Learning Safe Action Models

Brendan Juba

Washington University in St. Louis

Based on joint works with

Hai S. Le, Washington University in St. Louis

Roni Stern, Ben Gurion University

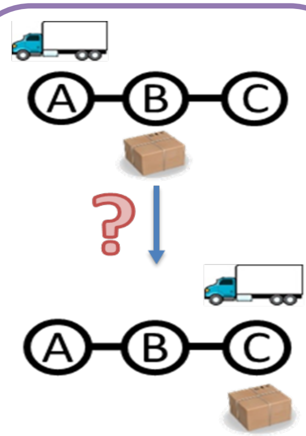
Argaman Mordoch, Ben Gurion University

Enrico Scala, University of Brescia

Outline

1. **Problem: learning a safe action model**
2. Algorithm for basic STRIPS models
3. Learnability of representations
4. Learnability of conditional effects

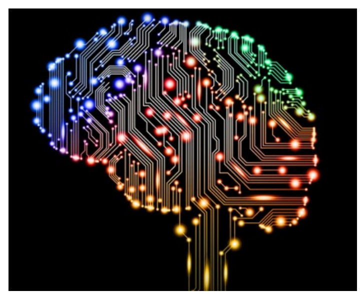
Domain Independent Planning



Real World Planning Problem

PDDL

Action: Move(A,B)
Pre: TruckAt=A
Eff: TruckAt=B
Action: Pickup(B)
Pre: (TruckAt=B,
PackageAt=B)
Eff: (PackageAt=Truck)
.....



AI Planner

Plan

1. Move(A,B)
2. Pickup(B)
3. Move(B,C)
4. Unload(C)

STRIPS action models

- **Domain**: *Actions* A and "*Fluents*" F (relations)
  - Parameterized by fixed set of (typed) arguments e.g., Move(truck t, location x, location y)
    At(truck t, location x)
- **Action Model**: *preconditions* and *effects* for each action in A
  - **Preconditions**: a conjunction of literals on fluents.
  - **Effects**: a set of literals on fluents.
  - We assume these fluents' arguments are action parameters (e.g., as above).

STRIPS problems

- **Domain**: *Actions* A and “*Fluents*” F (relations)
- **Action Model**: *preconditions* and *effects* for each action in A
  - **Preconditions**: a conjunction of literals on fluents.
  - **Effects**: a set of literals on fluents.
- **Problem**: *objects* O, initial *state* s_0, and *goal* g.
  - **Grounded fluents**: fluents bound to $o \in O$
  - **States**: Boolean valuations of grounded fluents
  - **Goal**: a conjunction of literals on grounded fluents
- **Plan**: a sequence of actions bound to $o \in O$.

STRIPS axioms

- **Trajectory**: sequence of states $s_0,\ldots,s_T$ and grounded actions $a_1,\ldots,a_{T-1}$ satisfying the *STRIPS axioms*: for each triple $\langle Pre, Action, Post \rangle$

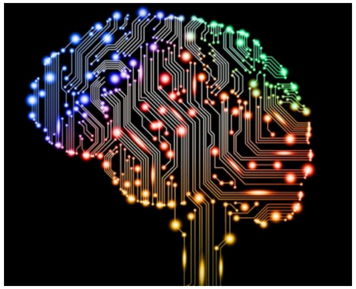
1. The precondition of action a_t is satisfied in s_{t-1}
2. Each effect literal of action a_t is satisfied in s_t
3. Each literal that is not an effect of action a_t and satisfied in s_{t-1} is satisfied in s_t

- **Solution** to a planning problem: a plan s.t. the trajectory with given s_0 satisfies $g(s_T)$

Domain Independent Planning

Real World
Planning Problem

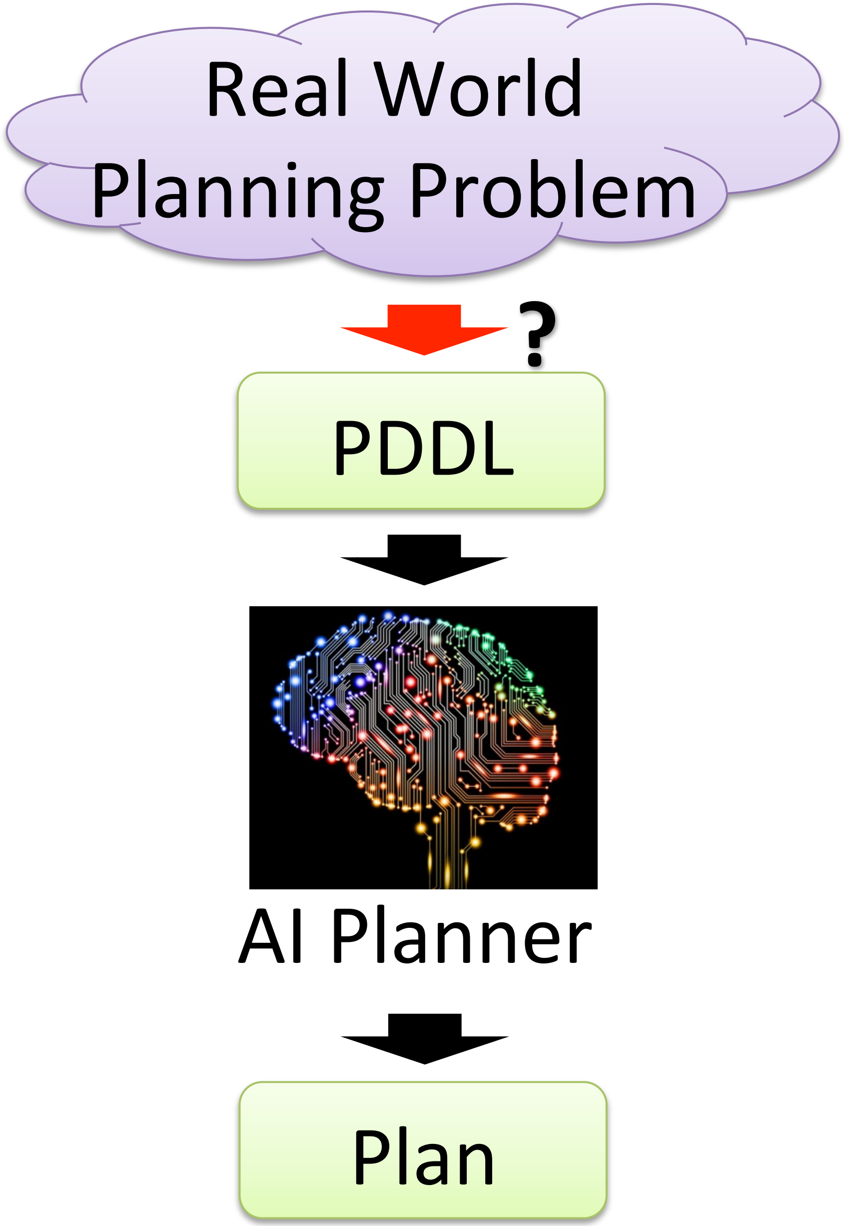
PDDL



AI Planner

Plan

The Problem: How to Model the World



Naïve approach 1

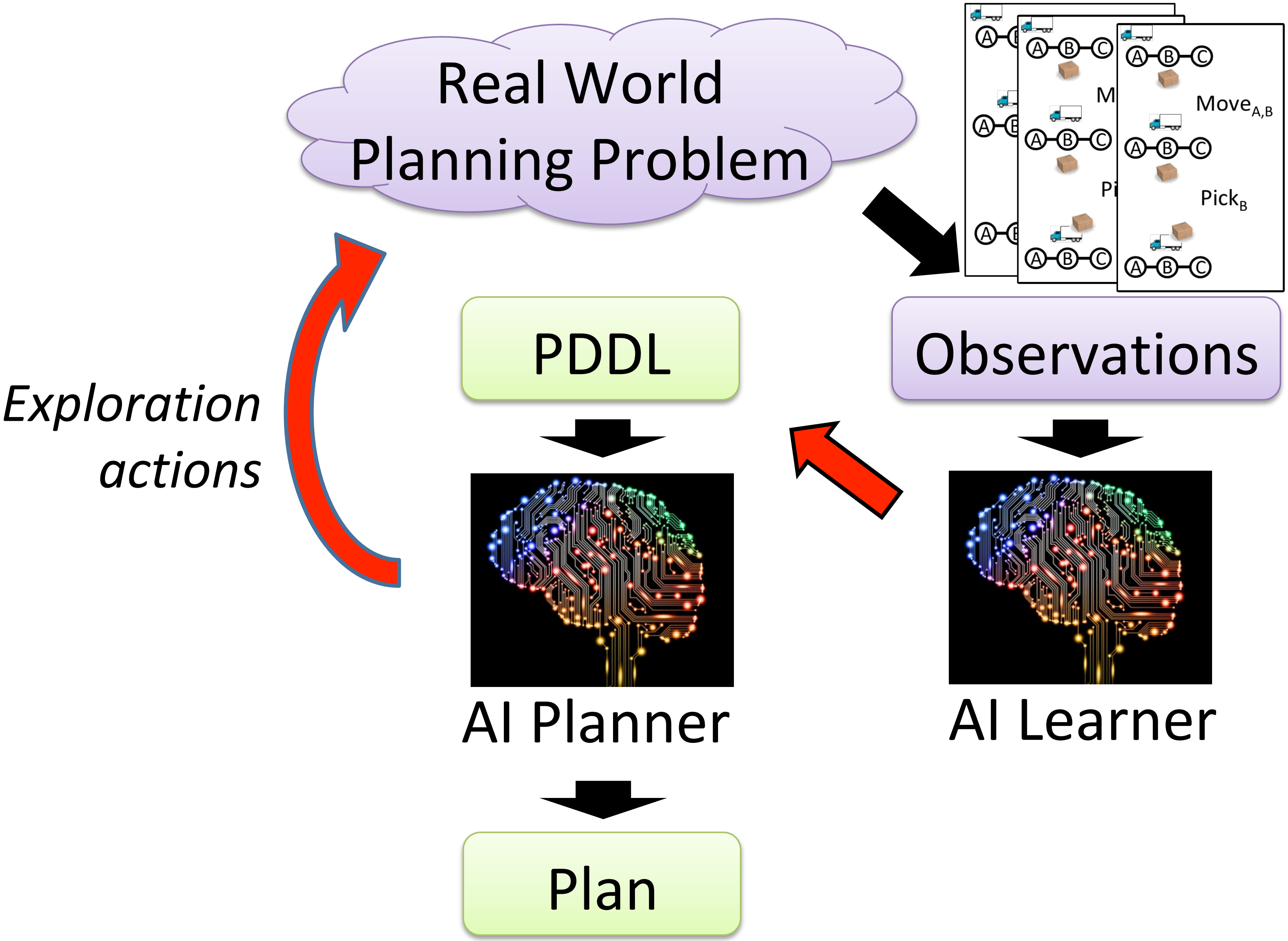
- Supervised learning? Pre-state ⟷ input, post-state ⟷ output, i.e., $s_{t+1} = a(s_t)$
  - ✖ Distribution shift: plans that reach erroneous states may appear better to planner
  - ✖ Preconditions are always true in trajectories!

Prior work on learning planning models

- **Learning a PDDL model** and planning with it, e.g., FAMA (*Aineto* et al.), ARMS (*Yang* et al.*),* LOCM (*Cresswell* et al.)
  - **Trial-and-error** + analysis of **past plan executions**

- **Reinforcement learning**

Learning to Plan



Prior work on learning planning models

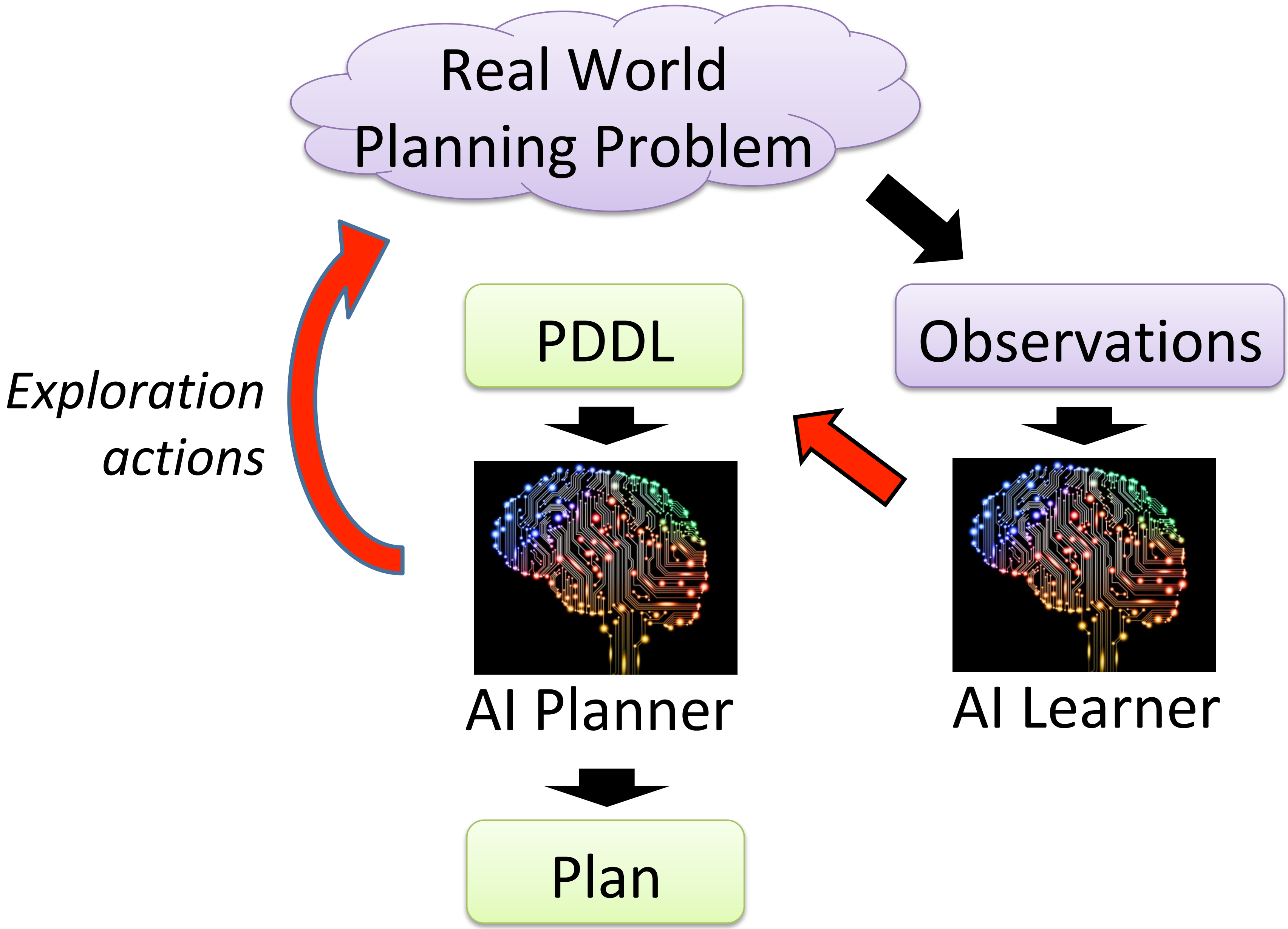
- **Learning a PDDL model** and planning with it, e.g., FAMA (*Aineto* et al.), ARMS (*Yang* et al.*),* LOCM (*Cresswell* et al.)
  - **Trial-and-error** + analysis of **past plan executions**

Not Safe

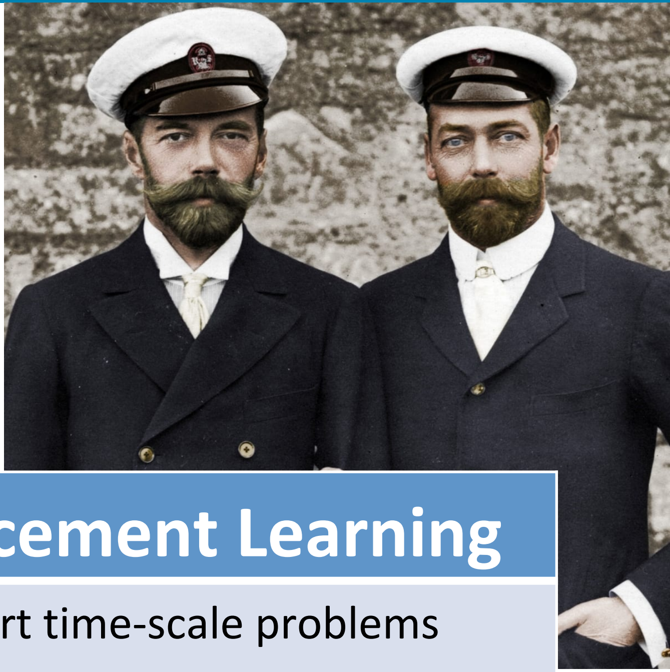
- **Reinforcement learning**

Not Safe

Risky exploration is not an option



A cousin: Offline Reinforcement Learning



- *Exploration is unsafe*: conservative learning with trajectories from an expert

Learning Planning Models	Reinforcement Learning
High-level, long time-scale problems	Low-level, short time-scale problems
Produces reusable domain model	Produces policy optimizing fixed reward
Large, declaratively described state space	Small unstructured state space *or* linear structured (small basis) state space
Little training data required	Data-intensive
Assumes known domain structure (predicates, signatures, etc.)	Few assumptions necessary: raw sensor data

Naïve approach 2

- Exact learning?
  - ✖ Impossible: example trajectories may not be adequate to identify action model
    E.g.: if fluent f is always true when action a is taken and remains true afterwards,
    f may or may not be an effect of a.

Key Assumption: Stationary Distribution

Train and test are drawn from the same distribution

What does this mean in planning?

- There is a **distribution D over planning problems**
- We are given **trajectories executing plans** solving problems **drawn from D**
- **Future problems** will also be **drawn from D**

Action Model Learning

- Given: trajectories of plans for problems from D

  Do **A** at state S_1 and get to state S_2

  Do **B** at state S_2 and get to state S_3

  …

- Not given: how actions change the world

  What does **A** do? **(effects)**

  When can I do **A**? **(preconditions)**

- **Task: learn an action model for planning**

Safe Action Model Learning Guarantees

- **Safe** action model: Only allow action *a* if
  - **you are sure it satisfies the precondition** and
  - **you know what the post state will be**

- Probably Approximately **Complete** action model:
  - **(Probably)** w.p. 1-δ learn an action model s.t.
  - **(Approximately Complete)** w.p. 1-ε action model permits solving a new problem drawn from D

Remark: action model = nondeterministic alg.

- *Fluents*: HeadAt(i), σAt(i) for each σ in Σ, Adjacent(i,j), Stateq for each q in Q.
- *Actions*: Δσqk(i,j) for each σ in Σ, q in Q, k = 1,…,[max relation size]
  - **Pre:** HeadAt(i), σAt(i), Adjacent(i,j) or Adjacent(j,i)
  - **Effect:** HeadAt(j),¬HeadAt(i), σ'At(i), ¬σAt(i)
- *Problem*: let o contain tape cells i=1,…,p(n), s_0 encode input x, Stateq_0, Adjacent(i,i+1) goal: StateAccept
- Plan exists ⟺ NTM accepts x in space p(n)

Outline

① Problem: learning a safe action model

② **Algorithm for basic STRIPS models**

③ Learnability of representations

④ Learning models of stochastic environments

Learning **Safe Models**

STRIPS axioms ➔ Conservative Learning

1. Each precondition literal of action *a* is satisfied in pre

1. Each literal <u>un</u>satisfied in pre <u>isn't</u> a precondition of *a*

2. Each effect literal of action *a* is satisfied in post

2. Each literal <u>un</u>satisfied in post <u>isn't</u> an effect of *a*

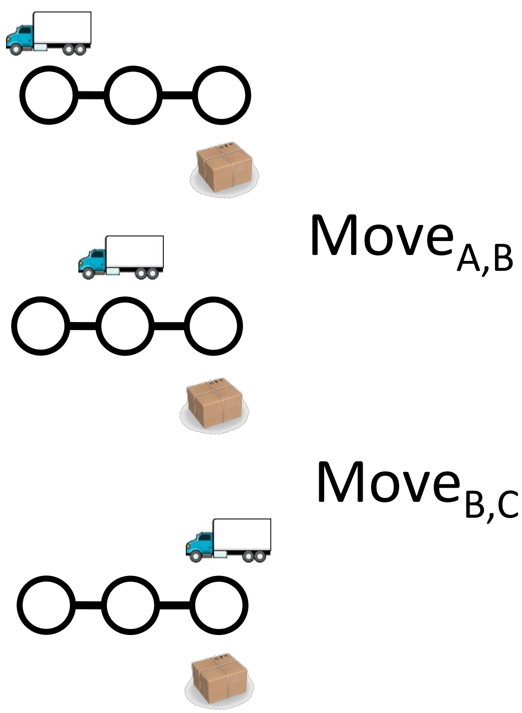
3. Each literal that is not an effect of action *a* and satisfied in pre is satisfied in post

3. Each literal that is satisfied in post but <u>not</u> pre <u>is</u> an effect of *a*

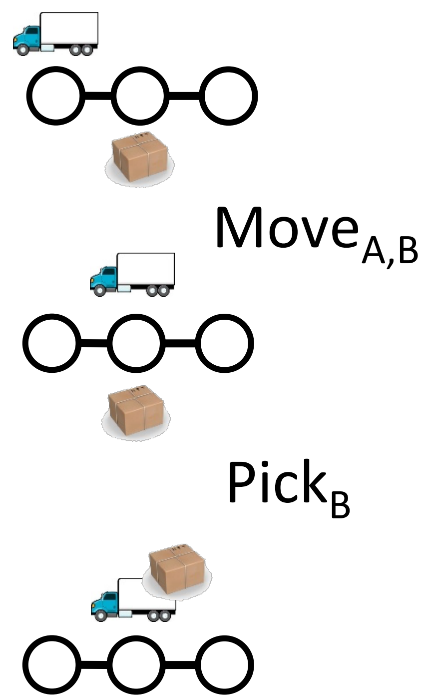
Learning **Safe Models**

STRIPS axioms ➔ Conservative Learning

Observation #1



Observation #2



Learned action model

$\mathrm{Move}_{A,B}$

Pre:

At(truck,A),

At(package,C)

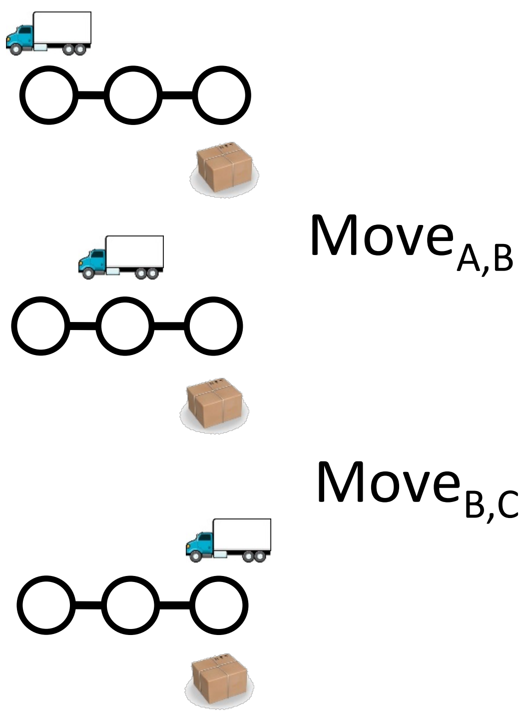
$\mathrm{Move}_{B,C}$

...

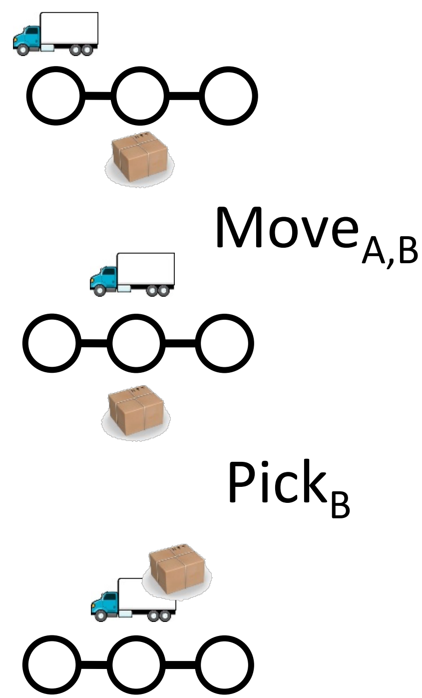
Learning **Safe Models**

STRIPS axioms ➔ Conservative Learning

Observation #1



Observation #2



Learned action model

$\mathbf{Move}_{A,B}$

Pre:

At(truck,A)

$\mathbf{Move}_{B,C}$

…

$\mathbf{Pick}_{B}$

Learning Safe Models

Conservative Learning ➔ Safe Models

1. ***l*** is a precondition of ***a*** if it **holds in all pre-states**

$$pre(a) \subseteq \bigcap_{\langle s,a,s' \rangle \in \mathcal{T}(a)} s$$

2. ***l*** is an effect of ***a*** if it holds in a **post-** but **not pre-state**

$$\bigcup_{\langle s,a,s' \rangle \in \mathcal{T}(a)} s' \setminus s \subseteq \mathit{eff}(a)$$

Key point: **every effect either appears in every pre-state or is absent in one.**
If it ever is absent from a pre-state, it is included in the effects of *a*.
If it is always present in the pre-state, it is in the conservative precondition.
Then since the negation is *not* also an effect, the fluent holds in the post-state.
***Therefore*: the model always predicts this fluent holds, so the model is *safe*.**

Safe models can be learned efficiently

Theorem: Given $m \geq {}^{1}/_{\varepsilon}(2 \ln 3\ |A|\,|F|\ + \ln{}^{1}/_{\delta})$ trajectories, with probability at least 1-δ the conservative action model permits solving a new problem with probability at least 1-ε.

Furthermore, we can find the conservative action model in $O(m|F| + |A|\,|F|)$ time.

(*A: set of actions, F: set of fluents*)

- **(Probably)** $\mathbf{1-\boldsymbol{\delta}}$ prob. to learn an action model s.t.
- **(Approximately)** $\mathbf{1-\boldsymbol{\epsilon}}$ prob. to solve a given problem

Proof Outline: Adequate Action Models

An action model is **adequate for plans Π from D** iff w.h.p. its pre-conditions allow executing Π

Lemma: with high probability,
the conservative action model is **adequate for D**

Proof Outline: Adequate Action Model

An action model is ε-***adequate*** for D iff the prob. $pre_L(\boldsymbol{a})$ doesn't hold for some ***a*** in **Π** is ≤ ε

The learned pre-conditions

Lemma: an action model that isn't ε-**adequate for D** is consistent with trajectories with probability ≤ $e^{-\varepsilon m}$

Proof outline:

- If the action model is not ε-adequate, some ***a*** was not observed in a state **s** with $pre_L(\boldsymbol{a}) \not\subset$ **s**

☞By definition, consistent w.p. **≤ $(1-\varepsilon)^m$ ≤ $e^{-\varepsilon m}$■**

Proof Outline: Counting Action Models

Theorem: Given $m \geq {}^{1}/_{\varepsilon}(2 \ln 3\ |A||F| + \ln {}^{1}/_{\delta})$ trajectories, with probability at least $1\text{-}\delta$ the conservative action model permits solving a new problem with probability at least $1\text{-}\varepsilon$

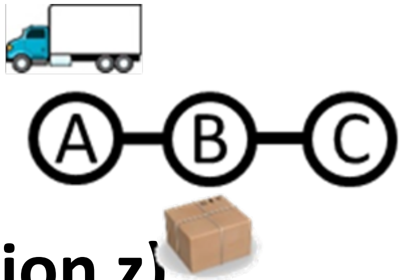
Lemma: an action model that isn't ε-**adequate for D** is consistent with trajectories with probability $\leq e^{-\varepsilon m}$

Proof Outline:

- There are $3^{|F|}$ possible preconditions and $3^{|F|}$ possible effects per action: $3^{2|F||A|}$ models.
- The conservative action model can be each of the ε-inadequate models with probability $\leq \delta/3^{2|F||A|} \Rightarrow \varepsilon$-adequate with probability $1\text{-}\delta$ ■

Lifted Planning Domain

- The number of grounded actions for a lifted action grows polynomially with the number of domain objects:



Move(truck x, location y, location z):

Move(truck, A, B)	Move(truck, C, A)
Move(truck, A, C)	Move(truck, C, B)
Move(truck, B, C)	Move(truck, B, A)

How do we learn a lifted representation?

Lifted Action Model Learning: Assumptions

- Action model is given by lifted literals but trajectories are given by grounded literals.
- Infer lifted literals for the action model from grounded literals that appear in trajectories using two assumptions:
  - All the preconditions and effects are bound to action parameters.
  - The bindings are unique: the same object is not bound to two different parameters of the action. (*injective binding assumption*)

Lifted Action Model Learning: Algorithm

- Since we assume bindings are injective, *bindings may be inverted*: ground literals can be mapped back to lifted literals with action parameters (*only parameter-bound literals in* eff *and* pre)
- Apply the inference rules on the lifted literals: For each (s,*a*,s') triplet in the trajectories,
  - Mark each lifted literal falsified in *s* as not in $\text{pre}_L(a)$
  - Mark each lifted literal satisfied in *s'* but not *s* as in $\text{eff}_L(a)$

Example

(pre-state, action, post-state)

At(package, B)	*At(package, B)*
At(truck, A)	*At(truck, B)*
Not(At(truck, B))	*Not(At(truck, A))*
Not(At(truck, C))	*Not(At(truck, C))*
Not(At(package, A))	*Not(At(package, A))*
Not(At(package, C))	*Not(At(package, C))*
Not(On(truck, package))	*Not(On(truck, package))*

Move(object x, loc y, loc z):
Precondition: At(x, y), ¬At(x, y),
At(x, z), ¬At(x, z)
Effect: ∅

SAM
Learning

Move(object x, loc y, loc z):
Precondition: At(x, y),
¬At(x, z)
Effect: At(x, z), ¬At(x, y)

Outline

1. Problem: learning a safe action model
2. Algorithm for basic STRIPS models
3. **Learnability of representations**
4. Learnability of conditional effects

Learning Theory Questions

- How **rich** a family of action models can be learned **safely** and **efficiently**?
  - What classes of **preconditions** can be learned safely and efficiently?
  - What classes of **effects** can be learned safely and efficiently?
  - Can we learn **probabilistic effects**? *(not today…)*
- *Other questions, also not today: other observation models? E.g., partial info, noise,…*

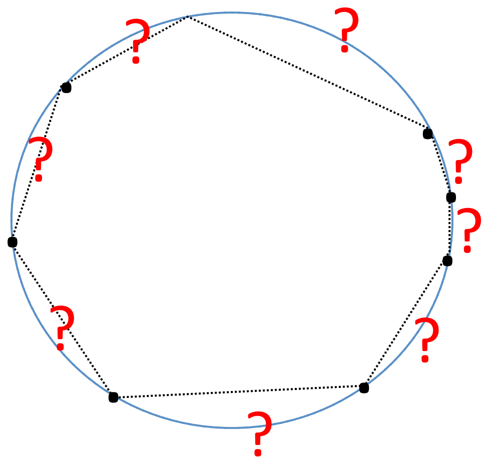
Preconditions: positive-only learning

Observation 1: supervised learning from positive examples for *C* reduces to safe action model learning with preconditions from *C*.

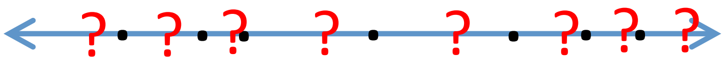
- *Natarajan '91*: Optimal safe concept is the intersection of all consistent concepts.
- E.g., halfspaces ➡ convex hull
- *Kearns-Li-Pitt-Valiant '87*: disjunctions not learnable

Learnability of numeric preconditions

- Axis-aligned boxes are learnable: $X_i[\min_i,\max_i]$
- *Goldberg '92*: Halfspaces are not learnable, even in two variables.



- *Kivinen '95*: Complements and unions of (two) intervals are not learnable.



Effects: VC-dimension/approx. interpolation

Observation 2: "approximate interpolation" for *C* reduces to safe action model learning with effects from *C*.

Anthony, Bartlett, Ishai, Shawe-Ta

Boolean *f*:
d is VC-dim(C)

If *d* is the VC-dimension of the fam

$$\left\{ I\left[\left\| f^*(x) - f(x) \right\|_\infty \le \varepsilon \right] : f \in C \right\}$$

(where also $f^* \in C$) then $\Omega\left(\frac{1}{\delta_1}\left(d + \log\frac{1}{\delta_2}\right)\right)$ examples are necessary to obtain with probability 1-δ_2 a ε-approx. interpolation of *C* on a 1-δ_1-probability set.

Open question: condition for safety in general?

Extension to numeric domains (char. 0)

Theorem. Suppose observations are errorless. Then, there is a polynomial time and sample complexity algorithm for learning safe action models where

- Preconditions: Conjunctions of univariate intervals with linear equality constraints
- Effects: Affine functions

Extension to numeric domains (char. 0)

- Preconditions: Conjunctions of univariate intervals with linear equality constraints
- Effects: Affine functions

Algorithm: For each action a,

- For each coordinate use max/min as interval
- Write subspace containing pre-states in examples as linear constraints
- Solve for affine effects on subspace
  - *Note: data has full rank on the subspace it spans*

Safety in Numeric Domains

- For each coordinate: max/min interval contained in true precondition interval
- Subspaces closed under span ⇒ true subspace contains span of observed data

✓ *Preconditions are safe*

- Observed data has full rank on its span ⇒ affine transformation for effects is uniquely determined on pre-states satisfying precondition.

✓ *Effects are also safe*

Completeness in Numeric Domains

- Subspaces have VC-dim. n in $\mathbf{R}^n$
- Intervals have VC-dim. 2
- VC-dim of conjunction ≤ sum of components
- Thus: *Preconditions have VC-dimension O(n)*
- **Standard VC-dimension bound** (*Vapnik-Chervonenkis/Blumer-Ehrenfeucht-Haussler-Warmuth*): Any candidate precondition consistent with $\Omega\left(\frac{1}{\delta_1}\left(n\log\frac{n}{\delta_1}+\log\frac{1}{\delta_2}\right)\right)$ trajectories is δ_1-adequate. *(Rest as before)*■

Outline

1. Problem: learning a safe action model
2. Algorithm for basic STRIPS models
3. Learnability of representations
4. **Learnability of conditional effects**

Conditional effects: definition

Back to Boolean domains.

- *Previously*: each *action* had a *precondition*, each *effect* was a *literal*
- A **conditional effect** is given by a pair: an **effect literal** and an **effect condition** where each *effect condition* is a **conjunction.**

Conditional effects: semantics

- A **conditional effect** is given by a pair:
  an **effect literal** and an **effect condition**
  where each *effect condition* is a **conjunction.**

STRIPS axioms with conditional effects:

1. Preconditions – as before

2. Each effect literal of action a_t **with a satisfied effect condition in s_{t-1}** is satisfied in s_t
3. Each literal that is not an effect of action a_t **with a satisfied effect condition in s_{t-1}** and satisfied in s_{t-1} is satisfied in s_t

Conditional effects are hard to learn safely

Theorem. Suppose an algorithm given *m* examples drawn from D for a domain with actions *A*, fluents *F*, and conditional effects of size up to *k* returns a safe action model such that with probability at least 1-δ, the model permits solving a new problem with probability at least 1-ε. Then

$$m \geq \Omega({}^{1}/_{\varepsilon}(|A||F|({}^{|F|}/_{3k})^{k} + \log {}^{1}/_{\delta}))$$

- *Exponential in precondition size k*

Construction for lower bound

- Fluents: three kinds
  - **Goal fluents**, one per action (except “no-op”)
  - **Forbidden fluents**, half of remaining
  - **Flag fluents**, half of remaining
- Initial states of problems (uniformly random):
  - All goal fluents false
  - Exactly one forbidden fluent false
  - k flag fluents true
- Goals: wp. 1-4ε, empty. Otherwise at random:
  - One goal fluent true, others false
  - False forbidden fluent must remain false

Sketch of lower bound

- Exactly one action achieves each nonempty goal; all others leave goal unattainable.
- If a set of flag fluents was not observed, consistent with a conditional effect that sets forbidden fluent for the goal's action

⇒Safe action models cannot solve the problem.

⇒Must observe ¾ of all settings of flag fluent/forbidden fluent/goal tuples.

- $\geq (^{|F|}/_{3k})^k \, ^{|F|}/_3 \, (|A|\text{-}1)$ such tuples.
- Only observe one per $^1/_{4\varepsilon}$ examples.■

Optimal algorithm for conditional effects

Theorem. For fixed k, for domains with conditional effects of size k and $(k+1)$-CNF preconditions, there is a polynomial-time algorithm that returns a safe action model with $(k+2)$-CNF preconditions (and effect conditions of size up to $|F|$). With probability at least $1-\delta$ the action model permits solving a new problem with probability at least $1-\varepsilon$ when given $\Omega(^1/_\varepsilon(|A||F|^{k+1} + \log {}^1/_\delta))$ example trajectories.

- *Asymptotically optimal sample complexity*

Overview of algorithm

- Track for each action:
  - (k+1)-width clauses that may be preconditions
  - For each candidate effect literal, k-width clauses
    - true whenever effect occurred and
    - false whenever effect failed to occur
- *Observation*: The action is safe if for all candidate conditional effects, either
  - The effect literal is satisfied in the pre-state
  - All remaining candidate clauses are false (conj.)
  - All remaining candidate clauses are true (k-CNF)
- Can rewrite as a (k+2)-CNF by distributing

The safe action model

- Each action has preconditions obtained by
  - Conjunction of all (k+1)-width clauses that may be preconditions with
  - (k+2)-CNF for each candidate conditional effect
- Each effect literal has a condition given by the conjunction of all remaining possible width-k conditions (*may be unsat. → remove it*)
- **Safety**: true precondition clauses included; when conditional effect clauses satisfied, each literal is determined, and conditional effect (only) fires when it occurs in true model.

Analysis of sample complexity

Idea: examine **state of data structure** tracking possible remaining clauses. If the corresponding precondition isn't 1-ε-adequate, then w.p. ε, the trajectory contains a state in which either

- An extraneous precondition clause is falsified
- Some (but not all) candidate conditions are false and the effect would be observed to occur
- Some (but not all) candidate conditions are true and the effect would be observed to not occur

- In each case, a clause is deleted (and clauses are never added back).

Analysis of sample complexity

- *Thus*: data structure states corresponding to inadequate action models eliminated by each example with probability 1-ε.
  - As before, only survive m examples w.p. $(1\text{-}\varepsilon)^m$
- There are $2^{O\left(|F|^{k+1}\right)}$ possible precondition states per action.
- There are $2^{O\left(|F|^{k}\right)}$ possible sets of clauses per effect literal, per action.

⇒Overall: $2^{O\left(|A||F|^{k+1}\right)}$ possible states.

⇒Union bound for $m \geq \Omega({}^1/_\varepsilon(|A|\,|F|^{k+1} + \log {}^1/_\delta))$ ■

Conclusion

- Task: learn a **safe planning model**
- Approach: learn a **conservative action model**
- Results: **safe**, **efficient** and **PAComplete** learning for
  - Relational STRIPS models
  - STRIPS models with conditional effects of small size
  - Numeric models with affine effects, interval and subspace preconditions

Future work

- Learning more general types of stochastic domains
  - E.g., distribution on small number of effect sets
- Learning from partial observations
- Learning in continuous state spaces, actions with duration, etc.