1

One-Shot Learning of Temporal Action Models with Constraint Programming

Antonio Garrido and **Sergio Jiménez**

Abstract. This work presents a constraint programming (CP) formulation for the learning of temporal planning action models. This paper focus on the extreme scenario where just a single partial observation of the execution of a temporal plan is available (i.e. one-shot) to evidence that this task is closely related to plan synthesis and plan validation. Our CP formulation models time-stamps for actions, causal link relationships (conditions and effects), threats and effect interferences that appear in the plan synthesis and validation tasks. Further our CP formulation can accommodates a different range of expressiveness, subsuming the PDDL2.1 temporal semantics and is solver-independent, meaning that an arbitrary CSP solver can be used for its resolution.

1 Introduction

Temporal planning is a expressive planning model that relaxes the assumption of instantaneous actions of *classical planning* [5]. Actions in temporal planning are called *durative*, i.e. they have durations so conditions/effects may hold/happen at different times. This means that actions in the temporal plannin model can be executed in parallel and overlap in several ways [4] and that valid solutions for temporal planning instances indicate the precise time-stamp when actions start and end [6].

Despite the potential of state-of-the-art planners, its applicability to the real world is still somewhat limited because of the difficulty of specifying correct and complete planning models [8]. The more expressive the planning model is, the more evident becomes this knowledge acquisition bottleneck, which jeopardizes the usability of AI planning technology. This has led to a growing interest in the planning community for the learning of action models [7]. Most approaches for learning planning action models are purely inductive and often require large datasets of observations, e.g. thousands of plan observations to compute a statistically significant model that minimizes some error metric over the observations [11, 10, 12, 9]. Defining model learning as an optimization task over a set of observations does not guarantee completeness (the learned model may fail to explain an observation), nor correctness (the states induced by the execution of the plan generated with the model may contain contradictory information).

This paper analyzes the application of *Constraint Programming* for the *one-shot learning* of temporal action models, that is, the extreme scenario where action models are learned from a single observation of the execution of a temporal plan. While learning an action model for classical planning means computing the actions' conditions and effects that are consistent with the input observations, learning temporal action models extends this to: i) identify how these conditions and effects are temporally distributed in the action execution, and ii) estimate the action duration. As a motivating example, let us assume a logistics scenario. Learning the temporal planning model will allow us: i) to better understand the insights of the logistics in terms of what is possible (or not) and why, because the model is consistent with the observed data; ii) to suggest changes that can improve the model originally created by a human, e.g. re-distributing the actions' conditions, provided they still explain the observations; and iii) to automatically elaborate similar models for similar scenarios, such as public transit for commuters, tourists or people in general in metropolitan areas —*a.k.a.* smart urban mobility.

The contributions of our CP formulation are two-fold:

1. This is the first approach for learning action models for temporal planning where plan observations can refer to the execution of overlapping actions. Learning classical action models from sequential plans has been addressed by different approaches [2]. Since pioneering learning systems like ARMS [11], we have seen systems able to learn action models with quantifiers [1, 15], from noisy actions or states [10, 12], from null state information [3], or from incomplete domain models [13, 14]. But, to our knowledge, none of these systems learns from the execution of overlapping actions, this makes our approach appealing for learning in multi-agent environments.
2. Our CP formulation evidences the strong relation of the *one-shot learning* of planning action models with planning and plan validation. This validation capacity is beyond the functionality of VAL (the standard plan validation tool [6]) because we can address *plan validation* of a partial (or even an empty) action model with a partially observed plan trace (VAL requires a full plan and a full action model for plan validation). Further, our CP formluation allows that an arbitrary CSP solver can be used for the learning, planning and validation tasks.

2 Background

This section formalizes the *temporal* planning model that we follow in this work.

2.1 Temporal Planning

Let F be a set of facts that represent propositional variables. A state s is a time-stamped full assignment of values to variables. The initial state is fully observable (i.e. $|I| = |F|$) and $G \subseteq F$ is a set of goal conditions over F.

A *temporal planning problem* is a tuple $\langle F, I, G, A \rangle$ where F, I and G are defined like in classical planning, and A represents the set of *durative actions*. There are several options that allow for a high expressiveness of durative actions. On the one hand, an action can have a fixed duration, a duration that ranges within an interval or a distribution of durations. On the other hand, actions may have conditions/effects at different times, such as conditions that must hold some time before the action starts, effects that happen just when the

[1] Universitat Politècnica de València
Camino de Vera s/n. 46022 Valencia, Spain
{agarridot,serjice}@dsic.upv.es.

action starts, in the middle of the action or some time after the action finishes [**?**].

We assume that actions are grounded from action schemas or operators. A popular model for temporal planning is given by PDDL2.1 [5, **?**], a language that somewhat restricts temporal expressiveness, which defines a durative action a with the following elements:

- $\mathsf{dur}(a)$, a positive value for the action duration.
- $\mathsf{cond}_s(a), \mathsf{cond}_o(a), \mathsf{cond}_e(a) \subseteq F$. Unlike the *pre*conditions of a classical action, now conditions must hold before a (*at start*), during the entire execution of a (*over all*) or when a finishes (*at end*), respectively. In the simplest case, $\mathsf{cond}_s(a) \cup \mathsf{cond}_o(a) \cup \mathsf{cond}_e(a) = \mathsf{pre}(a)$.[2]
- $\mathsf{eff}_s(a)$ and $\mathsf{eff}_e(a)$. Now effects can happen *at start* or *at end* of a, respectively, and can still be positive or negative. Again, in the simplest case $\mathsf{eff}_s(a) \cup \mathsf{eff}_e(a) = \mathsf{eff}(a)$.

The semantics of a PDDL2.1 durative action a can be defined in terms of two discrete events, $\mathsf{start}(a)$ and $\mathsf{end}(a) = \mathsf{start}(a) + \mathsf{dur}(a)$. This means that if action a starts on state s, $\mathsf{cond}_s(a)$ must hold in s; and ending a in state s' means $\mathsf{cond}_e(a)$ holds in s'. *Over all* conditions must hold at any state between s and s' or, in other words, throughout interval $[\mathsf{start}(a)..\mathsf{end}(a)]$. Analogously, *at start* and *at end* effects are instantaneously applied at states s and s', respectively —continuous effects are not considered. Fig. 1 shows two durative actions that extend the classical actions of Fig. **??**. Now `board-truck` has a fixed duration whereas in `drive-truck` the duration depends on the two locations.

```
(:durative-action board-truck
  :parameters (?d - driver ?t - truck ?l - location)
  :duration (= ?duration 2)
  :condition (and (at start (at ?d ?l)) (at start (empty ?t))
                  (over all (at ?t ?l)))
  :effect (and (at start (not (at ?d ?l))) (at start (not (empty ?t)))
               (at end (driving ?d ?t))))

(:durative-action drive-truck
  :parameters (?t - truck ?from - location ?to - location ?d - driver)
  :duration (= ?duration (driving-time ?from ?to))
  :condition (and (at start (at ?t ?from)) (at start (link ?from ?to))
                  (over all (driving ?d ?t)))
  :effect (and (at start (not (at ?t ?from))) (at end (at ?t ?to))))
```

Figure 1. PDDL2.1 schema for two durative actions from the *driverlog* domain.

A temporal plan is a set of pairs $\langle (a_1, t_1), (a_2, t_2) \dots (a_n, t_n) \rangle$. Each (a_i, t_i) pair contains a durative action a_i and $t_i = \mathsf{start}(a_i)$. This temporal plan induces a state sequence formed by the union of all states $\{s_{t_i}, s_{t_i + \mathsf{dur}(a_i)}\}$, where there exists a state $s_0 = I$, and $G \subseteq s_{end}$, being s_{end} the last state induced by the plan. Though a sequential temporal plan is syntactically possible, it is semantically useless. Consequently, temporal plans are always given as parallel plans.

2.2 The space of *durative* actions

We denote as $\mathcal{I}_{\xi,\Psi}$ the set of symbols (vocabulary) that can appear in the *conditions* and/or *effects* of a *durative* action schema ξ. Formally this set is defined as the FOL interpretations of Ψ over the action parameters $pars(\xi)$.

For a *durative* action schema ξ its space of possible action models is then $2^{5 \times |\mathcal{I}_{\xi,\Psi}|} \times D$ where D is the number of different values for the durations of any action shaped by the ξ schema. Note that this space is significantly larger than for learnnign STRIPS actions that is just $2^{2 \times |\mathcal{I}_{\xi,\Psi}|}$ [11].

With this regard an action schema can be coded by 5 bit vector of length $|\mathcal{I}_{\xi,\Psi}|$ and the *Hamming distance* can be used straigthforward to asses the similarity of two given models as well as to compute the *Precision and Recall* metrics as in Machine Learning. For instance for comparing a learned model with respecto to a given reference model that serve as baseline.

3 Learning of Temporal Action Models with Constraint Programming

This section defines the learning task we addres in this paper and our CP formulation for addressing it with off-the-shelf CSP solvers.

3.1 One-shot learning of temporal action models

We define our one-shot learning task of a temporal action model as a tuple $\langle F, I, G, A?, O \rangle$, where:

- $\langle F, I, G, A? \rangle$ is a temporal planning problem in which actions are partially specified. Actions in $A?$ are those observed in the plan trace. They are partially specified because we do not know the exact structure in terms of distribution of conditions/effects nor the duration. In the worst case, we only know the set of symbols (vocabulary) that can appear in the *conditions* and/or *effects* but we can also assume that expert or prior knowledge is available bounding this vocabulary fora given action.
- O is the sequence of observations corresponding to a plan trace which contains the time when every action a in $A?$ starts, i.e. all $\mathsf{start}(a)$ that have been observed (by a sensor or human observer).

A solution to this learning task is a fully specified model of temporal actions $\mathcal{A}$, with all actions of $A?$, where the duration and distribution of conditions/effects is completely specified. In other words, for each action $a \in A?$, we have its equivalent version in $\mathcal{A}$ where we have learned $\mathsf{dur}(a)$, $\mathsf{cond}_s(a)$, $\mathsf{cond}_o(a)$, $\mathsf{cond}_e(a)$, $\mathsf{eff}_s(a)$ and $\mathsf{eff}_e(a)$. Actions in $\mathcal{A}$ must be consistent with the partial specification given in $A?$, having exactly the same conditions and effects, starting as observed in O, and inducing a temporal plan from I that satisfies G. Intuitively, $\mathcal{A}$ is a solution to the learning task if it explains all the observations (completeness) and its subjacent temporal model implies no contradictions in the states induced by their execution (correctness).

3.2 The constraint programming model

4 Results

5 Conclusions

REFERENCES

[1] Eyal Amir and Allen Chang, 'Learning partially observable deterministic action models', *Journal of Artificial Intelligence Research*, **33**, 349–402, (2008).

[2] Ankuj Arora, Humbert Fiorino, Damien Pellier, Marc Métivier, and Sylvie Pesty, 'A review of learning planning action models', *The Knowledge Engineering Review*, **33**, (2018).

[2] Note that in classical planning, $\mathsf{pre}(a) = \{p, not - p\}$ is contradictory. In temporal planning, $\mathsf{cond}_s(a) = \{p\}$ and $\mathsf{cond}_e(a) = \{not - p\}$ is a possible situation, though very unusual

[3] S. N. Cresswell, T.L. McCluskey, and M.M West, 'Acquiring planning domain models using LOCM', *The Knowledge Engineering Review*, **28(2)**, 195–213, (2013).

[4] William Cushing, Subbarao Kambhampati, Daniel S Weld, et al., 'When is temporal planning really temporal?', in *Proceedings of the 20th international joint conference on Artifical intelligence*, pp. 1852–1859. Morgan Kaufmann Publishers Inc., (2007).

[5] Maria Fox and Derek Long, 'Pddl2.1: An extension to pddl for expressing temporal planning domains', *Journal of artificial intelligence research*, **20**, 61–124, (2003).

[6] Richard Howey, Derek Long, and Maria Fox, 'VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL', in *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pp. 294–301. IEEE, (2004).

[7] Sergio Jiménez, Tomás De la Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrajo, 'A review of machine learning for automated planning', *The Knowledge Engineering Review*, **27**(4), 433–467, (2012).

[8] Subbarao Kambhampati, 'Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models', in *Proceedings of the National Conference on Artificial Intelligence (AAAI-07)*, volume 22(2), pp. 1601–1604, (2007).

[9] Jirí Kucera and Roman Barták, 'LOUGA: learning planning operators using genetic algorithms', in *Pacific Rim Knowledge Acquisition Workshop, PKAW-18*, pp. 124–138, (2018).

[10] Kira Mourão, Luke S. Zettlemoyer, Ronald P. A. Petrick, and Mark Steedman, 'Learning STRIPS operators from noisy and incomplete observations', in *Conference on Uncertainty in Artificial Intelligence, UAI-12*, pp. 614–623, (2012).

[11] Qiang Yang, Kangheng Wu, and Yunfei Jiang, 'Learning action models from plan examples using weighted MAX-SAT', *Artificial Intelligence*, **171**(2-3), 107–143, (2007).

[12] Hankz Hankui Zhuo and Subbarao Kambhampati, 'Action-model acquisition from noisy plan traces', in *International Joint Conference on Artificial Intelligence, IJCAI-13*, pp. 2444–2450, (2013).

[13] Hankz Hankui Zhuo and Subbarao Kambhampati, 'Model-lite planning: Case-based vs. model-based approaches', *Artificial Intelligence*, **246**, 1–21, (2017).

[14] Hankz Hankui Zhuo, Tuan Anh Nguyen, and Subbarao Kambhampati, 'Refining incomplete planning domain models through plan traces', in *International Joint Conference on Artificial Intelligence, IJCAI-13*, pp. 2451–2458, (2013).

[15] Hankz Hankui Zhuo, Qiang Yang, Derek Hao Hu, and Lei Li, 'Learning complex action models with quantifiers and logical implications', *Artificial Intelligence*, **174**(18), 1540–1569, (2010).