Open challenges in learning action models for planning

Eva Onaindia
Valencian Research Artificial Intelligence Institute (VRAIN)
Universistat Politècnica de València

The goal of this talk is to walk the audience through the field of action model learning by using the learning system FAMA (Flexible Action Model Acquisition) as a reference.

FAMA is a joint work with Diego Aineto and Sergio Jimenez, and it is part of Diego's PhD dissertation.

Index

1. What is action-model learning?
2. The two roles in action model learning
3. Components of the action-model learning task
    - The space of possible action models
    - The learning examples
    - The error functions
4. Learning paradigms
5. Open issues

1. What is action model learning?

- Learning/acquisition of action models for planning

> Synthesize and update the knowledge of an agent about the **preconditions and effects of the actions** that can be executed in the environment

Action model: $\mathbf{M} = \langle X, A, Pre, Eff \rangle$

$X = \{x_1, \ldots, x_k\}$: a finite set of state variables; each x is associated with a finite domain $D(x)$

A: a finite set of actions

Pre: $A \rightarrow Constr(X)$ *applicabiity constraint* that describes the condition in terms of X

Eff: $A \rightarrow Constr(X \cup X^+)$ *effects constraint*; X^+ refers to the updated values of the state variables after execution

1. What is action model learning?

Action model: $M = \langle X, A, Pre, Eff \rangle$

$X = \{x_1, \ldots, x_k\}$: a finite set of state variables

A: a finite set of actions

Pre: $A \rightarrow Constr(X)$ *applicabiity constraint* that describes the condition in terms of X

Eff: $A \rightarrow Constr(X \cup X^+)$ *effects constraint*; X^+ refers to the updated values of the state variables after execution

Learning relational action models (STRIPS fragment of PDDL)

Common requirements in the formulation of the action model learning task:

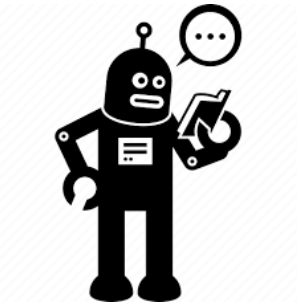
- X and A are known
- Pre and Eff are the parameters to estimate

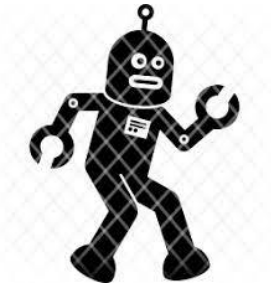
1. What is action model learning?

- Learning/acquisition of action models for planning

> Synthesize and update the knowledge of an **agent** about the preconditions and effects of the actions that can be executed in the **environment**

two entities/roles





learning agent

X, A

acting agent

M= <X, A, Pre, Eff>

1. What is action model learning?

Find a model in the space of possible action models of the acting agent $\mathcal{M}$

Learn an alphabet $\forall a \in A$: a set of Boolean variables that indicate whether a state variable belongs to the preconditions or effects the action

```
(:action move                                |
  :parameters (?v1 ?v2)                      |
  :precondition (and (at ?v1)                |   <move, pre, at(v1)>
                     (connected ?v1 ?v2))    |   <move, pre, connected(v1,v2)>
  :effect (and (not (at ?v1))                |   <move, eff, at(v1)>
               (at ?v2)))                    |   <move, eff, at(v2)>
```

<move,pre,at(v1)> **<move,pre,connected(v1,v2)>** <move,eff,connected(v1,v2)>
<move,pre,at(v2)> <move,pre,connected(v2,v1)> <move,eff,connected(v2,v1)>
<move,eff,at(v1)> <move,pre,connected(v1,v1)> <move,eff,connected(v1,v1)>
<move,eff,at(v2)> <move,pre,connected(v2,v2)> <move,eff,connected(v2,v2)>
. . .

1. What is action model learning?

Define $\mathbb{A}$ as the union of the alphabets $\forall a \in A$

Finding a model $M \in \mathcal{M}$ involves learning the alphabet of each action; that is, searching in the space of $\mathbb{A}$, the space of all the Boolean variables defined through the instantiation of predicates with the parameters of the actions

A model $M \in \mathcal{M}$ corresponds to an interpretation in $2^{\mathbb{A}}$

2. The two roles in action model learning

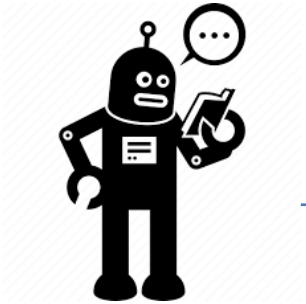
Objective: to learn the model of the acting agent

- Communication learning agent with acting agent
- Type of information that is accessible to the learner from the actor
- Acquisition mechanism

Two approaches:

1. Interrogation
2. Observation

2. The two roles in action model learning: interrogation [Verma21]



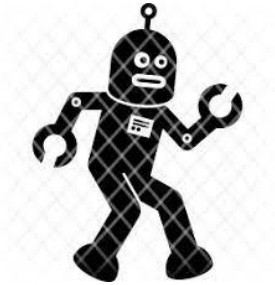
user

X, A

Agent Assessment Module

query

response



acting agent

M= <X, A, Pre, Eff>

Agent-Assessment Module:

$\mathcal{Q} : \mathcal{M} \longrightarrow \mathcal{R}$

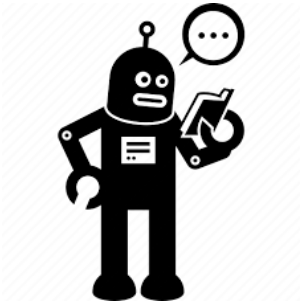
query: $< s_I, \pi >$

response: $<I, s_F >$

- Models are functionally equivalent
- Distinguishable models
- Consistency of a model with the ground truth model

Learning an action model in a user-interpretable representation

2. The two roles in action model learning: observation



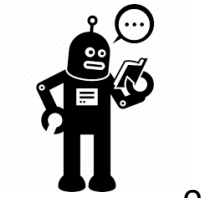
observer

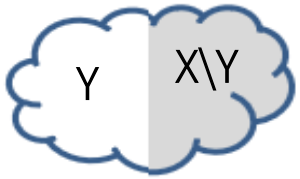
Y: observable variables

acting agent

M = <X, A, Pre, Eff>

$Y \subseteq X$





observable hidden

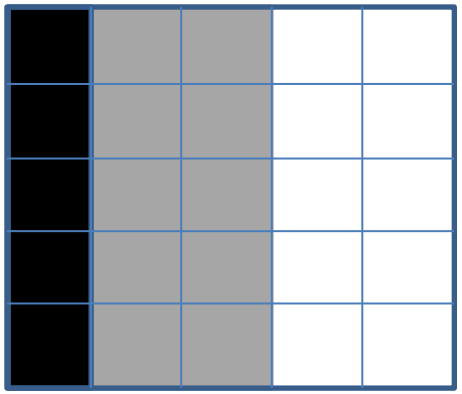
2. The two roles in action model learning: observation

Full observability: $Y=X$

Partial observability: $Y \subset X$ $\quad X=\{x_1, \ldots, x_k\}$ $Y=\{y_1, \ldots, y_j\}$ $\quad j \leq k$

Non-determinism: $D(y_i)= D(x_j) \cup \{unknown\}$

$loc_{x,y}$



$obs\_loc_{x,y}$

hidden: $\neg obs\_loc_{x,y}$ $x=1, y \in \{1,..,5\}$

observable: $obs\_loc_{x,y} = loc_{x,y}$ $x=\{4,5\}, y \in \{1,..,5\}$

noisy: $obs\_loc_{x,y}$=unknown, $x=\{2,3\}, y \in \{1,..,5\}$

3. Components of the action-model learning task

Formalization of the action-model learning task [Aineto22]:

1. The space of possible action models $\mathcal{M}$
2. The learning examples (from observations)
3. The error function

3. Learning task: model space

Define $\mathbb{A}$ as the union of the alphabets $\forall a \in A$

$\mathcal{M}$ includes all the action models that are definable as full assignments of the alphabet $\mathbb{A}$

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

full variable assignment of $\mathbb{A}$ that defines the STRIPS operator `stack(x,y)`

combinatorial search

$\langle \texttt{stack}, \texttt{pre}, holding(x) \rangle = \texttt{True}$
$\langle \texttt{stack}, \texttt{pre}, clear(y) \rangle = \texttt{True}$

$\langle \texttt{stack}, \texttt{pre}, holding(y) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{pre}, ontable(x) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{pre}, ontable(y) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{pre}, on(x,x) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{pre}, on(y,x) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{pre}, on(y,y) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{pre}, clear(x) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{pre}, handempty() \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{pre}, on(x,y) \rangle = \texttt{False}$

Pre (`stack`)

$\langle \texttt{stack}, \texttt{eff}, holding(x) \rangle = \texttt{True}$
$\langle \texttt{stack}, \texttt{eff}, clear(y) \rangle = \texttt{True}$
$\langle \texttt{stack}, \texttt{eff}, clear(x) \rangle = \texttt{True}$
$\langle \texttt{stack}, \texttt{eff}, handempty() \rangle = \texttt{True}$
$\langle \texttt{stack}, \texttt{eff}, on(x,y) \rangle = \texttt{True}$

$\langle \texttt{stack}, \texttt{eff}, holding(y) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{eff}, ontable(x) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{eff}, ontable(y) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{eff}, on(x,x) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{eff}, on(y,x) \rangle = \texttt{False}$
$\langle \texttt{stack}, \texttt{eff}, on(y,y) \rangle = \texttt{False}$

Eff (`stack`)

3. Learning task: model space

Bit-vectors provide a convenient encoding of the space of action models

1. A bit encodes whether a variable of the alphabet of an action is set to TRUE or FALSE
2. The four ways in which a schematic variable can appear in Pre and Eff are stored using two bits
3. Allow to quantify the distance between two models
4. Allow to sort the space of possible action models
5. Enable the activation/deactivation of the bit associated to a constraint when the corresponding precondition or effect is added/removed in the model of an operator

Many learning systems use a binary encoding of the space of action models:
ARMS [Yang07], **SLAF** [Amir08], **ALICE** [Mourão10], **FAMA** [Aineto19].

3. Learning task: the learning examples

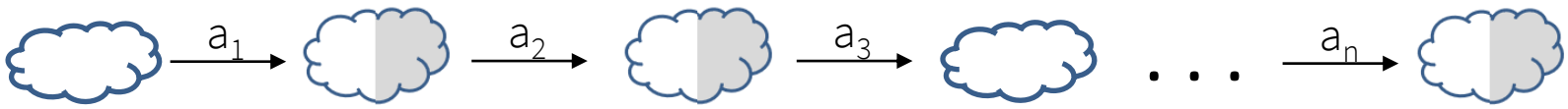
Data are collected from the acting agent typically in the form of observations of trajectories (observations of sequence of actions and states traversed during the agent execution)

$$\tau = \langle s_0, a_1, s_1, \ldots, a_n, s_n \rangle$$

$$\mathcal{O} = \langle s_0^o, a_1^o, s_1^o, \ldots, a_m^o, s_m^o \rangle$$

Common assumptions:

s_0 is fully observed: $s_0 = s_0^o$
$a_1, \ldots, a_n$ is fully observed



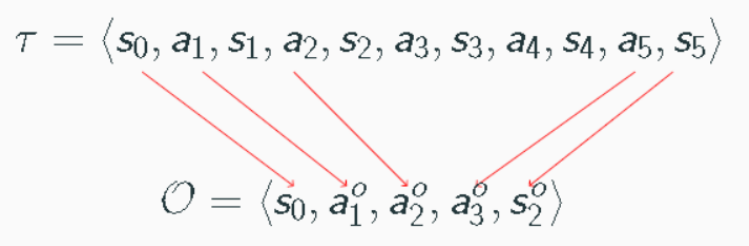
The observer captures at least one state variable at each execution step of the actor

The length of the agent execution τ is known

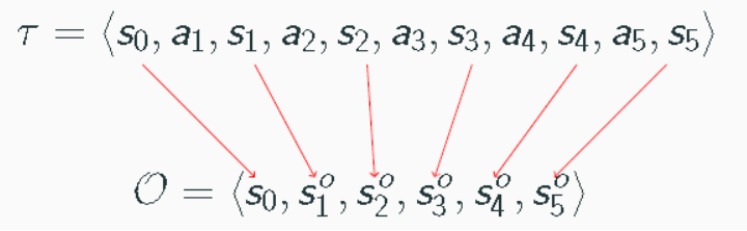
3. Learning task: the learning examples

Learning examples in FAMA:

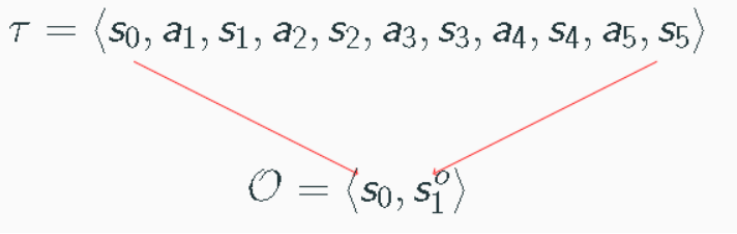
τ and $\mathcal{O}$ are **consistent** iff $\mathcal{O} \subseteq \tau$



partial plan



partially observed states

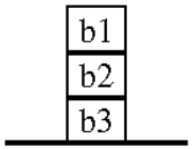


initial and final states

Intermittency =
Gapped observable sequences

3. Learning task: the learning examples

```
(trajectory (:objects b1 b2 b3)
(:state (clear b1) (handempty) (on b1 b2) (on b2 b3) (ontable b3))
(:action (unstack b1 b2))
(:state (clear b2) (holding b1) (on b2 b3) (ontable b3))
(:action (putdown b1))
(:state (clear b1) (clear b2) (handempty) (on b2 b3) (ontable b1) (ontable b3))
(:action (unstack b2 b3))
(:state (clear b1) (clear b3) (holding b2) (ontable b1) (ontable b3))
(:action (stack b2 b1))
(:state (clear b2) (clear b3) (handempty) (on b2 b1)  (ontable b1) (ontable b3))
(:action (pickup b3))
(:state (clear b2) (holding b3) (on b2 b1) (ontable b1))
(:action (stack b3 b2))
(:state (clear b3) (handempty) (on b3 b2) (on b2 b1) (ontable b1)))
```



initial state

Trajectory from the *Blocksworld* domain that inverts a 3-block tower from the initial state of the figure

```
(example (:objects b1 b2 b3)
(:state (clear b1) (handempty) (on b1 b2) (on b2 b3) (ontable b3))
(:action (unstack b1 b2))
(:action (putdown b1))
(:action (unstack b2 b3))
(:action (stack b2 b1))
(:action (pickup b3))
(:action (stack b3 b2))
(:state (clear b3) (handempty) (on b3 b2) (on b2 b1) (ontable b1)))
```

3. Learning task: the learning examples

Observations of trajectories of the acting agent are the most commonly used learning examples

Other types of learning examples: tuples <pre-state, post-state> of an executed action [Mourão12]

Graphs that encode a fragment of the transition system can be regarded as comprising several trajectories that share some segments [Bonet20].

3. Components of the action-model learning task: the error function

Given a planning frame <X, A>, the learning task is a tuple $< M[], \mathcal{L} >$

$M[]$ is an empty action model (alternatively *partially specified* when some fragments are known)

$\mathcal{L} = \{\mathcal{O}_i\}$ is the set of learning examples

Solution to a learning task $< M[], \mathcal{L} >$

the pair ϕ =<Pre, Eff> such that $M[\phi] \in \mathcal{M}$ is the action model consistent with $M[]$ that *best fits* the examples in $\mathcal{L}$

3. Components of the action-model learning task: the error function

We need an error function for guiding the search in the space of action models and evaluating the quality of the learned model

Error functions to evaluate the quality of the model $\mathbb{M}[\phi]$ with respect to $\mathcal{L}$ (data-based)

Error functions to evaluate the quality of the model $\mathbb{M}[\phi]$ regardless $\mathcal{L}$ (structural)

3. Components of the action-model learning task: the error function

Quantify error of τ with respect to $\mathcal{O}$

For $\mathcal{O} = < s_0^o , a_1^o , s_1^o , \ldots , a_n^o , s_n^o >$
that contain the complete action sequence, reproduce $a_1^o, \ldots, a_n^o$
with $\mathbb{M}$ and count the unsatisfied preconditions and redundant effects [Yang07]

Cost of fixing the trajectory τ to make it consistent with $\mathcal{O}$
(*edit operations* that add/remove variables from τ) [Kucera18].
Extending the idea of edit operations to model-centric interpretation [Aineto19b].

Cost functions to assess the fitness of a model to the set of $\mathcal{L}$ [Aineto20]

$$\theta^* = \arg min_\theta \left\{ \sum_{\mathcal{L}} min_{\tau \in \mathrm{T}}\, cost(\tau) \right\}$$

cost of the best trajectory

3. Components of the action-model learning task: the error function

Error functions to evaluate the quality of the model $\mathbb{M}[\phi]$ regardless $\mathcal{L}$

Structural error functions

Only consider the structure of the actions

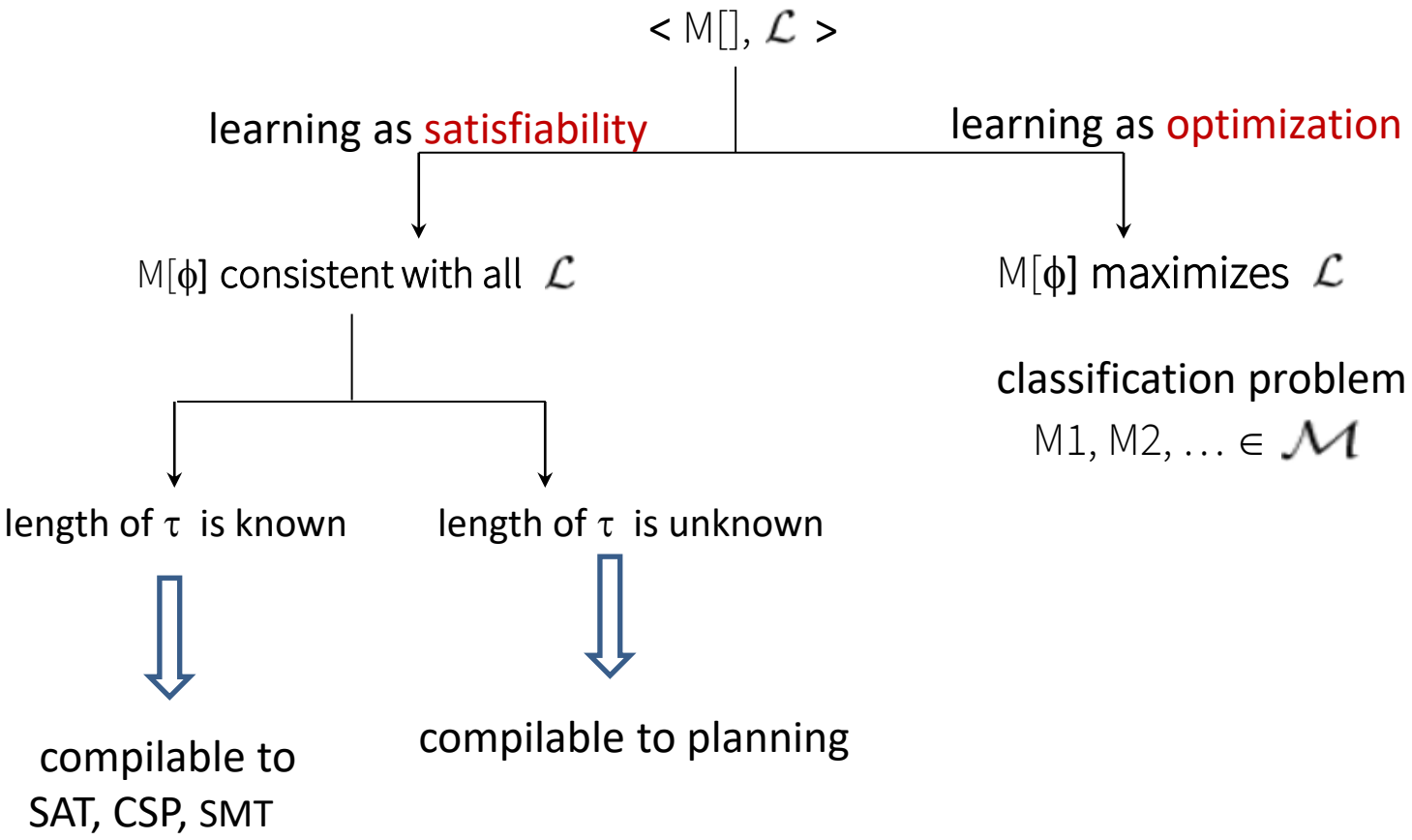
Represent regularization penalties to indicate preference for simpler models or to establish initial preferences for certain action models

Useful to bound the size of the action model space (models that contain up to *k* preconditions/effects or models that differ up to k preconditions/effects from a given solution)

Number of variables, their arity, number of actions and their parameters [Pasula07, Aineto19]

Weights for expressing preferences of the formulae that appear in Pre/Eff

4. Learning paradigms



Maturity of the techniques
Scalability issues
No noise in examples

Lack of guarantee that M[ϕ] is consistent with $\mathcal{L}$
Handling of noise

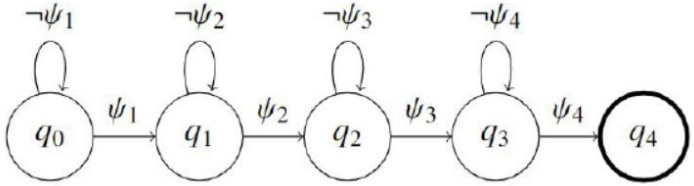
4. FAMA: Compilation to planning

Main idea: learn by editing (https://github.com/daineto/meta-planning)

Build a planning problem $P_{M_a}(\mathcal{L})$

M_a has empty precondition, add, and delete lists (all alphabet variables set to False)

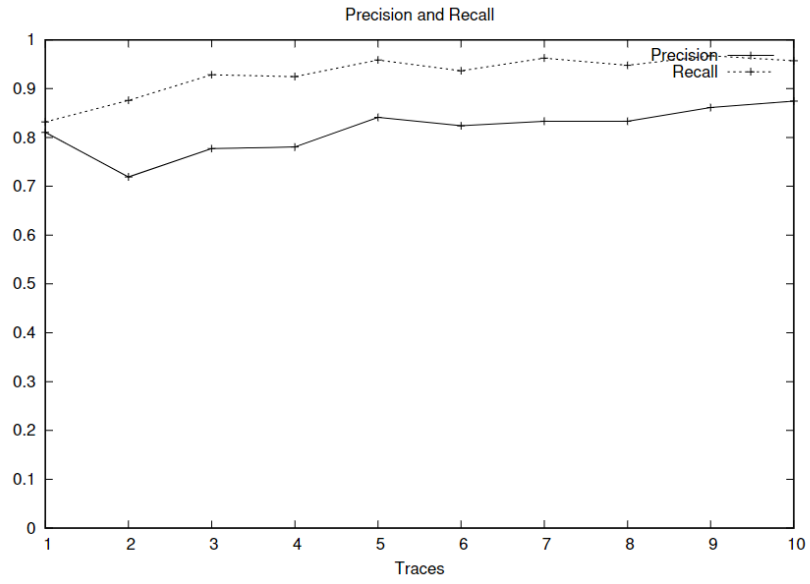
`insert` actions plus actions of a monitor automaton to validate the observations with the programmed action model

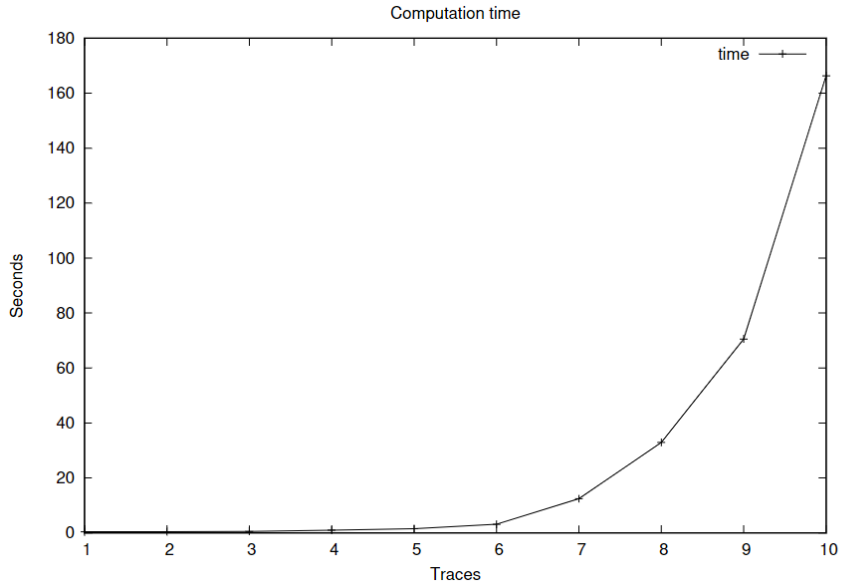


4. FAMA: some results

Known Horizon: Quality and Scalability

15 domains
10 learning examples per domain (each 10 actions and 10 intermediate states)
Learning examples with partial observability (10%) of intermediate states.
Timeout = 1000s (score 0 if process killed)



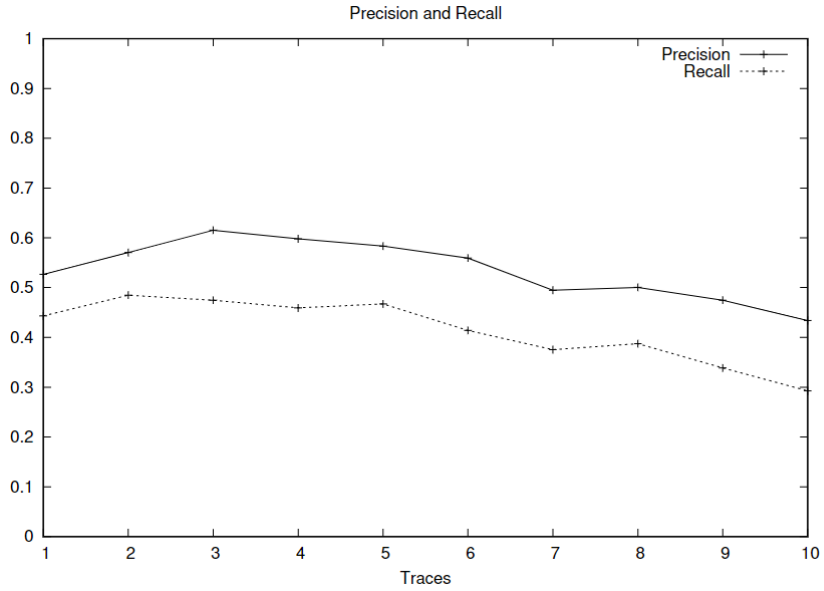


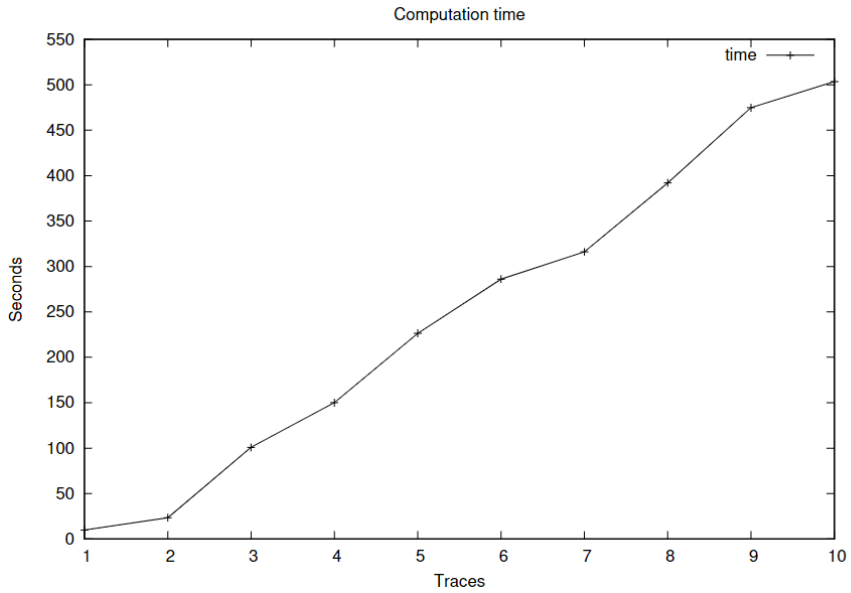
Metrics: precision and recall wrt. the true model.
Learning with few input examples yield models that contain 95% of the pre/eff of the GTM.

4. FAMA: some results

Unknown Horizon: Quality and Scalability

Learning examples contain **only initial and final states.**





Quality drop caused by the increasing number of timeouts.

5. Open issues

Informativeness vs Scalability

Compilation to planning is an informative approach that sacrifices scalability for the ability to be solved with off-the-shelf planners

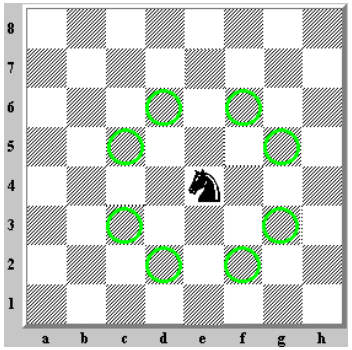
Generating trajectories consistent with the observations require a combinatorial search process

Cheaper but potentially less informative solution (heuristic approximation):

- interpretation of observations as landmarks
- use of landmark-based heuristics to generate the trajectory
- similar ideas used in Goal Recognition [Vered18, Pereira20])

5. Open issues

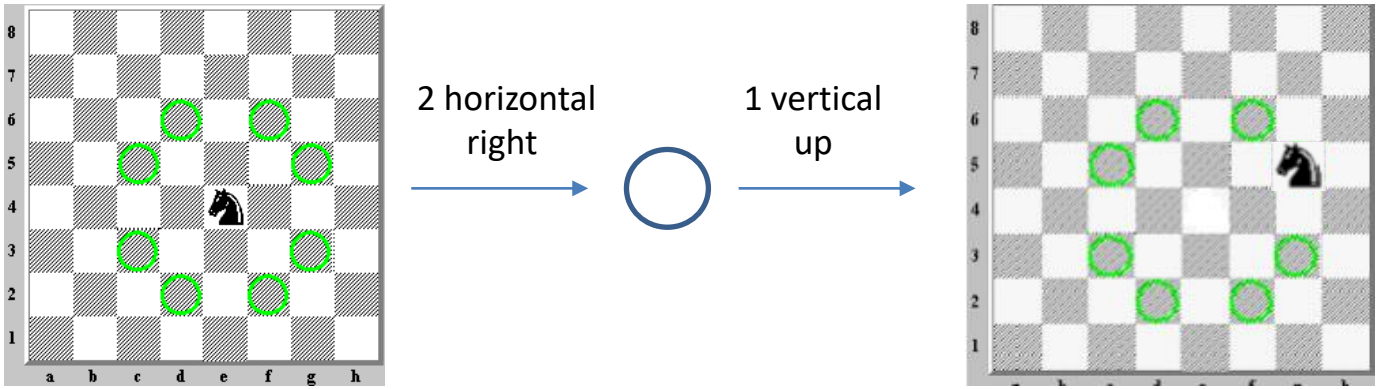
Exploiting intermittency to learn procedural action models



The knight moves two squares horizontally right and one square vertically up

The knight moves one square horizontally left and two squares vertically up

....

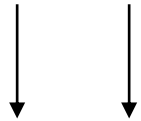


Intermittency enables to deal with latent states that are not part of the transition system but are helpful to understand what is happening

5. Open issues

Observations of the transition system of the acting agent: is the agent pursuing some goal?

Common requirements in action-model learning: $\mathbb{X}$ and $\mathbb{A}$ are known



predicates and actions of
the target model

$\mathbb{X}$ and $\mathbb{A}$ must contain the predicates/actions of the learning examples … and optionally more

extra-preconditions to capture when actions are applied to achieve a given goal

learning a policy: the extra-preconditions reflect the behaviour of the agent, not the physics of the world

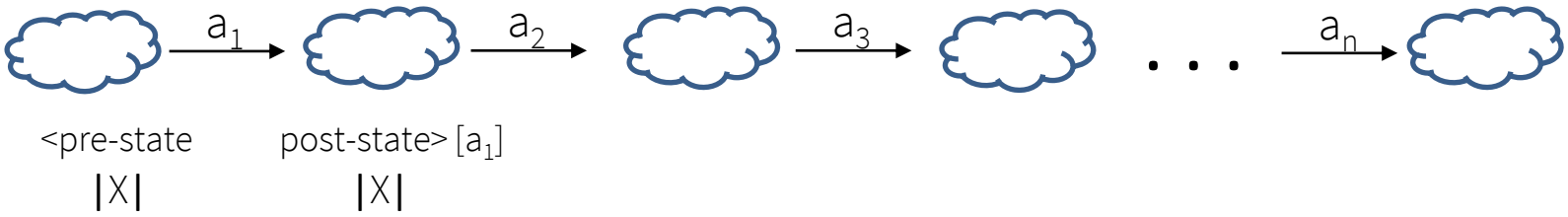
Usage in two steps:

1. learn the action model $\mathbb{M}[\phi]$
2. learn the extra-preconditions (policy)

5. Open issues

Can we use ML techniques to learn action models?

When learning examples provide full knowledge (fully observable)



Dealing with noise: create negative samples by adding noise in pre-state

<noisy-pre-state, noisy-pre-state> $[a_1]$ ⇨ a_1 is not executable and no changes are observed

Dealing with partial observability (intermittency):
can ML cope with irregular alternation of states and actions?

References

[Verma21] Pulkit Verma, Shashank Rao Marpally, Siddharth Srivastava. Asking the Right Questions: Learning Interpretable Action Models Through Query Answering. AAAI 2021.
[Aineto22] Diego Aineto, Sergio Jiménez, Eva Onaindia. A Comprehensive Framework for Learning Declarative Action Models. JAIR 2022.
[Yang07] Yang, Q., Wu, K., & Jiang, Y. (2007). Learning action models from plan examples using weighted MAX-SAT. Artificial Intelligence, 171(2-3), 107–143.
[Amir08] Amir, E., & Chang, A. (2008). Learning partially observable deterministic action models. Journal of Artificial Intelligence Research, 33, 349–402.
[Mourã010] Mourão, K., Petrick, R. P., & Steedman, M. (2010). Learning action effects in partially observable domains. ECAI, pp. 973–974.
[Aineto19] Aineto, D., Jiménez, S., & Onaindia, E. (2019). Learning action models with minimal observability. Artificial Intelligence Journal, 275, 104–137.
[Mourão12] Mourão, K., Zettlemoyer, L. S., Petrick, R. P. A., & Steedman, M. (2012). Learning STRIPS operators from noisy and incomplete observations. In Conference on Uncertainty in Artificial Intelligence, UAI-12, pp. 614–623.
[Bonet20] Bonet, B., & Geffner, H. (2020). Learning First-Order Symbolic Representations for Planning from the Structure of the State Space. In European Conference on Artificial Intelligence (ECAI), pp. 2322-2329.
[Vered18] Vered, M., Pereira, R. F., Kaminka, G., & Meneguzzi, F. R. (2018). Towards online goal recognition combining goal mirroring and landmarks. AAMAS, 2018.
[Pereira20] Pereira, R. F., Oren, N., & Meneguzzi, F. (2020). Landmark-based approaches for goal recognition as planning. Artif. Intell., 279.
[Kucera18] Kucera, J., & Barták, R. (2018). LOUGA: learning planning operators using genetic algorithms. In Pacific Rim Knowledge Acquisition Workshop, PKAW-18, pp. 124–138.
[Aineto20] Aineto, D., Jiménez, S., & Onaindia, E. (2020). Observation decoding with sensor models: recognition tasks via classical planning. ICAPS 2020.
[Pasula07] Pasula, H. M., Zettlemoyer, L. S., & Kaelbling, L. P. (2007a). Learning symbolic models of stochastic domains. Journal of Artificial Intelligence Research, 29, 309–352.
[Aineto19b] Aineto, D., Jiménez, S., Onaindia, E., & Ram´ırez, M. (2019). Model Recognition as Planning. ICAPS 2019, pp. 13–21.