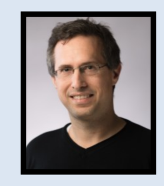
Safe Learning of Lifted Action Models

Brendan Juba[1] , Hai Le[1] , and Roni Stern[2,3] 

[1]Washington University in St. Louis, USA

[2]Ben Gurion Univeristy of the Negev, Israel, [3]Palo Alto Research Center, USA

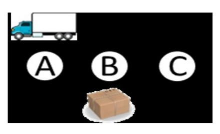
1. Background and Problem Definition

Classical Domain Independent Planning

Given:

- A ***domain*** specifying the *action model of every* actions
  - An **action model** specifies actions' precond. and effects
- A ***problem*** specifying the start state and goal conditions

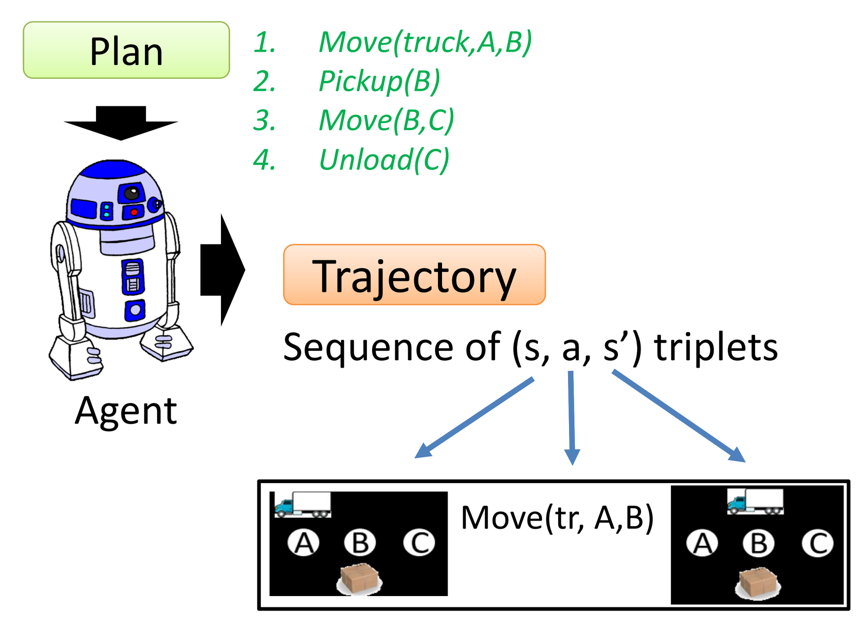
Output: a ***plan***, i.e., sequence of actions that achieve the goal

Example: the Logistics domain 

- Domain includes action model for move, pickup, & unload
- Problem can be "Package at location C" (At(Package, C))

Challenge: how to plan without a given action model?

Execution Traces (Trajectories)



Safe Model-Free Planning

Given:

- A ***problem* P** in some unknown domain D
- A set of **trajectories** executed in D

Not given: the real action model M* in D

Output: a ***plan** for P that is **sound** w.r.t M**

Prior work: Safe Action Model (SAM) Learning [Stern & Juba '17]

- Learns a **safe action model** M_{safe} from trajectories
- **Sound:** plans generated by M_{safe} are sound w.r.t M*
- **Incomplete:** M_{safe} may be too weak to find a plan
- Does not generalize between instances of the same action

➔ Move(A,B) and Move(C,D) are considered different actions

2. SAM Learning for Grounded Domains

Def: an action model M is safe if an action A is applicable in state S iff:
(1) A is applicable at S in the real action model (M*)
and (2) The effects of applying A in state S are the same for M and M*

Inference rules for grounded domains

Given a state-action-state triplet (s, a, s'):

- Rule 1. [Not a precondition] $\forall l \notin s: l \notin pre(a)$
- Rule 2. [Not an effect] $\forall l \notin s': l \notin \text{eff}(a)$
- Rule 3. [Must be an effect] $\forall l \in s' \setminus s: l \in \text{eff}(a)$

SAM Learning:

1. For each action a
1.1. Initialize pre(a) to be all literals
1.2. Initialize eff(a) to be an empty set
1.3. For each state-action-triplet (s,a,s')
1.3.1. Apply Rule 1 to remove literals from pre(a)
1.3.2. Apply Rule 3 to add literals to eff(a)

3. Lifted Domains in Classical Planning

A lifted domain defines

- A set of types (e.g., truck, package) and objects (truck1, package_A)
- Lifted literals (e.g., At(x?, y?), On(x?,y?))
- Lifted actions (e.g., Move(truck?, location?, location?))

Note: trajectories consists of **grounded actions and literals**

A **grounding** of a literal/action is a pair (lifted literal/action, **binding**)

- A binding maps literal/action parameters to concrete objects
- Ex.: At(tr1, A) = (At(truck?, loc?), [truck?:tr1, loc?:A])
- Ex.: Move(tr1, A,B) = (Move(truck?, loc1?, loc2?), [truck?tr1, A:loc1?, B:loc2?])

Preconditions and effects of lifted actions are **parameter-bound literals**

- A parameter-bound literal is a pair (lifted literal, parameters binding)
- Ex: precond. of Move(truck?,loc1?, loc2?) is (At(x?,y?), [x?: truck?, y?:loc1?]

Def: for a grounded action $\langle A, b_A \rangle$ and a grounded literal $\langle L, b_L \rangle$, let $\boldsymbol{bindings(b_A, b_L)}$ denote the set of all possible parameter-bindings between this action and literal, i.e., $\{b_{LA} | b_A \circ b_{LA} = b_L\}$.

4. SAM Learning for Lifted Domains

Inference rules for lifted domains

Given a state-action-state triplet $(s, (A, b_A), s')$:

Rule 1. $\forall (L, b_L) \notin s$:
$\forall \boldsymbol{b_{LA}} \in binding(b_A, b_L): (L, b_{LA}) \notin pre(a)$

Rule 2. $\forall (L, b_L) \notin s'$:
$\forall \boldsymbol{b_{LA}} \in binding(b_A, b_L): (L, b_{LA}) \notin \text{eff}(a)$

Rule 3. $\forall (L, b_L) \in s' \setminus s$:
$\exists \boldsymbol{b_{LA}} \in binding(b_A, b_L)$: $(L, b_{LA}) \in \text{eff}(a)$

If b_A is injective, this means $|bindings(b_A, b_L)| = \{0,1\}$
➔ Rule 3 becomes simply add (L, b_{LA}) to list of effects

5. Theoretical Properties

Under the injective binding assumption, it holds that

1. SAM learning returns the **strongest** possible **safe** action model
2. Planning with the returned safe action model is **sound** but **complete**
3. Approximate completeness: the number of trajectories required to learn the action model is linear in the size of the domain model
   ➔ Key: # trajectories does not depend on the number of domain objects.

Theorem 4. *Under the injective binding assumption, given* $m \geq \frac{1}{\epsilon}(2\ln 3|\mathcal{F}||\mathcal{A}|k^d + \ln\frac{1}{\delta})$ *trajectories sampled from* $\mathcal{T}_D$*, then with probability at least* $1-\delta$ *SAM learning for lifted domains (Algorithm 1) returns a safe action model* M_{SAM} *such that the probability of drawing from* $\mathcal{P}_D$ *a problem that is not solvable with* M_{SAM} *is at most* ϵ.

6. Multiple Action Bindings

What if b_A is not bijective?

- $\boldsymbol{bindings(b_A, b_L)}$ is a non-trivial set
- Example: consider a lifted action A(x?,y?)

T1= $\langle \{\}, A(o,o), \{L(o)\} \rangle$
T2= $\langle \{L(o_1)\}, A(o_1, o_2), \{L(o_1)\} \rangle$
➔ Either L(x) is an effect or L(y) or both (Rule 3)
➔L(y) is not an effect (Rule 3)
➔L(x) is an effect!

7. Extended SAM Learning

- Generate a CNF for every lifted action describing knowledge of effects
- Create proxy actions to maintain the predictability of the effects.

Example: assume we learn for action A the following CNF:

(**IsEff(L1(x))** Or **IsEff(L1(y))**) AND (**IsEff(L2(z))** Or **IsEff(L2(w))**)

The generated proxy actions are::

$A_{x=y}$: eff={L1(x)}, pre={L2(z), L2(w)}, merge x=y
$A_{z=w}$: eff={L2(z)}, pre={L1(X), L1(Y)}, merge z=w
$A_{x=y, z=w}$:eff={L1(x), L2(z)}, pre={}, merge x=y, z=w

Algorithm 3: Extended SAM Learning

Input : $\Pi_{\mathcal{T}} = \langle T, O, s_I, s_g, \mathcal{T} \rangle$
Output: (pre, eff) for a safe action model

```
A' ← all lifted actions observed in T
foreach lifted action A ∈ A' do
    (CNF_pre, CNF_eff) ← ExtractClauses(A, T(A))
    CNF^1_eff ← all unit clauses in CNF_eff
    SurelyEff← {l | IsEff(l) ∈ CNF^1_eff}
    SurelyPre← {l | IsPre(l) ∈ CNF_pre}
    /* Create proxy actions for non-unit effects clauses */
    CNF_eff ← CNF_eff \ CNF^1_eff
    foreach S ∈ Powerset(CNF_eff) do
        pre(A_S) ← SurelyPre; eff(A_S) ← SurelyEff
        foreach C_eff ∈ CNF_eff \ S do
            foreach IsEff(l) ∈ C_eff do
                Add l to pre(A_S)
        MergeObjects(S, pre(A_S), eff(A_S))
return (pre, eff)
```

8. Preliminary Experimental Results

- Domains: N-Puzzle (3x3 tiles, 3 predicates, 1 action), Blocksworld (8 blocks, 5 predicates, 4 actions)
- Baseline: FAMA (Aineto et al. 2019), outcome may be a unsafe

Results:

- N-Puzzle: both algorithms find the action model with a single (s,a,s') triplet.
- Blocksworld: SAM learning able to find the action model with fewer trajectories

9. Summary and Future Work

SAM learning can learn lifted action models!

- Number of trajectories is quasy-linear in number of lifted actions
- Does **not depend on the number of objects in the world!**

Direction for future work: safe model-free planning in domains with continuous state variables, non-determinism, and partial observability.