Goal Recognition as Planning with Unknown Domain Models

Diego Aineto[1] , **Sergio Jiménez**[1] , **Eva Onaindia**[1] and , **Miquel Ramírez**[2]

[1]Departamento de Sistemas Informáticos y Computación. Universitat Politècnica de València. Valencia, Spain
[2]School of Computing and Information Systems. The University of Melbourne. Melbourne, Victoria. Australia

{dieaigar,serjice,onaindia}@dsic.upv.es, miquel.ramirez@unimelb.edu.au

Abstract

The paper shows how to relax one key assumption of the *plan recognition as planning* approach for *goal recognition* that is knowing the action model of the observed agent. The paper introduces a novel formulation that fits together the *learning of planning action models* with *plan recognition as planning*. The empirical evaluation evidences that this novel formulation allows to solve standard goal recognition benchmarks without *a priori* knowing the action model of the observed agent.

1 Introduction

Goal recognition is a particular classification task in which each class represents a different goal and each example is an observation of an agent acting to achieve one of that goals. Despite there exists a wide range of diffenrent approaches for *goal recognition*, *plan recognition as planning* [Ramírez and Geffner, 2009; Ramírez, 2012] is one of the most appealing since it is at the core of various activity recognition tasks such as, *goal recognition design* [Keren *et al.*, 2014], *deceptive planning* [Masters and Sardina, 2017], *planning for transparency* [MacNally *et al.*, 2018] or *counterplanning* [Pozanco *et al.*, 2018].

Plan recognition as planning leverages the action model of the observed agent and an off-the-shelf classical planner to compute the most likely goal of that agent. In this paper we show that we can relax the key assumption of the *plan recognition as planning* approach for *goal recognition* that is having an action model of the observed agent. In particular, the paper introduces a novel formulation that fits together the *learning of planning action models* with the *plan recognition as planning* approach. The evaluation of our formulation evidences that it allows to solve goal recognition tasks, even when the action model of the observed is unknown, using an off-the-shelf classical planner.

2 Background

This section formalizes the *planning model* we follow as well as the kind of *observations* that are given as input to the *goal recognition* task.

2.1 Classical planning with conditional effects

Let F be the set of propositional state variables (*fluents*) describing a state. A *literal* l is a valuation of a fluent $f \in F$; i.e. either $l = f$ or $l = \neg f$. A set of literals L represents a partial assignment of values to fluents (without loss of generality, we will assume that L does not contain conflicting values). Given L, let $\neg L = \{\neg l : l \in L\}$ be its complement. We use $\mathcal{L}(F)$ to denote the set of all literal sets on F; i.e. all partial assignments of values to fluents. A *state* s is a full assignment of values to fluents; $|s| = |F|$.

A *classical planning frame* is a tuple $\Phi = \langle F, A\rangle$, where F is a set of fluents and A is a set of *actions*. Each classical planning action $a \in A$ has a precondition $\mathsf{pre}(a) \in \mathcal{L}(F)$, a set of effects $\mathsf{eff}(a) \in \mathcal{L}(F)$, and a positive action cost $cost(a)$. 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 $\mathsf{pre}(a) \subseteq s$, i.e. if its precondition holds in s. The result of executing an applicable action $a \in A$ in a state s is a new state $\theta(s, a) = (s \setminus \neg\mathsf{eff}(a)) \cup \mathsf{eff}(a)$. Subtracting the complement of $\mathsf{eff}(a)$ from s ensures that $\theta(s, a)$ remains a well-defined state. The subset of action effects that assign a positive value to a state fluent is called *positive effects* and denoted by $\mathsf{eff}^+(a) \in \mathsf{eff}(a)$ while $\mathsf{eff}^-(a) \in \mathsf{eff}(a)$ denotes the *negative effects* of an action $a \in A$.

A *classical planning problem* is a tuple $P = \langle F, A, I, G\rangle$, where I is the initial state and $G \in \mathcal{L}(F)$ is the set of goal conditions over the state variables. A *plan* π is an action sequence $\pi = \langle a_1, \ldots, a_n\rangle$, with $|\pi| = n$ denoting its *plan length* and $cost(\pi) = \sum_{a\in\pi} cost(a)$ its *plan cost*. The execution of π on the initial state I of P induces a *trajectory* $\tau(\pi, s_0) = \langle s_0, a_1, s_1, \ldots, a_n, s_n\rangle$ 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 plan π solves P iff the induced *trajectory* $\tau(\pi, s_0)$ reaches a final state $G \subseteq s_n$, where all goal conditions are met. A solution plan is *optimal* iff its cost is minimal.

An *action with conditional effects* $a_c \in A$ 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 result of applying action a_c in state s is $\theta(s, a_c) = \{s \setminus \neg\mathsf{eff}_c(s, a) \cup \mathsf{eff}_c(s, a)\}$ where

$\mathsf{eff}_c(s, a)$ are the *triggered effects* resulting from the action application (conditional effects whose conditions hold in s):

$$\mathsf{eff}_c(s, a) = \bigcup_{C \triangleright E \in \mathsf{cond}(a_c), C \subseteq s} E,$$

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 [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 Classical planing with observation constraints

2.4 Goal recognition as classical planning

Goal recognition is a particular classification task in which each class represents a different goal $g \in G[\cdot]$ and each example is an $\mathcal{O}(\tau)$ observation of an agent acting to achieve one of the input goals in $G[\cdot]$. Following the *naive Bayes classifier*, the *solution* to the *goal recognition* task is the subset of goals in $G[\cdot]$ that maximizes this expression.

$$argmax_{g \in G[\cdot]} P(\mathcal{O}|g)P(g). \tag{1}$$

The *plan recognition as planning* approach shows how to compute estimates of the $P(\mathcal{O}|g)$ likelihood leveraging the action model of the observed agent and an off-the-shelf classical planner. More precisely, given a *classical planning problem* $P = \langle F, A, I, G[\cdot]\rangle$, where $G[\cdot]$ represents the set of possible goals, then the *plan recognition as planning* approach estimates the $P(\mathcal{O}|g)$ by computing the cost difference of the solutions to these two different classical planning problems:

- $P^\top$, that constrains the classical planning problem $P = \langle F, A, I, g\rangle$ to achieve $g \in G[\cdot]$ through a plan $\pi^\top$ *consistent* with the input observation $\mathcal{O}(\tau)$.
- $P^\perp$, that constrains $P = \langle F, A, I, g\rangle$ to achieve $g \in G[\cdot]$ through a plan $\pi^\perp$ *inconsistent* with $\mathcal{O}(\tau)$.

The higher the value of this difference $\Delta(cost(\pi^\top), cost(\pi^\perp))$, the higher $P(\mathcal{O}|g)$ likelihood. *Plan recognition as planning* uses the *sigmoid function* to map the previous cost difference into a likelihood:

$$P(\mathcal{O}|g) = \frac{1}{1 + e^{-\beta\Delta(cost(\pi^\top), cost(\pi^\perp))}} \tag{2}$$

This expression is derived from the assumption that while the observed agent is not perfectly rational, he is more likely to follow cheaper plans, according to a *Boltzmann* distribution. The larger the value of β, the more rational the agent, and the less likely that he will follow suboptimal plans.

Recent works show that estimates of the $P(\mathcal{O}|g)$ likelihood can be faster computed using relaxations of the $P^\top$ and $P^\perp$ classical planning tasks [Pereira *et al.*, 2017].

3 Classical planning with unknown domain models

3.1 A propositional encoding for the space of STRIPS action 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. 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 $pars(\xi)$ and is denoted as $\mathcal{I}_{\Psi,\xi}$.

For instance, in the *blocksworld* the $\mathcal{I}_{\Psi,\xi}$ set contains five elements for a `pickup(`v_1`)` schemata, $\mathcal{I}_{\Psi,pickup}$={`handempty`, `holding(`$v_1$`)`, `clear(`$v_1$`)`, `ontable(`$v_1$`)`, `on(`$v_1$,$v_1$`)`} while it contains eleven elements for a `stack(`v_1,v_2`)` schemata, $\mathcal{I}_{\Psi,stack}$={`handempty`, `holding(`v_1`)`, `holding(`v_2`)`, `clear(`v_1`)`, `clear(`v_2`)`, `ontable(`v_1`)`, `ontable(`v_2`)`, `on(`v_1,v_1`)`, `on(`v_1,v_2`)`, `on(`v_2,v_1`)`, `on(`v_2,v_2`)`}.

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 bounded by constraints of three kinds:

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 $2^{2\times|\mathcal{I}_{\Psi,\xi}|}$. *Typing constraints* are also of this kind [McDermott *et al.*, 1998].
2. *Domain-specific constraints*. One can introduce domain-specific knowledge to constrain further the

```
(:action stack
   :parameters (?v1 ?v2)
   :precondition (and (holding ?v1) (clear ?v2))
   :effect (and (not (holding ?v1)) (not (clear ?v2))
                (clear ?v1) (handempty) (on ?v1 ?v2)))


(pre_holding_v1_stack) (pre_clear_v2_stack)
(eff_holding_v1_stack) (eff_clear_v2_stack)
(eff_clear_v1_stack) (eff_handempty_stack) (eff_on_v1_v2_stack)
```

Figure 1: PDDL encoding of the `stack(?v1,?v2)` schema and our propositional representation for this same schema.

space of possible schemata. For instance, in the *blocksworld* 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 $add(\xi)$ lists of an action schema ξ because, in this specific domain, a block cannot be on top of itself. *State invariants* are also constraints of this kind [Fox and Long, 1998].

3. *Observation constraints*. An observations $\mathcal{O}(\tau)$ depicts *semantic knowledge* that constraints further the space of possible action schemata.

In this work we introduce a propositional encoding of the *preconditions*, *negative*, and *positive* effects of a STRIPS action schema ξ using only fluents of two kinds pre_e_ξ and eff_e_ξ (where $e \in \mathcal{I}_{\Psi,\xi}$). This encoding exploits the syntactic constraints of STRIPS so is more compact that the one previously proposed by Aineto *et al.* 2018. In more detail, if pre_e_ξ and eff_e_ξ holds it means that $e \in \mathcal{I}_{\Psi,\xi}$ is a negative effect in ξ while if pre_e_ξ does not hold but eff_e_ξ holds, it means that $e \in \mathcal{I}_{\Psi,\xi}$ is a positive effect in ξ. Figure 1 shows the PDDL encoding of the `stack(?v1,?v2)` schema and our propositional representation for this same schema with `pre_e_stack` and `eff_e_stack` fluents ($e \in \mathcal{I}_{\Psi,stack}$).

3.2 A classical planning compilation for planning with domain models

3.3 The bias of the initial model

4 Goal recognition as planning with unknown domain models

We define the task of *goal recognition with unknown domain models* as a $\langle P, \mathcal{O}(\tau)\rangle$ pair, where:

- $P = \langle F, A[\cdot], I, G[\cdot]\rangle$ is a planning problem where $G[\cdot]$ is the set of possible goals and $A[\cdot]$ is a set of actions s.t., for each $a \in A[\cdot]$, the semantics of a is unknown (i.e. the functions ρ and/or θ of a are undefined).
- $\mathcal{O}(\tau)$ is an observation of a trajectory $\tau(\pi, I)$ produced by the execution of an unknown plan π that reaches a goal $g \in G[\cdot]$ starting from the given initial state.

The *solution* to the *goal recognition with unknown domain models* task is again the subset of goals in $G[\cdot]$ that maximizes expression (1).

Next we show that plans $\pi^\top$ and $\pi^\perp$, and hence an approximation to the $P(\mathcal{O}|g)$ likelihood, can also be computed with an off-the-shelf classical planner despite the action model of the observed agent (i.e., the semantics of actions $a \in A[\cdot]$) is *unknown*. Our approach to compute $\pi^\top$ and $\pi^\perp$ plans with unknown domain models is to adapt the classical planning compilation for the learning of STRIPS action models [Aineto *et al.*, 2018]. First we introduce how we encode a space of STRIPS action models as a set of propositional variables and associated constraints and then, we show how the classical planning compilation is adapted for the computation of the $\pi^\top$ and $\pi^\perp$ plans.

4.1 Explaining observations with unknown domain models

The classical planning compilation for learning STRIPS action models [Aineto *et al.*, 2018] uses a set of $\mathcal{O}(\tau)$ input observations to complete a given classical planning frame $P = \langle F, A[\cdot]\rangle$, where $A[\cdot]$ is a set of actions s.t., for each $a \in A[\cdot]$, the semantics of a is unknown (i.e. the functions ρ and/or θ of a are undefined).

The output of the compilation is a new classical planning problem P_Λ s.t a solution plan π_Λ for P_Λ is a sequence of actions that:

1. Builds the action models of the *learned* domain model.
2. Uses the *learned* domain model to build a plan that is consistent with the given input observations.

Hence, π_Λ will comprise two differentiated blocks of actions: a first set of actions each defining the preconditions and effects of an action model $\xi \in \mathcal{M}'$; and a second set of actions that determine the **application** of the learned ξs while successively **validating** the effects of the action application in every observable point of $\mathcal{O}(\tau)$.

Roughly speaking, in the *blocksworld* domain, the format of the first set of actions of π_Λ will look like `(insert_pre_stack_holding_v1)`,`(insert_eff_stack_clear_v1)`, where the first effect denotes a positive effect and the second one a negative effect to be inserted in $name(\xi)$ =`stack`; and the format of the second set of actions of π_Λ will be like `(apply_unstack blockB blockA)`,`(apply_putdown blockB)` and `(validate_1)`,`(validate_2)`, where the last two actions denote the points at which the states generated through the action application must be validated with the observed states of $\mathcal{O}(\tau)$.

4.2 Computing the $P(\mathcal{O}|g)$ with unknown domain models

Now we are ready to compute the target distribution $P(g|\mathcal{O})$ over the possible goals $g \in G[\cdot]$ given the observation $\mathcal{O}(\tau)$:

1. For each goal, we define the $P^\top$, that constrains the classical planning problem $P = \langle F, A[\cdot], s_0, g\rangle$ to achieve $g \in G[\cdot]$ through a plan $\pi^\top$ *consistent* with the input observation $\mathcal{O}(\tau)$. Note that s_0 is the initial state in the given observation $\mathcal{O}(\tau)$. Likewise we define $P^\perp$, that constrains $P = \langle F, A[\cdot], s_0, g\rangle$ to achieve $g \in G[\cdot]$ through a plan $\pi^\perp$ *inconsistent* with $\mathcal{O}(\tau)$.
2. We use the adapted compilation to compute the classcial planning tasks $P_\lambda^\top$ and $P_\lambda^\perp$ and solve them using an off-the-shelf-classical planner.
3. We compute the cost difference $\Delta(cost(\pi_\top), cost(\pi_\perp))$ where these costs are defined as the length of the postfix

of the $\pi_\lambda^\top$ and $\pi_\lambda^\perp$ plans and plug this cost difference into equation (2) to get the $P(g|\mathcal{O})$ likelihoods.

4. Finally the previous likelihoods are plugged into the Bayes rule from which the goal posterior probabilities are obtained. In this case the $P\mathcal{O}(\tau)$ probabilities are obtained by normalization (goal probabilities must add up to 1 when summed over all possible goals).

5 Evaluation

6 Conclusions

References

[Aineto *et al.*, 2018] Diego Aineto, Sergio Jiménez, and Eva Onaindia. Learning STRIPS action models with classical planning. In *International Conference on Automated Planning and Scheduling, (ICAPS-18)*, pages 399–407. AAAI Press, 2018.

[Fox and Long, 1998] Maria Fox and Derek Long. The automatic inference of state invariants in tim. *Journal of Artificial Intelligence Research*, 9:367–421, 1998.

[Keren *et al.*, 2014] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design. In *International Conference on Automated Planning and Scheduling, (ICAPS-14)*, pages 154–162, 2014.

[MacNally *et al.*, 2018] Aleck M MacNally, Nir Lipovetzky, Miquel Ramirez, and Adrian R Pearce. Action selection for transparent planning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1327–1335. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[Masters and Sardina, 2017] Peta Masters and Sebastian Sardina. Deceptive path-planning. In *IJCAI 2017*, pages 4368–4375. AAAI Press, 2017.

[McDermott *et al.*, 1998] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL – The Planning Domain Definition Language, 1998.

[Pereira *et al.*, 2017] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi. Landmark-based heuristics for goal recognition. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*. AAAI Press, 2017.

[Pozanco *et al.*, 2018] Alberto Pozanco, Yolanda E.-Martín, Susana Fernández, and Daniel Borrajo. Counterplanning using goal recognition and landmarks. In *International Joint Conference on Artificial Intelligence, (IJCAI-18)*, pages 4808–4814, 2018.

[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.

[Ramírez, 2012] Miquel Ramírez. *Plan recognition as planning*. PhD thesis, Universitat Pompeu Fabra, 2012.