on the most complex setting of input traces and the (seven) most complex domains in our benchmark, i.e., traces with both partially observable states and actions, an observability degree in $\{0.1, 0.2, 0.3\}$, and domains with a number of predicates and operators greater than 5 and 3, respectively.

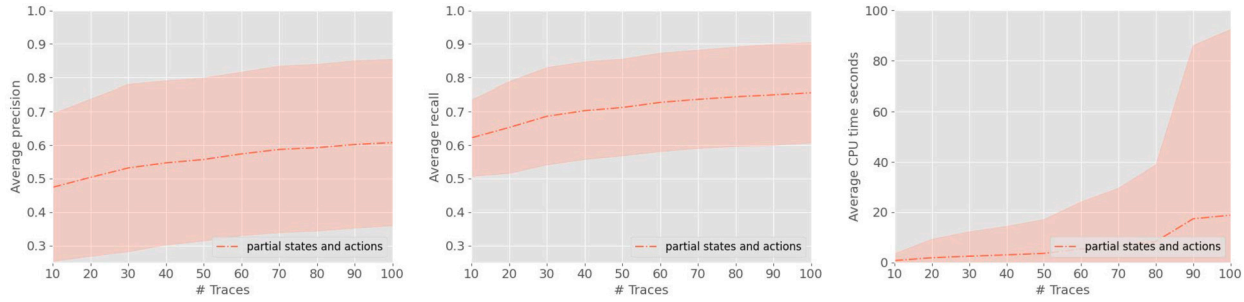


Fig. 2. Average precision and recall of the models learned by $\text{OffLAM}_{\text{PT}}$, and average CPU time required by $\text{OffLAM}_{\text{PT}}$ for different amounts of input traces having both partial states and partial actions. The shaped area represents the standard deviation.

For each considered number of input traces, $\text{OffLAM}_{\text{PT}}$ successfully learns all the 21 action models learned for the seven benchmark domains and the three observability degrees used in this experiment. Fig. 2 shows the CPU time used by $\text{OffLAM}_{\text{PT}}$ and the precision and recall of the $\text{OffLAM}_{\text{PT}}$'s models, averaged over the seven domains and three observability degrees, for different number of input traces. Remarkably, the CPU time required by $\text{OffLAM}_{\text{PT}}$ tends to degrade only when the number of input traces is quite high, i.e. greater than 80. As expected, the precision and recall of the learned models increase with the number of input traces. However, we empirically observed that, for a number of test problems, the models learned from a large set of input traces allow solving the same set of problems solved by the models learned from only 10 input traces. Therefore, for some complex settings, the effectiveness of the action models learned by $\text{OffLAM}_{\text{PT}}$ may not benefit from an increasing number of input traces.

8.4. Comparison with FAMA

In this section, we compare $\text{OffLAM}_{\text{PT}}$ with FAMA [2], which is a state-of-the-art method for learning action models from partially observable traces. ARMS is another approach proposed for learning action models, encoded using the STRIPS language [30]. However, we do not consider ARMS as an additional baseline for our experiment, because ARMS deals with plan traces having partially observable states and fully observable actions; i.e., ARMS solves a more specific problem than $\text{OffLAM}_{\text{PT}}$, which can deal with traces having both partially observable states and actions. Therefore, ARMS might be used as an additional baseline for only one over three settings of plan traces considered for our experiments. Moreover, Aineto et al. [2] show that FAMA outperforms ARMS for input traces with partially observable states and fully observable actions; therefore, we consider FAMA as the state-of-the-art baseline to compare with in our experiments.

FAMA fails to derive an action model from our plan traces, because FAMA requires that the initial and final states of the trace are complete, while we do not make such an assumption for the generation of our traces. Moreover, we observed that FAMA successfully compute a model from a very small set of traces but it does not scale well with larger set of traces. The benchmark described in Section 8.1 consists of sets of 10 plan traces from which we learn an action model; we empirically observed that, even when the initial and final state of our traces are complete, FAMA often fails to compute an action model from our sets of 10 traces. Therefore, for this comparison, we adopted the benchmark provided by the developers of FAMA. This benchmark consists of 15 domains, and for each of these domains 10 complete traces (from which we select smaller subsets of traces). The domains are the first 10 listed in Table 1 plus Floortile, Rovers, Visitall, Goldminer, and the single-agent version of the Gripper domain. We used the tool provided by the developers of FAMA to generate three sets of (partial) traces related to the settings: *(i)* partially observable states and fully observable actions; *(ii)* fully observable states and partially observable actions; and *(iii)* partially observable states and actions. Each of these sets contains traces with an observability degree ranging from 0.1 to 1. Each (partial) trace in the sets consists of 10 transitions. Similarly to Aineto et al. [2], we conducted two sets of experiments with $n = 2$ and $n = 5$: for 10 times, we randomly selected n traces from the set of 10 traces provided by Aineto et al. [2], run $\text{OffLAM}_{\text{PT}}$ and FAMA with the selected set of traces, and compared the required CPU times and the quality and efficacy of the learned action models.

It is worth noting that, for every set of selected traces, $\text{OffLAM}_{\text{PT}}$ computes an action model successfully, while FAMA sometimes fails to compute an action model within the given CPU time limit. We also observed that, even increasing the time limit to 1 hour for FAMA, the number of action models learned by FAMA has remained the same. The percentage of failure of FAMA with 2 and 5 traces is respectively 3 and 5.2; this confirms that $\text{OffLAM}_{\text{PT}}$ tends to scale better than FAMA w.r.t. an increasing number of input plan traces. Fig. 3 shows the average CPU time required by $\text{OffLAM}_{\text{PT}}$ and FAMA to compute an action model from a set of 2 traces. The CPU time is averaged over the 15 considered planning domains and 10 pairs of selected traces. When FAMA runs out of time we considered 1000 seconds as the CPU time of FAMA. For every considered observability degree and every setting of the input traces, $\text{OffLAM}_{\text{PT}}$ is faster than FAMA. The performance gap is higher when learning from traces with partially observable states and actions. For this setting of the input traces, when the observability is partial, $\text{OffLAM}_{\text{PT}}$ is about three orders of magnitude faster than FAMA; it is still one order of magnitude better when the observability is full. Finally, we empirically observed that the performance gap between $\text{OffLAM}_{\text{PT}}$ and FAMA is even greater, in favour of $\text{OffLAM}_{\text{PT}}$, when learning from 5 input traces.

The rest of the section focuses on the quality and efficacy of the learned action models. For this experimental study, we considered only the set of traces with which FAMA successfully learns an action model. The average precision and recall of the action models learned by $\text{OffLAM}_{\text{PT}}$ and FAMA with 2 partial traces are shown in Fig. 4. The curves represent the values averaged over (almost) 150

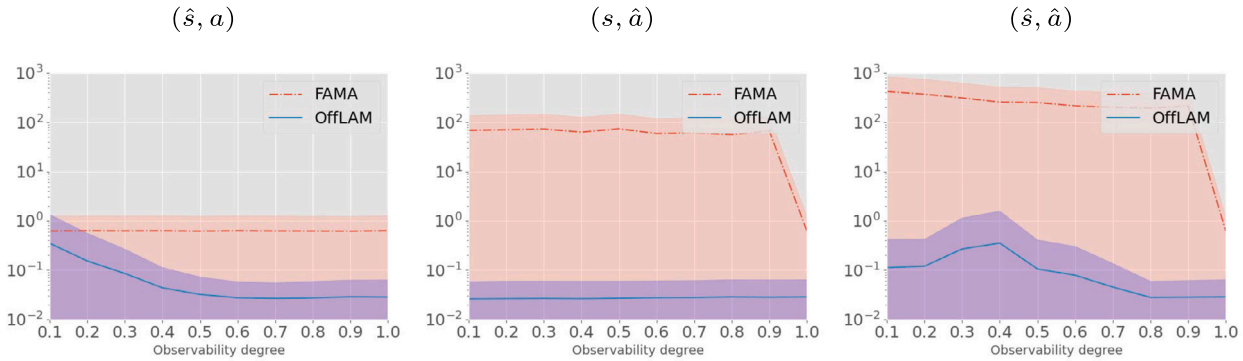


Fig. 3. Average CPU time of $\text{OffLAM}_{\text{PT}}$ and FAMA when learning models from a set of 2 partial traces for different degrees of observability, and for traces with partial states (denoted by $(\hat{s}, a)$), traces with partial actions (denoted by $(s, \hat{a})$), and traces with both partial states and partial actions (denoted by $(\hat{s}, \hat{a})$). The shaped area represents the standard deviation.

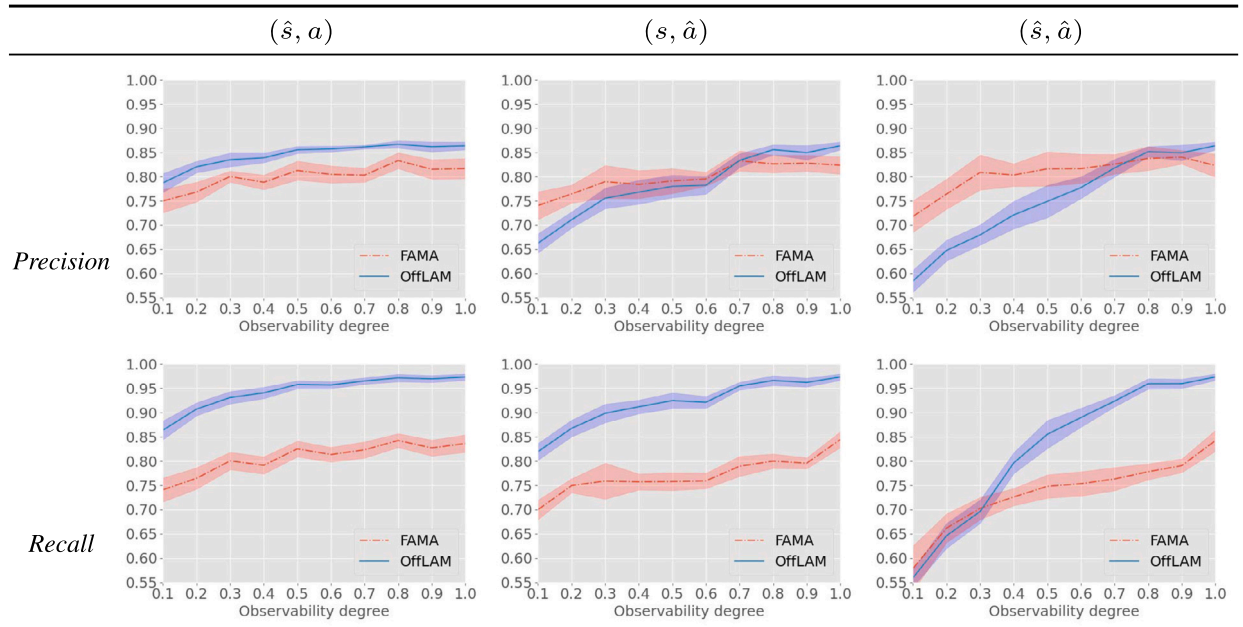


Fig. 4. Average precision and recall of the models learned by $\text{OffLAM}_{\text{PT}}$ and FAMA using 2 partial traces for different degrees of observability, and for traces with partial states (denoted by $(\hat{s}, a)$), traces with partial actions (denoted by $(s, \hat{a})$), and traces with both partial states and partial actions (denoted by $(\hat{s}, \hat{a})$). The shaped area represents the standard deviation.

action models, one for each of the 15 planning domains and 10 pairs of traces (excepting the pairs of traces from which FAMA fails to compute a model). The results in Fig. 4 show that when the states are partially observable and the actions are fully observable, $\text{OffLAM}_{\text{PT}}$ is clearly better than FAMA, as it is always the best in terms of both precision and recall. Notably, in terms of recall $\text{OffLAM}_{\text{PT}}$ performs better than FAMA by a large margin, i.e., more than 10%. When the states are fully observable and the actions are partially observable, $\text{OffLAM}_{\text{PT}}$ is still better than FAMA, as it is much better in terms of recall and is comparable in terms of precision. Specifically, the recall of $\text{OffLAM}_{\text{PT}}$ is always better than FAMA by a large margin; when the degree of observability is lower than 0.4, the precision of FAMA is slightly better than $\text{OffLAM}_{\text{PT}}$; when it is between 0.4 and 0.7, the precision of the two approaches is similar; when it is higher than 0.7, the precision of $\text{OffLAM}_{\text{PT}}$ is slightly better. Finally, when both states and actions are partially observable, the picture is less clear. Indeed, the recall obtained by $\text{OffLAM}_{\text{PT}}$ is almost always much better than FAMA, while FAMA is almost always much better in terms of precision. Indeed, the precision of $\text{OffLAM}_{\text{PT}}$ is better than or similar to FAMA only when the observability degree is greater than 0.6. The reason why the performance of $\text{OffLAM}_{\text{PT}}$ is sometimes very poor for this setting of traces is that rules (13) and (14) are more frequently applied when the observability degree is low, they introduce a large amount of dummy predicates from which Algorithm 3 fails to derive useful knowledge for the computation of the cautious model. Moreover, one can observe that overall the standard deviation of $\text{OffLAM}_{\text{PT}}$ is lower than or similar to FAMA. This means that the efficacy of FAMA changes more frequently than $\text{OffLAM}_{\text{PT}}$, i.e., the performance of FAMA is more affected from the input pair of traces w.r.t. $\text{OffLAM}_{\text{PT}}$.

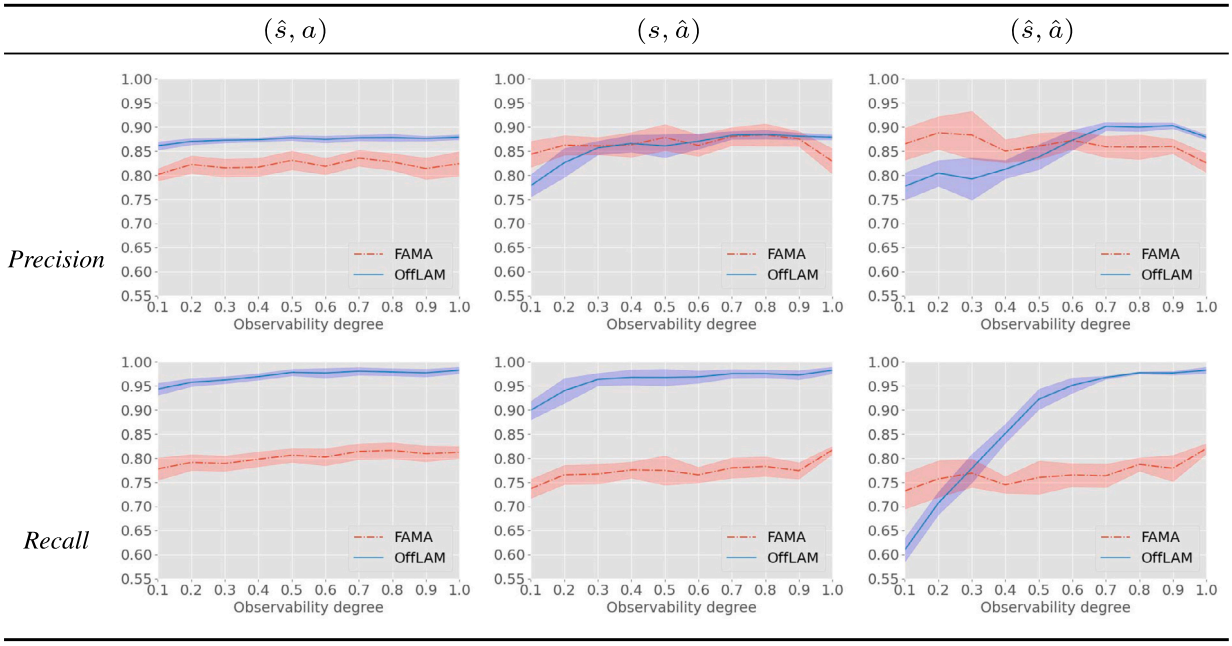


Fig. 5. Average precision and recall of the models learned by OffLAM$_{PT}$ and FAMA using 5 partial traces for different degrees of observability, and for traces with partial states (denoted by $(\hat{s}, a)$), traces with partial actions (denoted by $(s, \hat{a})$), and traces with both partial states and partial actions (denoted by $(\hat{s}, \hat{a})$). The shaped area represents the standard deviation.

Fig. 5 shows the performance of OffLAM$_{PT}$ and FAMA with 5 partial traces. The results in this figure lead to conclusions similar to those draw from Fig. 4. The main difference w.r.t. Fig. 4 regards the performance when both states and actions are partially observable. Indeed, with 5 traces the performance gaps are higher than with 2 traces. With 5 traces, we can draw that in terms of *both* precision and recall FAMA is better than OffLAM$_{PT}$ only when the observability degree is really low (between 0.1 and 0.2), while OffLAM$_{PT}$ is better when the observability degree is greater than 0.6.

We also compared the efficacy of the models computed by OffLAM$_{PT}$ and FAMA in terms of percentage of valid plans and lost plans. For this experiment, we generated 10 small and middle-size problem instances for every domain but Visitall and the single-agent version of Gripper, for which we could not find an available problem generator. Then, we run the FastDownward planner with a lazy greedy best-first search and a context-enhanced additive heuristic [16] using the action models learned from 2 traces. The results of FastDownward with the action models learned from 5 traces are similar. For each observability degree, the number of action models used for this experiment is (almost) 130, because we use OffLAM$_{PT}$ and FAMA to learn 10 action models (using the procedure mentioned before) for each of the 13 domains used for this experiment. For FAMA, the number of planning domains in slightly lower than 130, as sometimes FAMA fails to output an action model. Therefore, for each observability degree we considered (almost) 1300 planning problems.

Table 2 shows the percentage of valid plans computed by FastDownward using the action models learned by OffLAM$_{PT}$ and FAMA. A plan is considered valid if it is executable and reaches the goals using the ground truth model. The performance gap between OffLAM$_{PT}$ and FAMA is very large in favour of OffLAM$_{PT}$, as the percentage of valid plans with the action models learned by OffLAM$_{PT}$ is almost always greater than 90, while such a percentage with the action models learned by FAMA is at most 20.

Notably, when the observability degree is greater than or equal to 0.8 almost all the plans computed with the action models learned by OffLAM$_{PT}$ are valid. This is because, as mentioned before, for such observability degrees the recall is almost equal to 1. Sometimes it happens that a plan is not valid because some action models have a missing negative effect that compromise the validity of the plan. The percentage of valid plan is lower than 90 only when both states and actions are partially observable and the observability degree ranges from 0.2 to 0.4. However, even for these settings the action models learned by OffLAM$_{PT}$ for all domain but N-puzzle allow to compute a percentage of valid plan close to 100. For N-puzzle, a half of the learned action models allow to solve a large amount of problems with invalid plans. Indeed, in half of the N-puzzle models there is one missing negative effect that make the problem easy to solve with a plan that is invalid according to the ground truth model.

Table 3 gives the percentage of plans lost by using the action models learned by OffLAM$_{PT}$ and FAMA. The lost plan percentage is defined as the ratio of plans that are not valid for the learned models among the (1300) plans computed by FastDownward using the ground truth models. Plans that are not valid for a model cannot be computed using the model and, in this sense, we say that they are lost by the model. The results in Table 3 show that OffLAM$_{PT}$ loses much more plans than FAMA, and hence the OffLAM$_{PT}$'s models are much less permissive than those computed by FAMA.

The reason why the ratio of invalid plans is often quite high for OffLAM$_{PT}$ is that the models learned by OffLAM$_{PT}$ contain extra-preconditions that make the plans computed using the ground truth model not valid for the learned models. OffLAM$_{PT}$ fails to rule

Table 2
Average ratio of the valid plans computed by FastDownward using the models learned by OffLAM$_{PT}$ and FAMA from 2 partial traces for different degrees of observability, and for traces with partial states (denoted by $(\hat{s}, a)$), traces with partial actions (denoted by $(s, \hat{a})$), and traces with both partial states and partial actions (denoted by $(\hat{s}, \hat{a})$). Subscript indicates the number of plans over which the ratio is computed. The best results are in bold: the higher, the better.

Observability	$(\hat{s}, a)$		$(s, \hat{a})$		$(\hat{s}, \hat{a})$	
	OffLAM$_{PT}$	FAMA	OffLAM$_{PT}$	FAMA	OffLAM$_{PT}$	FAMA
0.10	$\mathbf{0.99}_{319}$	0.14_{1027}	$\mathbf{1.00}_{217}$	0.09_{1117}	$\mathbf{1.00}_{24}$	0.06_{614}
0.20	$\mathbf{0.93}_{598}$	0.12_{1061}	$\mathbf{1.00}_{261}$	0.12_{1158}	$\mathbf{0.62}_{39}$	0.11_{761}
0.30	$\mathbf{0.95}_{701}$	0.16_{1172}	$\mathbf{0.98}_{516}$	0.13_{1107}	$\mathbf{0.55}_{55}$	0.08_{872}
0.40	$\mathbf{0.93}_{763}$	0.13_{1068}	$\mathbf{1.00}_{539}$	0.10_{1087}	$\mathbf{0.83}_{119}$	0.08_{942}
0.50	$\mathbf{0.96}_{794}$	0.20_{1143}	$\mathbf{1.00}_{615}$	0.09_{1138}	$\mathbf{0.93}_{268}$	0.13_{998}
0.60	$\mathbf{0.99}_{906}$	0.18_{1132}	$\mathbf{1.00}_{641}$	0.11_{1145}	$\mathbf{0.97}_{329}$	0.11_{1017}
0.70	$\mathbf{0.95}_{918}$	0.15_{1130}	$\mathbf{1.00}_{816}$	0.12_{1143}	$\mathbf{0.94}_{580}$	0.14_{1167}
0.80	$\mathbf{1.00}_{911}$	0.18_{1148}	$\mathbf{0.99}_{918}$	0.14_{1159}	$\mathbf{0.97}_{846}$	0.11_{1149}
0.90	$\mathbf{1.00}_{924}$	0.17_{1108}	$\mathbf{1.00}_{852}$	0.11_{1074}	$\mathbf{1.00}_{828}$	0.13_{1072}
1.00	$\mathbf{1.00}_{908}$	0.15_{1089}	$\mathbf{1.00}_{908}$	0.15_{1086}	$\mathbf{1.00}_{908}$	0.14_{1122}

Table 3
Average ratio of the lost plans using the models OffLAM$_{PT}$ and FAMA from 2 partial traces for different degrees of observability, and for traces with partial states (denoted by $(\hat{s}, a)$), traces with partial actions (denoted by $(s, \hat{a})$), and traces with both partial states and partial actions (denoted by $(\hat{s}, \hat{a})$). The best results are in bold: the lower, the better.

Observability	$(\hat{s}, a)$		$(s, \hat{a})$		$(\hat{s}, \hat{a})$	
	OffLAM$_{PT}$	FAMA	OffLAM$_{PT}$	FAMA	OffLAM$_{PT}$	FAMA
0.10	0.83	**0.53**	0.89	**0.65**	0.98	**0.77**
0.20	0.70	**0.46**	0.85	**0.52**	0.96	**0.66**
0.30	0.60	**0.37**	0.75	**0.49**	0.94	**0.55**
0.40	0.58	**0.39**	0.73	**0.51**	0.93	**0.51**
0.50	0.58	**0.36**	0.64	**0.45**	0.88	**0.49**
0.60	0.51	**0.37**	0.63	**0.44**	0.84	**0.48**
0.70	0.51	**0.35**	0.61	**0.40**	0.72	**0.37**
0.80	0.51	**0.38**	0.53	**0.32**	0.59	**0.34**
0.90	0.49	**0.40**	0.56	**0.36**	0.58	**0.39**
1.00	0.52	**0.39**	0.52	**0.39**	0.52	**0.38**

out these extra preconditions from the learned models because of missing information in the input traces. For instance, for the Ferry domain, action `sail` contains the extra precondition `empty-ferry` in the models learned by OffLAM$_{PT}$; the presence of this extra precondition makes all the plans computed by FastDownward using the ground truth model invalid for the OffLAM$_{PT}$'s model; and the reason why such a precondition is part of the OffLAM$_{PT}$'s model is that in the input traces ferries always sail with an empty cargo. Notably, the quite high ratios for OffLAM$_{PT}$ in Table 3 do not mean that OffLAM$_{PT}$ loses the capability of solving problems. For instance, when the observability degree is greater than or equal to 0.8, although OffLAM$_{PT}$ loses more than one plan over two, we empirically also observed that the percentage of problems solved using the OffLAM$_{PT}$'s models is between 60 and 70 for every considered setting of traces. Fig. 4 shows that for such observability degrees the recall is almost equal to 1, and hence there are few or none missing effects. Therefore, we conjecture that the reason why about one out of three problems is unsolved by FastDownward using the OffLAM$_{PT}$'s models is the presence of extra preconditions in these action models.

9. Conclusions

Automated planning requires the specification of a model of the actions that the controlled agents can execute in their world. The hand-crafted specification of such a model is often a complex task, which can take a long time and may result in a wrong specification. Researching in AI proposed a number of approaches for automatically learning an action model from observations of the acting of the agents that one intends to control. In this work, we focused on learning an action model from a set of traces that provides knowledge about a sequence of actions successfully executed by the agents, and knowledge about the states of the world before and after the executions of these actions. Such traces can be partial, i.e., traces of partially observable states and actions.

There may be many action models that are consistent with a set of partial traces. The first contribution of our work is the preference criteria for determining the best action model among those consistent with the set of input partial traces: the smaller, the better. We consider a model $\mathcal{M}_1$ smaller than another model, $\mathcal{M}_2$, if from a given state the transitions enabled by $\mathcal{M}_1$ are a subset of those enabled by $\mathcal{M}_2$, and the changes performed by each of these transitions for $\mathcal{M}_1$ are a subset of those for $\mathcal{M}_2$. However, there may be still more than one minimal model consistent with the input set of traces. Thus, we compute a model, the cautious model, that

consists of the effects present in all the minimal models and the preconditions present in at least one minimal model. The cautious model is such that it contains the effects that are necessarily present in all the possible models consistent with the input traces, and the preconditions that are potentially present in at least one model consistent with the input traces.

The second contribution of our work is a set of rules for the problem of computing all the minimal models consistent with a set of traces. We prove that the iterative application of these rules is sound and complete for this problem. Therefore, we propose an algorithm that first computes all the minimal models for an input set of traces, and subsequently derives the cautious model. We conducted an extensive experimental study to evaluate the quality of the models learned using the proposed algorithm w.r.t. the ground truth model. Our study considers input plan traces ranging from having very little knowledge about the executed actions and traversed states to having fully observable states and actions. The comparison between the output model and the ground truth model used for the generation of the input traces shows that our approach performs very well. Briefly, when traces have either partially observable states or actions, our approach generates action models with very high values for precision and recall, which means that our models contain few extra preconditions and miss few effects of the ground truth model. With traces having both partially observable states and actions, the precision and recall are very high when at least a half of the knowledge about the executed actions and the traversed states is in the input traces.

Another contribution of the work is proposing to experimentally evaluate the efficacy of the learned models for planning, in addition to the evaluation of their quality w.r.t. the ground truth model. We propose to evaluate their effectiveness in terms of percentage of valid plans and percentage of lost plans. A plan computed using the learned action model is considered valid if it is executable and reaches the problem goals by using the ground truth model. A plan computed using the ground truth model is considered lost by a learned model if it is not executable or does not reach the problem goals by using the model. We compared the models learned by our approach with those learned by a state-of-the-art approach, FAMA. For the considered large set of benchmarks, our models are generally better than those learned by FAMA when the set of input traces have either partially observable states or partially observable actions, and when the input traces have partially observable states, partially observable actions, and the observed knowledge in the input traces is roughly at least as much as the missing knowledge. Most importantly, our approach outperforms FAMA in terms of percentage of valid plans, showing that the models learned using our approach are much more effective for acting in the world where the set of input traces was originated. Finally, FAMA loses less plans than our models, showing that the models learned using our approach are much less permissive than those computed by FAMA.

There are several research directions to extend our work. We intend to adopt our algorithms to learn from partial traces generated by an *online* approach, similarly to [22]. Another direction for the future work consists in learning from partial traces with states containing numerical variables or traces with states and actions labelled by times, which aims to output an action model with numerical preconditions and effects, or an action model with durative actions as in PDDL2.1 [14].

CRediT authorship contribution statement

Leonardo Lamanna: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Luciano Serafini:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Alessandro Saetti:** Writing – review & editing, Supervision, Methodology, Formal analysis, Conceptualization. **Alfonso Emilio Gerevini:** Writing – review & editing, Supervision, Resources, Methodology, Funding acquisition, Conceptualization. **Paolo Traverso:** Writing – original draft, Supervision, Resources, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU, and the EU H2020 project AIPlan4EU (GA n. 101016442). This work was also partially supported by EU ICT-48 2020 project TAILOR (No. 952215) and MUR PRIN project RIPER (No. 20203FFYLK).

Appendix A

The next tables show the precision and recall of the models learned by $\text{OffLAM}_{\text{PT}}$ using 10 partial traces with an observability degree ranging from 0.1 (poor) to 1.0 (full), and for traces with partially observable states (top table), traces with partially observable actions (middle table), and traces with both partially observable states and partially observable actions (bottom table). The best pair of values of precision and recall is in bold.

$(\hat{s}, a)$	0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1.0	
Domain	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*
ferry	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**
gripper	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
hanoi	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**
miconic	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
n-puzzle	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**
nomystery	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**
sokoban	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**
transport	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**
blocksworld	1.00	0.93	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
grid	0.78	0.94	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**
parking	0.89	0.97	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**
depots	0.94	0.86	0.97	0.97	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**
elevators	0.79	0.92	0.77	0.97	**0.80**	**1.00**	**0.80**	**1.00**	**0.80**	**1.00**	**0.80**	**1.00**	**0.80**	**1.00**	**0.80**	**1.00**	**0.80**	**1.00**	**0.80**	**1.00**
spanner	0.93	0.88	0.94	1.00	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**
tpp	0.80	0.77	0.82	0.87	0.81	0.84	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**
driverlog	0.82	0.82	0.92	0.86	0.93	0.89	0.93	1.00	0.93	0.96	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**
satellite	0.89	0.74	0.90	0.83	0.96	0.96	0.92	1.00	0.96	0.96	**0.96**	**1.00**	**0.96**	**1.00**	**0.96**	**1.00**	**0.96**	**1.00**	**0.96**	**1.00**
gold-miner	0.48	0.71	0.51	0.73	0.57	0.80	0.64	0.93	0.65	0.90	0.66	0.90	0.72	0.93	0.72	0.93	0.71	0.95	**0.72**	**0.95**

$(s, \hat{a})$	0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1.0	
Domain	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*
ferry	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**
miconic	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
n-puzzle	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**
nomystery	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**
sokoban	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**
transport	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**
blocksworld	0.96	1.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
depots	0.87	0.92	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**
gripper	0.92	0.86	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
parking	0.56	0.69	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**
spanner	0.82	0.88	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**
grid	0.74	0.90	0.64	0.94	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**
hanoi	0.80	1.00	0.80	1.00	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**
tpp	0.71	0.81	0.74	0.81	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**
driverlog	0.66	0.82	0.71	0.79	0.71	0.79	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**
gold-miner	0.47	0.83	0.62	0.90	0.52	0.83	0.62	0.88	0.53	0.83	0.61	0.88	**0.72**	**0.95**	**0.72**	**0.95**	**0.72**	**0.95**	**0.72**	**0.95**
elevators	0.51	0.95	0.76	1.00	0.76	1.00	0.74	1.00	0.76	1.00	0.77	1.00	**0.80**	**1.00**	0.79	1.00	**0.80**	**1.00**	**0.80**	**1.00**
satellite	0.74	0.74	0.87	0.87	0.82	0.78	0.96	0.96	0.96	0.96	0.92	1.00	**0.96**	**1.00**	0.96	0.96	**0.96**	**1.00**	**0.96**	**1.00**

$(\hat{s}, \hat{a})$	0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9		1.0	
Domain	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*
n-puzzle	0.43	0.43	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**	**0.88**	**1.00**
sokoban	0.44	0.50	0.82	0.88	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**
gripper	0.60	0.43	1.00	0.64	1.00	0.79	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
nomystery	0.21	0.59	0.29	0.59	0.79	0.88	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**	**0.85**	**1.00**
transport	0.38	0.50	0.77	0.50	0.94	0.75	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**	**0.95**	**1.00**
blocksworld	0.33	0.33	0.64	0.52	1.00	0.78	1.00	0.96	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.00
depots	0.55	0.49	0.77	0.65	0.91	0.78	0.97	0.92	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**	**0.97**	**1.00**
ferry	0.57	0.53	0.91	0.67	0.85	0.73	0.93	0.93	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**
grid	0.37	0.55	0.44	0.61	0.70	0.90	0.79	1.00	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**	**0.82**	**1.00**
miconic	0.39	0.56	0.71	0.62	0.88	0.88	1.00	0.94	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
parking	0.40	0.44	0.66	0.66	0.85	0.72	0.89	0.97	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**
driverlog	0.37	0.50	0.50	0.64	0.61	0.71	0.83	0.89	0.93	0.93	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**	**0.93**	**1.00**
hanoi	0.75	0.75	0.75	0.75	0.80	1.00	0.89	1.00	0.80	1.00	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**	**0.89**	**1.00**
spanner	0.64	0.56	0.85	0.69	0.93	0.88	0.93	0.81	0.94	0.94	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**	**0.94**	**1.00**
tpp	0.20	0.58	0.32	0.58	0.47	0.68	0.72	0.74	0.71	0.81	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**	**0.82**	**0.87**
satellite	0.58	0.61	0.70	0.61	0.73	0.70	0.86	0.78	0.87	0.87	0.86	0.78	0.96	0.96	**0.96**	**1.00**	**0.96**	**1.00**	**0.96**	**1.00**
elevators	0.17	0.57	0.26	0.59	0.56	0.78	0.67	0.97	0.74	0.95	0.79	1.00	0.79	1.00	0.77	1.00	**0.80**	**1.00**	**0.80**	**1.00**
gold-miner	0.20	0.51	0.28	0.59	0.33	0.63	0.38	0.66	0.47	0.83	0.55	0.80	0.57	0.83	0.71	0.95	**0.72**	**0.95**	**0.72**	**0.95**

Tables 4 and 5 show the percentage of plans computed by FastDownward, using the ground truth action models, that are valid in the action models learned by $\text{OffLAM}_{\text{PT}}$ and FAMA from a set of 2 and 5 traces, respectively. The experimental setting is the same adopted for comparing the efficacy of the action models learned by $\text{OffLAM}_{\text{PT}}$ and FAMA in Tables 2 and 3.

Table 4
Average ratio of the plans computed by FastDownward with the ground-truth models that are valid for the models learned by OffLAM$_{PT}$ and FAMA from 2 partial traces for different degrees of observability, and for traces with partial states (denoted by $(\hat{s}, a)$), traces with partial actions (denoted by $(s, \hat{a})$), and traces with both partial states and partial actions (denoted by $(\hat{s}, \hat{a})$). The best results are in bold.

Observability	$(\hat{s}, a)$		$(s, \hat{a})$		$(\hat{s}, \hat{a})$	
	OffLAM$_{PT}$	FAMA	OffLAM$_{PT}$	FAMA	OffLAM$_{PT}$	FAMA
0.10	0.17	**0.47**	0.11	**0.35**	0.02	**0.23**
0.20	0.30	**0.54**	0.15	**0.48**	0.04	**0.34**
0.30	0.40	**0.63**	0.25	**0.51**	0.06	**0.45**
0.40	0.42	**0.61**	0.27	**0.49**	0.07	**0.49**
0.50	0.42	**0.64**	0.36	**0.55**	0.12	**0.51**
0.60	0.49	**0.63**	0.37	**0.56**	0.16	**0.52**
0.70	0.49	**0.65**	0.39	**0.60**	0.28	**0.63**
0.80	0.49	**0.62**	0.47	**0.68**	0.41	**0.66**
0.90	0.51	**0.60**	0.44	**0.64**	0.42	**0.61**
1.00	0.48	**0.61**	0.48	**0.61**	0.48	**0.62**

Table 5
Average ratio of the plans computed by FastDownward with the ground-truth models that are valid for the models learned by OffLAM$_{PT}$ and FAMA from 5 partial traces for different degrees of observability, and for traces with partial states (denoted by $(\hat{s}, a)$), traces with partial actions (denoted by $(s, \hat{a})$), and traces with both partial states and partial actions (denoted by $(\hat{s}, \hat{a})$). The best results are in bold.

Observability	$(\hat{s}, a)$		$(s, \hat{a})$		$(\hat{s}, \hat{a})$	
	OffLAM$_{PT}$	FAMA	OffLAM$_{PT}$	FAMA	OffLAM$_{PT}$	FAMA
0.10	0.37	**0.68**	0.21	**0.48**	0.02	**0.20**
0.20	0.58	**0.68**	0.39	**0.47**	0.07	**0.34**
0.30	0.63	**0.69**	0.50	**0.55**	0.12	**0.45**
0.40	0.63	**0.69**	0.52	**0.58**	0.13	**0.41**
0.50	0.61	**0.71**	0.53	**0.57**	0.21	**0.46**
0.60	0.61	**0.67**	0.56	**0.59**	0.35	**0.47**
0.70	0.63	**0.68**	0.59	**0.60**	**0.51**	0.48
0.80	0.62	**0.69**	**0.62**	0.58	**0.62**	0.43
0.90	0.60	**0.69**	0.60	**0.62**	**0.60**	0.48
1.00	0.66	**0.68**	0.66	**0.68**	0.66	**0.70**

Appendix B. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.artint.2024.104256.

Data availability

Data and code are publicly available.

References

[1] D. Aineto, S. Jiménez, E. Onaindia, Learning STRIPS action models with classical planning, in: Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS-18), 2018, pp. 399–407.
[2] D. Aineto, S. Jiménez, E. Onaindia, Learning action models with minimal observability, Artif. Intell. 275 (2019) 104–137.
[3] E. Amir, A. Chang, Learning partially observable deterministic action models, J. Artif. Intell. Res. 33 (2008) 349–402.
[4] M. Asai, Unsupervised grounding of plannable first-order logic representation from images, in: Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS-19), 2019, pp. 583–591.
[5] M. Asai, A. Fukunaga, Classical planning in deep latent space: bridging the subsymbolic-symbolic boundary, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18), 2018, pp. 6094–6101.
[6] M. Asai, C. Muise, Learning neural-symbolic descriptive planning models via cube-space priors: the voyage home (to STRIPS), in: Proceedings of the 29th International Joint Conferences on Artificial Intelligence (IJCAI-21), 2021, pp. 2676–2682.
[7] P. Bachor, G. Behnke, Learning planning domains from non-redundant fully-observed traces: theoretical foundations and complexity analysis, in: Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI-24), 2024, pp. 20028–20035.
[8] B. Bonet, H. Geffner, Learning first-order symbolic representations for planning from the structure of the state space, in: Proceedings of the 24th European Conference on Artificial Intelligence (ECAI-20), 2020, pp. 2322–2329.
[9] M. Certicky, Real-time action model learning with online algorithm 3SG, Appl. Artif. Intell. 28 (2014) 690–711.
[10] R. Chitnis, T. Silver, J.B. Tenenbaum, L.P. Kaelbling, T. Lozano-Pérez, Glib: efficient exploration for relational model-based reinforcement learning via goal-literal babbling, in: Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI-21), 2021, pp. 11782–11791.
[11] S. Cresswell, P. Gregory, Generalised domain model acquisition from action traces, in: Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS-11), 2011, pp. 42–49.

[12] S. Cresswell, T.L. McCluskey, M.M. West, Acquiring planning domain models using LOCM, Knowl. Eng. Rev. 28 (2013) 195–213.
[13] P. Eyerich, R. Mattmüller, G. Röger, Using the context-enhanced additive heuristic for temporal and numeric planning, in: Towards Service Robots for Everyday Environments: Recent Advances in Designing Service Robots for Complex Tasks in Everyday Environments, Springer, 2012, pp. 49–64.
[14] M. Fox, D. Long, PDDL2.1: an extension to PDDL for expressing temporal planning domains, J. Artif. Intell. Res. 20 (2003) 61–124.
[15] P. Gregory, S. Cresswell, Domain model acquisition in the presence of static relations in the LOP system, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16), 2016, pp. 4160–4164.
[16] M. Helmert, The fast downward planning system, J. Artif. Intell. Res. 26 (2006) 191–246.
[17] J. Hoffmann, FF: the fast-forward planning system, AI Mag. 22 (2001) 57–62.
[18] B. Juba, H.S. Le, R. Stern, Safe learning of lifted action models, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-21), 2021, pp. 379–389.
[19] B. Juba, R. Stern, Learning probably approximately complete and safe action models for stochastic worlds, in: Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI-22), 2022, pp. 9795–9804.
[20] R. Karia, P. Verma, A. Speranzon, S. Srivastava, Epistemic exploration for generalizable planning and learning in non-stationary settings, in: Proceedings of the International Conference on Automated Planning and Scheduling, 2024, pp. 310–318.
[21] L. Lamanna, OffLAM: 1.0.0, https://doi.org/10.5281/zenodo.11634397, 2024.
[22] L. Lamanna, S. Alessandro, L. Serafini, A.E. Gerevini, P. Traverso, Online learning of action models for pddl planning, in: Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI-21), 2021, pp. 4112–4118.
[23] L. Lamanna, L. Serafini, Action model learning from noisy traces: a probabilistic approach, in: Proceedings of the International Conference on Automated Planning and Scheduling, 2024, pp. 342–350.
[24] H.S. Le, B. Juba, R. Stern, Learning safe action models with partial observability, in: Proceedings of the 38th AAAI Conference on Artificial Intelligence, 2024, pp. 20159–20167.
[25] A. Mordoch, R. Stern, B. Juba, Learning safe numeric action models, in: Proceedings of the 37th AAAI Conference on Artificial Intelligence, 2023, pp. 12079–12086.
[26] K. Mourão, L.S. Zettlemoyer, R.P.A. Petrick, M. Steedman, Learning STRIPS operators from noisy and incomplete observations, in: Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI-12), 2012, pp. 614–623.
[27] C. Rodrigues, P. Gérard, C. Rouveirol, H. Soldano, Active learning of relational action models, in: Proceedings of the 21st International Conference on Inductive Logic Programming (ILP-11), 2011, pp. 302–316.
[28] R. Stern, B. Juba, Efficient, safe, and probably approximately complete learning of action models, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17), 2017, pp. 4405–4411.
[29] J.Z. Xu, J.E. Laird, Instance-based online learning of deterministic relational action models, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10), 2010, pp. 1574–1579.
[30] Q. Yang, K. Wu, Y. Jang, Learning action models from plan examples using weighted max-sat, Artif. Intell. 171 (2007) 107–143.
[31] H.H. Zhuo, Crowdsourced action-model acquisition for planning, in: Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15), 2015, pp. 3439–3445.
[32] H.H. Zhuo, S. Kambhampati, Action-model acquisition from noisy plan traces, in: Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013), 2013, pp. 2444–2450.
[33] H.H. Zhuo, Q. Yang, D.H. Hu, L. Li, Learning complex action models with quantifiers and logical implications, Artif. Intell. 174 (2010) 1540–1569.