Explanation-based learning of Strips action models

Diego Aineto[1][], Sergio Jimnez[1][0000−0003−0561−4880], and Eva Onaindia[1][0000−0001−6931−8293]

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València.
Camino de Vera s/n. 46022 Valencia, Spain
{dieaigar,serjice,onaindia}@dsic.upv.es

Abstract.

Keywords: Learning action models · Planning and Learning · Classical Planning.

1 Introduction

2 Background

2.1 Classical planning with conditional effects

F is the set of *fluents* or *state variables* (propositional variables). A *literal* l is a valuation of a fluent $f \in F$, i.e. either $l = f$ or $l = \neg f$. L is a set of literals that represents a partial assignment of values to fluents, and $\mathcal{L}(F)$ is the set of all literals sets on F, i.e. all partial assignments of values to fluents. A *state* s is a full assignment of values to fluents. We explicitly include negative literals $\neg f$ in states and so $|s| = |F|$ and the size of the state space is $2^{|F|}$.

A *planning frame* is a tuple $\Phi = \langle F, A\rangle$, where F is a set of fluents and A is a set of *actions*. An action $a \in A$ is defined with *preconditions*, $\mathsf{pre}(a) \in \mathcal{L}(F)$, and *effects* $\mathsf{eff}(a) \in \mathcal{L}(F)$. The semantics of actions $a \in A$ is specified with two functions: $\rho(s, a)$ denotes whether action a is *applicable* in a state s and $\theta(s, a)$ denotes the *successor state* that results of applying action a in a state s. Then, $\rho(s, a)$ holds iff $\mathsf{pre}(a) \subseteq s$. And the result of applying a in s is $\theta(s, a) = \{s \setminus \neg\mathsf{eff}(a)) \cup \mathsf{eff}(a)\}$, with $\neg\mathsf{eff}(a) = \{\neg l : l \in \mathsf{eff}(a)\}$.

A *planning problem* is defined as a tuple $P = \langle F, A, I, G\rangle$, where I is the initial state in which all the fluents of F are assigned a value true/false and G is the goal set. A *plan* π for P is an action sequence $\pi = \langle a_1, \ldots, a_n\rangle$, and $|\pi| = n$ denotes its *plan length*. The execution of π in the initial state I of P induces a *trajectory* $\tau(\pi, P) = \langle s_0, a_1, s_1, \ldots, a_n, s_n\rangle$ such that $s_0 = I$ and, for each $1 \leq i \leq n$, it holds $\rho(s_{i-1}, a_i)$ and $s_i = \theta(s_{i-1}, a_i)$. A trajectory $\tau(\pi, P)$ that solves P is one in which $G \subseteq s_n$.

An action $a_c \in A$ with conditional effects is defined as a set of preconditions $\mathsf{pre}(a_c) \in \mathcal{L}(F)$ and a set of *conditional effects* $\mathsf{cond}(a_c)$. Each conditional effect $C \rhd E \in \mathsf{cond}(a_c)$ is composed of two sets of literals: $C \in \mathcal{L}(F)$, the *condition*, and $E \in \mathcal{L}(F)$, the *effect*. An action $a_c \in A$ is applicable in a state s if and only if $\mathsf{pre}(a_c) \subseteq s$, and the *triggered effects* resulting from the action application are the effects whose conditions hold in s:

$$triggered(s, a_c) = \bigcup_{C \rhd E \in \mathsf{cond}(a_c), C \subseteq s} E.$$

The result of applying a_c in state s follows the same definition of successor state, $\theta(s, a)$, but applied to the conditional effects in $triggered(s, a_c)$.

2.2 The observation model

Given a planning problem $P = \langle F, A, I, G\rangle$, a plan π and a trajectory $\tau(\pi, P)$, we define the *observation of the trajectory* as an interleaved combination of actions and states that represents the observation from the execution of π in P. Formally, $\mathcal{O}(\tau) = \langle s_0^o, a_1^o, s_1^o \ldots, a_l^o, s_m^o\rangle$, $s_0^o = I$, and:

- The **observed actions** are consistent with π, which means that $\langle a_1^o, \ldots, a_l^o\rangle$ is a sub-sequence of π. Specifically, the number of observed actions, l, can range from 0 (fully unobservable action sequence) to $|\pi|$ (fully observable action sequence).
- The **observed states** $\langle s_0^o, s_1^o, \ldots, s_m^o\rangle$ is a sequence of possibly *partially observable states*, except for the initial state s_0^o, which is fully observable. A partially observable state s_i^o is one in which $|s_i^o| < |F|$; i.e., a state in which at least a fluent of F is not observable. Note that this definition also comprises the case $|s_i^o| = 0$, when the state is fully unobservable. Whatever the sequence of observed states of $\mathcal{O}(\tau)$ is, it must be consistent with the sequence of states of $\tau(\pi, P)$, meaning that $\forall i, s_i^o \subseteq s_i$. In practice, the number of observed states, m, range from 1 (the initial state, at least), to $|\pi| + 1$, and the observed intermediate states will comprise a number of fluents between $[1, |F|]$.

We assume a bijective monotone mapping between actions/states of trajectories and observations [**?**], thus also granting the inverse consistency relationship (the trajectory is a superset of the observation). Therefore, transiting between two consecutive observed states in $\mathcal{O}(\tau)$ may require the execution of more than a single action ($\theta(s_i^o, \langle a_1, \ldots, a_k\rangle) = s_{i+1}^o$, where $k \geq 1$ is unknown but finite. In other words, having $\mathcal{O}(\tau)$ does not imply knowing the actual length of π.

Figure **??** illustrates a partial observation of a six-state trajectory {`<(xcoord 0)(ycoord 0)>, <(xcoord 1)(ycoord 0)>, <(xcoord 2)(ycoord 0)>, <(xcoord 3)(ycoord 0)>, <(xcoord 3)(ycoord 1)>, <(xcoord 2)(ycoord 1)>`}. This observation only contains fluents of the predicates `(xcoord ?v)` and `(ycoord ?v)`, and the value of the remaining fluents, corresponding to predicates `(next ?v1 ?v2)`, `(q0)` and `(q1)`, is unobservable in the six states.

3 Explanation-based learning of action models

The *one-shot* learning task to learn action models from *domain-specific knowledge* is defined as a tuple $\Lambda = \langle \mathcal{M}, \mathcal{O}, \Phi \rangle$, where:

- $\mathcal{M}$ is the *initial empty model* that contains only the header of each action model to be learned.
- $\mathcal{O}$ is a single learning example or plan observation; i.e. a sequence of (partially) observable states representing the evidence of the execution of an observed agent.
- Φ is a set of logic formulae that define *domain-specific knowledge*.

A *solution* to a learning task $\Lambda = \langle \mathcal{M}, \mathcal{O}, \Phi \rangle$ is a model $\mathcal{M}'$ s.t. there exists a plan computable with $\mathcal{M}'$ that is consistent with the headers of $\mathcal{M}$, the observed states of $\mathcal{O}$ and the given domain knowledge in Φ.

3.1 The space of Strips action models

We analyze here the solution space of a learning task $\Lambda = \langle \mathcal{M}, \mathcal{O}, \Phi \rangle$; i.e., the space of Strips action models. In principle, for a given action model ξ, any element of $\mathcal{I}_{\xi,\Psi}$ can potentially appear in $pre(\xi)$, $del(\xi)$ and $add(\xi)$. In practice, the actual space of possible Strips schemata is bounded by:

1. **Syntactic constraints**. The solution $\mathcal{M}'$ must be consistent with the Strips constraints: $del(\xi) \subseteq pre(\xi)$, $del(\xi) \cap add(\xi) = \emptyset$ and $pre(\xi) \cap add(\xi) = \emptyset$. *Typing constraints* would also be a type of syntactic constraint [**?**].
2. **Observation constraints**. The solution $\mathcal{M}'$ must be consistent with these *semantic constraints* derived from the learning samples $\mathcal{O}$, which in our case is a single plan observation. Specifically, the states induced by the plan computable with $\mathcal{M}'$ must comprise the observed states of the sample, which further constrains the space of possible action models.

Considering only the syntactic constraints, the size of the space of possible Strips models is given by $2^{2\times|\mathcal{I}_{\Psi,\xi}|}$ because one element in $\mathcal{I}_{\xi,\Psi}$ can appear both in the preconditions and effects of ξ. In this work, the belonging of an $e \in \mathcal{I}_{\Psi,\xi}$ to the preconditions, positive effects or negative effects of ξ is handled with a refined propositional encoding that uses fluents of two types, pre_ξ_e and eff_ξ_e, instead of the three fluents used in the BLS. The four possible combinations of these two fluents are sumarized in Figure 1. This compact encoding allows for a more effective exploitation of the syntactic constraints, and also yields the solution space of Λ to be the same as its search space.

3.2 The sampling space

The single plan observation of $\mathcal{O}$ is defined as $\mathcal{O} = \langle s_0^o, s_1^o \ldots, s_m^o \rangle$, a sequence of possibly *partially observed states* except for the initial state s_0^o which is a *fully*

Combination	**Meaning**
$\neg pre\_\xi\_e \wedge \neg eff\_\xi\_e$	e belongs neither to the preconditions nor effects of ξ $(e \notin pre(\xi) \wedge e \notin add(\xi) \wedge e \notin del(\xi))$
$pre\_\xi\_e \wedge \neg eff\_\xi\_e$	e is only a precondition of ξ $(e \in pre(\xi) \wedge e \notin add(\xi) \wedge e \notin del(\xi))$
$\neg pre\_\xi\_e \wedge eff\_\xi\_e$	e is a positive effect of ξ $(e \notin pre(\xi) \wedge e \in add(\xi) \wedge e \notin del(\xi))$
$pre\_\xi\_e \wedge eff\_\xi\_e$	e is a negative effect of ξ $(e \in pre(\xi) \wedge e \notin add(\xi) \wedge e \in add(\xi))$

Fig. 1. Combinations of the fluent propositional encoding and their meaning

observable state. As commented before, the predicates Ψ and the objects that shape the fluents F are then deducible from s_0^o. A partially observed state s_i^o, $1 \leq i \leq m$, is one in which $|s_i^o| < |F|$; i.e., a state in which at least a fluent of F was not observed. Intermediate states can be *missing*, meaning that they are unobservable, so transiting between two consecutive observed states in $\mathcal{O}$ may require the execution of more than one action ($\theta(s_i^o, \langle a_1, \ldots, a_k \rangle) = s_{i+1}^o$ (with $k \geq 1$ is unknown but finite). The minimal expression of a learning sample must comprise at least two state observations, a full initial state s_0^o and a partially observed final state s_m^o so $m \geq 1$.

Figure 2 shows a learning example that contains an initial state of the blocksworld where the robot hand is empty and three blocks (namely `blockA`, `blockB` and `blockC`) are on top of the table and clear. The observation represents a partially observable final state in which `blockA` is on top of `blockB` and `blockB` on top of `blockC`.

```
(:predicates (on ?x ?y) (ontable ?x) (clear ?x) (handempty) (holding ?x))

(:objects blockA blockB blockC)

(:init (ontable blockA) (clear blockA) (ontable blockB) (clear blockB)
       (ontable blockC) (clear blockC) (handempty))

(:observation (on blockA blockB) (on blockB blockC))
```

Fig. 2. Example of a two-state observationn for the learning STRIPS action models.

3.3 The domain-specific knowledge

One can introduce domain-specific knowledge to constrain further the space of possible schemata. For instance, back to the *blocksworld* domain, one can argue that on(v_1,v_1) and on(v_2,v_2) will not appear in the $pre(\xi)$, $del(\xi)$ and $add(\xi)$ of any action model ξ because, in this specific domain, a block cannot be on top of itself.

4 Learning Strips action models with classical planning

5 Experimental results

6 Conclussions