Computing *the least commitment* action model from observations of plan executions

Diego Aineto[1] , **Sergio Jiménez**[1] , **Eva Onaindia**[1] and , **Blai Bonet**[2]

[1]Departamento de Sistemas Informáticos y Computación. Universitat Politècnica de València. Valencia, Spain
[2]Departamento de Computación. Universidad Simón Bolívar. Caracas, Venezuela

{dieaigar,serjice,onaindia}@dsic.upv.es, bonet@usb.ve

Abstract

1 Introduction

Given the partial observation of a plan execution, this paper formalizes the task of computing the exact set of action models that are *conformant* with the given observation. This task is of interest because it allows to scallably learn action models from arbitrary large sets of observations generated with that models. The approach we follow for addressing this learning task is compiling it into *conformant planning*.

2 Background

This section formalizes the *classical* and the *conformant* planning models as well as the *observation* model we follow to observe the execution of sequential plans.

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*, $\mathsf{pre}(a) \in \mathcal{L}(F)$, *positive effects*, $\mathsf{eff}^+(a) \in \mathcal{L}(F)$, and *negative 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 $\theta(s, a) = \{s \setminus \mathsf{eff}^-(a)) \cup \mathsf{eff}^+(a)\}$.

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 set of goal conditions over the state variables in F. A *plan* π for P is an action sequence $\pi = \langle a_1, \dots, a_n\rangle$, and $|\pi| = n$ denotes its *plan length*. The execution of π in the initial state I of P induces a *trajectory* $\tau(\pi, s_0) = \langle s_0, a_1, s_1, \dots, a_n, s_n\rangle$ 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 plan π solves P iff the induced *trajectory* $\tau(\pi, s_0)$ reaches a final state $G \subseteq s_n$.

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 is applicable in a state s if $\rho(s, a_c)$ is true, 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 action a_c in state s is $\theta(s, a_c) = \{s \setminus \mathsf{eff}_c^-(s, a)) \cup \mathsf{eff}_c^+(s, a)\}$, where $\mathsf{eff}_c^-(s, a) \subseteq triggered(s, a)$ and $\mathsf{eff}_c^+(s, a) \subseteq triggered(s, a)$ are, respectively, the triggered *negative* and *positive* effects.

2.2 The observation model

Given a planning problem $P = \langle F, A, I, G\rangle$, a plan π and a trajectory $\tau(\pi, s_0)$, 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, $\mathcal{O}(\tau) = \langle s_0^o, a_1^o, s_1^o \dots, a_l^o, s_m^o\rangle$, $s_0^o = I$, and:

- The **observed actions** are consistent with π, which means that $\langle a_1^o, \dots, a_l^o\rangle$ 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** $\langle s_0^o, s_1^o, \dots, s_m^o\rangle$ 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 $\tau(\pi, s_0)$, 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 [Ramírez and Geffner, 2009], 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.

2.3 Conformant planning

Conformant planning refers to planning with incomplete information about the initial situation, no sensing, and where goals are to be achieved with certainty in spite of the uncertainty in the initial situation [Smith and Weld, 1998; Goldman and Boddy, 1996].

Syntactically, conformant planning problems are expressed in compact form through a set of state variables. A *conformant planning problem* is defined as a tuple $P_c = \langle F, A, \Upsilon, G\rangle$, where Υ is a set of clauses over literals $l = f$ or $l = \neg f$ (for $f \in F$) that define the set of possible initial states.

A solution to a conformant planning problem is an action sequence that maps each possible initial state into a goal state. More precisely an action sequence $\pi = \langle a_1, \ldots, a_n\rangle$ is a *conformant plan* for P_c iff for each possible *trajectory* $\tau(\pi, s_0) = \langle s_0, a_1, s_1, \ldots, a_n, s_n\rangle$ such that s_0 is a valuation that satisfies Υ and it holds that the trajectory reaches $G \subseteq s_n$.

3 Learning *the least commitment* action model from observations

3.1 The hypothesis space

STRIPS action schemata provide a compact representation for specifying classical planning models. *A* STRIPS *action schema* ξ is defined by four lists: A list of *parameters* $pars(\xi)$, and three list of predicates (namely $pre(\xi)$, $del(\xi)$ and $add(\xi)$) that shape the kind of fluents that can appear in the *preconditions*, *negative effects* and *positive effects* of the actions induced from that schema.

Definition 1 (Comparable STRIPS action schemata) *Two* STRIPS *schemata ξ and ξ' are* **comparable** *iff $pars(\xi) = pars(\xi')$, i.e, both share the same list of parameters.*[1]

For instance, the `stack(?v1,?v2)` and `unsstack(?v1,?v2)` schemata from a four operator *blocksworld* [Slaney and Thiébaux, 2001] are *comparable* while `stack(?v1,?v2)` and `pickup(?v1)` are not. Last but not least, we say that two STRIPS models $\mathcal{M}$ and $\mathcal{M}'$ are *comparable* iff there exists a bijective function $\mathcal{M} \mapsto \mathcal{M}^*$ that maps every action schema $\xi \in \mathcal{M}$ to a comparable schemata $\xi' \in \mathcal{M}'$ and vice versa.

Let be Ψ the set of *predicates* that shape the propositional state variables F, and a list of *parameters* $pars(\xi)$. The set of elements that can appear in $pre(\xi)$, $del(\xi)$ and $add(\xi)$ of the STRIPS action schema ξ is given by FOL interpretations of Ψ over the parameters $pars(\xi)$. We denote this set of FOL interpretations as $\mathcal{I}_{\Psi,\xi}$.

Despite any element of $\mathcal{I}_{\Psi,\xi}$ can *a priori* appear in the $pre(\xi)$, $del(\xi)$ and $add(\xi)$ of schema ξ, the space of possible STRIPS schemata is constrained by a set $\mathcal{C}$ that includes:

1. *Syntactic constraints*. STRIPS constraints require $del(\xi) \subseteq pre(\xi)$, $del(\xi) \cap add(\xi) = \emptyset$ and $pre(\xi) \cap add(\xi) = \emptyset$. Considering exclusively these syntactic constraints, the size of the space of possible STRIPS schemata is given by $2^{2\times|\mathcal{I}_{\Psi,\xi}|}$.
2. *Domain-specific constraints*. One can introduce domain-specific knowledge to constrain further the space of possible schemata. For instance, in a *blocksworld* one can claim that on(v_1,v_1) and on(v_2,v_2) will not appear in any of these lists because of the semantic of the `on` predicate that codes which block is in top of other block.

Definition 2 (Well-defined STRIPS action schemata) *Given a set of* predicates Ψ*, a list of action* parameters $pars(\xi)$*, and set of FOL constraints $\mathcal{C}$, ξ is a* **well-defined STRIPS action schema** *iff its three lists $pre(\xi) \subseteq \mathcal{I}_{\Psi,\xi}$, $del(\xi) \subseteq \mathcal{I}_{\Psi,\xi}$ and $add(\xi) \subseteq \mathcal{I}_{\Psi,\xi}$ only contain elements in $\mathcal{I}_{\Psi,\xi}$ and they satisfy all the constraints in $\mathcal{C}$.*

We say a planning model $\mathcal{M}$ is *well-defined* if all its STRIPS action schemata are *well-defined*.

3. *Observation constraints*. An observation of the execution of a plan further constraints the space of possible action schemata. This *semantic knowledge* included in the observations introduce a third type of constraints, that we will call *observation constraints*, and that can be added to the set $\mathcal{C}$.

3.2 The learning task

3.3 Learning the least commitment models with conformant planning

Given a STRIPS action schema ξ, a propositional encoding for the *preconditions*, *negative* and *positive* effects of that schema can be represented with fluents of the kind $[pre|del|add]\_e\_\xi$ such that $e \in \mathcal{I}_{\Psi,\xi}$ is a single element from the set of interpretations of predicates Ψ over the corresponding variable names Ω_ξ. Figure **??** shows the propositional encoding for the six action schema defined in Figure **??**.

The interest of having a propositional encoding for STRIPS action schema is that, using *conditional effects*, it allows to compactly define *editable actions*. Actions whose semantics is given by the value of the $[pre|del|add]\_e\_\xi$ fluents at the current state. Given an operator schema $\xi \in \mathcal{M}$ its *editable* version is formalized as:

$$\begin{aligned}
\mathsf{pre}(\mathsf{editable}_\xi) =& \{pre_e_\xi \implies e\}_{\forall e \in \mathcal{I}_{\Psi,\xi}} \\
\mathsf{cond}(\mathsf{editable}_\xi) =& \{del_e_\xi\} \rhd \{\neg e\}_{\forall e \in \mathcal{I}_{\Psi,\xi}}, \\
& \{add_e_\xi\}\} \rhd \{e\}_{\forall e \in \mathcal{I}_{\Psi,\xi}}.
\end{aligned}$$

[1] In STRIPS models, $pars(\xi) = pars(\xi')$ implies the number of parameters must be the same. For other planning models that allow object typing, the equality implies that parameters share the same type

```
(:action editable_inc-x
  :parameters (?v1 ?v2)
  :precondition
    (and (or (not (pre_xcoord_v1_inc-x)) (xcoord ?v1))
         (or (not (pre_xcoord_v2_inc-x)) (xcoord ?v2))
         (or (not (pre_ycoord_v1_inc-x)) (xcoord ?v1))
         (or (not (pre_ycoord_v2_inc-x)) (xcoord ?v2))
         (or (not (pre_q0__inc-x)) (q0))
         (or (not (pre_q1__inc-x)) (q1)))
         (or (not (pre_next_v1_v1_inc-x)) (next ?v1 ?v1)))
         (or (not (pre_next_v1_v2_inc-x)) (next ?v1 ?v2)))
         (or (not (pre_next_v2_v1_inc-x)) (next ?v2 ?v1)))
         (or (not (pre_next_v2_v2_inc-x)) (next ?v2 ?v2))))
    :effect (and
       (when (del_xcoord_v1_inc-x) (not (xcoord ?v1)))
       (when (del_xcoord_v2_inc-x) (not (xcoord ?v2)))
       (when (del_ycoord_v1_inc-x) (not (xcoord ?v1)))
       (when (del_ycoord_v2_inc-x) (not (xcoord ?v2)))
       (when (del_q0__inc-x) (not (q0)))
       (when (del_q1__inc-x) (not (q1)))
       (when (del_next_v1_v1_inc-x) (not (next ?v1 ?v1)))
       (when (del_next_v1_v2_inc-x) (not (next ?v1 ?v2)))
       (when (del_next_v2_v1_inc-x) (not (next ?v2 ?v1)))
       (when (del_next_v2_v2_inc-x) (not (next ?v2 ?v2)))

       (when (add_xcoord_v1_inc-x) (xcoord ?v1))
       (when (add_xcoord_v2_inc-x) (xcoord ?v2))
       (when (add_ycoord_v1_inc-x) (xcoord ?v1))
       (when (add_ycoord_v2_inc-x) (xcoord ?v2))
       (when (add_q0__inc-x) (q0))
       (when (add_q1__inc-x) (q1))
       (when (add_next_v1_v1_inc-x) (next ?v1 ?v1))
       (when (add_next_v1_v2_inc-x) (next ?v1 ?v2))
       (when (add_next_v2_v1_inc-x) (next ?v2 ?v1))
       (when (add_next_v2_v2_inc-x) (next ?v2 ?v2)))
```

Figure 1: Editable version of the `inc-x(?v1,?v2)` schema for robot navigation in a $n \times n$ grid.

Figure 1 shows the PDDL encoding of the *editable* `inc-x(?v1,?v2)` schema for robot navigation in a $n \times n$ grid (Figure **??**). Note that this editable schema, when the fluents of Figure **??** hold, behaves exactly as defined in Figure **??**.

4 Evaluation

5 Conclusions

References

[Goldman and Boddy, 1996] Robert P Goldman and Mark S Boddy. Expressive planning and explicit knowledge. In *AIPS*, volume 96, pages 110–117, 1996.

[Ramírez and Geffner, 2009] Miquel Ramírez and Hector Geffner. Plan recognition as planning. In *International Joint conference on Artifical Intelligence, (IJCAI-09)*, pages 1778–1783. AAAI Press, 2009.

[Slaney and Thiébaux, 2001] John Slaney and Sylvie Thiébaux. Blocks world revisited. *Artificial Intelligence*, 125(1-2):119–153, 2001.

[Smith and Weld, 1998] David E Smith and Daniel S Weld. Conformant graphplan. In *AAAI/IAAI*, pages 889–896, 1998.